E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Sistema online de disección de protocolos en trazas de paquetes

Grado en Ingeniería
en Tecnologías de Telecomunicación

Trabajo Fin de Grado

Autor: Fermín Lassa Iglesias

Tutor: Eduardo Magaña Lizarrondo

Pamplona, 27 de octubre de 2020

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

JORGE CHAM ©THE STANFORD DAILY

Resumen

Se pretende ampliar las prestaciones del prototipo analizador de tramas online que se presentó previamente. Esta necesidad surge del aumento del ancho de banda y la cantidad de datos que genera cada usuario, unido con el enorme incremento del despliegue de servicios e infraestructuras en la nube y la mejora de las herramientas disponibles. Queriendo aprovechar estas circunstancias, se va a desarrollar una mejora para poder aprovechar al máximo los recursos disponibles en la actualidad, ofreciendo información más detallada al usuario. Para ello, se utilizarán herramientas para tratar de mejorar el proceso de almacenamiento de los datos (bases de datos relacionales como MySQL). Para la visualización de estas tramas, se utilizarán herramientas como Grafana, ya que ofrecen muchas métricas distintas y la posibilidad de configurar alertas personalizadas.

Índice

Índice de figuras

Índice de tablas

1. Introducción

El aumento del número de dispositivos con acceso a Internet y sus reducidos costes, el incremento del ancho de banda de las redes y la gran cantidad de contenido disponible ha provocado que cada vez el volumen de información transmitida sea mayor. Es por ello que el análisis de las trazas paquete a paquete resulta imposible y se ha de utilizar otra técnica donde obtener información global sobre la captura para, a continuación, analizar la traza en detalle. Este proyecto se ha elaborado como un complemento a una herramienta [1] desarrollada previamente, permitiendo al usuario obtener una imagen global sobre una captura de tráfico o monitorizar el funcionamiento de su red en tiempo real mediante un sencillo sniffer.

2. Prototipo de disección de protocolos

El prototipo consiste en un lector de capturas de tráfico de red que son procesadas, almacenadas en una base de datos y visualizadas mediante un panel vía web. Se ha diseñado de manera que sea fácil de usar por cualquier persona sin necesidad de grandes conocimientos previos, ya que lo único que debe realizar el usuario es subir la traza y el programa se encarga del resto. Este proceso se muestra en la siguiente figura con una pequeña explicación de cada uno de los pasos.

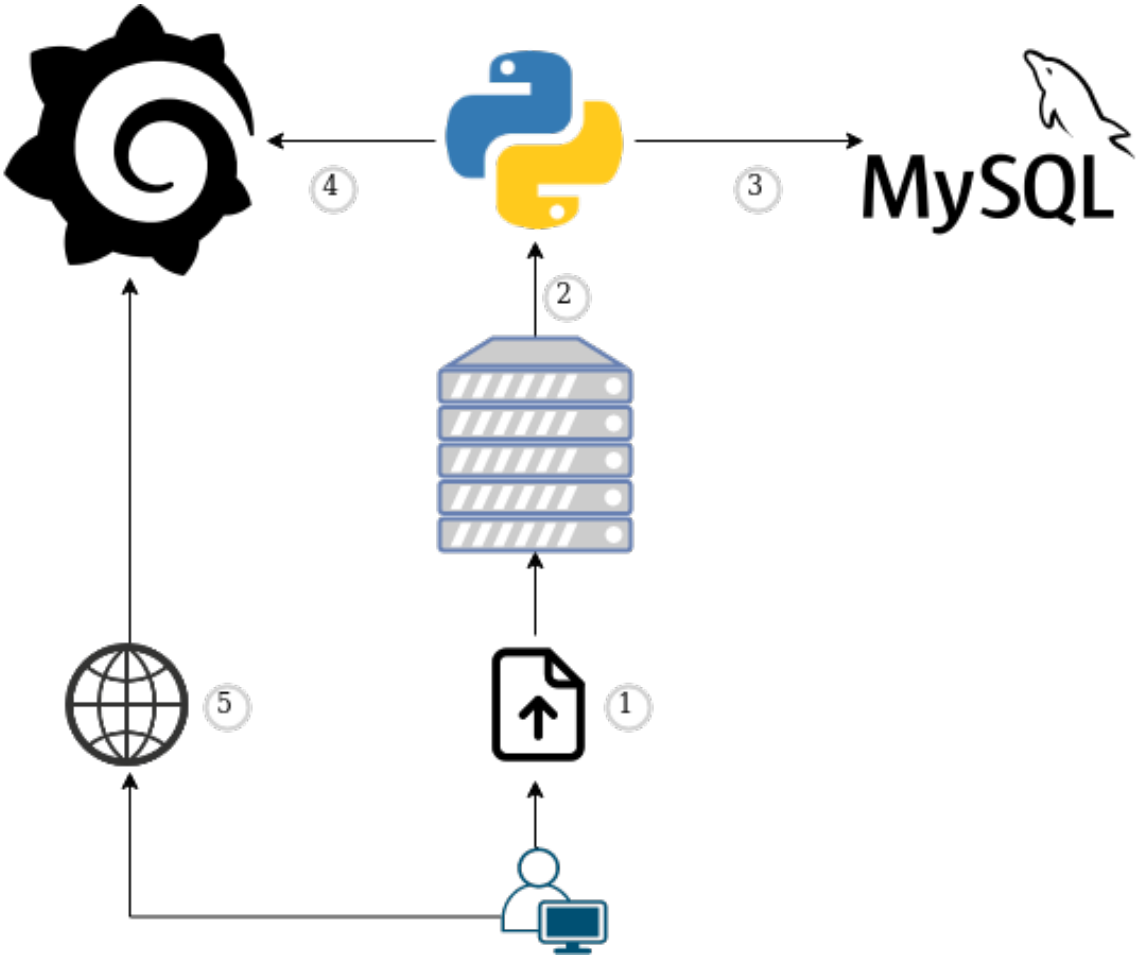

Figura 1: Flujo de funcionamiento del prototipo

1. El usuario sube la captura de tráfico al servidor.
2. Ejecuta el programa que procesa la captura.
3. Se introducen los paquetes en la base de datos.
4. Se añaden la base de datos y el panel para poder visualizar la traza.
5. El usuario se conecta al servidor vía web para ver el panel con la información de la traza.

Este prototipo se compone de tres partes bien diferenciadas y que se van a explicar con detalle en los siguientes apartados:

- Python: Se encarga de procesar los paquetes e introducirlos en la base de datos.
- MySQL: Almacena los paquetes de la captura.
- Grafana: Permite visualizar la captura mediante gráficas.

2.1. Python

Es un lenguaje de programación interpretado, de alto nivel, dinámico y multiplataforma que soporta programación orientada a objetos [2]. Se utiliza para prototipos y pruebas de concepto, entre otros muchos usos, debido a su sencillez y a la gran cantidad de librerías existentes. Es por ello que se ha decidido utilizar este lenguaje para el desarrollo del prototipo.
Este prototipo procesa los paquetes para obtener la información necesaria y los almacena en la base de datos. Los parámetros de interés de cada paquete son:

Capa 2			**Capa 3**		**Capa 4**		**Otro**
MAC origen	MAC destino	Ethertype	IP origen	IP destino	Puerto origen	Puerto destino	Timestamp

Tabla 1: Parámetros de interés por paquete

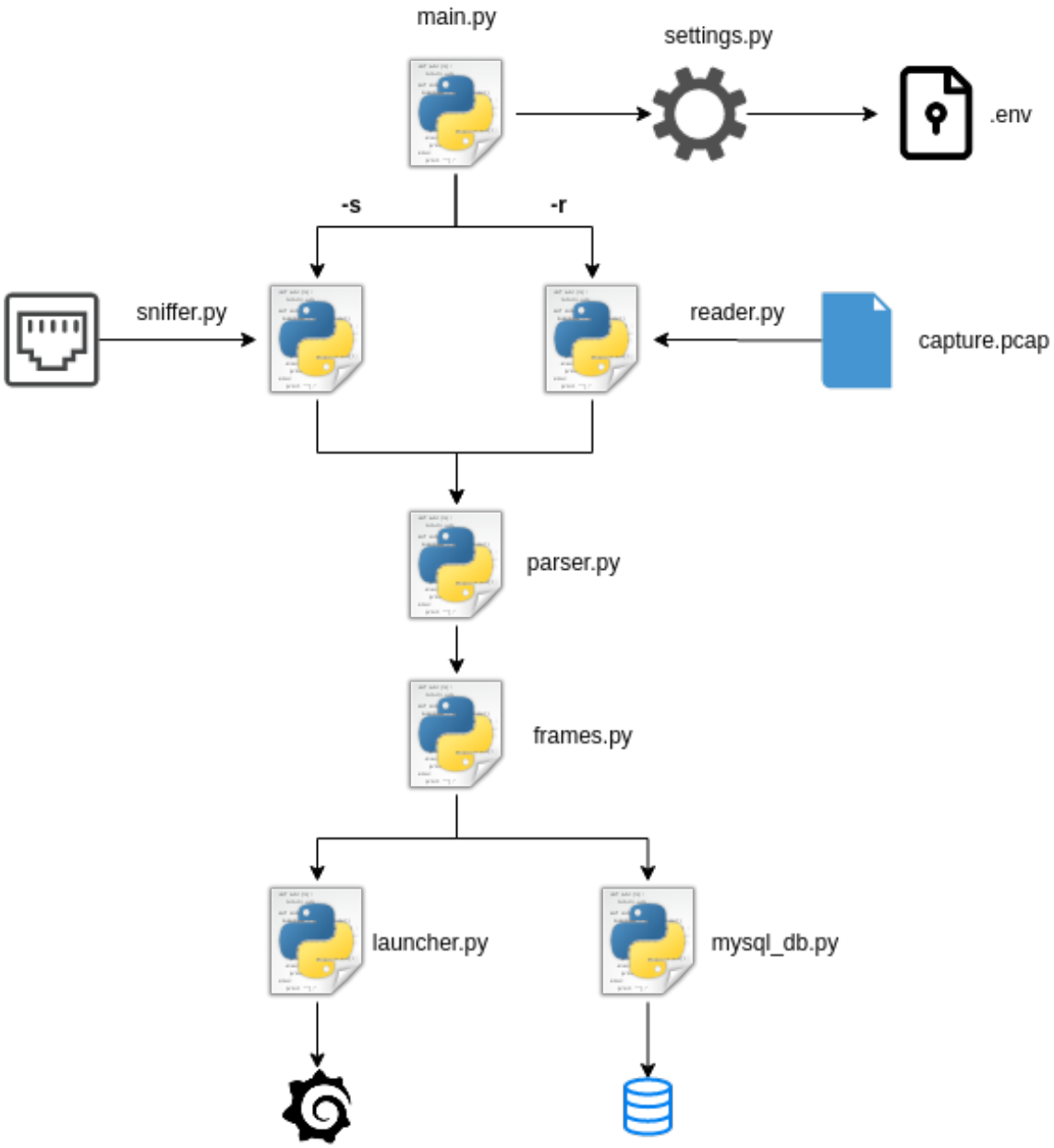

Figura 2: Flujo de funcionamiento del código de Python

El prototipo dispone de dos modos de funcionamiento:

- Sniffer de tráfico en tiempo real
- Lector de capturas de tráfico

El flujo de funcionamiento del código es similar, independientemente del modo seleccionado. El script obtiene las variables necesarias mediante el script *settings.py* y procesa los paquetes con la ayuda de los scripts *parser.py* y *frames.py*. Tras obtener los parámetros de interés de cada paquete, los introduce en la base de datos mediante el driver *mysql_ db.py*. Finalmente, añade el panel y la base de datos a Grafana y ejecuta un navegador para visualizar los datos mediante *launcher.py*.

2.2. MySQL

Es un gestor de bases de datos relacional de gran popularidad [3]. Permite crear distintas bases de datos, tablas y usuarios para el correcto acceso y gestión de los datos almacenados. Se ha diseñado una base de datos (OSniffy) con dos usuarios (OSniffy, OSniffy_Grafana) y una tabla (Packets) donde almacenar la información de los paquetes.
La tabla **Packets** se ha definido con los siguientes parámetros:

Field	Type	Null	Key	Default	Extra
packetID	int(11)	NO	PRI	NULL	auto_increment
srcMAC	varchar(17)	NO		NULL	
dstMAC	varchar(17)	NO		NULL	
etherType	varchar(6)	NO		NULL	
srcIP	varchar(15)	NO		NULL	
dstIP	varchar(15)	NO		NULL	
protocol	tinyint(1) unsigned	YES		NULL	
srcPort	smallint(2) unsigned	YES		NULL	
dstPort	smaillint(2) unsigned	YES		NULL	
timestamp	timestamp	NO		CURRENT_TIMESTAMP	

Tabla 2: Definición de la tabla **Packets**

El usuario **OSniffy** tiene todos los privilegios en la base de datos y es el encargado de insertar los datos.

```
+-----------------------------------------------------------------+
| Grants for OSniffy@localhost                                    |
+-----------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'OSniffy'@'localhost'                     |
| GRANT ALL PRIVILEGES ON `OSniffy`.* TO 'OSniffy'@'localhost' |
+-----------------------------------------------------------------+
```

El usuario **OSniffy_Grafana** tiene permiso de lectura en la base de datos y es el encargado de hacer las peticiones que serán visualizadas en Grafana.

```
+-----------------------------------------------------------------+
| Grants for OSniffy_Grafana@localhost                            |
+-----------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'OSniffy_Grafana'@'localhost'             |
| GRANT SELECT ON `OSniffy`.* TO 'OSniffy_Grafana'@'localhost' |
+-----------------------------------------------------------------+
```

2.3. Grafana

Es una aplicación web de código abierto y multiplataforma que permite visualizar y analizar de manera interactiva métricas almacenadas en distintas fuentes de datos [4]. Sirve para crear paneles de monitorización en tiempo real mediante gráficos, estadísticas, alertas personalizadas y consultas interactivas a las bases de datos.
Se ha diseñado un panel con dos secciones diferentes. La primera contiene estadísticas generales sobre la traza y la segunda aporta información sobre las conexiones con mayor tráfico.

Figura 3: Detalle del primer panel

El primer panel se compone de 4 subpaneles:

- Paquetes por protocolo: Número de paquetes en la traza, divididos por protocolo.
- Número de direcciones IP y MAC únicas: Cantidad de direcciones MAC e IP únicas en la traza.
- Últimos 50 paquetes: Listado con el detalle de los últimos 50 paquetes.
- Paquetes por segundo: Cantidad de paquetes por segundo, permite estimar el throughput.

Figura 4: Detalle del segundo panel

El segundo panel se compone de 4 subpaneles:

- Direcciones IP origen: Las 5 direcciones IP origen que más paquetes mandan y la cantidad.
- Direcciones IP destino: Las 5 direcciones IP destino que más paquetes reciben y la cantidad.
- Puertos origen: Los 5 puertos origen que más paquetes mandan y la cantidad.
- Puertos destino: Los 5 puertos destino que más paquetes reciben y la cantidad.

3. Tiempos de ejecución

Una vez que se ha definido la estructura y el funcionamiento del prototipo, se quiere analizar la capacidad de procesado y el tiempo necesario para visualizar las trazas. Se ha diferenciado el tiempo de almacenamiento de las trazas y el tiempo de visualización.

3.1. Tiempos de almacenamiento

El proceso de almacenamiento de los paquetes es el que más tiempo consume, aunque tiene la ventaja de que es un proceso único. Una vez se procesa y almacena la captura, ya no es necesario volver a hacerlo. Se puede realizar de varias maneras distintas, aunque en este escenario se ha optado por probar dos:

- Insertar los paquetes uno a uno: Cada vez que hay un paquete para insertar, se crea una nueva conexión a la base de datos, se añade y se cierra la conexión.
- Insertar los paquetes en bloques: Se almacena un número de paquetes determinado y por cada bloque se crea una conexión a la base de datos, se añaden y se cierra la conexión.

Se han realizado varias pruebas para comprobar que método es más eficiente y el número de paquetes óptimo por bloque para reducir el tiempo de inserción. Se ha utilizado tres tipos de trazas [5]: navegación web (59.635 paquetes, 39Mbs), vídeo a 1080p (136.797 paquetes, 120Mb) y descarga de un fichero (1.061.088 paquetes, 1.1Gb) para poder abarcar tres de los escenarios más comunes.

	1 paquete	10 paquetes	100 paquetes	1.000 paquetes	10.000 paquetes	20.000 paquetes	50.000 paquetes	100.000 paquetes
39Mb	150.07s	27.68s	11.08s	8.35s	7.40s	9.02s	7.63s	6.78s
120Mb	443.89s	114.98s	39.17s	19.52s	19.21s	20.00s	19.20s	18.05s
1Gb	3599.27s	865.21s	323.95s	165.18s	137.14s	140.2s	124.38s	118.88s

Tabla 3: Tiempos de inserción por número de paquetes y tamaño de traza

Mediante estos experimentos, se observa que la manera más eficiente de insertar los datos es en bloques de 100.000 paquetes, ya que reduce en gran medida el número de conexiones a realizar en la base de datos.

3.2. Tiempos de visualización

Grafana utiliza sentencias de MySQL para obtener los datos y poder visualizarlos. Es por ello que el tiempo necesario para poder visualizar los datos se ve influido por la complejidad de las búsquedas y el número de registros en las tablas de la base de datos. Se ha analizado las distintas gráficas y peticiones para poder detectar cuales consumen más tiempo.

	Paquetes por protocolo	Nº de IPs y MACs	Últimos 50 paquetes	Paquetes por segundo
Peticiones (39Mb)	186ms	394ms	300ms	298ms
Procesado (39Mb)	2ms	<1ms	1ms	<1ms
Peticiones (120Mb)	318ms	493ms	531ms	468ms
Procesado (120Mb)	3ms	<1ms	1ms	<1ms
Peticiones (1Gb)	1.02s	2.44s	2.43s	2.46ms
Procesado (1Gb)	2ms	<1ms	<1ms	<1ms

Tabla 4: Tiempos de visualización - Panel 1

	Direcciones IP origen	Direcciones IP destino	Puertos origen	Puertos destino
Peticiones (39Mb)	398ms	327ms	244ms	297ms
Procesado (39Mb)	1ms	<1ms	1ms	<1ms
Peticiones (120Mb)	646ms	645ms	420ms	411ms
Procesado (120Mb)	<1ms	<1ms	<1ms	<1ms
Peticiones (1Gb)	2.83s	2.84s	1.46s	1.56s
Procesado (1Gb)	<1ms	<1ms	<1ms	<1ms

Tabla 5: Tiempos de visualización - Panel 2

Se observa que el tiempo de las peticiones es el más elevado y aumenta considerablemente en función de la cantidad de datos almacenados. También se observa que las peticiones que incluyen algún tipo de condición son las que más tiempo consumen.

4. Conclusiones

Se ha diseñado un prototipo sencillo pero potente, fácil de usar y aplicando buenas prácticas para el desarrollo de software seguro y escalable. Permite obtener una imagen global de una traza en un tiempo aceptable y no requiere grandes conocimientos por parte del usuario, siendo un buen complemento a la herramienta comentada previamente.
Esta escrito en Python, un lenguaje de alto nivel, así que una mejora sería la traducción completa del código a C/C++. Este cambio permitira optimizar y reducir los tiempos de procesado y almacenamiento de las trazas. Utiliza una base de datos relacional, que son muy potentes y rápidas con un número reducido de entradas y pequeñas cantidades de datos, pero que no funcionan bien con grandes capturas de tráfico. Es por ello que se puede sustituir por una base de datos no relacional, ya que operan bien con grandes cantidades de datos, permitiendo mejorar los tiempos de visualización de los datos. Solo procesa 4 protocolos, aunque se pueden añadir más con pequeñas modificaciones del código, gracias al diseño estructurado del código.
Es un prototipo que, con un desarrollo posterior, puede incluir mejoras y nuevas herramientas que permitan un análisis más completo y veloz del trafico de red de una empresa o institución.

Referencias

[1] Arellano Lahuerta, L. (2017). *Prototipo analizador de tramas online*. Trabajo Fin de Grado. Pamplona: Escuela Técnica Superior de Ingenieros Industriales y de Telecomunicación.

[2] Wikipedia. (2020). Definición de Python. Recuperado de `https://en.wikipedia.org/wiki/Python_(programming_language)`

[3] Wikipedia. (2020). Definición de MySQL. Recuperado de `https://en.wikipedia.org/wiki/MySQL`

[4] Wikipedia. (2020). Definición de Grafana. Recuperado de `https://en.wikipedia.org/wiki/Grafana`

[5] Labayen, Víctor & Magaña, Eduardo & Morató, Daniel & Izal, Mikel. (2020). Online classification of user activities using machine learning on network traffic. Computer Networks. 181. 107557. 10.1016/j.comnet.2020.107557.