UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Programari de monitorització de comunicacions VoIP

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Alex Llobet Martínez-Pons

In partial fulfilment

of the requirements for the degree in

Ciències i Tecnologies de les Telecomunicacions
ENGINEERING

Advisor: Alfonso Rojas

Barcelona, June 2014

1- Abstract

This project solves a common problem of the companies which provide *VoIP* services to their clients: traffic voice monitoring. To do so, two different monitoring solutions have been configured, tested and compared: *SIPCAPTURE*, which is a free software and *VoIPmonitor*, which is distributed under a commercial license. Once the first program architecture has been set and the second program has been configured and installed, both of them have been released in a preproduction scenario so as to analyse simultaneously real *VoIP* traffic flowing through the enterprise's servers and to show, for each of them, the call information and its quality.

The comparison of both software solutions has been based on several criteria: the information shown, the scalability, the use of the host resources in which each software has been installed and finally, the costs assumed by the Enterprise for each solution.

Once the comparison has been done, it has been concluded that *VoIPmonitor* is slighter better tan *SIPCAPTURE*, but a future line has been set in which a transition to the free software in medium term is forecasted.

Finally, it is worth mentioning until the transition is not done, both software have already been set in a production scenario.

2- Resum

Aquest treball dóna una solució a un problema comú en les empreses que proveeixen serveis de veu *VoIP* als seus clients: la monitorització del tràfic de veu. Per a això, s'ha configurat, executat i comparat dues solucions de monitorització de veu radicalment diferents: *SIPCAPTURE,* de codi obert i *VoIPmonitor*, de llicencia comercial. Una vegada s'ha dissenyat i configurat l'arquitectura del primer programari i s'ha instal·lat i configurat el segon, tots dos programaris s'han posat en preproducció per a que de forma simultània analitzin el tràfic *VoIP* real que circula pels servidors de l'empresa i així aportin cadascun la informació sobre les trucades i llur qualitat.

La comparació de totes dues solucions s'ha fet sota diversos criteris: la informació aportada, l'escalabilitat, l'ús de recursos de la màquina on està instal·lat cadascun i finalment, els costos que suposen cadascun per a l'empresa.

Una vegada feta la comparació, s'ha conclòs que *VoIPmonitor* està lleugerament per sobre de *SIPCAPTURE,* però s'ha establert una línia de futur en què a mig termini l'empresa farà una transició al programari lliure.

Finalment, és necessari fer palesa de que fins que no conclogui aquesta transició, tots dos programes han estat ja posats en producció.

3- Resumen

Este trabajo soluciona un problema común en las empresas que proveen servicios de voz VoIP a sus clientes: la monitorización del tráfico de voz. Para ello, se ha configurado, ejecutado y comparado dos soluciones de monitorización radicalmente diferentes: *SIPCAPTURE*, de código abierto y *VoIPmonitor*, de licencia comercial y privada. Una vez se ha diseñado y configurado la arquitectura del primer programa y se ha instalado y configurado el segundo, ambos programas se han puesto en preproducción para que de forma simultánea analicen el tráfico *VoIP* real que circula por los servidores de la empresa y aporten cada uno la información acerca de las llamadas y su calidad.

La comparación de ambas soluciones se ha hecho en base a múltiples criterios: la información aportada, la escalabilidad, el uso de recursos de la máquina donde está instalada y finalmente, los costes que suponen cada uno para la empresa.

Una vez hecha la comparación, se ha concluido que *VoIPmonitor* está ligeramente por encima de *SIPCAPTURE*, pero se ha establecido una línea de futuro en la que a medio plazo la empresa hará la transición al *software* libre.

Finalmente, cabe mencionar que hasta que no se concluya la transición, ambos programas han sido ya puestos en producción.

Dedico este trabajo a todas las personas que me han apoyado durante la carrera y el desarrollo de este trabajo, dentro de las cuales hago especial mención a mi familia.

4- Acknowledgements

Quiero manifestar mi expreso agradecimiento a mi tutor académico en este trabajo, el doctor Alfonso Rojas, así como a mi tutora empresarial e ingeniera, Anna Dorca; por haberme guiado y aconsejado durante todo el proyecto. También a la comunidad de desarrolladores de *GitHub*, por haber respondido todas mis preguntas y sin los cuales, el *software* libre estaría mucho menos evolucionado.

5- Revision history and approval record

Revision	Date	Purpose
0	20/12/2019	Document creation
1	24/1/2020	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Alex Llobet	alex.llobet@alu-etsetb.upc.edu
Alfonso Rojas	alfonso@entel.upc.edu
Anna Dorca	anna.dorca@tecsens.com

Written by: Alex Llobet		Reviewed and approved by: Alfonso Rojas	
Date	24/1/2020	Date	24/1/2020
Name	Alex Llobet	Name	Alfonso Rojas
Position	Project Author	Position	Project Supervisor

6- Table of contents

7- List of Figures

8- List of Tables:

9- Introducción

La proliferación de las nuevas T.I. y la sustancial mejora calidad de las conexiones ha cambiado nuevamente el paradigma de las diferentes tecnologías presentes. En el caso de la voz, la aparición de la fibra óptica ha permitido pasar de una obsoleta red telefónica pública conmutada a transmitir la voz digitalmente por Internet en forma de paquete de datos, creando nuevos protocolos y programas que permiten establecer y mantener llamadas mediante el protocolo IP. Todo esto ha permitido economizar sustancialmente el servicio telefónico y desarrollar nuevas funcionalidades telefónicas como colas de llamadas, transcripciones de mensajes en los buzones de voz y otras características típicas de centralita que mejoran notablemente la productividad particular y empresarial.

Esta actualización, sin embargo, no está exenta de inconvenientes. En efecto, la integración de los servicios telefónicos con Internet ha hecho depender la calidad del servicio de voz del estado de las redes telemáticas que transportan los paquetes de voz y dada la naturaleza estocástica de las redes en general, esta dependencia puede resultar en una degradación del servicio que reduce la calidad percibida por los usuarios que, en el peor de los casos, imposibilita la comunicación.

Es necesario, por lo tanto, acondicionar las redes en la medida de lo posible para que el tráfico de voz no sufra degradaciones que afecten al servicio. Por otro lado, y añadiendo el hecho de que el servicio de voz es un servicio en tiempo real, es importante monitorear este servicio para poder identificar estas degradaciones y actuar consecuentemente. Esta monitorización debe ayudar a detectar, acotar y solventar cualquier problema sucedido durante una comunicación VoIP.

En este trabajo se muestra la comparación de dos herramientas de monitorización de voz: *SIPCAPTURE* y *VoIPmonitor*. Ambas herramientas son muy diferentes: *SIPCAPTURE* es un proyecto *open source* que permite combinar varios programas libres para confeccionar el sistema de monitorización. Por otro lado, VoIPmonitor es un programa comercializado por la empresa con su mismo nombre. Durante el proyecto, se mostrará la instalación, configuración y los parámetros de calidad que se puede extraer de cada herramienta. En el caso de *SIPCAPTURE*, también se justificará qué programas se han usado para formar el servidor de monitorización y se mostrará su arquitectura y funcionamiento.

Finalmente, se compararán ambos sistemas de monitorización y se concluirá, en base a la información que se puede extraer de cada uno, qué sistema resulta más apropiado dado el *statu quo* de la empresa.

10- Estado del arte

Actualmente existen multitud de herramientas que permiten realizar una monitorización efectiva de una red *VoIP*. En este proyecto se han considerado únicamente y por motivos de infraestructura telemática, las que son compatibles con alguna distribución *linux*, por lo que se omitirá, también en este apartado, cualquier herramienta que no pueda funcionar en *linux*.

Dentro de las herramientas de monitorización *VoIP*, las hay de dos tipos: de código abierto y cerrado. Estas herramientas, en general, no suelen ser meramente programas de monitorización *VoIP*, sino que están integradas en un analizador de tráfico complejo[1]. La única solución de código abierto desarrollada y que cuenta con una comunidad importante de usuarios es *SIPCAPTURE*. Por otro lado, hay varias herramientas de código cerrado, todas ellas muy eficientes: *ManageEngine VoIP Monitor* es una solución web de análisis de tráfico de red que tiene un *plugin* con todas las funcionalidades de un servidor de monitorización *VoIP*: análisis de llamadas, métricas de calidad, gráficos, alertas… Además, la monitorización puede ser a nivel *WAN* y proactiva, inyectando tráfico a la red para prever situaciones de alta latencia. Otra solución privada es *VoIPSpear*, una plataforma en línea que monitoriza la red de forma remota para conocer el estado de las comunicaciones *VoIP*, por lo que es un servicio automático en el cual no hace falta instalar *software* para obtenerlo.

Por último, *VoIPmonitor* es una herramienta disponible para *Windows* y *Linux* específicamente diseñada para monitorizar redes VoIP mediante sondas y un servidor central.

En este trabajo, se han considerado solamente los motivos económicos y prácticos; por ende, se han descartado todas las soluciones integradas, aunque sus funcionalidades fueran muy avanzadas. Finalmente, se ha optado por *VoIPmonitor*, ya que sus funcionalidades son básicas pero potentes, a la par que suficientes para el tipo de monitorización que se quiere realizar.

[1] Véase el estudio de Luca Deri "*Open Source VoIP Traffic Monitoring"*, de la empresa NTOP. Disponible en https://www.cosmocom.gr/wp-content/uploads/2013/05/VoIP-2.pdf

11- Presentación del *software* de monitorización

11.1- SIPCAPTURE

SIPCAPTURE es un conjunto modular independiente de programas *libres* que interactúan entre ellos para proveer el servicio de monitorización de voz. El conjunto es indefinido, es decir, no hay una lista exacta u oficial de programas que conformen *SIPCAPTURE*, sino que hay unas directrices de configuración y multitud de programas de código abierto recomendados para satisfacer cada parte de la configuración. Algunos de los programas sí se han desarrollado para la monitorización de voz, pero otros ya habían sido creados previamente y para otros fines. En consecuencia, la elección de qué programas van a usarse es esencial, pues los habrá que tendrán afinidad, ya sea porque han sido desarrollados por la misma comunidad o empresa o porque fueron desarrollados para ser instalados conjuntamente; otros que serán independientes, y que por lo tanto podrán interactuar, pero de manera limitada; y finalmente habrá combinación de *software* que será incompatible.
Por otro lado, el problema de la monitorización más elemental está dividido en 4 etapas básicas, como puede verse en la siguiente figura:

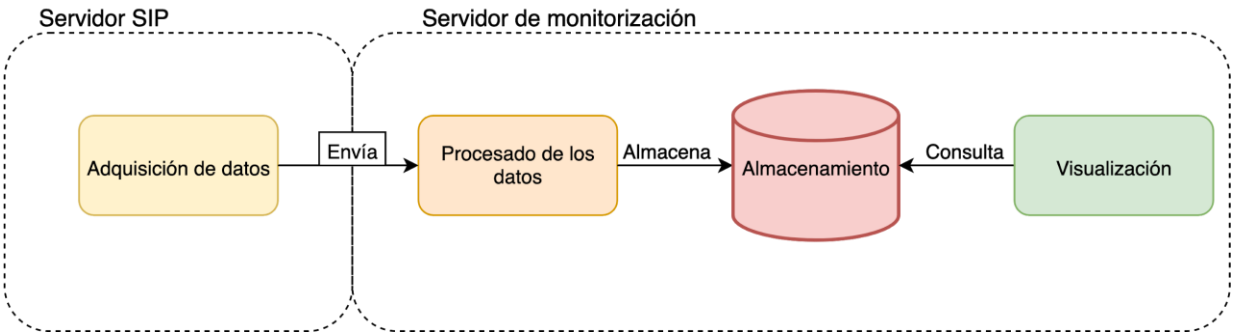

Ilustración 1: Esquema básico de monitorización

Dado que el servidor SIP *proxy* es el encargado de enrutar los paquetes SIP, RTC y RTCP[2] para establecer y mantener las llamadas, ahí es donde ha de haber un agente de captura que intercepte y reenvíe los paquetes al servidor de monitorización para que este pueda contar con la misma información del servidor SIP y así poder monitorizar la calidad de las llamadas, así como el rendimiento del propio servidor.

El servidor de monitorización, por otro lado, debe contar con un módulo que procese los datos recibidos por la sonda y los formatee adecuadamente para ser guardados en la siguiente etapa: el almacenamiento ordenado y eficiente de toda la información formateada. Finalmente, debe haber un módulo que permita filtrar y representar estos datos mediante consultas al módulo de almacenamiento.

[2] Véase Anexo 1:"*Protocolo SIP"* y Anexo 2: "*Protocolos RTP, RTCP y SDP"* para información detallada.

Cada etapa puede estar dividida en una o diversas subetapas, según los programas que se hayan escogido y las funcionalidades con las que se las quiere dotar. En este proyecto, se ha decidido utilizar dos programas para la etapa de visualización, y cada uno de estos programas consulta dos bases de datos diferentes, por lo que cada una de las dos últimas etapas ha sido dividida en dos subetapas. En concreto, el esquema anterior se ha adaptado a este proyecto de la manera que se representa a continuación:

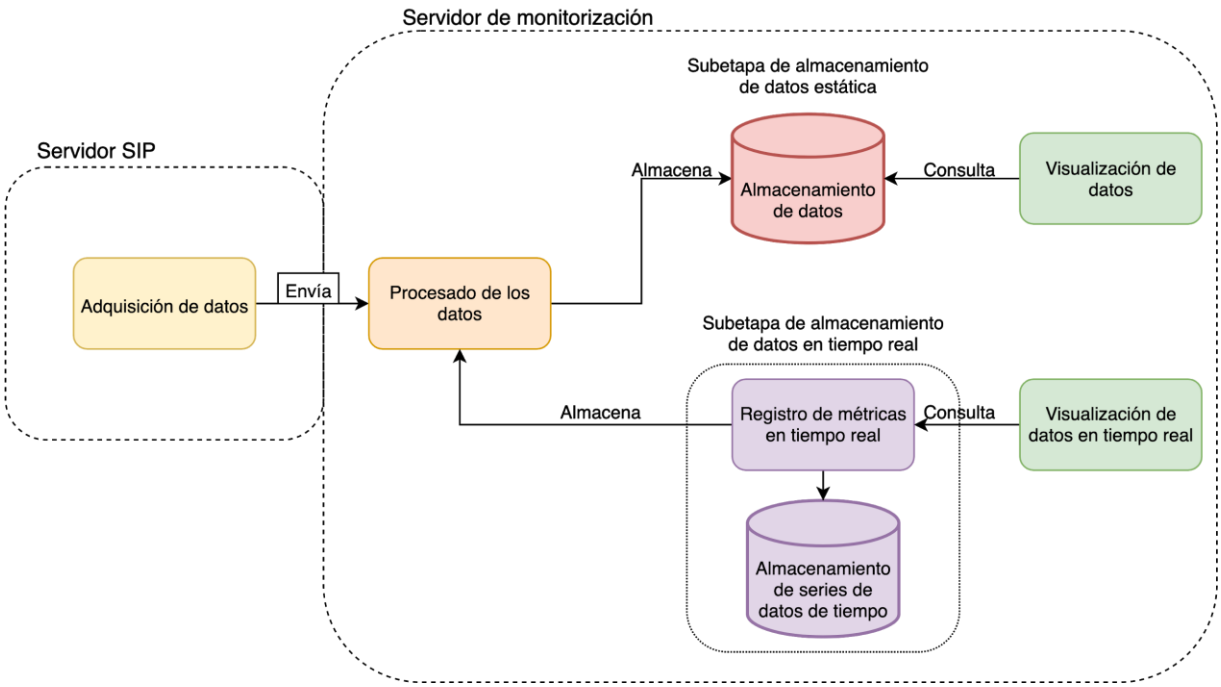

Ilustración 2: Esquema de monitorización en tiempo real

Se ha añadido una subetapa de almacenamiento de datos en tiempo real, ya que se ha considerado que, al ser la voz un servicio en tiempo real, cualquier degradación sustancial en este servicio que imposibilite la comunicación debe ser alertada inmediatamente para reducir el tiempo de reacción y minimizar el impacto que ha supuesto al servicio. Además, algunas métricas de calidad de calidad de voz y de rendimiento del servidor SIP aportan más información cuando son reportadas y visualizadas en tiempo real. Sin embargo, la visualización de datos en tiempo real puede limitar las características de la visualización de datos. En efecto, las herramientas usadas en este proyecto para obtener los datos en tiempo real no permiten la discriminación de estos datos por llamadas, aportando información de manera general: los paquetes perdidos, *jitter*, RTT y otras métricas de calidad[3] son calculadas a partir de todas las llamadas en cada tiempo. Esto permite tener una visión del rendimiento del servidor SIP al enrutar los paquetes y el estado de los *SIP trunks* que encaminan los paquetes a los operadores.

[3] Véase Anexo 3: "*Métricas de calidad en VoIP*"

Por todo esto, se ha mantenido el almacenamiento y visualización de datos convencional, es decir, sin tener el factor de tiempo como prioritario: a medida que van llegando paquetes se procesan y se almacenan, y la base de datos será accedida cuando el usuario lo requiera. De esta manera se puede acceder a todas las llamadas almacenadas y ver la evolución de los parámetros de calidad a lo largo de la llamada, pudiendo hacer así *troubleshooting* para llamadas y clientes en particular.

Para seleccionar los programas que forman cada etapa, se han considerado **solamente** los esquemas tales que los programas tengan el máximo grado de compatibilidad[4]. Estos esquemas pueden verse en el **anexo 5**.

Finalmente, se ha seleccionado el esquema que se ha considerado superior a los demás, basándose en criterios de complementariedad, recursos del *host* usados, velocidad y simplicidad. La selección final de *software* que formará parte de la *SIPCAPTURE stack* de este proyecto es:

- **Etapa de adquisición de datos**: *Heplify Agent Capture* frente a *Captagent o Kamailio*. *Heplify* es una versión más ligera y simple de *Captagent*, por lo que se gana en recursos computacionales. Por otro lado, se pretende que el servidor de monitorización disponga todos los recursos para la monitorización, por lo que la opción *Kamailio* queda descartada, al tratarse este de un servidor SIP.
- **Etapa de procesado de datos**: *Heplify Server*. Al haber escogido *Heplify* como agente de captura, es natural pensar que *heplify server* es el programa que tenga máxima complementariedad. De todas maneras, la decisión se toma también en base a la estabilidad que ofrecen ambos programas: *Heplify* se encuentra actualmente en una versión estable mientras que *Hepop* es un prototipo y aún se encuentra en desarrollo[5], tal y como se indica en el anexo 6.
- **Etapa de almacenamiento**:
    - **Registro de datos en tiempo**: *Prometheus*. Las herramientas alternativas eran *InfluxDB* y *Graphite*. Respecto a *InfluxDB*, las características son muy similares a *Prometheus* y, en algunos indicadores de rendimiento concretos, mejor[6]. Sin embargo, como se ha optado por utilizar *VictoriaMetrics* para almacenar la información, reduciendo el papel de *Prometheus* a conseguir la información de *Heplify*, almacenarla en *VictoriaMetrics* y finalmente recuperarla cuando sea necesario, las métricas de rendimiento de base de datos deben ser comparadas con *VictoriaMetrics*.
      El uso de *Prometheus* junto con *VictoriaMetrics* está sin embargo justificado: *Prometheus* ofrece un lenguaje mucho más sencillo que el "*SQL-like"* que usa *InfluxDB*. Además, *InfluxDB* tiene su almacenamiento basado en un sistema *PUSH*, es decir, para alimentar a *InfluxDB* se debería enviar la información desde *Heplify server*, por otro lado, *Prometheus* está

[4] Véase Anexo 5: "*Esquemas principales de instalación SIPCAPTURE"*
[5] Véase Anexo 5, apartado 5.2.2: "*Hepop"*
[6] Véase el estudio sobre la comparación de bases de datos en tiempo, por la Universidad de Stuttgard. Disponible en: http://btw2017.informatik.uni-stuttgart.de/slidesandpapers/E4-14-109/slides.pdf

basado en un sistema *PULL*[7]: es la propia base de datos quien registra el valor de las métricas disponibles en otro programa. Dado que *Heplify server* expone las métricas SIP y RTCP, resulta mucho más sencillo el sistema *PULL*.

Finalmente, *Prometheus* tiene una cantidad de librerías para soportar el reconocimiento de métricas de otros *software* muy superior a *Graphite*, por lo que se ha considerado, dado que solamente se realiza registro de datos y no almacenamiento, que *Prometheus* era más conveniente.

- **Almacenamiento de serie de datos de tiempo**: *VictoriaMetrics*, que es una solución concretamente optimizada para almacenar métricas que varían en el tiempo con *Prometheus*. La alta integración de ambos *software* ha sido el principal motivo, ya que de esta manera se puede seguir consultando la información mediante *PromQL*[8], mucho más simple que *SQL*. Se han considerado otras características, en las que *VictoriaMetrics* ha resultado ser superior a *InfluxDB*: la rapidez de operaciones de escritura y consulta, escalabilidad y poco consumo de recursos del *host*[9].
- **Almacenamiento de datos estático**: *PostgreSQL*. Para un almacenamiento convencional mediante tablas relacionales se ha considerado *Mysql* y *PostgreSQL*. Sin embargo, dado que se trabaja con numerosas particiones de tablas[10] y bastante de la información almacenada será de tipo *JSON*, se ha optado por *PostgreSQL*, dada su funcionalidad nativa de soportar *JSON* y tener mejor rendimiento y características respecto las particiones[11].

- **Etapa de visualización**:
  - **Visualización de datos**: *Homer v7*. Era la única solución *software* disponible y que contara con una comunidad de soporte importante.
  - **Visualización de datos en tiempo real**: *Grafana Server* frente a soluciones como *Kibana* o *Graphite*. *Kibana* posee una poderosa integración con *Elasticsearch* y *Homer*, pero se usa cuando la cantidad de datos es abrumadora y deben recurrirse a técnicas de *Big data*, así que, dado el volumen de tráfico que registramos y a la situación actual, se descarta tanto el uso de *Elasticsearch* como *Kibana*[12]. *Graphite*, por otro lado, se descarta al no ser usado en la subetapa de registro de datos.

[7] Véase Anexo 8, apartado 8.2: "*Prometheus"*
[8] Véase Anexo 6: *"Prometheus Query Language (PromQL)*".
[9] Véase el estudio de comparación de estas dos bases de datos. Disponible en https://www.slant.co/versus/6000/33519/~influxdb_vs_victoriametrics
[10] Véase Anexo 7*: "Estructura de la base de datos PostreSQL"*
[11] Véase el estudio: "*Analysis of DBMS: MySQL Vs PostgreSQL*", de Yang Xiaojie por la Universidad de Kemi-Tornio. Disponible en "https://core.ac.uk/download/pdf/38029082.pdf"
[12] Véase el proyecto de *GitHub*: "https://github.com/sipcapture/homer/wiki/Homer-Bigdata"

El esquema que representa el funcionamiento de la *SIPCAPTURE stack* usada en este trabajo es el que representa la figura 3:

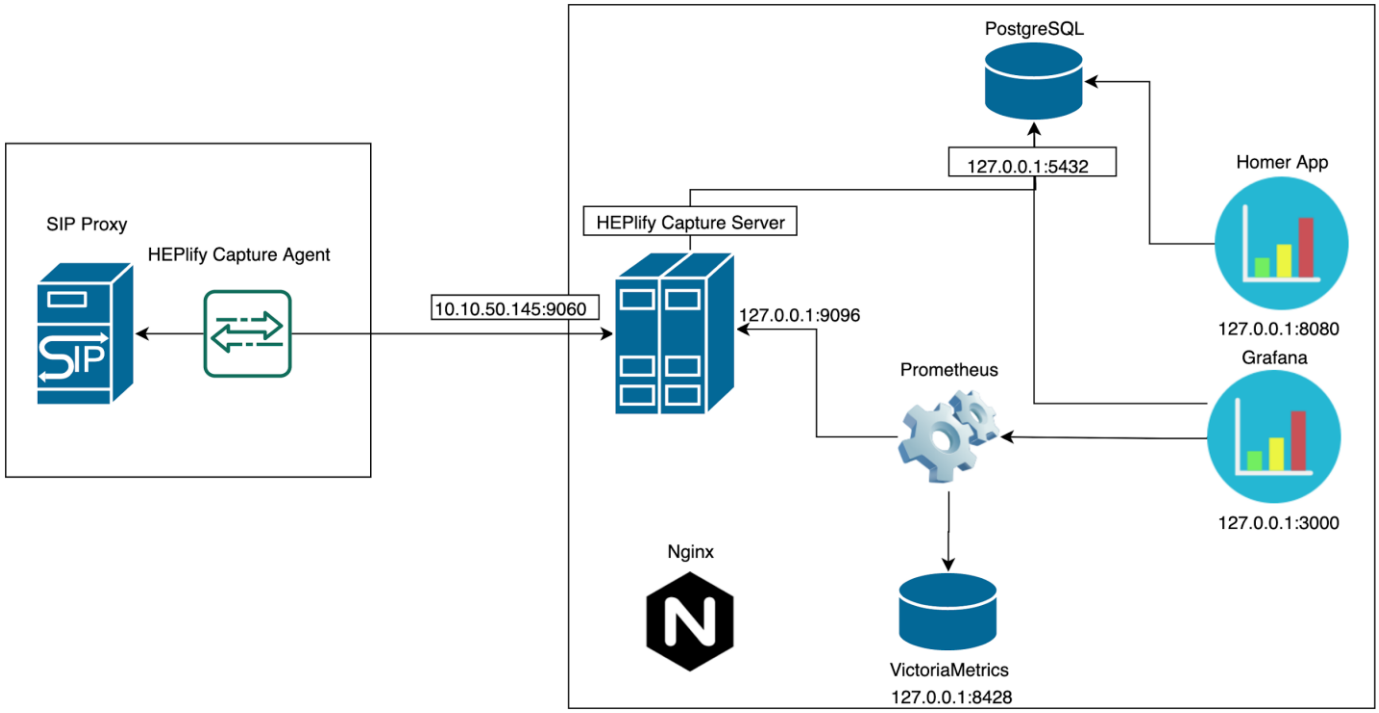

Ilustración 3: Esquema final de SIPCAPTURE

El tráfico SIP, RTP y RTCP que llega al servidor SIP *proxy* es interceptado por el agente de captura *Heplify Capture Agent*, que procesa cada paquete de acuerdo con el protocolo con el cual ha sido enviado[13] y envía otro paquete mediante el protocolo HEP/EEP (*Extensible Encapsulation Protocol*) con los datos formateados para que el servidor de monitorización los entienda. El protocolo HEP/EEP básicamente encapsula un paquete IP dentro de otro paquete IP para ser enviado mediante un *socket* TCP/SCTP/UDP.

El motivo de uso de este protocolo es para no alterar el paquete IP original ni ninguna parte de su cabecera, así como poder transmitir información adicional del paquete IP original dentro de su encapsulado. Se diseñó para el duplicado de paquetes por motivos de monitorización; de esta manera, por ejemplo, el agente de captura puede reenviar los paquetes de voz RTC, aportando nueva información y sin modificar el paquete original. En general este protocolo tiene más sentido cuando el agente simplemente reenvía los paquetes recibidos al servidor y ahí se procesan, pero en este proyecto se ha optado por partir el procesado entre agente y servidor.

Cuando el paquete llega al servidor de monitorización, *Heplify server* lo recibe y desencapsula. Obtiene la información formateada por *Heplify* (el agente de captura) expone cada una de las métricas obtenidas por los paquetes RTCP y SIP, es decir, toda la

[13] Véase Anexo 8: "*Arquitectura interna de los programes de SIPCAPTURE"*

información que extrae de los paquetes se etiqueta de acuerdo con unas métricas predefinidas y se hacen accesibles para ser obtenidas por otro programa[14].

Por otro lado, *Heplify-server* también actualiza la base de datos *PostgreSQL*[15] con los datos de los paquetes, pudiendo ser estos accesibles desde otros programas mediante consultas SQL.

Paralelamente, *Prometheus* tiene establecida una conexión con el puerto apropiado de *Heplify server* y periódicamente va obteniendo los valores de las métricas y les asigna una marca de tiempo (en inglés *timestamp*) creando así series de datos de tiempo y guardándolas en la base de datos *VictoriaMetrics.* Cada serie de datos de tiempo es de una métrica determinada y representa como ha cambiado su valor con respecto del tiempo.

Finalmente, la visualización de estos datos se puede hacer a través de dos servidores web: *Homer web* y *Grafana*. Junto con el servidor web *Nginx*, proporcionan una *web user interface (WUI)* para requerir, filtrar y visualizar los datos de monitorización.

La visualización de los datos se ha dividido ya que *Homer web* realiza consultas a la base *PostgreSQL*, y por lo tanto tiene acceso a los datos en tiempo no real. Sin embargo, mediante consultas SQL el usuario que monitoriza es capaz de ver un listado de todas las llamadas y acceder a la información específica, de protocolo y métricas de calidad, de cada una de las llamadas.

Por otro lado, *Grafana* realiza consultas *PromQL* a *VictoriaMetrics*, cuyos datos son almacenados en tiempo real, pero de manera general, es decir, cada métrica se evalúa por separado y sin hacer distinciones de llamada, por lo que mediante *Grafana* se podrá ver la variación de las métricas de calidad del sistema *VoIP* en general y en tiempo real.

[14] Véase Anexo 9: *"Métricas expuestas por Heplify"* para ver las métricas expuestas.
[15] Véase Anexo 7: "Estructura de *PostrgreSQL"* para ver las tablas de la base de datos.

11.2- VoIPmonitor

VoIPmonitor es un capturador y analizador de paquetes (*sniffer*) de código abierto, pero con un *front-end* (una interfaz web para poder interactuar) para analizar las comunicaciones SIP, RTP y RTCP comercializado y distribuido bajo una marca registrada. Mediante la implementación de este *front-end*, *VoIPmonitor* analiza la calidad de las llamadas *VoIP* basándose en los parámetros de red aportados por el protocolo RTCP. Al ser solamente el *sniffer* de código abierto, no se podrá proporcionar mucha información acerca de la arquitectura de VoIPmonitor, aunque si resumir cada componente y proporcionar un esquema general.

Los componentes básicos de VoIPmonitor son:

- *Web user Interface (WUI):* es un servidor web que proporciona una interfaz en PHP para poder controlar la monitorización, ver los gráficos y realizar el filtrado de los datos. Esta aplicación web puede estar alojada en un *host* con independencia de las otras partes de VoIPmonitor, aunque en este trabajo se ha optado por alojarla junto a la base de datos *MySQL*.
- *MySQL server*: almacena toda la información estadística que el *sniffer* extrae de las llamadas. Tanto la comunicación con la interfaz web como con los *sniffer* es mediante TCP y está cifrada.
- *Sniffer*: también llamado sensor, es un capturador de paquetes de red de código abierto que analiza en tiempo real las llamadas SIP con sus correspondientes flujos de paquetes RTP y almacena la información en forma de archivo CDR (*Call Detail Record*). Estos archivos son enviados mediante TCP a un servidor *MySQL*.

A continuación, se presenta un esquema que manifiesta resumidamente la interacción entre estos tres componentes

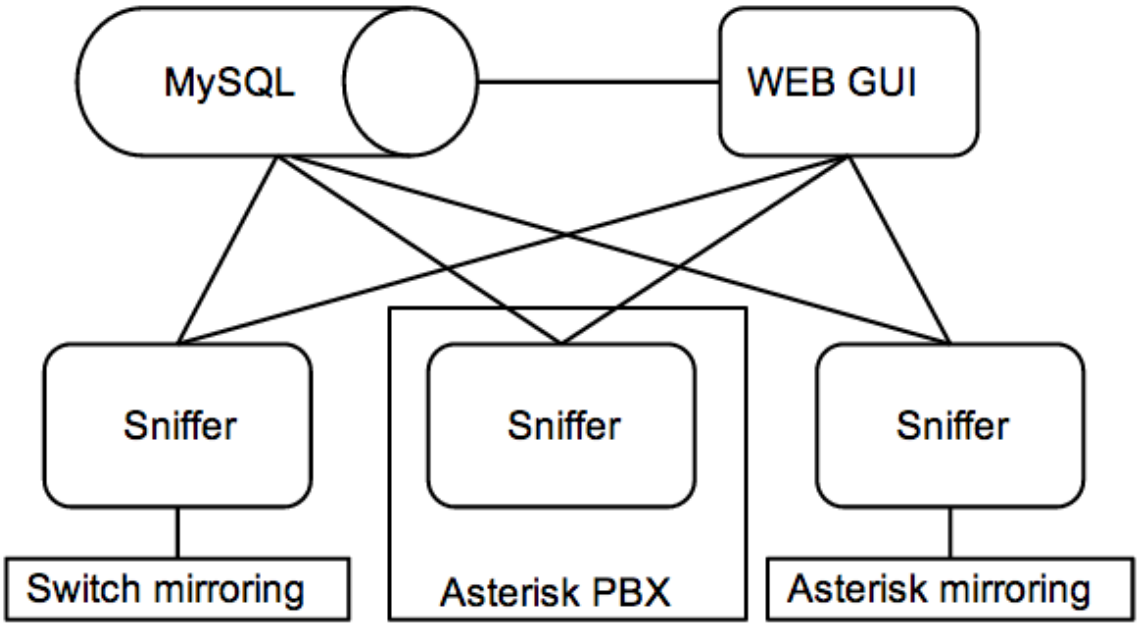

Ilustración 4: Esquema general de la arquitectura de VoIPmonitor

Cada *sniffer* o sensor está asociado a un servidor o sistema donde circula tráfico SIP, RTP y RTCP. El *sniffer* está conectado mediante el protocolo TCP tanto con la base de datos como con la interfaz web. Los archivos CDR generados por cada *sniffer* se guardan en la base de datos bajo una clave que identifica el sensor que ha aportado la información. La representación de estas estadísticas se realiza en la aplicación web PHP mediante consultas SQL a la base de datos.

Por otro lado, cuando desde la WUI se solicita escuchar la llamada en tiempo real, el *sniffer* envía todos los paquetes RTC al servidor web para que la voz pueda ser reproducida.

El *sniffer* tiene varios modos de configuración, cada cual modificará su comportamiento.

12- Comparación

La comparación que se procederá a realizar de ambas soluciones se hará en base a varios criterios. A saber, el coste de cada una, su escalabilidad, la información estadística de métricas QoS y el consumo de recursos del *host*. Una vez realizada la comparación, se concluirá qué servidor de monitorización resulta más apropiado para la empresa teniendo en cuenta el estado actual y las líneas de futuro.

12.1 Coste

VoIPmonitor tiene varios tipos de licencia, difiriendo cada una de la otra por el número de canales, es decir, llamadas activas simultáneas, que es capaz de monitorizar. Dado el *statu quo* del tráfico actual, la licencia adecuada sería de 200 canales, cuyo precio es 134 €/mes[16]. Nótese que la escalabilidad y el coste están estrechamente relacionados, dado que cuantos más canales haya, más cara es la licencia.

Por otro lado, la *stack* o conjunto apilado de programas de *SIPCAPTURE* es integralmente gratuito y puede distribuir bajo una licencia de *software* libre. Esto supone un gasto fijo nulo para la empresa, a diferencia de *VoIPmonitor*. Sin embargo, también se deben tener en cuenta los costes humanos o de persona que suponen ambos programas.

En efecto, el proceso de instalación, configuración y mantenimiento, incluyendo en este último la resolución de errores y, si cabe, investigación y desarrollo, es muy sencillo en *VoIPmonitor*, pues la instalación solamente es la ejecución de dos instaladores y la configuración solamente la modificación de un archivo, es decir, es una instalación de *software* convencional y por ello el tiempo que se debe dedicar a ello es bajo. Adicionalmente, *VoIPmonitor* cuenta con una plataforma de tiques para la resolución de incidencias muy rápida y efectiva. Por todo ello esta solución de monitorización no requiere casi tiempo por parte de la empresa y por lo tanto los gastos de personal son bajos.

Por otro lado, *SIPCAPTURE* requiere un tiempo considerable para su instalación y configuración, mucho mayor que en *VoIPmonitor*. Además, su mantenimiento también requiere más atención, ya que al haber múltiples programas interactuando, el proceso de identificación de errores y fallos es más costoso en tiempo, y se requiere de alguien que tenga conocimiento amplio del sistema y su configuración. Tampoco hay una plataforma de tiques privada como en *VoIPmonitor*, aunque *SIPCAPTURE* cuenta con una gran comunidad en *GitHub* de desarrolladores que reportan y contestan incidencias en cada uno de los programas que lo conforman.

En lo que concierne al desarrollo de nuevas funcionalidades, *VoIPmonitor* no requiere ni tiempo ni coste de investigación e implementación, ya que, al ser *software* privado, se encarga la empresa. El punto negativo es la falta de adaptabilidad que esto puede suponer en muchos casos, ya que no todas las características y cambios que puede requerir la empresa van a ser implementados. *SIPCAPTURE* en cambio sí puede adaptarse,

[16] Información disponible en https://www.voipmonitor.org/whmcs/cart.php?gid=1

incluyendo o cambiando algún programa de su *stack* o simplemente modificándolo; aunque entonces se debe tener en cuenta el coste del tiempo del desarrollo.

En síntesis, *VoIPmonitor* supone un gasto fijo mensual para la empresa al adquirir y mantener la licencia, pero este gasto supone también estabilidad y facilidad en el proceso de instalación y configuración, así como resolución de incidencias y por ello se minimizan todos los gastos de personal asociados a estos procesos. El desarrollo tampoco supone un gasto, ya que lo realiza la empresa distribuidora, pero esto implica una pérdida importante de adaptabilidad y autonomía por parte de la empresa que adquiere el programa.

Por otro lado, *SIPCAPTURE* no supone ningún gasto fijo, pero si tiene numerosos gastos variables de tipo humano desde su instalación hasta su mantenimiento y desarrollo. Aun contando con una comunidad que puede minimizar el tiempo de resolución de errores, no es deber de la misma solucionarlos luego este tiempo nunca será despreciable y por ende el gasto que supone tampoco. Además, se debe contar con personal que conozca el sistema para poder modificarlo, por lo que de no disponer se debería formar al trabajador, lo cual indirectamente también supone un gasto.

12.2 Escalabilidad

La escalabilidad es un factor importante, pues la empresa debe contar con un servidor de monitorización que satisfaga el crecimiento de los clientes previsto en cada trimestre y consecuentemente ambas soluciones deben soportar los incrementos del tráfico de voz previstos e imprevistos y sin sobredimensionar la infraestructura.

Ambos programas son escalables: en el caso de *VoIPmonitor*, ya se ha comentado en el apartado anterior que existe la posibilidad de ampliar la capacidad de monitorización de 200 canales a un máximo de 8000 llamadas concurrentes, cuya licencia supondría un coste de 948 € mensuales. Existe, por lo tanto, un límite de escalabilidad que actualmente dado el tráfico que se enruta no supone ningún problema, pero sí podría convertirse en uno en un futuro si las previsiones de crecimiento se cumplen. Por ende, esta solución presenta un problema de escalabilidad a largo plazo.

Por otro lado, los programas claves de *SIPCAPTURE* en el procesado de los datos de también se usan en otras soluciones de otro tipo de monitorización con un volumen de tráfico mucho más sobrecogedor que el máximo que presenta *VoIPmonitor*[17] [18] así que en principio el único factor que se debe tener en cuenta es el almacenamiento de los datos estadísticos de las llamadas en las bases de datos. En caso de crecimiento, el remodelado de la infraestructura sería relativamente sencillo, ya que simplemente se deberían externalizar las bases de datos (*VictoriaMetrics* y *PostgreSQL*) en un servidor distinto y cambiar los parámetros de configuración para que *Heplify server* almacene los datos en el

[17] Véase los artículos de "*Cloud Native Computing Fundation"*. Disponibles en https://www.cncf.io/blog/2016/08/24/prometheus-user-profile-monitoring-the-worlds-largest-digital-festival-dreamhack/ y en https://www.cncf.io/announcement/2016/05/09/cloud-native-computing-foundation-accepts-prometheus-as-second-hosted-project/

[18] Véase el estudio sobre la escalabilidad de varias bases de datos. Disponible en https://medium.com/@valyala/measuring-vertical-scalability-for-time-series-databases-in-google-cloud-92550d78d8ae

nuevo, servidor y la configuración de *Homer* y *Prometheus* para consultar las bases de datos ahora externalizadas. Luego la escalabilidad se reduciría a un problema de almacenamiento en este nuevo servidor, pero tampoco supone un problema grave, ya que hoy en día el almacenamiento es un recurso barato en comparación con otros recursos *hardware*.

Aunque ya se haya comentado en este trabajo, en este apartado es propicio destacar que *SIPCAPTURE* cuenta con un proyecto para adaptarlo a un esquema de *Big Data*, luego el problema de la escalabilidad no es nuevo para el proyecto *SIPCAPTURE* y actualmente ya está lo suficientemente desarrollado para que no suponga ningún problema a largo plazo.

12.3 Métricas QoS

En los **anexos 10 y 11** se ha expuesto las métricas de calidad que cada solución ha sido capaz de obtener y representar, tanto generales del sistema como particulares para cada llamada y con las representaciones pertinentes. Ambas son igual de importantes ya que las generales permiten evaluar el rendimiento del servidor SIP *proxy* monitoreado y las particulares facilitan un correcto *troubleshooting* para las llamadas degradadas por las condiciones de red u otros problemas.

En los siguientes cuadros se presenta una lista de métricas, tanto generales como particulares y se determina, de acuerdo con los anexos ya dichos, si cada una de las soluciones objeto de la comparación pueden calcularlas y representarlas.

Los gráficos de cada métrica se encuentran en el **anexo 12,** pero se explica cada uno en los **anexos 10 y 11**.

12.3.1 Métricas generales

En la siguiente tabla están las métricas de calidad y la información general de la comunicación que es capaz de enseñar cada una de las soluciones:

Métrica	*VoIPmonitor*	*SIPCAPTURE*
ASR	Sí	Sí
NER	Sí	Sí
IRA	No	Sí
SER	No	Sí
SEER	No	Sí
SCR	No	Sí

ISA	No	Sí
PDD (latencia)	Sí	No
MOS	Sí, adaptado a diferentes *jitter buffers*	No
Jitter	Sí, tanto de los paquetes RTP como RTCP.	Sí, de RTP.
RTT	Sí	Sí
Packet Loss	Sí	Sí
Fraction Loss	Sí	Sí
Llamadas concurrentes	Sí	Sí
User Agent	Sí	No
Códecs usados	Sí	No
Filtrado por número, países y cabeceras	Sí	No
Alertas	Sí	Sí

Tabla 1: Comparación de las métricas generales

En la siguiente tabla están las diferentes ratios que aportan información acerca del rendimiento del SIP *proxy* y el estado de la red:

Ratio	***VoIPmonitor***	***SIPCAPTURE***
Errores de establecimiento de sesión	Sí	Sí
Errores de registro	Sí	Sí
Paquetes SIP	Sí	Sí
Llamadas exitosas y fallidas	Sí	Sí
Registros exitosos y fallidos	Sí	Sí

Tabla 2: Comparación de las diversas ratios

12.3.2 Métricas particulares a cada llamada

Estas son las métricas de calidad particulares de la comunicación entre dos usuarios:

Métrica	*VoIPmonitor*	*SIPCAPTURE*
PDD	Sí	No
PDV (latencia)	Sí, por intervalos, total y en media	No
MOS	Sí, calculado para diferentes *jitter buffers.* Además, hay un gráfico con la variación de la métrica calculada a lo largo de la llamada.	No
Jitter	Sí, con gráficos de humo.	Sí
Packet Loss	Sí	Sí
Fraction Loss	Sí	No

Tabla 3: Comparación de las métricas particulares

Y en la siguiente tabla se representa la información particular de la llamada que puede aportar cada solución:

Información	*VoIPmonitor*	*SIPCAPTURE*
Números *caller* y *callee*	Sí	Sí
Call ID	Sí	Sí
Flujograma de paquetes	Sí	Sí
Archivo PCAP	Sí	Sí
Escucha/Transcripción de la llamada	Se puede escuchar la llamada en tiempo real.	No, aunque se puede implementar un sistema de transcripción.
Mapa de la llamada/red	Sí	No
Códecs usados	Sí	Sí
User Agent	Sí	No
Alertas	Sí	Sí

Tabla 4: Información expuesta por cada solución

En la comparación debemos diferenciar la información de la calidad del *SIP proxy* aportada y la información acerca de la calidad de una llamada en particular. Con respecto a la primera, ambas soluciones ofrecen una amplia información acerca de las llamadas y la capacidad y efectividad del servidor para establecerlas y mantenerlas.

Por un lado, tenemos métricas de calidad que dan información certera acerca del rendimiento del servidor SIP *proxy* y su capacidad para enrutar. A saber, ASR, NER, SER, SCR e ISA dan la información acerca de este rendimiento en el caso de las llamadas. Lo mismo ocurre con los registros en el caso de la métrica IRA, que se puede utilizar para ver los ataques SIP que recibe el servidor en tiempo real. En el caso de *VoIPmonitor*, algunas de las métricas mencionadas no están disponibles, pero este hecho no impide hacer un diagnóstico certero de la red *VoIP*, ya que se prescinde de las métricas que redundan la información de otras. *SIPCAPTURE,* por su lado, expone estas métricas y esto permite tener una amplia visión en el proceso de identificar y descartar problemas en el servidor.

Otro factor del cual se debe hacer hincapié es la manera que tienen ambas soluciones de filtrar esta información: mientras que *SIPCAPTURE* no realiza ningún filtrado, *VoIPmonitor* permite filtrar toda la información general por número de destino u origen, pudiendo filtrar por rangos de número para conocer la estadística *VoIP* de una empresa cliente, o para acotar el problema viendo si las degradaciones de calidad son sistemáticas de un número en concreto o han surgido en una llamada determinada. Esta es una gran ventaja que tiene *VoIPmonitor* sobre *SIPCAPTURE,* dado que este último solamente aportará información cuando el servidor se desvíe de su funcionamiento nominal y no puede discriminar por números. En otras palabras, *SIPCAPTURE* puede manifestar solamente cuánto éxito tienen las sesiones enrutadas, sin diferenciarlas, por él y en caso de mal funcionamiento puede alertar. *VoIPmonitor* no sólo puede realizar las mismas funciones, sino que además permite acotar el problema mediante el filtrado de estas estadísticas generales.

Por otro lado, cabe destacar dos puntos fuertes de *VoIPmonitor* frente a *SIPCAPTURE*: la representación de la puntuación *MOS.* La puntuación *MOS* es considerada la métrica más importante a la hora de evaluar la calidad de una llamada a lo largo del tiempo y *VoIPmonitor* no solamente calcula esta métrica, sino que además lo hace para varios *jitter buffers,* permitiendo conocer si, en caso de mala calidad, qué *jitter buffer* podría haber subsanado las degradaciones sufridas. En este trabajo **no** se ha podido configurar *SIPCAPTURE* para poder representar esta métrica.

Por lo que respecta al resto de la información general y las ratios, *VoIPmonitor* vuelve a estar ligeramente por encima al poder mostrar los *códecs*[19] y los *User Agent,* aunque cuando se ve de manera general esta información es de menor importancia que la anteriormente dicha.

En cuanto a la información particular de cada llamada, aquí *VoIPmonitor* destaca por su capacidad de exponer mucha información y representarla muy adecuadamente. En efecto,

[19] Véase Anexo 4: "*Códecs de audio"*

se manifiestan todas las métricas de calidad de una llamada (*packet loss, jitter,* latencia o PDV y MOS) y además en el caso del MOS también se calcula para varios *jitter buffer*. Para cada una de las métricas se puede hacer una representación gráfica de su variación en el tiempo. Por otro lado, *VoIPmonitor* representa 2 gráficos para cada SIP *endpoint* de la pérdida y latencia de los paquetes de sendos protocolos RTCP y RTP, por lo que se puede acotar y ver dónde se pierden los paquetes y la latencia con la que llegan a cada nodo final. Además, dos de los cuatro gráficos son de humo, es decir, para cada marca de tiempo hay una zona gris que representa el máximo y mínimo de tiempo con el cual han llegado los paquetes a ese *endpoint* y por lo tanto el *jitter* también es conocido mediante estos gráficos.

SIPCAPTURE, por otro lado, solamente ha podido ser configurado para calcular *packet loss* y *jitter*, las métricas más importantes a la hora de evaluar la calidad de la llamada después de la puntuación MOS. En este caso, ambas métricas son representadas en un gráfico de barras a lo largo del tiempo con valores cada pocos segundos. Además, se dan los valores mínimo, máximo y medio para cada métrica.

No solo la información es más completa en *VoIPmonitor*, nótese que la información básica de la calidad se complementa con variables como el PDD, que se usa como indicador congestión de red; sino que además la representación gráfica que hace *VoIPmonitor* de estas variables se antoja más propicia para realizar una detección de errores en la llamada. La representación de *SIPCAPTURE* también permite realizar un *troubleshooting*, pero es mucho más simple y la información extraída no es tan detallada.

Por último, la información básica de la llamada (números, flujograma…) que da la última tabla es dada por ambas soluciones. La información suplementaria como mapas de red, escuchar la llamada en tiempo real o los *User Agent* solamente se da en *VoIPmonitor*.

12.4 Consumo de recursos

Cada solución se ha instalado y configurado en una máquina virtual dedicada exclusivamente a la monitorización, por lo que se ha procedido a ver cuánta memoria y almacenamiento global se ha ocupado durante el proceso de monitorización mediante el programa *htop*, sin discriminar los procesos, ya que todos los procesos, o bien son comunes para ambos servidores de monitorización o bien son exclusivos para cada una de las soluciones. En consecuencia, la memoria y almacenamiento usados en ambas máquinas determinarán el uso que hacen cada una de las soluciones de estos recursos.

En el caso de la memoria, el *SIPCAPTURE* emplea 716 MB de los 7,8 GB que dispone la máquina virtual. En este caso se ha sobredimensionado ya que se desconocía de antemano el uso que iba a hacer esta herramienta de monitorización. Por otro lado, *VoIPmonitor* emplea 1,5 GB de los 7,8 GB de los que dispone el sistema. En este caso *VoIPmonitor* utiliza un poco más del doble de memoria que *SIPCAPTURE*.

Por otro lado, por lo que respecta a almacenamiento, se ha usado el comando *df -h* para ver cuánto espacio hay en la partición donde se usa cada herramienta. Además, para hacer equitativa la comparación, se ha calculado cada uso de almacenamiento 2 meses después del inicio de ingesta de datos por parte de cada solución. En el caso de *SIPCAPTURE* se

emplean 4,8 GB de los 38 GB disponibles en la partición, de nuevo, se ha sobredimensionado. Por otro lado, *VoIPmonitor* ha empleado 5,6 GB de 38.

Estos resultados son bastante previsibles, ya que la *stack SIPCAPTURE* guarda los datos en dos bases de datos distintas: *PostgreSQL* y *VictoriaMetrics*, por lo que, aunque estén optimizadas, si los datos que se guardan son aproximadamente los mismos por parte de ambas herramientas, *SIPCAPTURE* siempre empleará más almacenamiento. Además, cuanto más tiempo transcurra, más datos se almacenarán y esta diferencia será cada vez mayor.

En cuanto a la memoria usada, ambas herramientas están optimizadas y usan bastante menos de un máximo común de memoria. Por ende, no debe ser un factor de alta importancia a la hora de escoger. Por otro lado, el almacenamiento es un recurso barato y fácilmente *externalizable* a otro *host*, así que tampoco supone un factor decisivo.

13 Conclusiones y líneas de futuro

En este trabajo se han presentado, explicado y comparado en base a varios criterios dos soluciones de monitorización de voz: una solución privada (*VoIPmonitor*) y una de código abierto (*SIPCAPTURE*). Pese a tener un objetivo común, ambas soluciones son diferentes tanto a nivel de arquitectura como de configuración e información aportada. Dada la naturaleza de su licencia y a estas diferencias, cada programa tiene una ventaja fundamental que lo diferencia del otro. En el caso de *VoIPmonitor*, es un programa distribuido bajo licencia de pago y resulta una solución robusta, fiable y con un equipo técnico que realiza una resolución de tiques efectiva. Estas ventajas, sin embargo, no están exentas de inconvenientes consecuentes de la naturaleza privada del software. En efecto, se depende totalmente de una empresa y de los cambios que aplica y esto implica que, si bien es cierto que puede haber una idoneidad inicial entre esta solución y los requisitos del cliente, si estos cambiaran, hay una falta de adaptabilidad importante, ya que se depende del departamento de I+D de la empresa vendedora. Naturalmente, otro inconveniente es el coste de la licencia en sí, que supone un gasto fijo mensual para el cliente.

SIPCAPTURE, por otro lado, es un software de código abierto, y esto ya constituye una ventaja de por sí, dado que es el propio cliente quien puede, mediante su departamento de I+D, adaptar el programa a las necesidades de la empresa dinámicamente e incluso investigar y programar nuevas funcionalidades. Adicionalmente, *SIPCAPTURE* cuenta con una numerosa comunidad de programadores y usuarios en *GitHub* que desarrollan continuamente e investigan errores del programa u otros comportamientos que se desvíen del funcionamiento nominal. Sin embargo, al igual que *VoIPmonitor*, esta solución cuenta con otros inconvenientes connaturales a su naturaleza libre: requiere un coste humano de configuración, mantenimiento y desarrollo por parte de la empresa cliente, por lo que toda la adaptabilidad y gratuidad de licencia iniciales se pagan con un equipo técnico que tenga que realizar el trabajo.

El criterio fundamental para poder escoger qué *software* es más adecuado para la empresa es la información que cada solución es capaz de aportar. Tal y como se ha visto en el apartado de comparación, es claro que *VoIPmonitor* es superior a *SIPCAPTURE* tanto en la exposición de métricas de calidad generales como particulares para cada llamada.

En realidad, el *quid* de la comparación no es tanto el número de métricas que cada solución es capaz de exponer, sino cuáles. Las métricas consideradas más importantes a la hora de evaluar la calidad de un sistema o comunicación VoIP son *jitter*, *packet loss* y MOS; siendo la puntuación MOS la de mayor importancia. Es precisamente esta métrica la que descalifica a *SIPCAPTURE*, puesto que no se ha podido configurar de manera que la pueda exponer de manera general ni particular. Sin embargo, hay dos cosas de *SIPCAPTURE* que cabe resaltar:

- Se exponen las métricas *jitter* y *packet loss*.
- La información general se expone en *dashboards* y se ve en tiempo real. En *VoIPmonitor* se puede solicitar y configurar alertas en tiempo real, pero no se puede visualizar la información en este tiempo.

Es por todo ello que, aunque la diferencia de información aportada es decisiva, no es abrumadora.

Por otro lado, también se debe tener en cuenta la escalabilidad. Como se ha visto anteriormente, *VoIPmonitor* presenta límites, que, aunque lejanos, existentes de escalabilidad. Estos límites no suponen un problema ni a corto ni a medio plazo si no hay un crecimiento abrupto de tráfico de voz en los servidores SIP de la empresa, cosa que es improbable que suceda. Aun así, sí supone un problema a largo plazo si la empresa crece monótonamente y este hecho da una ligera ventaja a *SIPCAPTURE*, que es fácilmente escalable más allá del límite que presenta *VoIPmonitor*.

Se concluye, por lo tanto, que la solución propicia para la empresa a corto y medio plazo es *VoIPmonitor,* dada su robustez e información. Sin embargo, es una solución transitoria, ya que al mismo tiempo se investigará e implementará la exposición de la métrica MOS en *SIPCAPTURE* y una vez se haya logrado, se podrá hacer la transición a esta solución. Por lo que respecta a los costes, es cierto que *SIPCAPTURE* implica tener un coste de personal, pero este se estabilizará cuando el programa funcione adecuadamente para la empresa, es decir, exponiendo todas las métricas. Una vez conseguido, solamente supondrá un coste la implementación o cambio de una funcionalidad, pero es un precio justo por toda la adaptabilidad que se ofrece. Por otro lado, *VoIPmonitor* supone un coste sin garantía de adaptación y fijo, que es el precio a pagar por la garantía de robustez que tiene.

En conclusión, a medio plazo se puede conseguir la exposición de todas las métricas y la robustez suficiente en *SIPCAPTURE* para prescindir de *VoIPmonitor*, por lo que habrá una diferencia ventajosa de costes para la solución libre al tener que invertir solamente cuando se requiera modificar alguna parte.

Una vez implementado el sistema *SIPCAPTURE*, se podrá adaptar dinámicamente a las necesidades de la empresa. Además, existen algunos proyectos, aún en desarrollo, pioneros con *Machine Learning* para implementar con *SIPCAPTURE* y hacer el salto a la monitorización proactiva…

Bibliography:

Johnson, D. Schutzer. "*SIP: Understaing the Session Initiation Protocol"*, 1ª. edición, Artech House Publishers, 2009. ISBN 1607839954.

M. Tawfiq Al-Akhras, I. Almonani. "*VoIP Quality Assessment Technologies"*, DOI 10.5772/14438, febrero 2011. Disponible en "https://www.researchgate.net/publication/221910733_VoIP_Quality_Assessment_Technologies"

Basic Telephony SIP End-to-End Performance Metrics. IETF RFC 6076, enero 2011.

RTP: A Transport Protocol for Real-Time Applications. IETF RFC 3550, julio 2003.

SDP: Session Description Protocol. IETF RFC 4566, julio 2006.

Luca Deri, "*Open Source VoIP Traffic Monitoring"*, NTOP, 2013. Disponible en https://www.cosmocom.gr/wp-content/uploads/2013/05/VoIP-2.pdf

Bader, O. Kopp, M. Faklenthal, *"Survey and Comparison of Open Source Time Series Databases"*, Universidad de Stuttgart, 2017. Disponible en http://btw2017.informatik.uni-stuttgart.de/slidesandpapers/E4-14-109/slides.pdf

Y. Xiaojie "*Analysis of DBMS: MySQL Vs PostgreSQL*", Universidad de Kemi-Tornio, 2014. Disponible en "https://core.ac.uk/download/pdf/38029082.pdf"

"SERIES P: TELEPHONE TRANSMISSION QUALITY, TELEPHONE INSTALLATIONS, LOCAL LINE NETWORKS", ITU-T P.10/G.100, 2017.

Glossary

ASR: *Answer Seizure Ratio,* el porcentaje de llamadas exitosas.
CDR: *Call Detail Record*, archivo que contiene toda la información de una llamada.
DNS: *Domain Name Service*, servicio que traduce direcciones IP a nombres de dominio.
EEP: *Extensible Encapsulation Protocol*, protocolo usado en la monitorización en teimpo real.
HEP: siglas por las que se conoce formalmente al protocolo EEP.
HTTP: *Hypertext Transfer Protocol*, protocolo que se usa para la transferencia de información por Internet.
IP: *Internet Protocol*, protocolo de comunicaciones para hacer lo propio por Internet.
IRA: *Ineffective Registration Attempts*, porcentaje de registros SIP fallidas en el servidor.
ISA: *Ineffective Session Attempts*, porcentaje de llamadas fallidas.
ITU: *International Telecommunication Union*, organismo regulador de las telecomunicaciones.
JSON: *Javascript Object Notation*, formato de texto para intercambiar datos entre programes.
KPI: *Key Performance Indicators,* indicadores de rendimiento clave.
MOS: *Mean Opinion Score*, puntuación subjetiva de la calidad de una sesión.
NER: *Network Effectiveness Ratio,* ratio que describe la habilidad de la red para establecer y mantener la llamada.
NTP: *Network Time Protocol*, protocolo para sincronizar correctamente el reloj de un servidor o equipo.
PCAP: *Packet Capture,* extensión para los archivos que contienen tráfico de red.
PDD: *Post Dial Delay*, tiempo transcurrido entre enviar la señal de inicio de llamada y oir el primer pitido.
PDV: *Packet Delation Variation,* equivalente, a efectos prácticos, al *jitter*.
PHP: *Hypertext Preprocessor*, lenguaje de programación para el procesado de texto, normalmente presente en un servidor web.
PromQL: *Prometheus Query Lenguage*, lenguaje de consultas para *Prometheus.*
RFC: *Request Form Comments*, monográfico que varios expertos en una materia hacen llegar a IETF para consensuar un tema.
RTP: *Real Time Transport Protocol*, protocolo para el transporte de datos en tiempo real.
RTCP: *Real Time Transport Control Protocol,* protocolo que controla las comunicaciones con RTP.
RTT: *Round Trip Time,* tiempo transcurrido entre que un paquete sale de una màquina, llega al destino y vuelve.
SCR: *Session Completation Ratio,* igual que la métrica ISA, pero excluyendo causes que típicamente se atribuyen al usuario.
SCTP: *Stream Control Transmission Protocol*, protocolo usado para el intercambio de datos multimèdia.
SEER: *Session Establishment Effectiveness Ratio,* igual que SER, pero excluyendo del cálculo las razones típicamente atribuidas al usuario.
SER: *Session Establishment Ratio,* métrica que mide la capacidad de un *Proxy* o *User Agent* para establecer sesiones.
SIP: *Session Initiation Protocol,* protocolo usado para establecer, mantener y desconectar sesiones multimedia entre dos o más usuarios.

SQL: *Structured Query Language*, lenaguaje usado para realizar consultas en algunes bases de datos.
TCP: *Transmission Control Protocol*, protocolo para intercambiar datos contando con técnicas de recuperación de los mismos.
UDP: *User Datagram Protocol*, protocolo para intercambiar datos.
VoIP: *Voice over IP*, paradigma en el que la información de voz se propaga mediante Interent y no por una red telefónica pública conmutada.
WUI: *Web User Interface*, interfaz de usuario web.