Método para la Construcción de Modelos de Tráfico de Red para la Evaluación de la Seguridad de Sistemas de Control de Procesos

Iñaki Garitano, Roberto Uribeetxeberria y Urko Zurutuza
Depto. de Electrónica e Informática
Mondragon Unibertsitatea
Email: igaritano, ru, uzurutuza@mondragon.edu

***Resumen*—Las Infraestructuras Críticas (ICs) de hoy en día se basan en las Tecnologías de la Información y Comunicación (TIC). Debido a la simplicidad y al ratio eficiencia–coste de las TIC, los Sistemas de Control y Adquisición de Datos (SCADA) han encontrado su camino en los Sistemas de Control de Procesos (SCPs) mediante el uso de hardware y software estándar. A pesar de ello, incidentes recientes como *Stuxnet, Duqu* o *Night Dragon* han revelado nuevas vulnerabilidades y escenarios de ataque contra los SCPs. Como consecuencia, se han realizado numerosos estudios de seguridad para los sistemas SCADA. Sin embargo, tal y como muestran los recientes eventos, la investigación sobre la seguridad de los sistemas SCADA resulta un desafío debido a la falta de entornos de experimentación apropiados. Este trabajo presenta un método para generar tráfico de red real en condiciones de laboratorio sin la necesidad de instalaciones de SCP. La mayor contribución de este trabajo consiste en ser la base de un futuro sistema de detección de anomalías y en ofrecer apoyo a la experimentación mediante la recreación de tráfico realista en medios simulados. La precisión y la fidelidad del enfoque propuesto ha sido validado mediante varios métodos estadísticos que comparan el tráfico predicho con el tráfico obtenido en una instalación real.**

I. Introducción

Las Infraestructuras Críticas como plantas de energía, redes inteligentes, plantas de tratamiento de agua o refinerías se gestionan mediante Sistemas de Control de Procesos. Al principio los SCPs eran entornos aislados que utilizaban hardware y software propietario. Hoy en día, los SCPs se basan en el uso de TICs, y utilizan protocolos estándar permitiendo conexiones remotas desde Internet. Esto aporta ventajas como la reducción de costes, el tiempo de implementación, el incremento de la eficiencia y de la interoperabilidad entre componentes. Pero también tiene desventajas. En términos de seguridad, la adopción de las TIC introducen una amplia gama de amenazas y vulnerabilidades a los SCPs. Eventos recientes como los gusanos *Stuxnet* [1], *Duqu* [2] y *Night Dragon* [3] demuestran claramente el impacto de los cyber ataques en los SCPs.

Los sistemas de detección de anomalías y específicamente los Sistemas de Detección de Intrusiones (IDSs) basados en modelos, utilizan modelos que describen el comportamiento esperado/aceptable de los sistemas o las redes de comunicación. Aunque estos modelos podrían ser utilizados de forma efectiva para detectar ataques que podrían causar violaciones contra las políticas de seguridad establecidas, su construcción requiere una gran cantidad de datos libres de ataques. Además, pequeños cambios en las instalaciones pueden requerir no solo la creación de nuevos modelos, sino también su validación en entornos reales. Estas operaciones pueden resultar difícilmente aceptables en entornos de producción que necesitan trabajar de manera fiable y con un ratio bajo de costo-eficiencia.

En este trabajo se propone una nueva técnica para la creación de modelos de tráfico real de sistemas SCADA. La mayor contribución de este trabajo es un algoritmo que genera modelos de tráfico basándose en aplicaciones SCADA reales desarrolladas por ingenieros. Una aplicación SCADA se puede considerar como la combinación de lógica y valores. Estás aplicaciones son utilizadas por el algoritmo como entrada, de manera que se generan descripciones de modelos incluyendo paquetes, clases de variables y tasas de actualización. Una de las principales ventajas de este enfoque es que los modelos de tráfico pueden ser regenerados de manera sencilla, sin tener que analizar el tráfico real. En términos de aplicabilidad, este método puede ser utilizado como base para desarrollar IDSs, apoyando la experimentación mediante la recreación de tráfico realista en medios simulados. La precisión y la fiabilidad del enfoque propuesto han sido validados mediante el uso de varios modelos estadísticos que comparan el tráfico predicho con el tráfico obtenido en instalaciones reales.

El resto del trabajo está estructurado de la siguiente manera. La Sección II comienza con la descripción de trabajos relacionados con IDSs basados en modelos. En la Sección III se presenta el enfoque propuesto introduciendo la arquitectura típica de los SCPs, se define la estructura general de las aplicaciones para los SCPs y se presenta el algoritmo. La Sección IV presenta el procedimiento que se ha seguido para la validación del algoritmo y los resultados experimentales obtenidos. Finalmente, se presentan las conclusiones y las líneas futuras en la Sección V.

II. Trabajos relacionados

Mahamood *et al.* [4] analiza los métodos existentes para medir el tráfico y propone distintas soluciones para aplicar técnicas de monitorización de tráfico de red para la seguridad de los sistemas SCADA. Este trabajo se centra en el análisis de las cabeceras de los protocolos y de los flujos de tráfico, agrupa los datos y extrae la información útil mediante algoritmos de

minería de datos. Continuando con las técnicas basadas en modelos, Roosta *et al.* [5] presentan un diseño de un IDS basado en modelos de redes de sensores utilizados en los SCPs. Define modelos en las distintas capas de la pila de red TCP, como la física, la de enlace y la de red. De esta forma, identifican los siguientes patrones de comunicación: master a esclavo, administrador de red a dispositivos de campo, HMI a master y comunicación de nodo a nodo. Aunque los trabajos previos no necesitan inspeccionar los datos útiles de los paquetes para la creación de los modelos, sus dos principales desventajas son la necesidad de datos de aprendizaje libres de ataques y el hecho de que no son capaces de predecir el tamaño de los paquetes ni la capacidad de red. Estos parámetros juegan un papel importante en la detección de ataques capaces de alterar los paquetes, e.g. cambiando registros adicionales sin afectar el flujo del tráfico.

Cheung *et al.* [6] describen tres técnicas de detección de anomalías. La primera, basada en modelos a nivel de protocolo, especifica los campos independientes del protocolo Modbus/TCP, las dependencias entre distintos campos y la relación entre múltiples preguntas y respuestas con el fin de extraer reglas del IDS Snort. La segunda, genera reglas de Snort describiendo los modelos de comunicación utilizados para la detección de ataques que violan patrones específicos. Finalmente, una técnica de disponibilidad servidor/servicio, crea un modelo de servidores y servicios disponibles y detecta cambios que puedan indicar reconfiguraciones maliciosas de los dispositivos Modbus. De la misma forma, Düssel *et al.* [7] proponen un detector de anomalías en tiempo real basado en datos útiles de los paquetes. Este sistema no depende de ningún protocolo y es capaz de detectar ataques desconocidos. Este método toma en cuenta la similitud entre los mensajes de la capa de comunicación de la red SCADA. Aunque las soluciones presentadas por Cheung *et al.* and Düssel *et al.* inspeccionan los datos útiles de los paquetes, son similares al trabajo realizado por Mahamood *et al.* en el sentido de no necesitar datos de entrenamiento libres de ataque.

El trabajo realizado por Valdes *et al.* [8] se enfoca en la generación de patrones de comunicación y propone un sistema de detección de anomalías para los SCPs. Los patrones que se toman en consideración incluyen las direcciones IP origen/destino además de los puertos origen/destino. El sistema detecta una anomalía en el caso de encontrar un nuevo patrón con una probabilidad inferior al umbral especificado. Al contrario de los métodos anteriores, el trabajo presentado por Valdes *et al.* al igual que la propuesta presentada en este artículo, no necesita datos de entrenamiento libres de ataque. Aun así, no es capaz de crear los modelos requeridos sin datos reales ni de predecir el tamaño de los paquetes ni la capacidad de la red.

III. Enfoque propuesto

En esta sección se presenta el algoritmo propuesto para la generación de modelos de tráfico de la comunicación entre los servidores SCADA y los PLCs. Primero se describe la arquitectura típica de los SCPs, las aplicaciones típicas de los SCPs y a continuación la metodología propuesta. Finalmente se presenta una visión general del algoritmo.

III-A. Arquitectura Típica de los Sistemas de Control de Procesos

Las arquitecturas modernas de los SCPs están compuestas por dos capas de control: (i) la capa física, compuesta por dispositivos hardware que actúan sobre el sistema; (ii) la capa lógica, compuesta por todos los componentes TIC y el software. La capa lógica utiliza protocolos SCADA para controlar y gestionar los dispositivos físicos. Esta capa está típicamente dividida en dos redes distintas: la *red de control* y la *red de proceso*. La red de proceso la componen los servidores SCADA (conocidos también como Masters SCADA) y las Interfaces Hombre Maquina (HMI). La red de control se compone de dispositivos que por una parte controlan los actuadores y los sensores de la capa física y por otra parte, ofrecen la "interfaz de control" a la red de proceso. Una red de control típica se compone por varios Controladores Lógicos Programables (PLCs). Desde un punto de vista operacional, los PLCs reciben datos desde la capa física, elaboran una "estrategia de actuación local", y envían comandos a los actuadores. Los PLCs ejecutan las órdenes que reciben desde los servidores SCADA (Masters) y adicionalmente proveen, cuando se les solicita, datos detallados de la capa física.

III-B. Aplicaciones de los Sistemas de Control de Procesos

Los SCPs están controlados por aplicaciones especialmente diseñadas para cada instalación industrial. Una aplicación es la combinación de lógica y valores. La lógica se representa como acciones condicionales dependientes de los valores que el controlador debe de ejecutar. Estas aplicaciones, a la vez que los sistemas de control de procesos son únicos. Incluso si la aplicación propuesta es la misma, dependiendo del programador, puede tener una lógica totalmente diferente y utilizar valores distintos, dando lugar a una amplia variedad de posibilidades. Aunque el número posible de aplicaciones es enorme, el tráfico resultante puede ser descrito mediante los siguientes tres componentes: clases de variables, número de variables y tasa de actualización (TA) de las variables.

- **Clases de variables:** tipo de variables que el sistema o el software de desarrollo de la aplicación es capaz de gestionar, e.g. `boolean`, `integer`, `real` y `string`. Dependiendo de la clase, las variables reservadas necesitan un número específico de bytes.
- **Número de variables:** con el objetivo de ahorrar recursos, los desarrolladores intentan usar el mínimo número de variables debido a que las variables consumen recursos de memoria y tiempo de proceso.
- **Tasa de actualización de las variables:** el tiempo en el cual el valor de cada variable es actualizada. Dependiendo del propósito, las tasas de actualización pueden ser las mismas para todas las variables, o pueden estar definidas para cada clase de variable o incluso para cada variable. Los desarrolladores deberían usar la máxima TA posible con el fin de ahorrar recursos.

En los SCPs, los servidores SCADA interrogan a los PLCs mediante protocolos de comunicación SCADA para actualizar el valor de las variables locales. Los servidores SCADA almacenan los datos para su presentación de manera entendible a operadores que se encuentran detrás de los dispositivos HMI.

Los protocolos de comunicación SCADA como Modbus/TCP y DNP3 pueden ser protocolos orientados a objetos o variables. En el caso de los protocolos orientados a variables, los servidores almacenan la dirección de memoria en el cual las variables están almacenadas. Con el objetivo de preservar el ancho de banda y los recursos del sistema, los servidores SCADA agrupan en una única petición la dirección de múltiples variables. Esto da lugar a que el tamaño de los paquetes sea proporcional al número de variables.

III-C. Metodología

El objetivo del algoritmo propuesto es generar un modelo del tráfico de red partiendo de una descripción de una aplicación real. Previamente se ha realizado un análisis exhaustivo del tráfico real entre servidores SCADA y PLCs de ABB. El tráfico analizado se ha recogido en la Plataforma EPIC (*Experimental Platform for Internet Contingencies*), banco de pruebas del Instituto para la Protección y la Seguridad de los Ciudadanos del *Joint Reseach Center* de la Comisión Europea.

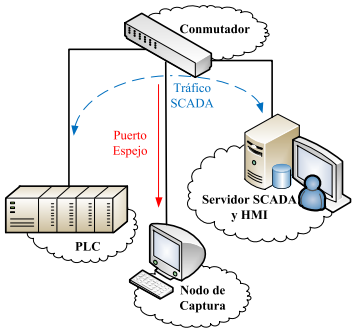

Figura 1. Topología de red experimental

Usando la plataforma EPIC, se ha construido la topología que se muestra en la Figura 1. Esta topología se compone de un PLC real, un servidor SCADA junto con el software HMI, un conmutador de red y un nodo dedicado a capturar el tráfico de la red. Todos los dispositivos han sido conectados a la misma red conmutada.

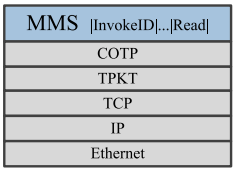

Figura 2. Capas del protocolo MMS

El PLC que se ha utilizado es el modelo ABB 800M con el protocolo MMS (*Manufacturing Message Specification*). Como se muestra en la Figura 2, el protocolo MMS se situa por encima de otros protocolos como COTP y TPKT. El tráfico analizado se ha generado utilizando varias aplicaciones SCADA. El Cuadro I resume el número de aplicaciones y sus especificaciones. Cada captura ha sido analizada en los siguientes términos:

- **Número de paquetes:** número de paquetes capturados.
- **Número de peticiones distintas:** número de peticiones distintas enviadas al PLC por el servidor SCADA.
- **Tamaño de cada petición (bytes):** tamaño en bytes de cada petición, incluyendo todas las capas.
- **Tiempo entre llegadas de paquetes (segundos):** tiempo promedio en segundos entre llegadas de paquetes.
- **Tiempo entre llegadas de paquetes para cada petición (segundos):** tiempo promedio entre los paquetes recibidos para cada tipo de petición.
- **Tamaño de cada paquete basado en las clases de variables (bytes):** tamaño del paquete según el número de variables y sus especificaciones (clase, TA).
- **Número de peticiones distintas basado en las clases de variables:** número de peticiones distintas teniendo en cuenta la combinación de varias clases de variables en la misma petición.
- **Número de peticiones distintas basado en las TAs:** número de peticiones distintas teniendo en cuenta que varias TAs de variables pueden combinarse en una misma tasa de actualización del sistema.

III-D. Generación de modelos de tráfico

El algoritmo genera modelos de tráfico basándose en las especificaciones de las aplicaciones en vez de en el análisis estadístico del tráfico. Como se ha explicado en la Sección III-B, cada aplicación puede ser descrita en términos de variables, clases de variables y tasas de actualización. Estas especificaciones son dependientes de cada aplicación y se utilizan para calcular el patrón de comunicación entre el servidor SCADA y el PLC. Aunque el análisis que se ha realizado se enfoca en el protocolo MMS, puede ser adaptado para otros protocolos, como por ejemplo Modbus.

El algoritmo propuesto toma como entrada los siguientes conjuntos y variables: U_v es el conjunto de las tasas de actualización de las variables; C es el conjunto de las clases de variables, donde cada elemento está formado por la clase de la variable y la longitud en bytes, expresado como (c, l), e.g. $(\texttt{integer}, 2)$; V es el conjunto de variables, donde cada elemento está formado por la clase de la variable y su correspondiente tasa de actualización, expresado como (c, u_v); U_s es el conjunto de las tasas de actualización del sistema; d es la duración de la traza (en segundos) que genera el algoritmo; y b es el ancho de banda. Como salida, el algoritmo genera un conjunto de descripción de paquetes P, donde cada paquete es definido por el tiempo de transmisión y el tamaño del paquete, expresado como (θ, σ). Por simplicidad, a lo largo del artículo se utiliza la notación X_y^z para expresar el componente y del elemento z del conjunto X. Por ejemplo, en el conjunto C, el elemento j se expresa como C^j, y el componente c del elemento j se expresa como C_c^j.

Cuadro I
DESCRIPCIÓN DE APLICACIONES UTILIZADAS PARA LA CONSTRUCCIÓN DEL ALGORITMO

Tamaño de aplicación	Número de variables	Clases de variables	Variables por TA	Número de TA de sistema	Número de aplicaciones
Pequeña	1-6	4	13	6	7
Media	7-60	4	12	6	4
Grande	61-120	4	6	6	8
Total					**19**

Algorithm 1 Generador modelo tráfico

1: $input :< U_v, C, V, U_s, d, b >$, $output :< P >$
2: **function** $Packet(i, c)$
3: $X := 0$
4: **for** ($j := 1$ to $|V|$) **do**
5: **if** ($V_c^j = c$) **then**
6: **if** (($i = 1$) AND ($V_u^j < U_s^2$)) **then**
7: $X := X + 1$
8: **end if**
9: **if** (($1 < i < |U_s|$) AND
10: ($U_s^i \leq V_u^j < U_s^{i+1}$)) **then**
11: $X := X + 1$
12: **end if**
13: **if** (($i = |U_s|$) AND ($U_s^i \leq V_u^j$)) **then**
14: $X := X + 1$
15: **end if**
16: **end if**
17: **end for**
18: **return** X
19: **end function**
20: **function** $MAIN$
21: $P := \emptyset$, $numP := 0$
22: **for** $j := 1$ to $|U_s|$ **do**
23: **for** $k := 1$ to $|C|$ **do**
24: $\sigma := C_l^k \cdot Packet(j, C_c^k)$
25: **for** $g := 1$ to (d/U_s^j) **do**
26: $numP := numP + 1$
27: $r := rand('Gauss', 0, 10^{-3})$
28: $\theta := (g-1)U_s^j + r$
29: $P := P \cup (\theta, \sigma)$
30: **end for**
31: **end for**
32: **end for**
33: time_sort(P)
34: **for** $j := 2$ to $numP$ **do**
35: $t := P_\sigma^{j-1}/b$
36: **if** $P_\theta^j < P_\theta^{j-1} + t$ **then**
37: $P_\theta^j := P_\theta^{j-1} + t$
38: **end if**
39: **end for**
40: **return** P
41: **end function**

El algoritmo comienza con la función *MAIN* inicializando el conjunto de las descripciones de los paquetes P y $numP$, donde este último se utiliza para contar el número de paquetes. A continuación, el algoritmo entra en un bucle y por cada tasa de actualización del sistema U_s^j y por cada clase de variable C_c^k llama a la función $Packet(j, C_c^k)$ donde obtiene el número de variables que se deben solicitar por cada paquete. Después, en la línea #24 basándose en el valor devuelto y el tamaño de la clase de variable C_l^k, se calcula el tamaño del paquete σ.

Los usuarios pueden especificar el tamaño de la traza, i.e. mediante la variable d, se calcula el número de transmisiones mediante la división de d por cada tasa de actualización U_s^j. En cada transmisión se incrementa la variable $numP$ y se genera un número aleatorio r utilizando una función de distribución Gaussiana. Este número aleatorio es utilizado para introducir una desviación realista a la tasa de actualización configurada. Las desviaciones tienen su origen en sistemas operativos multi-proceso, retrasos en las comunicaciones de red, etc., y son frecuentes en entornos reales. Basándose en este valor se calcula el tiempo de transmisión predicho θ y se añade una nueva descripción de paquete a P.

En la línea #33 se ordena el conjunto P respecto al tiempo y se actualiza el tiempo de transmisión de los paquetes con el fin de tener en cuenta el ancho de banda del sistema. Específicamente, se calcula el tiempo t requerido para enviar cada paquete P^{j-1} dividiendo el tamaño del paquete P_σ^{j-1} con el ancho de banda b. Una vez realizado el cálculo, si el tiempo requerido para enviar el paquete anterior excede el tiempo de transmisión del paquete actual, se actualiza este último como se muestra en la linea #37.

La función $Packet(i, c)$ se utiliza para calcular el número de variables enviadas en la i-esima tasa de actualización del sistema. Esta función devuelve el número de variables de la clase c para el cual la tasa de actualización satisface una de las siguientes tres condiciones. La *primera condición*, mostrada en la linea #6 es un caso particular y cuenta todas las variables cuya tasa de actualización es menor que la segunda tasa de actualización del sistema. La petición para todas estas variables será enviada en la primera tasa de actualización del sistema. La *segunda condición*, mostrada en la línea #9, cuenta las variables que se encuentran entre dos tasas de actualización consecutivas del sistema, i.e. U_s^i y U_s^{i+1}. Finalmente, la *tercera condición* cuenta todas las variables que tienen una tasa de actualización mayor que la última tasa de actualización del sistema, i.e. $U_s^i \leq V_u^j$ para $i = |U_s|$.

IV. VALIDACIÓN Y RESULTADOS EXPERIMENTALES

El algoritmo propuesto genera una secuencia ordenada de descripciones de paquetes, definidas por el tiempo y el tamaño.

Cuadro II
RESULTADOS EXPERIMENTALES DE LA COMPARACIÓN DEL TRÁFICO REAL Y PREDICHO

Aplicación		Número de paquetes	Tamaño de paquetes	Tasa de transferencia (bytes/seg.)			Error de tiempo entre llegadas de paquetes				
				Med.	Max.	Min.	50ms	500ms	1s	3s	6s
1	Predicho	29400	114	2793	2964	2736	**4.88 %**	**0.70 %**	**0.32 %**	**0.37 %**	**0.12 %**
	Real	29165	113/114	2758	5928	113					
	Error	**0.80 %**	**48.16 %**	**1.27 %**	**50 %**	**2321 %**					
2	Predicho	29400	132	3234	3432	3168	**8.34 %**	**0.28 %**	**0.32 %**	**0.06 %**	**0.20 %**
	Real	29421	132	3236	6336	660					
	Error	**0.07 %**	**0 %**	**0.07 %**	**46 %**	**380 %**					
3	Predicho	29400	152	3724	3952	3648	**4.53 %**	**0.15 %**	**0.08 %**	**0.01 %**	**0.01 %**
	Real	29407	152	3725	6688	760					
	Error	**0.02 %**	**0 %**	**0.02 %**	**41 %**	**380 %**					

Mediante la herramienta Matlab [9] se ha implementado un prototipo del algoritmo que se ha utilizado en el proceso de validación. La validación se ha realizado mediante varios métodos estadísticos comparando el modelo predicho con las trazas capturadas en una instalación SCADA real.

Se han utilizado tres aplicaciones distintas para validar el enfoque propuesto. Cada aplicación se compone de un número distinto de variables, clases de variable y tasas de actualización de variables. El tráfico real de cada aplicación se ha comparado con los modelos de tráfico generados en términos de número de paquetes, tamaño de paquetes, tasa de transferencia y tiempo entre llegadas de paquetes.

Con el fin de evaluar los errores entre el tráfico predicho y el tráfico real, se ha utilizado el Error Porcentual Absoluto de la Media (MAPE). De acuerdo a Lewis (1982) [10], cuanto menor sea el valor de MAPE más precisa es la predicción. De esta manera, se considera que un valor menor que el 10 % representa una precisión alta, mientras que un valor entre el 11 % y el 20 % se considera un buen resultado. Un valor entre el 21 % y el 50 % se considera razonable y los valores mayores que el 51 % muestran una clara imprecisión.

IV-A. Número de paquetes

Como se muestra en el Cuadro II el error calculado para el número de paquetes predicho es menor que el 1 %. Esto se debe a que incluso si el número de variables, clases de las variables y las TAs de las variables son diferentes para cada aplicación, una misma petición puede actualizar distintas variables generando el mismo número de paquetes.

IV-B. Tamaño de los paquetes

Para las aplicaciones dos y tres, el tamaño de los paquetes es idéntico entre las aplicaciones reales y las predichas, siendo el error del 0 %. Debido a que el protocolo analizado contiene un campo dinámico, i.e. el número de secuencia, el cual varía su tamaño, la primera aplicación muestra un error del 48.16 %. El algoritmo propuesto no tiene en cuenta la variación de los tamaños de los paquetes y debido a ello, casi la mitad de los paquetes reales son de un tamaño distinto. Sin embargo, una versión mejorada del algoritmo puede tener en cuenta los tamaños dinámicos, pero ello requiere más investigación, considerándose parte del trabajo futuro.

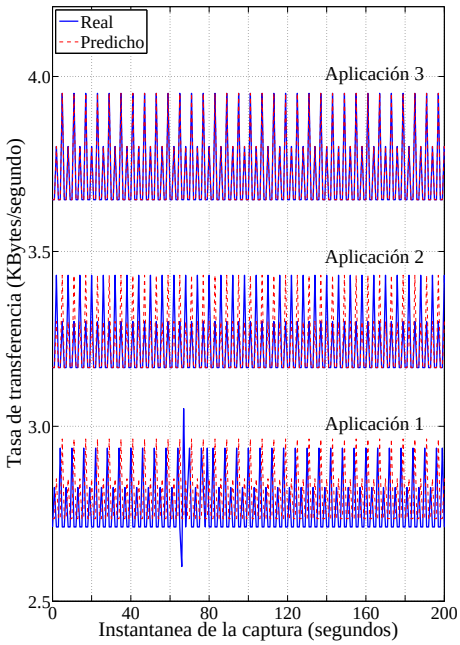

Figura 3. Comparación de las tasas de transferencia

IV-C. Tasa de transferencia

La tasa de transferencia predicha puede ser extraída dividiendo la salida del algoritmo en bloques de un segundo y sumando el número de bytes. Como se muestra en el Cuadro II, en todos los casos el error del promedio de la tasa de transmisión es menor que el 2 %, mostrando el nivel de precisión del algoritmo propuesto. Sin embargo, los errores de la tasa de transmisión máxima son mayores que el 41 %, mientras que el error mínimo de la tasa de transmisión de la primera aplicación es de 2321 %. Estos valores son debidos a las operaciones de los sistemas reales, en los cuales el tiempo de transmisión puede sufrir retrasos, causando una caída en el valor de la tasa de transmisión. Los paquetes omitidos en el segundo anterior son transmitidos en el instante posterior, causando un incremento en el valor de la tasa de transmisión. Sin embargo, en los segundos posteriores el error vuelve a valores menores que el 2 %. La Figura 3 muestra un claro

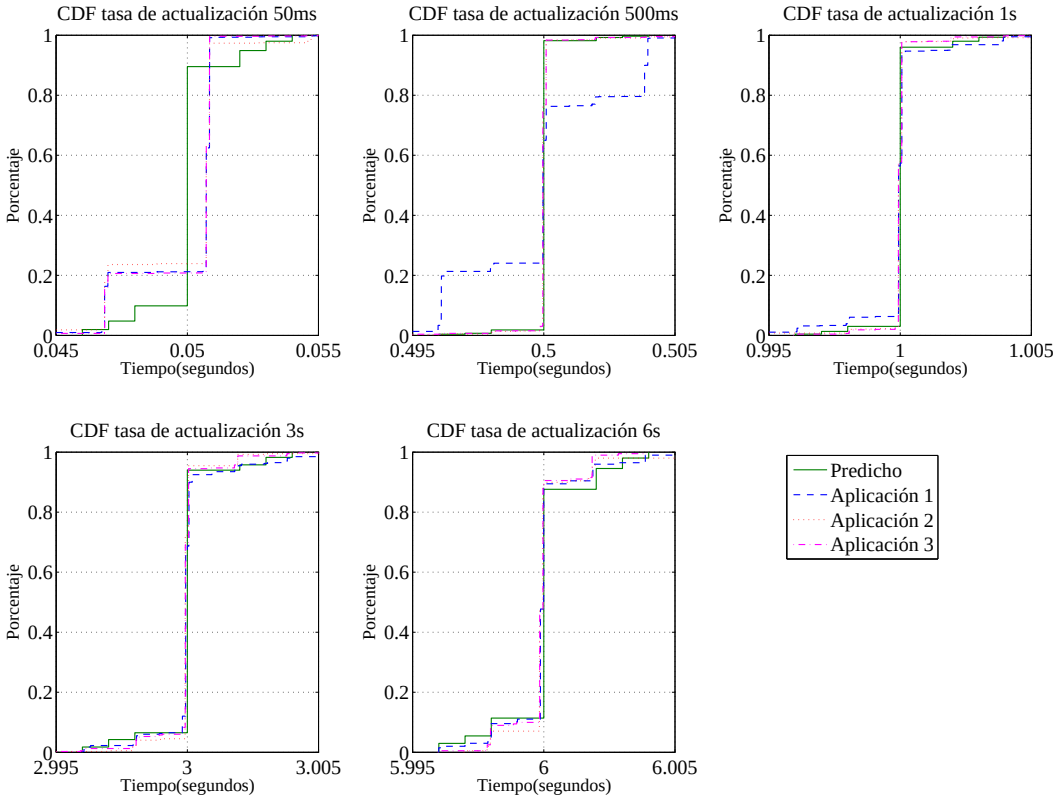

Figura 4. Función de Distribución Acumulativa (CDF)

ejemplo de este comportamiento, donde se pueden observar caídas en la tasa de transferencia seguidos por picos en el caso de la aplicación 1.

IV-D. Tiempo entre llegadas de paquetes

Todos los errores medidos para el tiempo entre llegadas de paquetes son menores que el 10 % y en la mayoría de los casos estos errores son menores que el 1 %. El mayor error se asocia a la menor TA, i.e. 50ms, que se debe mayormente al uso de sistemas operativos que no son de tiempo-real y a los retrasos de la red. Con el fin de estimar la distribución de los paquetes del tráfico real y predicho, se ha calculado la Función de Distribución Acumulativa (CDF), que se muestra en la Figura 4. Estas figuras muestran la precisión de las aplicaciones reales y predichas para las tasas de actualización del sistema. El eje horizontal de cada figura muestra un intervalo de diez milisegundos (en segundos), mientras que el eje vertical representa el porcentaje de paquetes para cada TA del sistema.

La Figura 4 muestra claramente que la mayor diferencia entre el valor predicho y el valor real de la distribución de los paquetes es para la TA de 50ms. Esto ha sido discutido en el párrafo previo y puede ser justificado también mediante los errores mostrados en el Cuadro II. Las siguientes figuras muestran un incremento en la precisión de la CDF a mayor TA. Así, con el fin de aumentar la precisión de la predicción, se ha introducido una variación en la distribución mediante el uso de la función de distribución Gaussiana. Así se ha aumentado la precisión del enfoque propuesto obteniendo en la mayoría de los casos errores menores del 1 %. Aun así, una versión mejorada del algoritmo puede tener en cuenta más variaciones específicas de las aplicaciones y reducir todos los errores por debajo del 1 %.

V. Conclusiones y líneas futuras

Los recientes enfoques en el campo del modelado del tráfico en los Sistemas de Control de Procesos (SCP) [4], [5], [6], demuestran que la construcción de modelos precisos del tráfico de red es una tarea compleja. El tiempo y los recursos necesarios para la construcción de modelos realistas constituye una de los mayores retos que los ingenieros deben resolver. La construcción automática de modelos basados en las características de las aplicaciones SCADA pueden eliminar efectivamente este problema. Así, este artículo propone un enfoque para la construcción de modelos de tráfico de red para los SCPs, mediante datos específicos de aplicaciones como el número de variables, clases de variables y tasas de actualización de las variables. La mayor contribución de este trabajo es un algoritmo que recibe como entrada la descripción de la aplicación y genera un modelo de tráfico de la duración especificada. El algoritmo ha sido construido mediante el análisis del tráfico de red generado por varias aplicaciones y obtenido en una instalación SCADA real. Como muestran los resultados experimentales, el algoritmo es muy preciso y

puede predecir el número de paquetes con un error $< 1\,\%$, el tamaño de los paquetes con un error de un $0\,\%$, la tasa de transferencia con un error $< 2\,\%$ y el tiempo entre llegadas de paquetes con un error $< 10\,\%$.

La principal ventaja del enfoque propuesto es que los modelos de tráfico se pueden regenerar de manera sencilla, sin la necesidad de analizar tráfico real para la construcción de nuevos modelos. Además, puede ser aplicado en varias direcciones, empezando desde los Sistemas de Detección de Intrusiones, hasta la recreación del tráfico de red en entornos simulados/experimentales. Como trabajo futuro, se pretende aplicar el enfoque propuesto en los campos mencionados anteriormente y mejorarlo con el análisis de las especificaciones de las aplicaciones con el fin de reducir por debajo del $1\,\%$ los errores medidos.

AGRADECIMIENTOS

Iñaki Garitano está financiado mediante la beca BFI09.321 del Departamento de Investigación, Educación y Universidades del Gobierno Vasco.

REFERENCIAS

[1] N. Falliere, L. O. Murchu, y E. Chien. Dosier W32.stuxnet. http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf, Febrero, 2011. [Online; último acceso Febrero 28, 2012].

[2] Symantec. Dosier W32.duqu the precursor to the next stuxnet. http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_duqu_the_precursor_to_the_next_stuxnet.pdf, Noviembre 23, 2011. [Online; último acceso Febrero 28, 2012].

[3] McAfee Foundstone Professional Services y McAfee Labs. Global energy cyberattacks: "night dragon". http://heartland.org/sites/all/modules/custom/heartland_migration/files/pdfs/29423.pdf, Febrero 10, 2011. [Online; último acceso Marzo 23, 2012].

[4] A. N. Mahmood, C. Leckie, J. Hu, Z. Tari, y M. Atiquzzaman. Network traffic analysis and scada security. En *Handbook of Information and Communication Security*, páginas 383–405, 2010.

[5] T. Roosta, D. K. Nilsson, U. Lindqvist, y A. Valdes. An intrusion detection system for wireless process control systems. En *5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, 2008. MASS 2008*, páginas 866–872, IEEE, 2008.

[6] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, y A. Valdes. Using model-based intrusion detection for scada networks. En *Actas de SCADA Security Scientific Symposium*, 2006.

[7] P. Düssel, C. Gehl, P. Laskov, J.-U. Bußer, C. Störmann, y J. Kästner. Cyber-critical infrastructure protection using real-time payload-based anomaly detection. En *Critical Information Infrastructures Security. CRITIS'09*, 6027:85–97, 2010.

[8] A. Valdes y S. Cheung. Communication pattern anomaly detection in process control systems. En *IEEE Conference on Technologies for Homeland Security, 2009. HST'09.*, páginas 22–29, IEEE, 2009.

[9] The MathWorks Inc. Matlab - the language of technical computing. http://www.mathworks.com/products/matlab/, 2012. [Online; último acceso Marzo 04, 2012].

[10] K. D. Lawrence, R. K. Klimberg, y S. M. Lawrence. En *Fundamentals of forecasting using excel*. Industrial Press, 2008.