Sistemas de monitorización de latencias en redes de visibilidad

J. Álvaro Garrido López

Universidad de Granada

Tutores: Javier Díaz y Miguel Jiménez

Trabajo de Fin de Grado

September 10, 2019

Índice

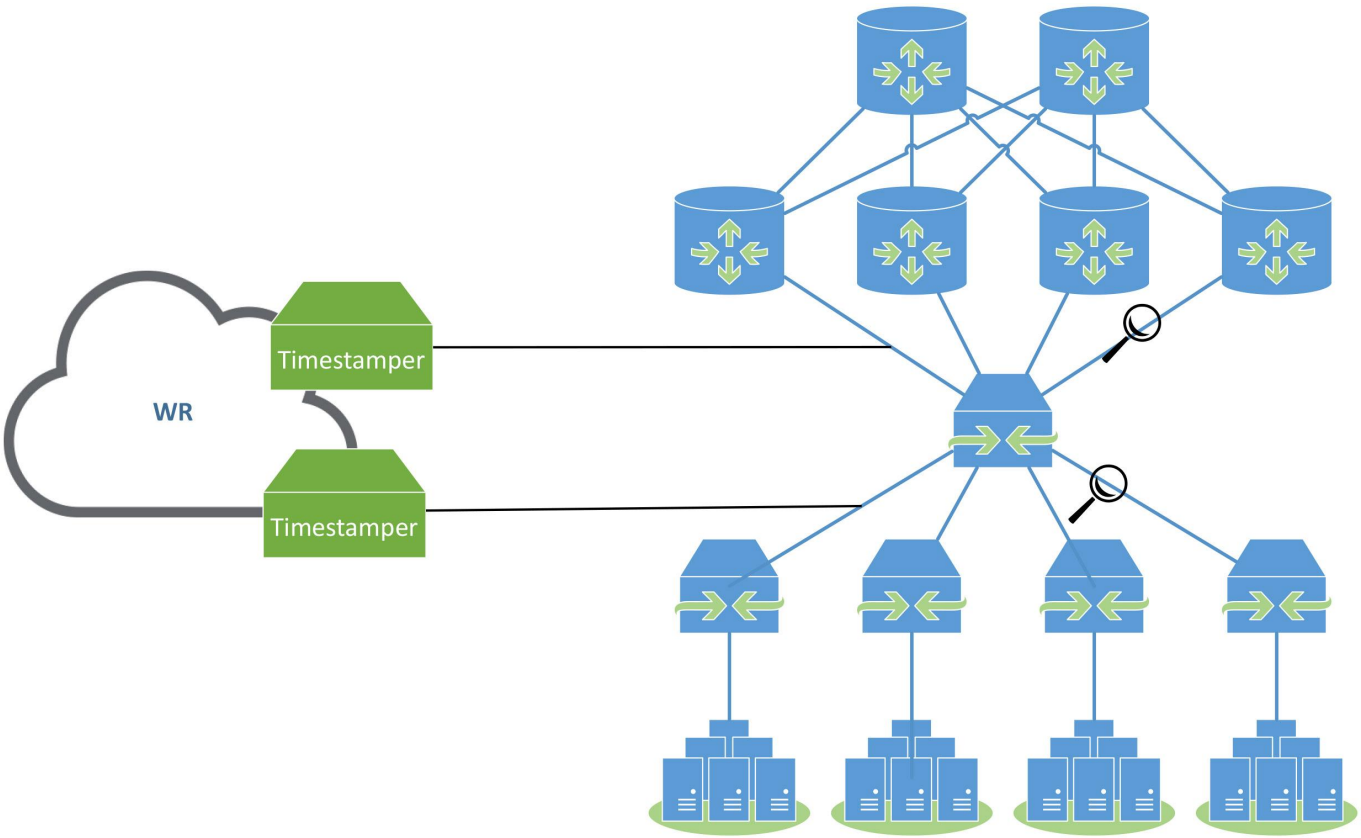
Introducción

Redes de visibilidad y aplicaciones

¿Qué son las redes de visibilidad?

Son la infraestructura en una red que permite la monitorización de la misma, con el fin de conocer el estado sobre su rendimiento y de detectar posibles fallos de seguridad.

Redes de visibilidad y aplicaciones



Motivación y contexto

Contexto

- Volúmenes ingentes de datos
- Preocupación por la seguridad
- Servicios de altas prestaciones (telecom y finance)
- Necesidad de controlar constante y eficientemente el tráfico
- Auge del *Big Data*

- Ingredientes perfectos para que se requiera de una recopilación, distribución y entrega de datos eficaz y escalable.
- De este punto parte la **visibilidad en redes**.

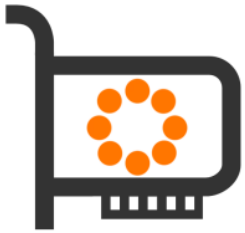
Objetivos (I)

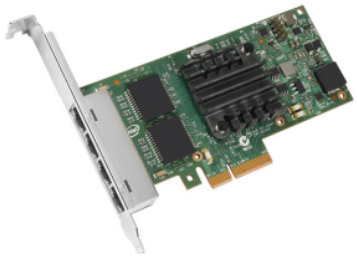
- Estado de la técnica sobre captura eficiente
- Aplicaciones comerciales y libres para visibilidad
- Análisis sobre las características de las tecnologías encontradas
- Evaluación del funcionamiento lógico de tecnologías

Objetivos (II)

- Diseño y desarrollo del sistema. Favorecer escalabilidad y flexibilidad
- Integración de los componentes *hardware* y *software*
- Integración de un sistema de alerting

Material y métodos



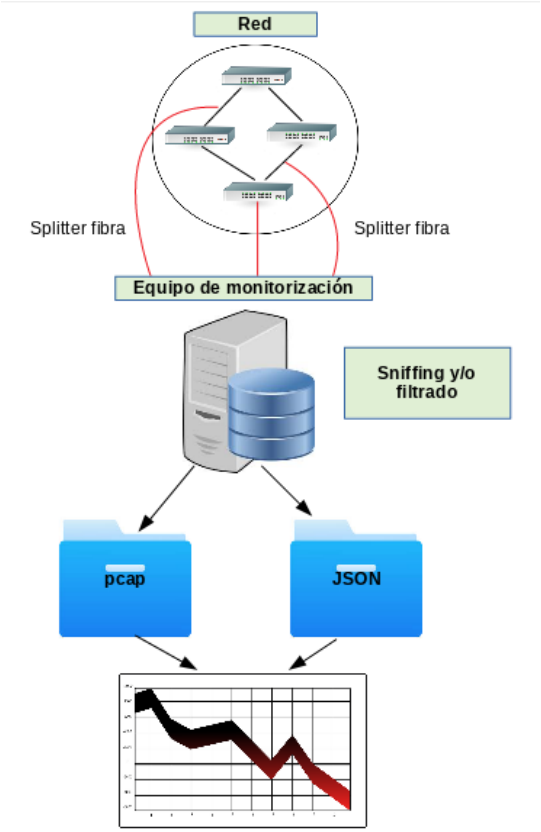


Estado de la técnica

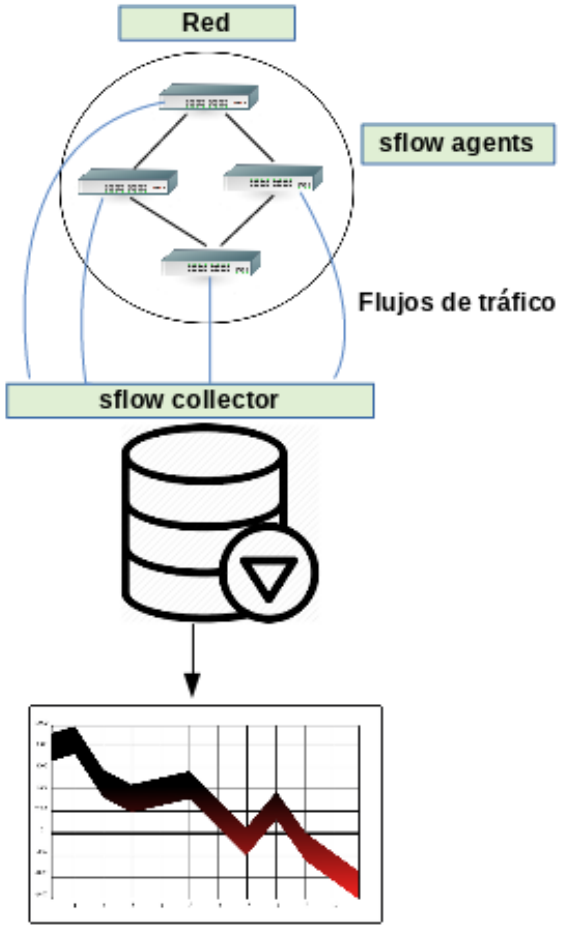
Métodos para implementar visibilidad (I)

- Mediante peticiones **SNMP**
- A través de **gestión directa** del tráfico (e.g. mediante **TAP**)
- Gestión del tráfico **por flujos** (e.g. **sflow**)

Métodos para implementar visibilidad (II)



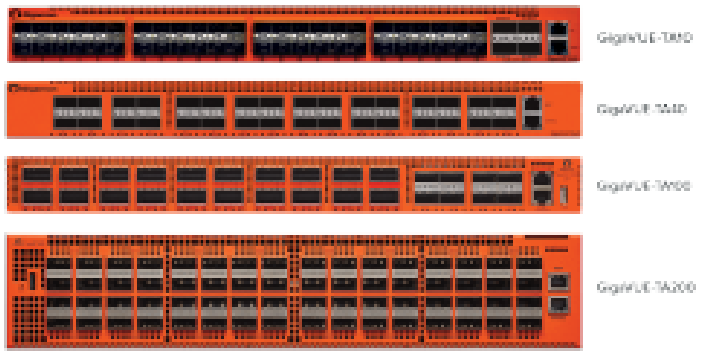
Métodos para implementar visibilidad (III)

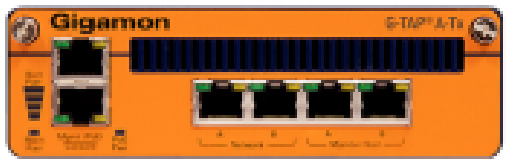


Hardware específico para visibilidad (I)

- **Divisores ópticos**. Dividen el haz de luz en varios caminos. No consumen electricidad
- **SPAN**. Puerto en un *switch* a donde se replica el tráfico. Existe pérdida de paquetes
- **TAP**. Replican el tráfico asegurando 0 pérdida de paquetes
- **Agregadores**. Agregan el flujo de tráfico de diferentes puertos en uno escogido

Hardware específico para visibilidad (II)





Implementación

Pruebas preliminares (I)

Gestión del tráfico basado en flujos

¿Qué es un flujo?

Es una secuencia de paquetes que comparten las mismas propiedades que son enviados entre un *host* emisor y un receptor. Por ejemplo, en una emisión de *streaming*, los paquetes son enviados por el servidor forman parte del mismo flujo.

Pruebas preliminares (II)

Tecnologías existentes

- **NetFlow**. Propietaria de **CISCO**. Necesario exportar flujos cada cierto tiempo
- **sflow**. Tecnología compatible con múltiples fabricantes. Datagramas enviados en tiempo real. Más recomendado para visibilidad

A pesar de escoger **sflow** por su mayor compatibilidad, y comprobar sus virtudes, nos interesa la opción más escalable y versátil. Por tanto, descartamos trabajar con gestión del tráfico por flujos.

Captura de paquetes

- **libpcap**. Biblioteca por defecto en Linux. Búffer lineal. Tamaño reducido
- **pf_ring ZC**. Búffer circular con **DMA**
- **Sniffer 10G**. Tecnología propietaria 10G. Depende de *hardware* específico del fabricante.

Escogemos **pf_ring** por ser mayoritariamente de código abierto, versátil, y no depender de *hardware* específico.

Filtrado

- **Hardware**. Existen **NIC** que ofrecen la posibilidad de aplicar reglas de filtrado *hw*
- **Bloque IP en FPGA**. También existe la posibilidad de implementar un bloque IP en una FPGA para el filtrado
- **Software**. **BPF** es una biblioteca que permite el filtrado *software* haciendo uso de cualquier herramienta de captura que hayamos escogido anteriormente

No escogemos la opción *hardware* por la limitación de compatibilidad, tampoco la opción de implementar un bloque IP por la complejidad de la solución.
Entonces, la más escalable es la opción *software*

Almacenamiento

- **InfluxDB**. Base de datos rendimiento *real-time*. Escalabilidad buena
- **elasticsearch**. Base de datos rendimiento *near real-time*. Alta flexibilidad del *input* y *output* de datos haciendo uso del *stack ELK*

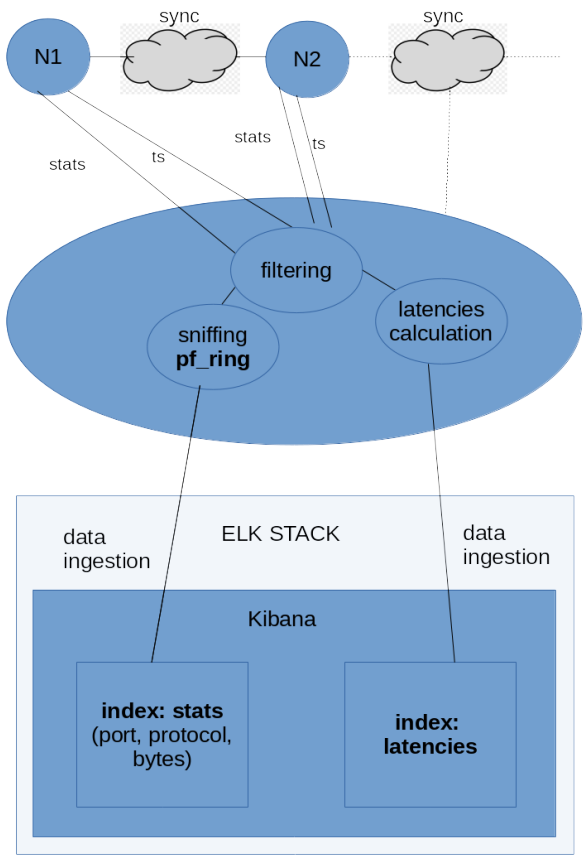
Escogemos *elasticsearch* por su flexibilidad y escalabilidad apoyándose en el *stack ELK*

Visibilidad

- **Grafana**. Escalabilidad buena. Diferentes opciones para visualizar parámetros
- **Kibana**. Gran escalabilidad. Compatibilidad con *JSON*. Apoyo en *stack ELK*. Tratamiento previo de los datos con *Logstash*
- **Graphite**. Buena compatibilidad y escalabilidad

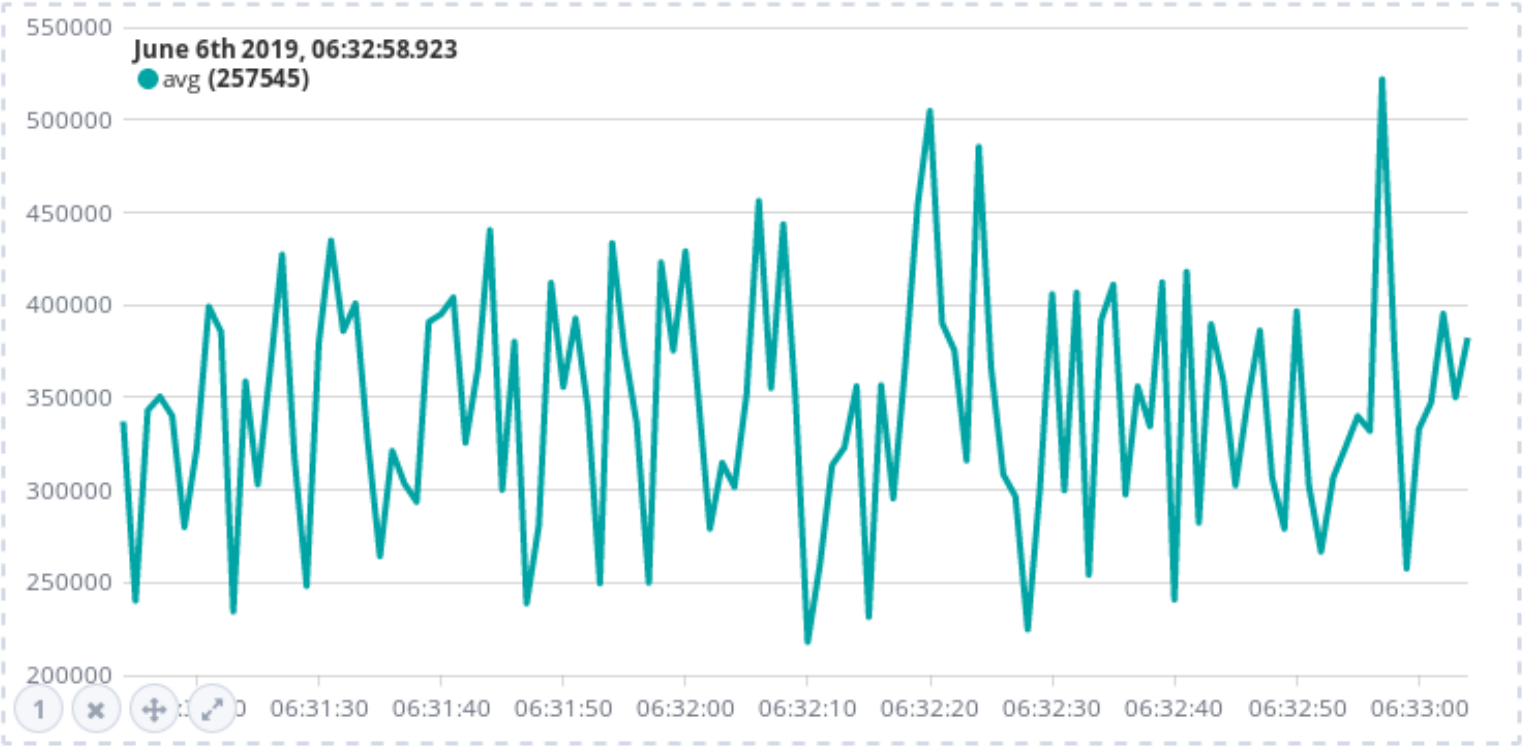
Nos quedamos con **Kibana** por haber demostrado ser la opción más escalable por su compatibilidad y flexibilidad con *JSON*, por las etapas previas y tratamiento de datos con *Logstash*, y por la multitud de opciones para visualización que ofrece

Setup final

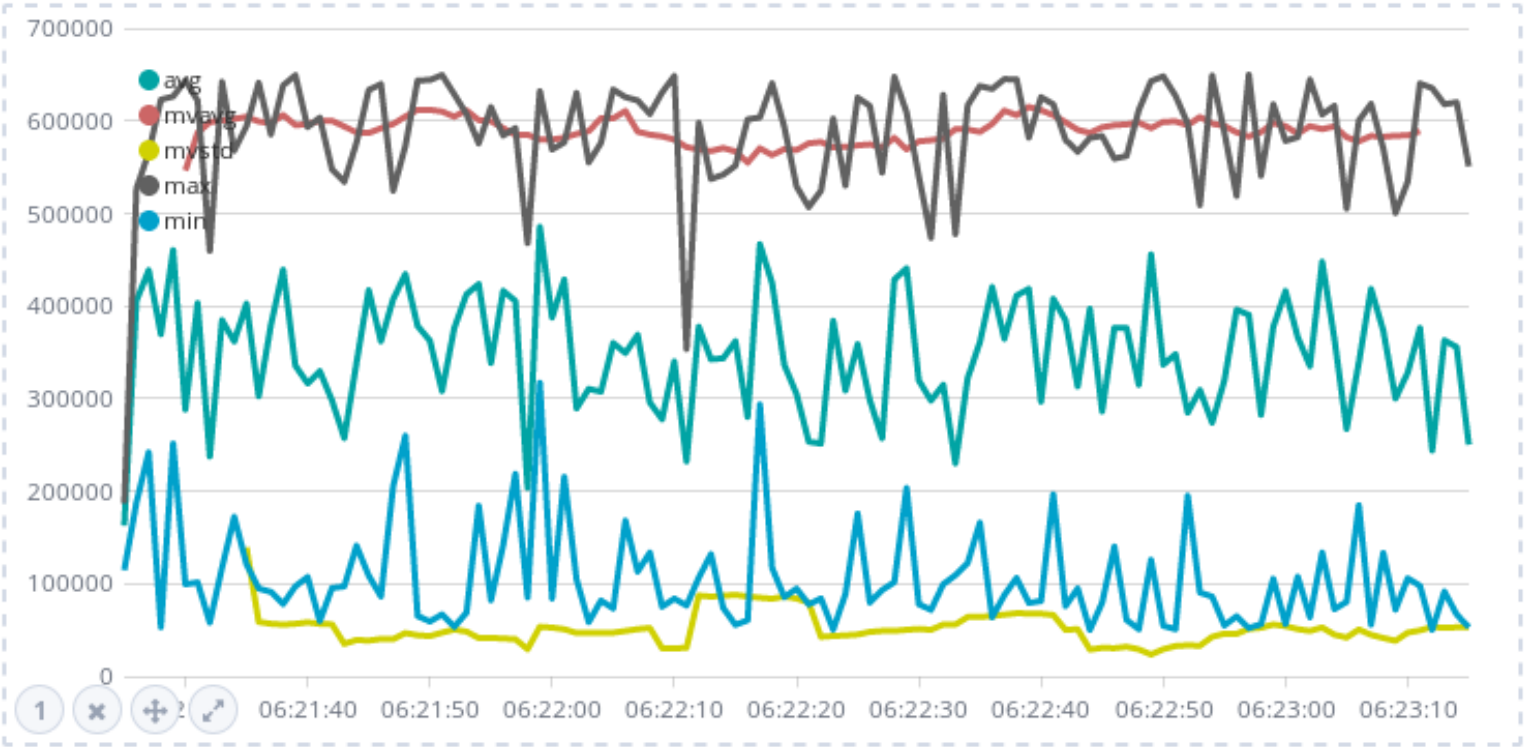

N1
sync
N2
sync
stats
ts
stats
ts
filtering
sniffing
pf_ring
latencies
calculation
data
ingestion
ELK STACK
data
ingestion
Kibana
index: stats
(port, protocol,
bytes)
index:
latencies

Resultados

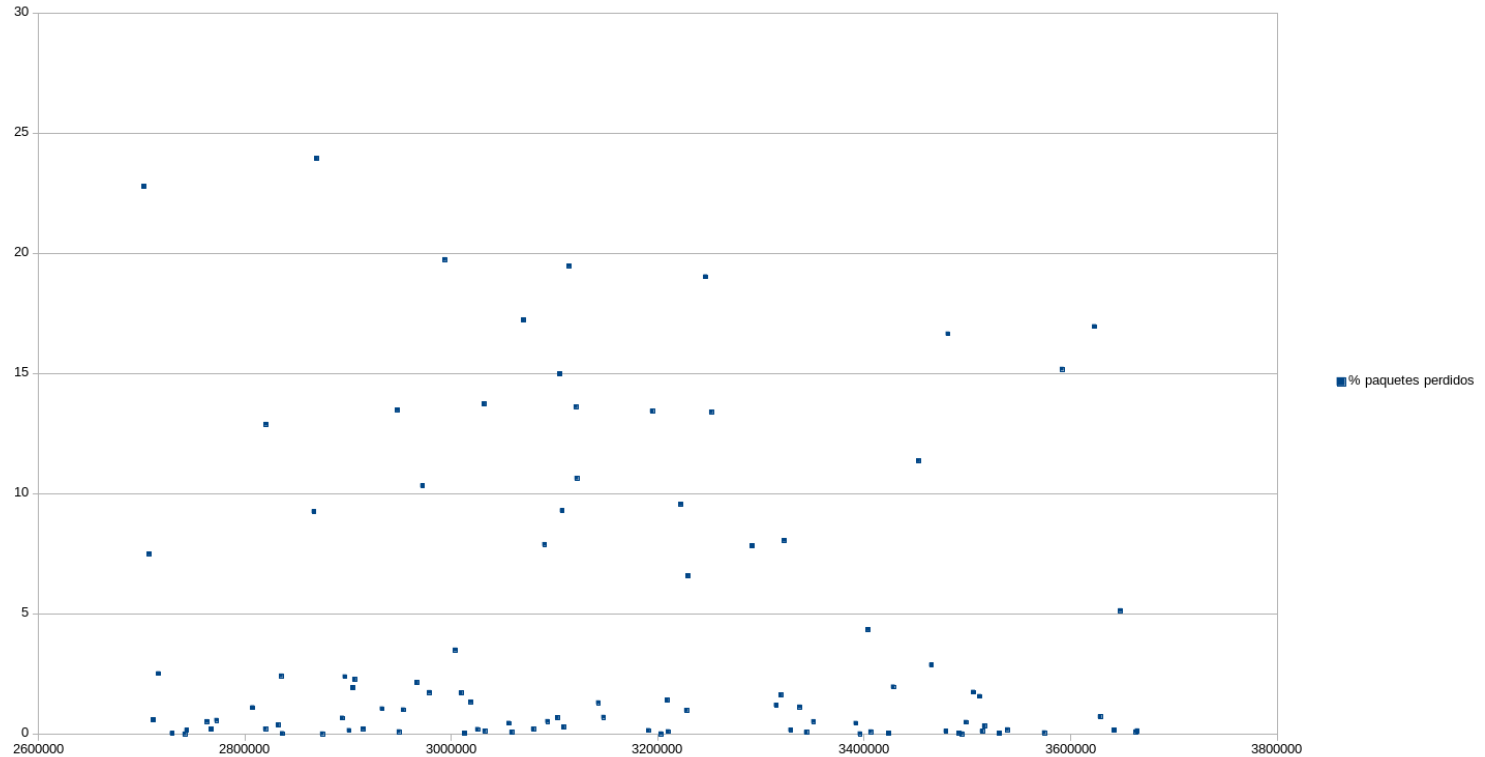
Visualización de estadísticas (I)



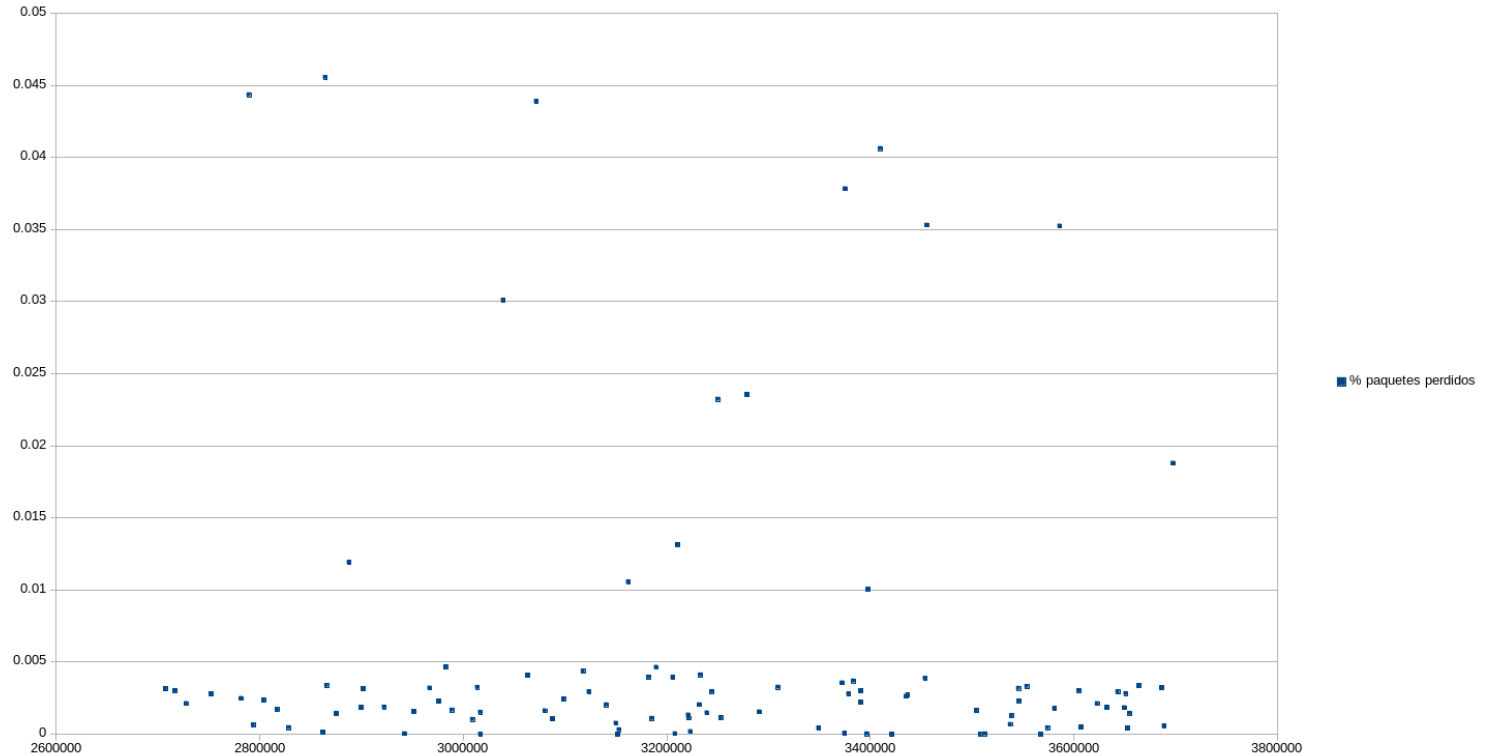
Visualización de estadísticas (II)



Comparativa captura de paquetes (I)



Comparativa captura de paquetes (II)



Conclusiones

Conclusiones

Hemos construido una herramienta que:

- Integra soluciones a diferentes niveles para construir una herramienta para medir latencias y otras estadísticas
- Es posible sustituir componentes de la misma para satisfacer necesidades más específicas, por la **alta escalabilidad** y **flexibilidad** del sistema
- Mejora de hasta un 30% en la etapa de captura de paquetes (pf_ring) sobre *libpcap*)
- Permite implementar visibilidad en una red
- Permite mucho trabajo futuro
- Ya tiene asignada una futura **aplicación** en el mercado (Seven)

Trabajo futuro

- Apoyo *hardware* para filtrado (FPGA)
- Creación de campos para medición de latencias con mayor precisión
- Data Analytics / Big Data

Referencias