ISSN 1688-2806

Universidad de la República
Facultad de Ingeniería

Desarrollo de Plataforma de Monitoreo de Redes para Plan Ceibal

Tesis presentada a la Facultad de Ingeniería de la Universidad de la República por

Iván Barbot, Santiago Bentancur, Santiago Saralegui

en cumplimiento parcial de los requerimientos para la obtención del título de Ingeniero Electricista.

Tutores

Gabriel Gómez ..... Universidad de la República
Germán Capdehourat ..... Universidad de la República

Tribunal

Eduardo Grampín ..... Universidad de la República
Eduardo Cota ..... Universidad de la República
Pablo Flores ..... Universidad de la República

Montevideo
miércoles 24 junio, 2020

Desarrollo de Plataforma de Monitoreo de Redes para Plan Ceibal, Iván Barbot, Santiago Bentancur, Santiago Saralegui.

ISSN 1688-2806

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 161 páginas.
Compilada el miércoles 24 junio, 2020.
`http://iie.fing.edu.uy/`

Agradecimientos

En primer lugar queremos agradecer a la Universidad de la República, al Instituto de Ingeniería Eléctrica y a nuestros tutores Gabriel Gómez y Germán Capdehourat por guiarnos y estar a nuestra disposición durante el desarrollo de este proyecto. Siguiente queremos agradecer a Plan Ceibal por ser parte del mismo y brindarnos elementos de su infraestructura. En especial a Maximiliano Perez por ser nuestra contraparte y al resto del equipo de ingeniería de Plan Ceibal. Por último queremos agradecer a nuestras familias pues sin ellos nada de esto hubiera sido posible y a nuestros amigos por su apoyo a lo largo de este proceso.

Esta página ha sido intencionalmente dejada en blanco.

A nuestras familias.

Esta página ha sido intencionalmente dejada en blanco.

Resumen

En la actualidad existen múltiples sistemas de monitoreo capaces de evaluar determinadas métricas de una red de datos, con el propósito de brindar información acerca del estado de salud de la red.

Algunos de estos sistemas se concentran principalmente en métricas relacionadas con el rendimiento interno de los equipos y no en aspectos vinculados a la conectividad de la red.

Por lo tanto este proyecto se basó en principio en el diseño e implementación de una plataforma cuyos módulos de medición puedan evaluar aspectos tales como la calidad de servicio de conectividad o calidad de experiencia de los usuarios. Para la toma de mediciones, la plataforma debe interactuar con una variedad de herramientas informáticas de código abierto. Al seleccionar dichas herramientas, se priorizaron aspectos tales como: el bajo consumo de recursos en los equipos, la facilidad de su manejo e instalación, su exactitud y la cantidad de tráfico inyectado en la red al tomar medidas.

En segundo lugar la plataforma posee la cualidad de poder ser enteramente gestionada desde un punto central de la red. Desde dicho punto se pueden realizar tareas tales como seleccionar nuevos nodos donde tomar mediciones o cambiar la configuración en los nodos donde ya se están tomando.

Entre las principales virtudes del producto final se puede destacar que se logró un diseño liviano, escalable, intuitivo y poco invasivo para la red. El hecho de que los módulos sean capaces de funcionar en base a bajos niveles de tráfico, hace que las mediciones puedan ser tomadas en horarios de gran demanda en la red, sin perjudicar la calidad de experiencia de los usuarios.

Como último punto importante acerca de la plataforma se debe mencionar que su código fuente es fácil de leer e intuitivo. Esto permite que futuros usuarios puedan adaptarlo a diversos tipos de escenarios o que el mismo sirva como punto de partida para nuevos proyectos. Hay que destacar que ésta plataforma se pensó de cara al futuro, es decir se planificó e implementó para que permitiera agregar nuevos módulos aún no desarrollados.

En particular, la contraparte interesada de este proyecto es Plan Ceibal, quien posee una red con las características necesarias para poner en funcionamiento la plataforma desarrollada, donde ya se encuentran instaladas algunas de las herramientas de monitoreo utilizadas. Por lo tanto se cree que el monitoreo que se realiza actualmente sobre dicha red podría verse enriquecido por los resultados de este proyecto, pues los módulos desarrollados podrían eventualmente expandirse a todos los centros educativos del país vinculados a Plan Ceibal. Cabe mencio-

nar que tanto con la plataforma, como con los módulos se realizaron pruebas en producción en la red Ceibal, lo que permitió no solo validar el proyecto sino que obtener un producto final más depurado y por lo tanto mejor.

Tabla de contenidos

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 1

Introducción

1.1. Antecedentes

El rápido crecimiento de las redes en la actualidad hace que la monitorización de las mismas sea un aspecto fundamental. Para esto existe una amplia variedad de herramientas que permiten comprobar la salud de la red y de sus equipos, evaluando determinadas métricas. La red de una organización puede ser considerada uno de sus recursos más críticos, poder administrar las redes para obtener su mayor rendimiento es un gran desafío. La caída o mal funcionamiento de un equipo o servicio de la red puede tener grandes repercusiones y el monitoreo es una de las principales herramientas para prevenir o estar informado de este tipo de escenarios. Para escoger la herramienta de monitoreo a utilizar es necesario tener en cuenta diversos factores tales como el tamaño de la red, la cantidad de tráfico que circula por la misma, aspectos concretos que se desea evaluar, entre otros. Puede ocurrir que la herramienta seleccionada no termine de satisfacer las necesidades del usuario. Puede que esta funcione de forma rápida e inteligente pero aún así no permita por ejemplo evaluar una métrica concreta. En la actualidad algunas de las herramientas más usadas son Zabbix, Cacti, Nagios, entre otras a las que se hace referencia en el capítulo 2.

En este proyecto, la contraparte interesada es Plan Ceibal, el cual cuenta con una red de datos que abarca todos los centros educativos públicos del país y que podría verse beneficiada con los resultados obtenidos.

A continuación se dará una breve descripción de dicha red. A agosto de 2019 esta cuenta con 2850 centros educativos que disponen de una red Wi-Fi y un 97 % de usuarios acceden a Internet mediante banda ancha [1]. A su vez de estos centros educativos, 1350 que se encuentran en zonas urbanas poseen equipamiento de videoconferencia. Todos los centros educativos se encuentran conectados al servidor central de Plan Ceibal que se encuentra ubicado en un *datacenter* de Antel y por lo tanto pueden ser monitorizados desde ese punto.

Los centros educativos poseen una estructura de red similar a la de la figura 1.1, en la cual existe un *router* de borde que es el punto de egreso/ingreso que brinda a los usuarios finales acceso a la red Ceibal y al resto del mundo. A este

router se conectan por un lado uno o más puntos de acceso (AP), los cuales sirven como anclaje para los usuarios. Por otro lado puntos de monitoreo de la red Ceibal, los cuales hacen uso de equipos tales como PCs. Actualmente se está tendiendo a remplazar estos equipos por Mini-PC, los cuales se tratan de PCs pero de dimensiones pequeñas, útiles para una amplia gama de tareas. El modelo de Mini-PC que se está empezando a utilizar en la red Ceibal es la Odroid-C2.

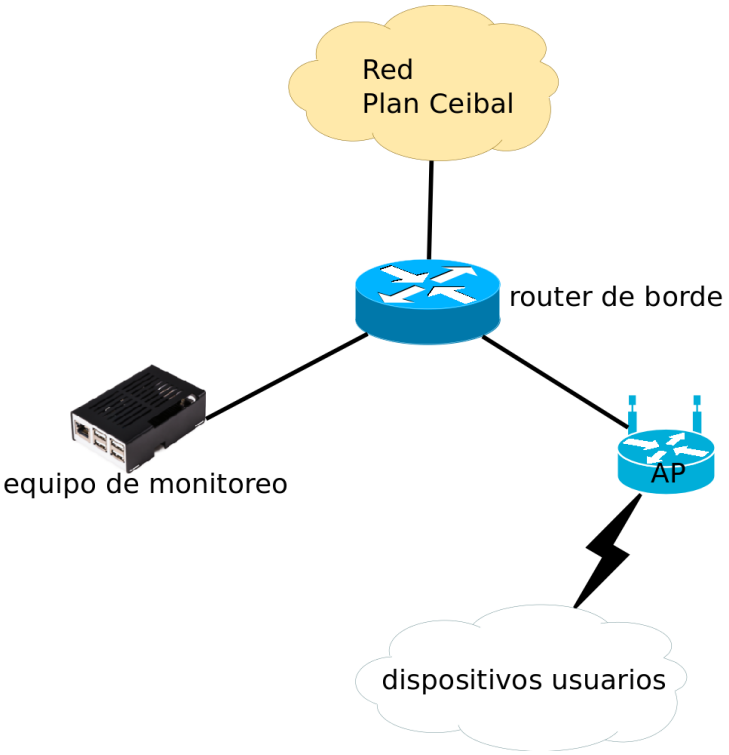


Figura 1.1: Estructura de red de un centro educativo.

Con una red de este tamaño es esperable que Plan Ceibal requiera de una buena monitorización sobre sus equipos. Por lo tanto uno de los programas que utiliza para tal fin es Zabbix [2]. Este ofrece una forma de monitorear una gran cantidad de equipos de forma centralizada y eficiente a través de distintas vías, tales como *scripts* y SNMP [6], entre otros. Sin embargo hay módulos de medición que no tiene implementados y por lo tanto hay aspectos referidos a la calidad de conectividad de la red que en la actualidad no se miden. El monitoreo que realiza Plan Ceibal actualmente sobre los servidores de los centros educativos está mayoritariamente enfocado en aspectos relacionados con el rendimiento interno de los equipos. Algunas de estas medidas son por ejemplo carga de CPU y uso de memorias en equipos tales como: *routers*, *switches*, APs, UPSs, PoEs, entre otros. Así como una medida esporádica del ancho de banda en horarios de bajo tráfico.

Por lo tanto se busca implementar módulos que permitan tomar nuevas medidas y de esta forma generar datos que brinden información que pueda dar un mejor

panorama de la salud de la red. De esta forma se podrá obtener datos sobre otros aspectos relacionados a la red, por ejemplo la latencias de un sitio web o ancho de banda de un determinado punto de la red. Aspectos que están relacionados a la calidad de servicio o calidad de experiencia y que pueden afectar a cualquiera de las dos. De esta forma se podrán generar alarmas o detectar fallos, permitiendo en un futuro tomar decisiones para mejorar el rendimiento de la red, logrando brindarle un mejor servicio a los usuarios.

Uno de los mayores desafíos es poder diseñar e implementar una plataforma que pueda integrar de forma eficiente las herramientas ya usadas por Ceibal y que permitan interactuar con los módulos alojados en los centros educativos. Es importante notar que en este proyecto una de las partes de mayor dificultad y que involucra una gran cantidad de tiempo es el diseño de la plataforma. La misma debe cumplir con las expectativas de Ceibal y a su vez debe agregar funcionalidades que permitan mejorar el monitoreo que ya lleva Ceibal sobre su red. Esto involucra un profundo estudio de las herramientas Zabbix y Ansible [5] (permite la distribución, configuración y administración de archivos dentro de una red). A su vez es necesario tomar varios criterios de diseño en los cuales se busca eficiencia pero a sus vez brindar soluciones con un cierto grado de simplicidad.

1.2. Objetivos

El objetivo principal de este proyecto es desarrollar una plataforma, la cual se llamará Moniceibal, que permita fortalecer el monitoreo que Plan Ceibal realiza actualmente sobre su red. Para esto se buscará crear módulos de medición aún no implementados en Zabbix que permitan obtener mediciones acerca del estado de conectividad de la red.

Las mediciones que se incorporan son:

- *Tiempo DNS:* tiempo que toma en obtener la dirección IP de determinado servidor, dada cierta URL.
- *Tiempo DHCP:* tiempo que toma el servidor DHCP en asignar una IP a un nuevo equipo que ingresa a la red.
- *Tiempo de Carga de Sitio Web:* tiempo que se tarda en acceder al contenido de una página web seleccionada por el administrador.
- *Ancho de Banda:* Estimación en forma regular de la capacidad disponible del enlace entre cada centro educativo y el servidor central.

  Las medidas de ancho de banda pueden generar tráfico excesivo que perjudique el desempeño de la red, en este caso se buscará utilizar un *software* que haga un monitoreo "liviano" que sea capaz de estimar la capacidad disponible en base a inyectar poco tráfico, permitiendo realizar dicha medida de forma periódica en cualquier momento del día.

Esta plataforma deberá poner en funcionamiento los módulos que sean creados en cada centro educativo seleccionado, gestionar su distribución y a su vez poder administrarlos de forma centralizada. Para poder lograr estos objetivos Moniceibal deberá hacer uso de herramientas como Ansible y a su vez interactuar con Zabbix, para poder aprovechar las funcionalidades ya brindadas por este.

Como objetivo general se planteó poder realizar un aporte al sistema de monitoreo de Zabbix dado que se implementaron nuevos módulos que permiten a dicha aplicación recabar nuevas mediciones. A su vez se busca que sirva como guía para trabajos futuros en los que se busque adaptar Zabbix u otras herramientas de monitoreo de red con aplicaciones que distribuyan archivos de forma remota como lo es Ansible.

1.3. Generalidades de la plataforma

Aunque el funcionamiento de la plataforma y la interacción entre sus componentes no será la misma para todos los módulos, en esta sección se pretende dar una idea general.

En primer lugar es necesario acondicionar cada uno de los equipos que se utilizará para tomar medidas, ya sea por ejemplo, instalando programas, creando o modificando archivos, entre otras acciones. Cada una de estas tareas puede ser realizada por el programa Ansible, que como se ve en la figura 1.2 será el encargado de preparar los equipos para la toma de mediciones.

Posteriormente se utilizará el programa Zabbix el cual debe estar instalado en el servidor de Plan Ceibal que se encuentra dentro del *datacenter* de Antel y en cada centro educativo a monitorizar. Este programa permite agendar las frecuencias o fechas de medición en el servidor central para luego dar la orden de ejecución correspondiente a cada centro educativo y almacenar los resultados obtenidos en su base de datos como se muestra en la figura 1.2. Por último los resultados pueden ser desplegados en forma de texto o gráfica para un usuario administrador desde el servidor central.

1.4. Requerimientos

La plataforma fue desarrollada con el objetivo primordial de funcionar en Plan Ceibal, no obstante esta sección se dedica a desarrollar en qué tipo de entorno o red podría llegar también a funcionar.

Como se puede ver en la figura 1.2 debe tratarse de una red con un punto denominado nodo principal que sea capaz de acceder a todos los otros nodos de la red con diversos propósitos. En el mismo deben estar instalados los servidores de Zabbix y Ansible y en el resto de la red deben estar instalados los agentes de Zabbix. En el capítulo 4 se desarrollan los conceptos de servidor y agente de Zabbix y Ansible.

Por una descripción más detallada de los requerimientos necesarios para instalar la plataforma puede consultarse el manual de usuario disponible en apéndice

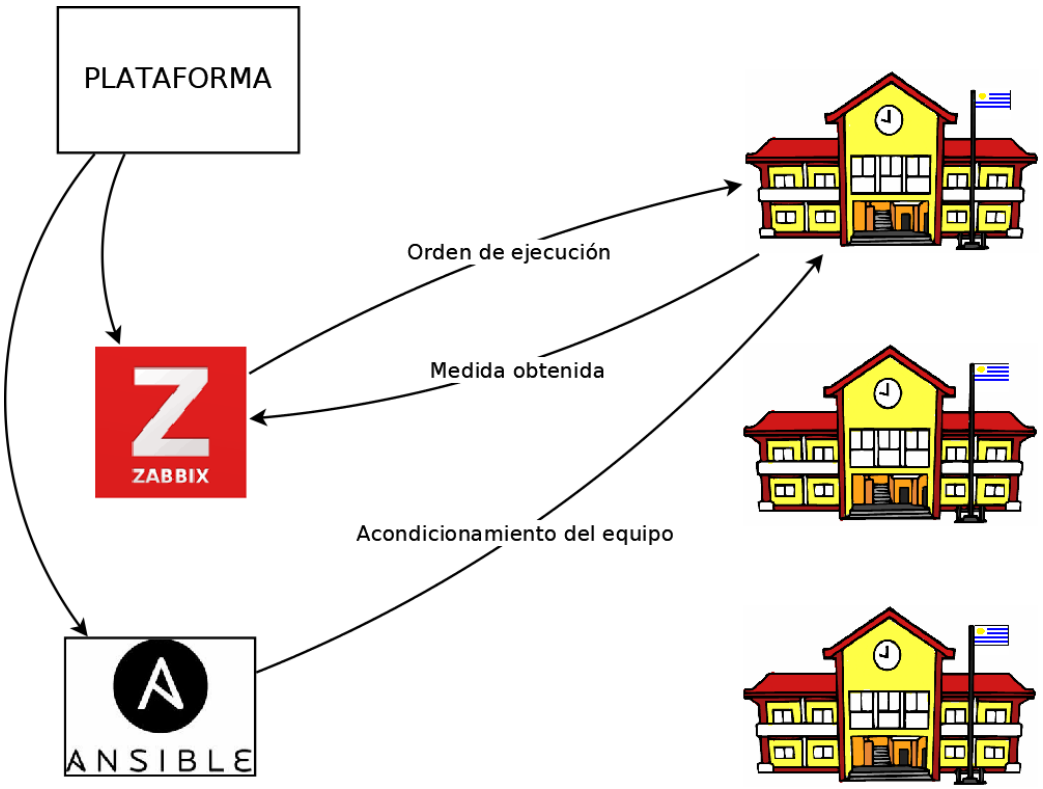


Figura 1.2: Funcionamiento de la Plataforma.

C.

1.5. Alcance

El proyecto se centra por un lado en lograr que los módulos realicen las mediciones de las métricas de interés, la generación y transmisión de datos desde cada centro educativo al servidor central y en que a su vez los módulos estén funcionando según las indicaciones del usuario.

La función principal de los módulos será el tomar mediciones para generar datos. No se incluye en el alcance que en caso de que surjan problemas con la red o con las mediciones, los mismos efectúen acciones para solucionarlos, sí se incluye el informar a Zabbix que hubo un problema.

La plataforma deberá distribuir de forma correcta tanto los módulos como sus configuraciones seleccionadas por el usuario. La misma podrá detectar en qué centros ya se encuentran instalados los mismos y en cuales no. De ya encontrarse instalados en cierto equipo, la plataforma ofrecerá la opción de modificar la configuración introducida por el usuario.

Una vez instalados los módulos en los centros educativos, la plataforma no brindará una forma de eliminarlos de los centros, puesto que Ansible (herramienta usada por Ceibal) se podría utilizar para hacer esto sin la intervención de Moniceibal. No obstante sí se brindará la opción de desactivarlos, donde con desactivarse se entiende que el o los *scripts* de medición dejen de ejecutarse o enviar medidas a Zabbix.

Por otro lado, el alcance de este proyecto se centra en lograr una plataforma escalable y que deje abierta la posibilidad de agregar nuevos módulos interactuando de forma correcta tanto con Ansible como con Zabbix.

En caso de que se desee modificar alguno de los módulos de medición instalados en los centro educativos ya sea para generar una versión mejorada de los mismos o para realizar otro tipo de modificaciones, la plataforma ofrecerá la opción de volver a instalar los módulos en los centros donde ya se encuentran instalados. Por otro lado existe información referida a la plataforma y a los módulo que sería de interés mantener en una base de datos. Dado el tiempo que se estimó que llevaría realizarla, se decidió dejarla para trabajos futuros ya que pondría en riesgo el cumplimiento de los objetivos principales de este proyecto.

Capítulo 2

Monitoreo de Redes de Datos

Una herramienta de monitoreo es un programa que permite obtener información de los nodos presentes en una red de datos. Esta información es normalmente enviada a un nodo central desde donde se administra dicha red y generalmente es guardada en una base de datos. Los datos colectados permiten evaluar el correcto funcionamiento de la red, detectar fallas y en mucho de los casos el origen de las mismas.

Algunas herramientas de monitoreo más avanzadas presentan funcionalidades como *triggers*, los cuales puede ser programados para alertar al administrador de la red ante la ocurrencia de cierto evento. Una popular vía que normalmente usan para colectar datos es el protocolo SNMP.

A continuación se describirán brevemente las principales características de algunos de los sistemas de monitoreo más populares. Esto se hará con la meta de brindar un panorama del funcionamiento de tales herramientas en la actualidad. Una descripción más profunda sobre los sistemas que se mencionan a continuación se puede encontrar en [14].

2.0.1. Zabbix

Este programa consiste en una plataforma de código abierto de monitoreo de redes creada en el 2005 por Alexie Valdishev [2]. La misma funciona en una modalidad *Agente-Servidor*. El servidor se instala en un equipo concreto y es el encargado de agendar los horarios o frecuencias de medición con la meta de darle la orden a los agentes de tomar la medida en el instante correspondiente. Una vez realizada una medida, es trabajo del agente enviar el dato obtenido al servidor central para que este lo guarde en una base de datos, la cual puede ser MySQL, SQLite, Oracle o MariaDB.

Los agentes son instalados en cada nodo de la red a monitorear y permiten a la plataforma realizar acciones en forma remota que sin su existencia no serían posibles. No obstante, en los casos en que no es posible instalar el agente de Zabbix en los equipos a monitorear, este también permite que el servidor central realice chequeos vía SNMP, TCP, ICMP o HTTP.

Este programa puede ser administrado tanto desde una interfaz gráfica denominada Zabbix Web Frontend, como a través de la línea de comandos usando la llamada Zabbix API.

Al igual que otras herramientas, para detectar fallos en la red este permite monitorear aspectos como carga del CPU de los nodos de la red o estado del servidor DNS [7] [8] entre otros. Por otro lado, no permite evaluar ciertas métricas como por ejemplo el ancho de banda o estado de servidores DHCP [10]. Como cualidad importante a resaltar de esta herramienta es la implementación de *templates*, los cuales permiten definir un conjunto de acciones que se pueden aplicar a múltiples *hosts* o grupos de *hosts*.

2.0.2. Nagios

Nagios es una herramienta de monitoreo de redes en tiempo real, también de código abierto al igual que Zabbix, creada y lanzada por Ethan Galstad en 1999 [14]. Esta herramienta al igual que Zabbix puede funcionar en base a agentes o sin ellos. Para realizar dichas acciones es posible crear grupos de *hosts* al igual que en Zabbix, de forma de realizar la misma acción con la misma configuración a todos los miembros del grupo.

Una de sus principales fortalezas es que cuenta con una librería en constante crecimiento de diversos *plugins* que permiten realizar una gran cantidad de tareas. Es importante resaltar que dentro de estas tareas se encuentra la posibilidad de monitorear lo servidores DHCP [10] funcionalidad con la que Zabbix no cuenta. Por otro lado, al igual que Zabbix este también permite comprobar el estado de los servidores DNS [11], carga del CPU, cantidad de procesos corriendo en cada equipo, entre otros.

2.0.3. Cacti

Cacti se trata de otra herramienta de monitoreo de código abierto, la cual fue creada por Ian Berry [4] en el 2001. A diferencia de Zabbix y Nagios trabaja mayoritariamente a través de mensajes SNMP y no de agentes. El no contar con los mismos hace que no pueda realizar ciertas tareas como por ejemplo que los nodos ejecuten *scripts* y envíen los resultados al servidor central. Por otro lado, sí es posible utilizar Cacti para ejecutar *scripts* externos agregados por el usuario, en el servidor central donde está instalado Cacti y de esta forma generar datos de interés. Entre sus principales fortalezas se puede destacar su robustez y su rica interfaz gráfica. Dicho entorno gráfico permite realizar gráficas de los datos obtenidos durante cierto período de tiempo y desplegarlas de varias maneras. Por otro lado, también puede establecer vínculos entre los datos obtenidos de distintos equipos para luego poder visualizar las gráficas simultáneamente y realizar comparaciones.

Capítulo 3

Métricas de interés

Mediante este proyecto se propone incorporar métricas que evalúen la calidad de los servicios de conectividad prestados a los centros educativos. A continuación se da una breve descripción de las métricas de interés, su importancia, cómo afectan las mismas el desempeño de la red y por qué su evaluación fortalece el monitoreo actual.

3.1. DHCP

El protocolo Dynamic Host Configuration Protocol (DHCP) descrito en la RFC 2131 [9], es un protocolo de red cliente-servidor el cual se utiliza para asignar de forma dinámica direcciones IP y otros parámetros de configuración a dispositivos de una red. Un servidor DHCP cuenta con una lista de direcciones IP dinámicas que va asignando a los clientes conforme estas van quedando libres, sabiendo en todo momento quién ha estado en posesión de esa IP, cuánto tiempo la ha tenido y a quién se la ha asignado después. Así los clientes de una red IP pueden conseguir sus parámetros de configuración automáticamente. Puesto que el servidor DHCP es el responsable de asignar direcciones IP en su red, un fallo en este aspecto podría ocasionar problemas tales como que a un nuevo dispositivo que intenta conectarse a la red no se le asigne una dirección IP. En tal caso dicho dispositivo no podrá comunicarse ni ser alcanzado por los demás dispositivos.

Para evaluar el funcionamiento del servidor DHCP se desea medir el tiempo que se demora en otorgarle una IP a un nuevo equipo que se conecta a la red. Si bien otras herramientas de monitoreo de redes como Nagios permiten monitorear varios parámetros de un servidor DHCP tales como su disponibilidad, direcciones de IP libres o tiempo durante el cual las ofrece, estos no miden, por lo menos de forma nativa, el tiempo de respuesta a un pedido de IP. Dicho indicador es una clara medida del desempeño del servidor DHCP y del tiempo que le toma a un nuevo equipo conectarse a la red.

3.2. DNS

El Domain Name System (DNS) o servidor de nombres de dominio es un sistema de nomenclatura jerárquico descentralizado utilizado por computadoras y diferentes dispositivos que se conectan al Internet o redes privadas [7] [8] [3]. Cuando uno de estos equipos realiza una consulta, se utiliza lo que se denomina una Uniform Resource Locator (URL), la misma sirve para identificar un recurso, como por ejemplo una página web determinada. Dicho sitio o página web pertenecen a un cierto dominio, el cual tiene asociado una dirección IP. La principal función del servidor DNS consiste en dada un *hostname*, traducirla en la dirección IP del servidor correspondiente, tal información se encuentra en una base de datos descentralizada.

Por lo que para resolver una consulta DNS, es necesario averiguar el espacio de nombres. Para esto debe realizarse una búsqueda a través de los distintos nombres de dominio, los cuales forman una estructura de árbol. Dicha consulta comienza desde la raíz y va siendo delegada a lo largo del árbol a través de distintos nodos hasta conseguir la dirección del servidor buscado. Tal tarea es normalmente realizada por un servidor DNS al que se le realizan las consultas.

Para evaluar el desempeño de tal servidor se propone medir su tiempo de respuesta, es decir el tiempo desde que se le realiza una o varias peticiones a determinas *hostname* hasta que proporciona la dirección IP de cada uno de los servidores correspondiente. Si bien Zabbix permite verificar el estado de un servidor DNS, es decir si este se encuentra activo o no, el mismo no permite medir su tiempo de respuesta.

Un mal desempeño de este factor puede afectar de forma directa la calidad de la navegación por Internet de los usuarios. El tiempo que toma realizar una consulta DNS, es tiempo extra que un usuario deberá esperar para acceder al contenido de un sitio web. En caso de que un servidor no sea capaz de proporcionar la dirección IP donde se aloja una pagina web, entonces tal sitio sera inaccesible para el usuario.

3.3. Tiempo de Descarga de un Sitio Web

Es de interés también realizar mediciones que ayuden a evaluar la calidad de la navegación por Internet para los usuarios de la red. Una forma de evaluar esto es medir el tiempo que toma acceder al contenido parcial o completo de una página web determinada. Por lo tanto se propone la creación de un módulo que mida el tiempo que toma descargar el contenido de cierto sitio web seleccionado por el administrador, desde el instante inicial en que se introduce la URL.

Esta medida servirá como estimativa de los tiempos de espera diarios de los usuarios al intentar acceder a la web. Dicho tiempo depende de un conjunto de factores tales como la latencia, o sea la suma de retardos temporales dentro de una red. Un retardo es producido por la demora en la propagación y transmisión de paquetes dentro de la red. Otros factores que también pueden influir son el ancho de banda disponible, congestión de la red, tiempo de resolución de DNS, etc.

3.4. Ancho de Banda

El ancho de banda es la máxima cantidad de bits que se puede transferir en un canal por unidad de tiempo, se puede expresar en bits por segundo o múltiplos de esta unidad. Las herramientas de medida normalmente nos permiten saber cuál es el ancho de banda disponible entre dos puntos de una red. Esto es útil ya que pueden ocurrir problemas ya sea porque un enlace no está funcionando como debería, o que el tráfico excesivo generado por cierto nodo de la red no permita el funcionamiento correcto de cierto servicio o aplicación. Por estas causas las mediciones se deben realizar de forma periódica, para poder comprobar que la calidad de la conectividad en la red en general sea la deseada y que cada usuario disponga del ancho de banda correspondiente. En general las aplicaciones de medida de ancho de banda disponibles inyectan tráfico en la red, y mientras se realizan estas medidas en ocasiones puede empeorar la calidad del servicio de forma notoria para los usuarios. Es por esto que en algunas ocasiones se opta por realizar medidas cada lapsos de tiempo muy largos o en horarios de baja demanda. Como contrapartida se desconoce el estado de la red en horarios de interés, o de más alta demanda. En este caso se optará por buscar y usar una herramienta que sea capaz de estimar el ancho de banda disponible a nivel de capa de transporte, teniendo en cuenta el tráfico total de la red. Esta herramienta debe de realizar las medidas en forma no invasiva, de forma de poder realizarlas con mayor frecuencia, esto brindará un mejor panorama de la calidad del servicio de la red o de sus puntos de falla.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 4

Herramientas a Utilizar

Para el desarrollo de esta plataforma es necesario trabajar con herramientas informáticas de código abierto elaboradas por terceros. Este capítulo da una breve descripción de cada una, su funcionamiento y los conceptos que resultan clave para el funcionamiento de la plataforma. De todos estos programas, algunos ya son utilizados por Plan Ceibal y otros habrán de ser incorporados para poner a Moniceibal en funcionamiento. A continuación, se incluye una comparación entre Ansible y CFEngine, dos herramientas de código abierto útiles para realizar cambios en equipos remotos, ambas utilizadas por Plan Ceibal. Por último en este capítulo se describen algunas herramientas a utilizar que en general son parte del sistema operativo Linux.

4.1. Zabbix

Como se mencionó antes, este programa funciona en una modalidad agente-servidor. Estos agentes se encuentran instalados en equipos de cada centro educativo, dado que esos son los lugares donde se desea recabar los datos.

De esta forma el servidor central situado en el *datacenter* de Pando puede dar órdenes de ejecución a los agentes instalados en los centros educativos.

4.1.1. *host*

El servidor central guarda una lista con todos los equipos que administra, la cual se denomina lista de *host*, en este caso dichos equipos son los que se encuentran en cada centro educativo. En esta lista cada *host* se identifica con un nombre, una IP, un número de identificación (*host* ID) y un número de interfaz. En Zabbix es posible realizar grupos de *hosts*, con la meta de por ejemplo establecer una configuración común en todos esos equipos.

4.1.2. *items*

Las tareas que puede realizar un *host* mediante Zabbix son llamadas *items*, el servidor central contiene una lista donde aloja los *items* de cada *host*. En dicha

lista cada uno se identifica por un nombre, un número de identificación, así como una frecuencia a la cual debe ejecutarse y un período de tiempo durante el cual se almacenan los datos recabados. Existe una gran variedad de tipos de *items* que Zabbix incluye por defecto. Estos pueden verificar aspectos de los equipos de la red tales como: porcentaje de carga de CPU, cantidad de memoria RAM en uso, contactos vía *ping* con los agentes para comprobar la conectividad, entre otros.

Entre las modalidades de funcionamiento de un *item*, se distinguen dos en particular. La primera es llamada Zabbix *agent*, en ella el servidor central da la orden al agente de tomar una medida y espera la respuesta durante una lapso de tiempo máximo de 30 s. En caso de que la respuesta no llegue a tiempo, Zabbix considera que existe algún problema en tal *item*, por lo tanto interrumpe el proceso y cesa de tomar dicha medida. Dado que las medidas a realizar en este proyecto pueden tener un tiempo de ejecución mayor a dicho lapso de tiempo, ha de buscarse una solución que permita evadir este límite de tiempo, la cual se profundizará en la sección 4.1.5.

El otro tipo de *item* que se manejará en este proyecto es llamado *trapper item*. En este caso el servidor central nunca da órdenes de ejecución, sino que espera que las medidas sean activadas mediante algún otro mecanismo o por parte del usuario y se remite a esperar que los datos le lleguen para almacenarlos en su base de datos.

Por otro lado, sea cual sea el tipo de *item*, los datos son almacenados junto con la fecha y hora a la cual fueron recibidos para luego poder ser visualizados.

4.1.3. Zabbix Sender

Puede que la medición a realizar se accione en el *host* mediante algún otro mecanismo que no sea Zabbix. Pero que se quiera enviar la información colectada al servidor central para que el mismo la almacene en la base de datos. Esto puede realizarse ejecutando la siguiente línea de comandos en cualquier equipo que esté conectado al servidor central:

```
zabbix_sender -z [IP del Servidor Central] -s [Nombre host] -k [Tipo de item]
-o [Dato Obtenido]
```

Para que el servidor central sea capaz de recibir este dato se debe crear un *item* del tipo *trapper* bajo la lista de *items* de dicho *host*. Este tipo de *item* es capaz de recibir un dato en cualquier instante sin necesidad de configurar ningún tipo de frecuencia de ejecución.

4.1.4. UserParameters

Si la medida que se desea realizar no se encuentra dentro de la lista que Zabbix ofrece por defecto, es posible crear nuevos tipos de *items* a partir de *scripts* los cuales son llamados UserParameters. Para crear un nuevo tipo de *item* es necesario editar el archivo de configuración del agente disponible en cada *host* agregándole

una línea con el siguiente formato:

```
UserParameter= [key del nuevo tipo de item], [Comandos o script a ejecutar]
```

Notar que hay una coma y un espacio luego de la key, esta coma es obligatoria ya que de otra forma habrá un error de sintaxis. El nuevo *item* se instancia en el servidor central mediante su nombre o *key*. Hay que destacar que cada *host* no puede tener creados dos *items* iguales, o si intenta instanciar un *item* que no se encuentre definido en el archivo de configuración de ese agente, ocurrirá un error. Por otro lado, los comandos o *scripts* a ejecutar recaban datos mediante factores externos a Zabbix y la salida que emitan es enviada al servidor central, esta puede ser alfanumérica o simplemente numérica.

Hay que destacar que dicha línea sólo debe ser agregada en caso de querer crear un nuevo *item* del tipo Zabbix *agent* y no al crear un *item* del tipo *trapper*. Esto se debe a que los *trapper item* son esencialmente todos iguales ya que no ejecutan *scripts* o comandos, sólo se limitan a recibir datos provenientes del exterior.

4.1.5. Evasión del Límite de Tiempo en la Ejecución de un *item*

Como se dijo anteriormente, Zabbix es muy útil para agendar horarios o frecuencias para tomar una medida, pero si la misma no finaliza su ejecución en un tiempo menor a 30 s, este interrumpe el proceso. La solución encontrada durante este proyecto consiste en que cuando llegue el momento de tomar la medida, Zabbix no tome la medida por sí mismo, sino que le encargue la tarea al usuario daemon mediante el comando *at*, el cual se describirá más en la sección 4.4.1. Para esto el UserParameter asociado a tal medida debe estar creado en el *host* mediante la siguiente línea:

```
UserParameter= [key del nuevo tipo de item], echo [Comandos o script a
ejecutar] | at now
```

De esta forma la ejecución del *script* o comando ya no está en manos de Zabbix y por ende este ya no puede interrumpir el proceso por más que haya pasado el límite de tiempo. Una vez finalizada la medida, el usuario daemon necesita una forma de enviar el dato a Zabbix para que este lo almacene. Dicho dato puede ser enviado usando el comando *zabbix_sender*, para lo cual se necesita que exista un *trapper item* en el servidor, el cual se encargará de recibir dicho dato. En conclusión, para la toma de una medida que excede el tiempo de ejecución de 30 s se necesita crear dos *items* en el servidor de Zabbix, uno del tipo Zabbix *agent* que de la orden de ejecución y otro del tipo *trapper item* que será el encargado de recibir el dato.

4.1.6. Zabbix API

Si bien existe una interfaz gráfica denominada Zabbix Front End mediante la cual Zabbix puede ser enteramente gestionado. La misma no permite una administración automatizada o por medio de *scripts*. En este caso, se puede usar la API REST de Zabbix, en la que el cuerpo de los mensajes *post* esta en formato JSON. Los *scripts* de la plataforma contendrán un mensaje escrito en dicho lenguaje para cada una de las acciones relacionadas con Zabbix que los mismos deseen realizar, ya sea para obtener información del servidor o realizar cambios en el mismo.

Para comenzar a usar la API de Zabbix es necesario autentificarse en la misma. Para esto debe enviarse un mensaje inicial conteniendo el nombre de usuario y la contraseña de Zabbix y el mismo devolverá una clave constituida por una larga secuencia de caracteres. Después de esta etapa, cada mensaje que se envíe debe estar acompañado de dicha clave con el fin de indicar que la autenticación fue realizada y que se está logueado en el servidor correcto.

Siempre que se envía un mensaje mediante la API se recibe una respuesta conteniendo la información deseada o indicando si la acción se pudo o no realizar.

4.2. Ansible

Ansible es un programa de código abierto escrito en Python y patrocinado por la multinacional estadounidense Red Hat [5]. Esta herramienta sirve para realizar cambios en una gran cantidad de equipos remotos de forma automática. Normalmente pueden realizarse cambios en un equipo distante a través de conexiones como *ssh*, pero en caso de tratarse de una gran cantidad de equipos, realizar dicha tarea manualmente se vuelve engorrosa si se la puede hacer automáticamente. Es en esta situación donde puede programarse Ansible de forma fácil y simple para que realice la misma tarea de forma automática en una gran cantidad de equipos.

De esta forma Ansible se instala en un servidor central y realiza los cambios deseados en cada nodo de la red mediante conexiones *ssh*, las cuales el mismo se encarga de establecer. De esta forma permite el despliegue de aplicaciones, ejecución de tareas, cambios de configuración o actualizaciones continuas en cada *host*.continuación se presentan brevemente los elementos que componen el programa.

4.2.1. Inventario

El inventario cuenta con una lista de *hosts*, en el cual se encuentran registrados todos los nodos de la red con los cuales se puede conectar. En el inventario existe la posibilidad de definir grupos con la meta de en lugar de encargar cierta tarea sobre un *host* concreto, encargarla sobre un grupo. Si bien este programa cuenta con un inventario oficial el cual se usa por defecto, también existe la posibilidad de crear otros extra. Al ejecutar la tarea, Ansible consultará el inventario proporcionado para verificar que el *host* o grupo seleccionado se encuentra en el inventario, de lo contrario la misma no se podrá llevar a cabo.

4.2.2. *playbook*

Puede que en lugar de quererse ejecutar una sola acción sobre un *host* o grupo, se trate de una lista de acciones, para esto existen los denominados *playbooks* los cuales se escriben en formato Yaml. En ellos se guarda la lista de tareas a realizar y si se quiere, la lista de equipos en los cuales se desea realizarlas. Una vez redactado el *playbook*, sólo es necesario darle a Ansible la orden de ejecutarlo para que este comience a realizar todas las tareas de forma automática y sin intervención del usuario. Cada tarea es llevada a cabo por un módulo de Ansible, los cuales se describen en la siguiente sección.

Dentro de un *playbook* pueden encontrarse algunas de las siguientes secciones:

- *hosts:* Nombre o grupo de equipos sobre el cual se realizará la lista de tareas.
- *become:* Variable booleana que de ser verdadera se ejecutan las tareas como administrador, se puede colocar al comienzo del archivo o justo antes de una tarea específica.
- *tasks:* Aquí comienza la lista de tareas que se ejecutarán sobre la lista de *hosts* ingresada. Para cada tarea deben ingresarse los siguientes campos:
  - *name:* Nombre de la tarea.
  - *Módulo:* Nombre del módulo que llevará a cabo la tarea.
  - *Parámetros:* Parámetros auxiliares que utiliza el módulos durante su ejecución.

4.2.3. Módulos de Ansible

Cada módulo de Ansible permite realizar una acción en particular en los equipos remotos. A continuación se brinda una breve descripción de los módulos a usar en este proyecto, para una descripción más profunda se puede recurrir a la documentación disponible en [5].

- **APT:** Gestiona paquetes APT. Para poder utilizar los módulos de la plataforma es necesario instalar dichos paquetes en cada equipo a ser monitoreado. Usándolo con Ansible permite la instalación de los paquetes en forma remota. Recibe como parámetro el nombre del programa o paquete que se quiere instalar.
- **copy:** Permite copiar archivos de forma remota desde el servidor central a cada nodo de la red, lo cual brinda una forma de distribuir archivos. En este caso será usado principalmente para distribuir los módulos de la plataforma o sus archivos de configuración. Dispone de los siguientes parámetros:
  - **Src:** Ruta de donde se encuentra ubicado el archivo en el servidor central.
  - **Dest:** Directorio a donde se desea copiar el archivo en cada nodo.

    - **force:** Este argumento recibe una variable booleana, en caso de ser verdadera, si ya existe un archivo con el mismo nombre en el destino, este lo remplaza a pesar de que los contenidos sean distintos. En caso contrario el archivo no se sobrescribe.

- **shell:** Este módulo permite ejecutar cualquier comando directamente sobre la terminal del equipo remoto. Será usado principalmente para ejecutar comandos en los centros educativo para poder correr los módulos creados. Recibe como parámetro la línea de comandos que desea ejecutarse.

- **Line in File:** Permite agregar, eliminar y actualizar líneas de texto en archivos alojados en los nodos. Recibe los siguientes parámetros:

    - **Path:** Ruta al archivo el cual se quiere modificar.
    - **Line:** línea de texto que se desea escribir en el archivo.
    - **Regexp:** En este parámetro puede ingresarse un fragmento de la línea que se desea sobrescribir en el archivo de destino, de no utilizar este parámetro la nueva línea será añadida al final de archivo.

- **File:** Dicho módulo permite cambiar los atributos o borrar un cierto archivo o directorio. En este proyecto se usa para cambiar los permisos de ejecución de un archivo, para que el mismo pueda ser ejecutado por la plataforma y cuenta con los siguientes parámetros:

    - **Dest:** Ruta hacia el archivo.
    - **Mode:** Tipo de permisos que se desea otorgar sobre el archivo.

4.3. Comparación entre Ansible y CFEngine

Ansible es el sistema que se está comenzando a utilizar actualmente en Plan Ceibal para realizar cambios a gran escala en equipos remotos. Es por esto y su simplicidad de manejo que será usado en este proyecto. A la vez el mercado también ofrece otras alternativas libres y gratuitas capaces de realizar las mismas tareas como lo es el caso de CFEngine, herramienta que Plan Ceibal está pasando a dejar en desuso. Esta sección pretende ofrecer una comparación entre los dos programas que pueda ser consultada en el futuro para conocer las fortalezas y diferencias de cada programa. Para más detalles se recomienda visitar la documentación de cada uno [5] [24].

Dado que Ansible ya ha sido definido en secciones anteriores, se comenzará por dar una definición de CFEngine. Se trata de una plataforma que permite la automatización, configuración e implementación de tareas en un gran número de equipos poniendo como prioridad la seguridad tanto de los equipos como de los cambios a realizar. Fue inicialmente desarrollado por Mark Burgess y cuenta con dos versiones. La primer versión se denomina, Community Edition, está disponible gratuitamente y es distribuida bajo una licencia de *software* libre. La segunda

versión se denomina edición empresarial o Enterprise Edition y también es gratuita si se utiliza para administrar hasta 25 *hosts*.

4.3.1. Descripción de cada programa

- *Ansible:* A continuación se dará una breve descripción de Ansible que menciona nuevamente algunas de las características del mismo dado que serán útiles como punto de partida para la comparación.

  Como se muestra en la figura 4.1, al ejecutarse el programa el mismo ha de leer dos archivos, el inventario, que contiene la lista de *hosts* con los cuales le es posible conectarse y por otro lado, el *playbook* que contiene la lista de tareas a ejecutar. Una vez hecho esto, el programa se encarga de contactar a el o los *hosts* vía *ssh* y así realiza todas las acciones indicadas. Ante cada tarea el programa puede responder que la misma se llevó a cabo con éxito, ya se encontraba realizada o que no se pudo realizar.

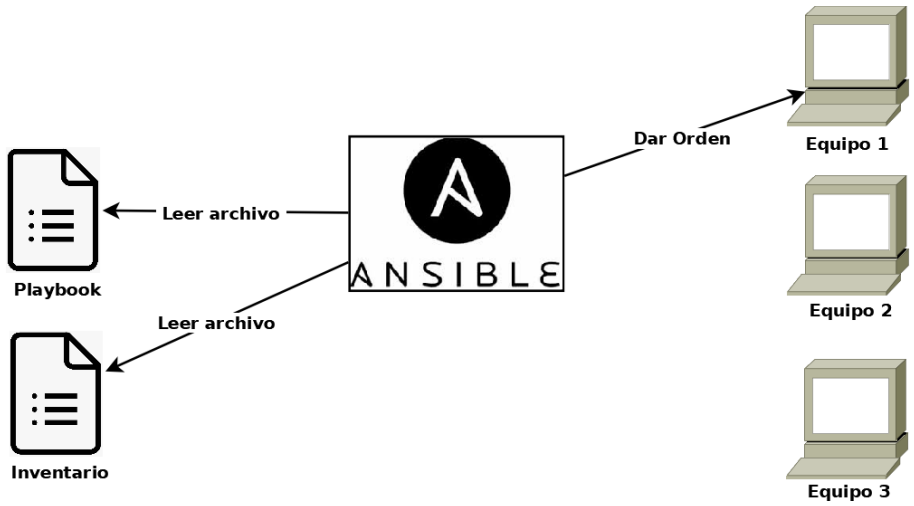


Figura 4.1: Funcionamiento de Ansible.

- *CFEngine:* A diferencia de Ansible, este programa es un sistema distribuido y cuenta con un agente corriendo en cada uno de los *hosts* que administra. El usuario utiliza el servidor central para definir las políticas, en ellas consta el estado al que se quiere llevar a cada uno de los equipos de la red. Las tareas necesarias para llevar cada equipo a dicho estado son llamadas Promesas y pueden tratarse de la instalación o actualización de un programa, modificación de un archivo, entre muchas otras cosas. El archivo de políticas es redactado en un lenguaje propio de CFEngine llamado Lenguaje específico de dominio o en inglés Domain Specific Language (DSL) y su contenido puede ser aplicado a un grupo de *hosts* o a un *host* concreto.

  Una vez definidas las políticas, el agente instalado en cada equipo consulta al servidor central cada 5 minutos como se muestra en la figura 4.2 para

informarse de las mismas y así intentar hacerlas cumplir. En caso de no poder conectarse con el servidor, el agente siempre sigue corriendo e intentando hacer cumplir las últimas políticas adquiridas. Aquí pueden darse tres casos, o bien que las promesas ya se encuentren cumplidas, que no se encuentren cumplidas pero el agente logre hacerlas cumplir o también puede que no se encuentren cumplidas y el mismo tampoco logre hacer que se cumplan. En este último caso el mismo continuará intentando una y otra vez hasta lograr su meta.

La información acerca del cumplimiento de las promesas queda guardada en un archivo *log* dentro de cada agente para la Community Edition y también en una base de datos si se utiliza la Enterprise Edition. Para el caso de usuarios Enterprise la base de datos no sólo contendrá la información relevante al cumplimiento de las promesas sino también otros datos como por ejemplo programas instalados, carga del CPU y memorias, espacio en los discos duros, actividad de la red, etc. Por último es el trabajo del servidor central el contactar a los agentes para obtener la información de sus archivos *log* y hacerla disponible al usuario administrador como se puede apreciar en la figura 4.2.

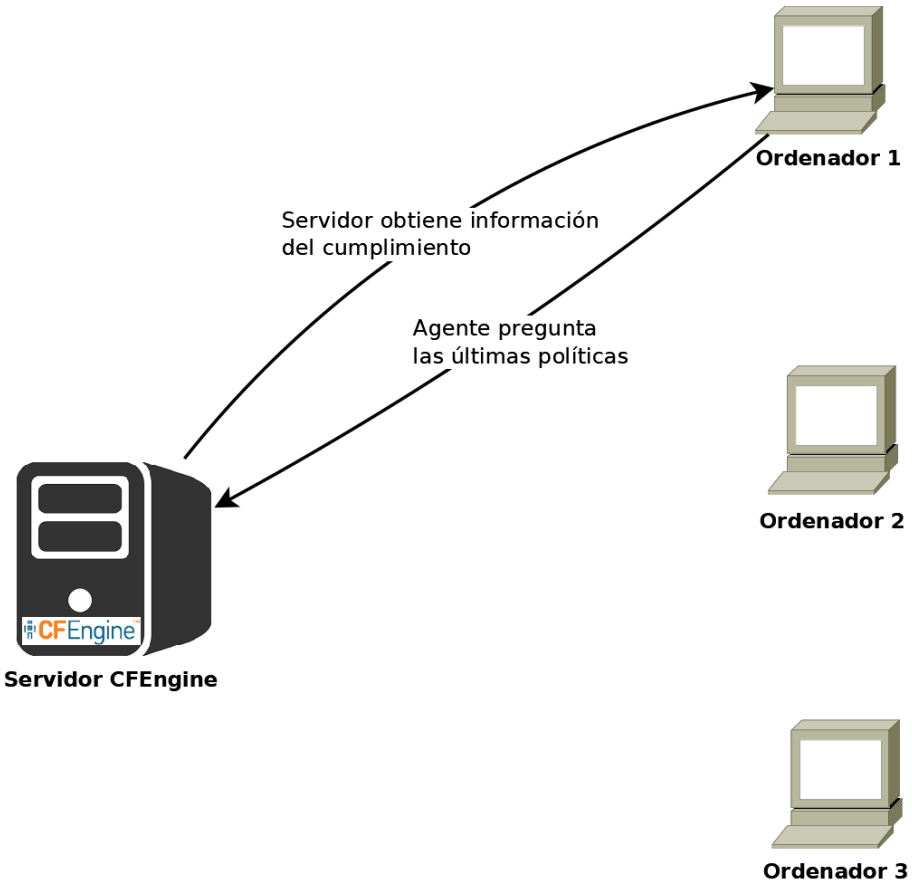


Figura 4.2: Funcionamiento de CFEngine.

4.3.2. Instalación y configuración

- *Ansible:* Antes de instalarlo sólo es necesario agregar su repositorio a la lista de repositorios del servidor y luego proseguir con la instalación desde la línea de comandos. El único detalle importante a tener en cuenta es que el mismo requiere una versión de Python 2.7 o posterior. Este sistema se encuentra enteramente desarrollado en dicho lenguaje y hace uso del paquete Paramiko [19] el cual es capaz de realizar conexiones *ssh* desde Python y mantenerlas abiertas durante toda la ejecución del *script* para las diferentes tareas. Finalizada la instalación es necesario modificar su archivo inventario para agregar cada uno de los *hosts* con los que se desea trabajar a la lista. El mismo utiliza conexiones *ssh* como medio para conectarse a cada uno de los *hosts* que desea configurar, si bien este permite introducir las contraseñas para autenticarse en cada uno de los equipos. Por razones de seguridad, tiempos de ejecución y escalabilidad, se recomienda realizar un intercambio de claves públicas antes de comenzar a trabajar con el fin de agilizar dichos procesos.

- *CFEngine:* Luego de asegurarse que los equipos están listos y cumplen con los requisitos para la instalación, el siguiente paso es descargar los paquetes de los repositorios e instalarlos. Existen dos tipos de paquetes, el que se utiliza para instalar los agentes en cada *host* y el que se utiliza para instalar el servidor. Una vez instalado el programa en todos los equipos es necesario configurar cada agente para que tengan conocimiento de la IP de su servidor y sepa de dónde obtener las políticas configuradas por el usuario. Después es necesario establecer un intercambio de clave pública entre servidor y agentes con el fin de establecer confianza y que cada uno pueda obtener la información necesario del otro. Por último CFEngine ofrece la posibilidad de recibir notificaciones por correo electrónico del estado del servidor y los agentes. Si se desea utilizar esta prestación entonces debe de realizarse la configuración correspondiente para que el programa envíe dichos correos a las casillas indicadas.

En cuanto a la instalación en un equipo concreto no hay grandes diferencias, ambos programas pueden ser instalados desde repositorios y ambos necesitan realizar intercambios de claves. Por otro lado, hay que considerar que CFEngine debe ser instalado y configurado en todos los equipos con los que se desea trabajar lo cual por supuesto insume más tiempo.

4.3.3. Sistemas operativos

- *Ansible:*

    - *Servidor:* Si bien de forma nativa es soportado por los sistemas operativos RHEL, CentOS, Fedora, Ubuntu, Debian, Gentoo, FreeBSD, macOS, Solaris, Arch Linux, Slackware Linux o Clear Linux, existen dos formas en las que puede correr también en Windows. La primera

de acuerdo a la documentación, es usando el subsistema Linux actualmente disponible en las últimas versiones de Microsoft Windows y la otra es usando el programa Cygwin [23]. Este programa contiene una amplia variedad de herramientas libres de origen GNU que proveen una funcionalidad similar a la de una distribución de Linux en Windows.

  - *hosts:* De acuerdo a la documentación es capaz de administrar equipos RHEL, CentOS, Fedora, Ubuntu, Debian, Gentoo, FreeBSD, macOS, Solaris, Arch Linux, Slackware Linux, Clear Linux y Windows. Este contiene dos librerías de módulos, la primera, más amplia y general que ha sido utilizada a lo largo de este proyecto y que funciona en la mayor parte de los sistemas operativas mencionados y la segunda que es exclusiva de Windows. Si bien la segunda librería es más pequeña, ésta contiene una amplia variedad de módulos capaces de satisfacer las necesidades básicas del usuario.

- *CFEngine:*

  - *Servidor:* Su versión Entreprise puede ser instalada en sistemas operativos RHEL, CentOS, Debian, Ubuntu o Windows Server 2003 en adelante. La Community Edition puede ser instalada en los mismos sistemas operativos con la diferencia de que para instalarla en Windows se necesita el programa Cygwin.
  - *hosts:* La versión Enterprise soporta los sistemas AIX, RHEL, CentOS, Debian, HP-UX, SLES, Solaris 11, Solaris 10, UltraSparc, Ubuntu o Windows XP SP2 o posterior. Nuevamente la Community Edition también soporta Windows XP SP2 o posterior a través del programa Cygwin, pero sin disponer de las características específicas diseñadas para Windows.

Como se puede ver ambos programas tienen cubierta una amplia gama de sistemas operativos Unix y Linux. En cuanto a Microsoft Windows, Ansible provee el soporte completo de forma gratuita, mientras que en CFEngine la versión gratuita sólo dispone de las características generales del programa, las cuales no son totalmente funcionales en dicho sistema operativo.

4.3.4. *hardware*

- *Ansible:* Los requerimientos son tan sólo un procesador que soporte sistemas operativos de 64 bits, un mínimo de 512 MB de memoria RAM, un disco duro de aproximadamente 10 GB y por supuesto una conexión a la red.

- *CFEngine:*

  - *Servidor:* En este caso también se necesita un equipo que soporte un sistema operativo de 64 bits. Un mínimo de 3 GB de RAM para propósitos generales o un mínimo de 8 MB por agente para usos de prueba

dentro de un laboratorio. Por otro lado, aunque en teoría no debería afectar el funcionamiento del sistema, se recomienda que el mismo se instale en una única partición. En cuanto a espacio en disco duro se debe prever un mínimo de 100 MB por agente y en cuanto a la conexión a la red, se prevé 30 Mbps para una red de 5000 agentes.

- *hosts:* Se considera un mínimo de 100 MB de espacio libre en disco duro y alrededor de 500 para que el sistema pueda correr ligeramente sin problemas. Para sistemas Windows los requerimientos de espacio en disco duro son algo mayores y se aconseja un mínimo de 1 GB y alrededor de 5 GB para que el sistema corra sin dificultades. Al igual que para el servidor, aquí también se aconseja que el sistema esté instalado en una única partición. Tratándose de memoria RAM el requisito mínimo es de 30 MB, sin embargo debido a picos causados en el uso del microprocesador por comandos corridos directamente desde el archivo de políticas, se recomiendan por lo menos 256 MB.

Como se puede observar, los requerimientos para Ansible en el servidor central son mucho menores a los de CFEngine. Por otro lado, también se debe considerar que este último debe instalarse en cada nodo de la red que se desea monitorizar.

4.3.5. Agentes

La diferencia más marcada entre un sistema y el otro es el uso de agentes. El hecho de que Ansible trabaje enteramente en base a *ssh* y no necesite disponer de agentes, suma en gran parte a la simplicidad del programa.

El uso de agentes de CFEngine lleva a aumentar los tiempos de instalación y configuración. CFEngine aprovecha la existencia de los mismos y el hecho de que pueden actuar por cuenta propia para ofrecer ciertas prestaciones que de otra manera no serían posibles. Pero con la arquitectura basada en agentes, los equipos deben tener una comunicación permanente con el servidor central. Este tipo de arquitectura puede aumentar la carga de la red.

Con la arquitectura sin agentes los equipos no necesitan instalar ni ejecutar en segundo plano ningún proceso que se comunique con el servidor central.

4.3.6. Lenguajes

Como ya se dijo antes, para redactar la lista de acciones que se desea realizar en cada *host*, Ansible utiliza el lenguaje Yaml y CFEngine el lenguaje Domain Specific Language (DSL). En un principio el lenguaje específico de dominio puede aparentar una sintaxis más compleja, lo cual para el usuario puede reflejarse en una curva de aprendizaje más empinada al comienzo. Una vez alcanzado un nivel intermedio de dominio de ambos lenguajes, se puede decir que ambos ofrecen relativamente las mismas posibilidades, más allá de detalles específicos. Como ventaja se puede

resaltar la simplicidad de Ansible y cómo este permite realizar una misma tarea dedicando menos tiempo a la parte de implementación.

Considerando CFEngine, este brinda la posibilidad de utilizar sus agentes para agendar la ejecución de sus tareas, o sea que al implementar cierta política, también se puede indicar en qué día, hora o con qué frecuencia se desea ejecutar la misma.

4.3.7. Interfaz gráfica

- *Ansible:* Este cuenta con dos interfaces gráficas, AWX y Ansible Tower. La primera se trata de una alternativa gratuita, de código abierto y desarrollada por la comunidad, la cual actualmente es utilizada por Plan Ceibal, y la segunda se trata de una edición que no es gratuita. Mientras que AWX es un proyecto de rápido cambio y desarrollo, dirigido por la comunidad, Ansible Tower se limita a tomar los aspectos más prometedores de AWX, mejorarlos y brindar una capacidad de soporte a largo plazo muy útil a nivel empresarial. Esto significa que ambas interfaces gráficas no son independientes, sino que AWX es la fuerza innovadora detrás de Ansible Tower. Por lo tanto no hace falta aclarar que todas las prestaciones de Ansible Tower están disponibles en AWX, no obstante, una diferencia importante es que Ansible Tower ofrece soporte pago mientras que para obtener soporte de AWX es necesario recurrir a la comunidad. continuación se mencionan algunas de las características más importantes de ambos.

  Al abrir una interfaz gráfica, en primer lugar se puede ver un tablero con el estado de todos los *hosts* del inventario y últimos trabajos realizados. Esta lista por supuesto puede ser configurada para sólo visualizar cierta información y no toda.

  Otra facultad de la interfaz gráfica es la implementación de flujos de trabajo. Esta permite unir distintos trabajos combinando distintos *playbooks* o inventarios residentes en diferentes servidores de Ansible.

  Desde el punto de vista de seguridad la interfaz gráfica también proporciona toda la información acerca de qué tareas se han realizado, a qué hora y por cuál usuario.

  Una importante característica es la integración de distintos servidores, corriendo desde distintos nodos de la red con el fin de unificar distintas tareas o tener acceso a recursos tales como *playbooks* o inventarios disponibles en distintos equipos.

- *CFEngine:* En este caso la interfaz gráfica sólo está disponible en la edición Enterprise. La misma permite crear tableros en donde el usuario puede seleccionar qué información relevante desplegar acerca del estado de los equipos y el cumplimiento de sus políticas. Esta información desplegada puede utilizarse para crear alertas que avisen cuando determinada política deja de cumplirse en determinado *host*. A la vez las alertas pueden dividirse en distintos niveles según su gravedad.

Mientras un mal funcionamiento es reparado, su alerta correspondiente puede ser pausada con el fin de no generar notificaciones innecesarias.

Ésta también ofrece la presencia de *widgets* que muestran la historia de eventos relevantes seleccionados por el usuario, tales como alertas reparadas o *hosts* que se incorporan o abandonan el sistema a lo largo del tiempo.

A la vez, la interfaz contiene otra sección de reportes donde el usuario puede seleccionar de qué atributos fuera de lo común quiere ser informado. De esta forma el usuario administrador puede elegir qué información desea compartir con otros usuarios que tienen diferentes roles.

Hay que destacar el hecho de Ansible provee una interfaz gráfica gratuita que realmente brinda nuevas prestaciones al usuario que sin ella no son posibles, además de desplegar toda la información necesaria acerca del estado de la red. La interfaz de CFEngine no es gratuita y se concentra más que nada en informar acerca del estado de la red y sus alarmas.

4.4. Herramientas de Linux

En esta sección se describen algunas herramientas utilizadas por la plataforma que en general forman parte del sistema operativo Linux. Estos programas resultan indispensables para el funcionamiento de la plataforma debido a las funciones que son capaces de cumplir. Entre ellas agendar tareas para que se realicen de forma automática, acceder al contenido de sitios web o examinar la configuración de las interfaces de red de los equipos.

4.4.1. *at*

Este es un programa que normalmente corre en los sistemas Linux o Unix y permite agendar la ejecución de tareas en un horario o fecha a especificar por el usuario. Cuando llega el momento la tarea es automáticamente ejecutada por un usuario llamado daemon, quien hereda los permisos de quien lo haya invocado. Por ejemplo si el usuario que lo ha invocado le encomienda ciertas tareas las cuales él mismo tiene permiso de realizar, entonces daemon tampoco tendrá problemas al llevarlas a cabo, de lo contrario el sistema no le permitirá realizarlas. En este proyecto es el usuario Zabbix quien le encomiende tareas a daemon, por esto es que es importante asegurarse que Zabbix cuente con todos los permisos necesarios.

El formato de este comando es el siguiente:

```
[comandos a ejecutar] | at now
```

Hay que destacar que como momento de ejecución puede introducirse *at now*, en ese caso el comando es ejecutado automáticamente en ese momento por el usuario daemon.

4.4.2. cron

Al igual que el programa *at*, agenda la ejecución de un comando o *script* con la diferencia de que éste también permite hacerlo de forma periódica. Un comando puede ser ejecutado cada cierto lapso de tiempo, a ciertos horarios del día, días de la semana, mes o año. Para agregar una nueva tarea es necesario escribir una nueva línea al archivo *crontab* el cual se encuentra en la carpeta */etc/cron.d/* del sistema operativo Linux o Unix. La línea a agregar debe ser la siguiente:

```
m h dom mon dow [Comando a ejecutar]
```

Los asteriscos se usan para escribir el momento o lapso de ejecución, el primer asterisco representa los minutos, el segundo las horas, el tercero los días del mes, el cuarto el mes y quinto los días de la semana. Por información más completa acerca de la sintaxis de este comando se recomienda el sitio web de la referencia [12].

4.4.3. *wget*

El programa *wget* es una herramienta de *software* libre que permite la descarga de contenidos desde servidores web en una forma simple [13]. Actualmente soporta descargas mediante los protocolos HTTP, HTTPS y FTP.

Los contenidos de las páginas web descargadas pueden ser guardados en una estructura de directorios si se utiliza la opción de recursión, la cual recrea la estructura del sitio web original. Dicha descarga permite hacer una copia exacta de cualquier sistema de archivos parcial o completa de un sitio web vía HTTP así como también de un repositorio. Dicho comando tiene la siguiente estructura:

```
wget [hostname] [opciones]
```

Su amplia gama de opciones le permiten por ejemplo seleccionar qué tanto de la página web se desea descargar o limitar la velocidad de bajada.

4.4.4. *dns utils*

Consiste en un paquete que contiene algunos programas clientes relacionados con DNS como por ejemplo *dig*, *nslookup* y *nsupdate*.

En este proyecto se usa especialmente el comando *dig*, el cual permite hacer peticiones DNS de varias maneras y así obtener la dirección del servidor solicitado. La salida de este comando no sólo incluye la dirección del servidor deseado sino también otra información de interés, así como lo es el *query time*, tiempo desde que se realizó la consulta hasta que se obtuvo la respuesta.

4.4.5. *net tools*

Es un paquete que contiene algunas herramientas básicas para trabajar con redes, en este proyecto resulta de particular interés la herramienta *ifconfig*. La

misma constituye una vía rápida para obtener información de las interfaces de red del equipo, así como su dirección IP o dirección MAC.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 5

Módulos

En este capítulo se da una breve descripción de los módulos que se pretende hacer correr en los centros educativos y los *scripts* encargados de realizar las mediciones, dejando para el capítulo siguiente la explicación de como interactúan los mismos con la plataforma y el usuario. Todos los *scripts* descritos en este capítulo necesitan comunicarse con Zabbix, ya sea para enviar datos colectados o informar de errores ocurridos, y realizan esta tarea utilizando el comando *zabbix_sender* discutido en la sección 4.1.3. En el capítulo 6 se discute cómo se crean los *items* necesarios para recibir la información enviada por estos códigos y cómo se realizan las modificaciones en los equipos remotos para tomar las medidas.

Para los módulos de Tiempo DHCP, Tiempo DNS, y Tiempo de descarga de sitios web, se mencionan los *scripts* que son activados por Zabbix y que deben correr en los equipos de los centros educativos en los cuales se desea tomar las medidas. Por el contrario para el módulo de medición de ancho de banda, los programas fueron mayoritariamente implementados para correr en el servidor central y son activados por el sistema *cron*, lo cual se explica más adelante en este capítulo. Cabe aclarar que para todos los módulos es Zabbix el encargado almacenar los datos colectados, por más que este sea el que da las órdenes de ejecución o no.

5.1. Tiempo DHCP

En este módulo se pretende evaluar el desempeño del servidor DHCP midiendo su tiempo de respuesta.

Se escribió un *script* que genera un alias de la interfaz mediante la cual cada equipo se conecta a la red. Dicho alias es luego usado para pedir una IP al servidor de DHCP y de esta forma medir el tiempo que este demora en responder.

Un alias de interfaz de red es una forma de asignar más de una IP a una misma interfaz física y así lograr múltiples conexiones a la red. Una característica de los alias es que tienen la misma dirección de capa de enlace [3] (MAC address) que tiene la interfaz de red original. Por lo tanto si el servidor DHCP ya ha asignado una IP a la interfaz original, el mismo detecta que está recibiendo múltiples pedidos desde la misma dirección de capa de enlace y se niega a otorgar una nueva dirección

de capa de red. Gracias a que las interfaces de red en los servidores de los centros educativos están configuradas de forma estática, su IP no fue asignada por el servidor DHCP, entonces tal problema no es un obstáculo. De otra forma no sería posible pedir una nueva IP al servidor DHCP mediante un alias, ya que éste sólo asigna una IP a pedidos provenientes de la misma dirección MAC.

En primer lugar se utiliza el comando *ifconfig*, que es parte del paquete *net-tools* con la meta de obtener el nombre de la interfaz de red del equipo. Dicho nombre es luego utilizado para crear un alias de la misma y pedir una IP mediante el comando *dhclient* disponible en Linux y luego liberarla.

Con el fin de medir el tiempo que toma obtener una dirección IP se utiliza el comando *time* de Linux, cuya salida después de ser procesada es enviada al servidor de Zabbix mediante el comando *zabbix_sender*. Podría darse el caso de que el servidor DHCP no responda, por lo cual se espera 20 s después de hacer el pedido y si para ese momento no hay respuesta por parte del servidor, se interrumpe el proceso con el comando *timeout* de Linux. En este caso envía a Zabbix un mensaje de error proveniente del módulo DHCP el cual informa que el pedido de IP no ha sido respondido.

El algoritmo 1 muestra resumidamente el funcionamiento de este módulo.

Algoritmo 1: Tiempo DHCP.

```
Obtener Nombre de la Interfaz de Red;
Crear Alias de la Interfaz de Red;
Pedir IP para el Alias Creado al Servidor DHCP;
si Tiempo_de_respuesta_del_servidor ≥ 20 Segundos entonces
    Interrumpir Pedido;
    Enviar Mensajes de Error a Servidor de Zabbix: Servidor DHCP No
     Responde;
en otro caso
    Enviar Tiempo_de_respuesta_del_servidor al Servidor de Zabbix;
    Liberar IP Obtenida;
fin
```

5.2. Tiempo DNS

La función de este módulo es brindar una estimativa de la calidad del servicio DNS en los centros educativos, lo cual se realiza midiendo el tiempo que toma resolver una o varias consultas DNS. Para esto se realizó un *script* en el lenguaje Bash el cual lee de un archivo de configuración una cantidad arbitraria de *hostnames* definidos por el usuario. En primer lugar el *script* ejecuta el comando *dig* disponible en el paquete *dnsutils* para cada una de los *hostnames* ingresados. Este comando realiza una consulta DNS con el fin de obtener la dirección IP del o los servidores donde se aloja tal sitio. La salida de este comando consiste de una cadena de caracteres donde entre otras cosas se encuentra el tiempo en milisegundos que le tomó realizar la consulta DNS. De esta manera el *script* realiza

una búsqueda dentro la cadena para extraer dicho tiempo. Una vez realizado este procedimiento para cada uno de los sitios web, se calcula el promedio de los tiempos, lo cual se envía como dato al servidor de Zabbix utilizando el comando *zabbix_sender* para que este lo almacene en su base de datos. Por último el *script* se encarga de borrar el cache DNS del equipo, ya que de no ser así, al volver a realizar la medida en un corto período de tiempo, el resultado sería cero ya que la dirección de dicho servidor web se encontraría almacenada en el cache del equipo y el mismo no tendría la necesidad de realizar una nueva consulta. Por otro lado, también existe la posibilidad de que el resultado de la búsqueda DNS quede almacenada en un *router* o servidor DNS. En este caso la plataforma no tiene acceso a borrar el contenido del mismo, lo cual puede reducir los resultados de las medidas al menos por un corto período de tiempo. Como se ve más adelante en la sección de validación, este aspecto no tiene un gran impacto sobre los resultados.

Existe la posibilidad de que en la búsqueda DNS no se pueda encontrar el servidor donde se aloja uno o más de los sitios web ingresados. Para saber si esto ha ocurrido, el *script* se encarga de revisar la parte de la salida del comando *dig* donde se encuentran las direcciones de los servidores encontrados. Si esta sección se encuentra vacía para uno o más *hostnames*, en lugar de devolver el promedio de los tiempos como salida, se envía un mensaje a Zabbix avisando que ocurrió un error en el módulo DNS seguido por el número del lugar donde se encontraba tal sitio web en la lista ingresada por el usuario.

El algoritmo 2 muestra una versión muy simplificada del *script* usado en este módulo.

Algoritmo 2: Tiempo DNS.

```
Lista_de_hostnames = [lista de ejemplo];
Número_de_hostname = 0;
Tiempo_Total = 0;
para hostname in Lista_de_hostnames hacer
    Número_de_hostname = Número_de_hostname + 1;
    Realiza Consulta DNS;
    si Servidor Encontrado entonces
        Tiempo = Tiempo que tomó realizar la consulta;
        Tiempo_Total = Tiempo_Total + Tiempo;
    en otro caso
        Enviar Mensajes de Error al Servidor de Zabbix: Servidor del
          hostname Número [Número_de_hostname] No Encontrado;
        Tiempo_Total = 0;
        Break For;
    fin
fin
Tiempo_Promedio = Tiempo_Total / Número_de_hostname;
Enviar Tiempo_Promedio al Servidor de Zabbix;
Borrar Cache DNS
```

5.3. Tiempo de Descarga de Sitios Web

El *script* de Bash implementado para este módulo permite descargar el contenido de un sitio web seleccionado por el usuario a través del comando *wget* disponible en Linux con el fin de medir el tiempo de descarga. Inicialmente dicho *script* lee de un archivo de configuración el *hostname* que se desea utilizar y con qué opción del comando *wget*.

Dicho comando tiene una variedad de opciones que permiten seleccionar qué cantidad del sitio web se quiere descargar, puede que el sitio web completo o sólo sus encabezados, este módulo permite seleccionar cualquiera de ellas. Para conocer todas las opciones se recomienda visitar el sitio web [13] de las referencias.

En primer lugar se realiza una búsqueda DNS para asegurarse de que el *hostname* se encuentra alojada en algún servidor de la web y que realmente exista, de no ser así, no se realiza ninguna descarga y se envía un mensaje al servidor de Zabbix explicando el error ocurrido.

Se procede a realizar la descarga utilizando las opciones que haya introducido el usuario y se almacena el tiempo que esto tomó en una variable. Puede que ocurran distintos errores durante la descarga, como que el sitio web deje de estar disponible o se pierda la conexión, entre otras cosas. Por esto es que si la descarga no se completa en menos de 5 minutos el proceso es interrumpido usando el comando *timeout* de Linux y se envía un mensaje de error al servidor de Zabbix informando de lo ocurrido.

Es importante destacar que dicho comando realiza una descarga de archivos los cuales han de ser borrados para no ocupar espacio en el equipo, esto es posible usando las opciones *-nd* y *–delete-after* del comando *wget*. La opción *-nd* se asegura de que el comando no genere directorios para almacenar los archivos descargados y la otra opción se encarga de borrar los archivos. El *script* utiliza ambas, ya que la opción *–delete-after* se encarga de borrar los archivos generados durante la descarga pero no los directorios que puedan haber sido generados durante la misma.

Por otro lado, en el sistema operativo Linux no permite guardar archivos en un directorio a no ser que el usuario tenga los permisos pertinentes sobre el mismo. Por esto es necesario que el *script* redirija la descarga hacia cierto directorio sobre el cual el usuario Zabbix, quien es el encargado de ejecutar el *script* tenga permisos. Esto es posible usando la opción *–directory-prefix=* del comando *wget*.

El algoritmo 3 da una idea del funcionamiento de este módulo.

Algoritmo 3: Tiempo de Descarga de Sitio Web.

```
Realizar consulta DNS;
si Servidor Encontrado entonces
    Realizar Descarga almacenando los archivos en la carpeta
     seleccionada;
    si Tiempo_de_descarga ≥ 5 Minutos entonces
        Interrumpir descarga;
        Enviar Mensajes de Error al Servidor de Zabbix: Error en el
         módulo Wget;
    en otro caso
        Enviar Tiempo_de_descarga al servidor de Zabbix;
    fin
    Borrar Archivos Descargados;
en otro caso
    Enviar Mensajes de Error al Servidor de Zabbix: Servidor de el
     hostname No Encontrado;
fin
```

5.4. Ancho de Banda

5.4.1. Herramienta de Medida

El ancho de banda describe la máxima cantidad de información que puede ser enviada a través de un canal de transmisión. Para evaluar la calidad del servicio de una red, la medición del ancho de banda disponible es una métrica esencial, ya que son muchas las aplicaciones que pueden verse afectadas por problemas de congestión o embotellamiento. A su vez estas medidas permiten saber en qué puntos de la red está siendo necesario controlar el tráfico utilizado.

En la actualidad existen diversas herramientas capaces de estimar el ancho de banda disponible de la red, algunas realizan la estimación en base al tráfico que actualmente circula por ella y otras generan el suyo propio. En este proyecto se tienen en consideración únicamente herramientas generadoras de tráfico. Esto se debe a que si bien estas herramientas pueden resultar invasivas en relación a las otras, las mismas no dependen del tráfico que exista actualmente en la red con el fin de poder realizar su estimación.

Uno de los factores a tener en cuenta a la hora de elegir la herramienta, es la cantidad de paquetes que la misma necesita inyectar en la red, si ésta necesita inyectar demasiados entonces consumirá una parte importante del ancho de banda disponible y puede afectar la calidad de la conectividad para los usuarios.

Otros dos criterios importantes son la exactitud de la estimación y el tiempo que le tome al programa realizarla.

Entre las herramientas generadoras de tráfico existen algunas que necesitan estar instaladas en ambos extremos del enlace cuyo ancho de banda se desea medir y otras que no. Las herramientas que sólo necesitan instalarse en uno de los extre-

mos de la ruta son útiles en caso de que por ejemplo no exista la posibilidad de instalar programas en algunos de los nodos. Durante este proyecto se contempló la posibilidad de usar algunas de ellas, como por ejemplo Pchar [20]. Que además tiene otras ventajas, como que por ejemplo no necesita reservar ningún puerto del equipo. Pero fue descartada debido a la larga cantidad de tiempo que le toma realizar una medida.

La mayoría de las herramientas generadoras de tráfico se basan o bien en el modelo Probe Gap Model (PGM) [17] o en el modelo Probe Rate Model (PRM) [17].

PGM se propuso como un método de estimación de ancho de banda disponible ligero y rápido. Las herramientas de medición como Delphi y Spruce se basan en PGM. En comparación con los métodos de estimación que requieren múltiples iteraciones con diferentes velocidades de sondeo, PGM utiliza una sola velocidad de sondeo e infiere el ancho de banda disponible de una relación directa entre las velocidades de entrada y salida de los pares de paquetes de medición. Una suposición importante detrás del modelo PGM es que la ruta medida tiene un enlace de cuello de botella único que determina el ancho de banda disponible de la ruta de extremo a extremo. Aunque PGM es precisa en el caso de una sola cola, no puede estimar el ancho de banda disponible de las rutas de múltiples saltos, incluso si hay un sólo cuello de botella en la ruta. PGM es preciso cuando el tráfico cruzado sigue el mismo camino que el tráfico de medición. En el caso general, sin embargo, PGM puede subestimar significativamente el ancho de banda disponible de una ruta de extremo a extremo.

Por otro lado, el modelo PRM se basa en el principio de congestión autoinducida. Este plantea que si se envía un tren de paquetes espaciados con una frecuencia menor al ancho de banda disponible entonces los paquetes deberían llegar al mismo ritmo al receptor sin exhibir anomalías. Pero si se envía un tren de paquetes espaciados con una frecuencia superior al ancho de banda disponible, los mismos deberían alinearse en una fila y se experimentaría cierto retraso en el receptor. Por lo tanto, la medida se realiza a través de la investigación del punto de inflexión donde la tasa de paquetes de llegada al receptor es diferente al de la enviada. El modelo PRM ha demostrado ser preciso y se utiliza en muchas herramientas de estimación, como Assolo, TOPP, Pathload, pathChirp, FEAT y BART.

5.4.2. Funcionamiento Assolo

La herramienta que se utiliza en este proyecto para realizar las medidas de ancho de banda se denomina Assolo [16]. Esta ha sido seleccionada debido a diversos factores tales como: su facilidad de uso, la exactitud de sus medidas, la baja cantidad de tráfico que inyecta en la red para realizarlas y el hecho de que permite seleccionar la cantidad tiempo que dura cada una. Como ya se dijo antes, algunas herramientas producen medidas exactas pero pueden llegar a tomar una gran cantidad de tiempo en realizarlas, lo cual no sería viable en caso de querer tomar medidas en una gran cantidad de centros educativos.

A continuación se da una breve descripción teórica de su funcionamiento, una

versión más detallada de la misma puede encontrarse en [17].

Assolo se basa en el modelo de PRM definido en la sección anterior. Una forma de establecer una medida usando este método es por ejemplo partir de una frecuencia inicial y comenzar a incrementar linealmente la frecuencia con la que se envían los paquetes hasta encontrar el punto donde se experimentan anomalías en el receptor. O sea el punto en el cual la frecuencia a la cual se reciben los paquetes en el receptor, no es la misma a la que fueron enviados.

Debe tenerse en cuenta que existe una dificultad con incrementar linealmente la frecuencia con la que se envían los paquetes. Esta es que si el ancho de banda es muy lejano a la frecuencia inicial, puede que se tenga que enviar una gran cantidad de paquetes hasta llegar a tal punto en que la red experimente una sobrecarga. Además de la gran cantidad de tiempo que esto podría consumir. Es por eso que los investigadores han desarrollado múltiples maneras de llegar a la frecuencia deseada partiendo de una frecuencia inicial.

En el caso de Assolo, se ha desarrollado el concepto de chirrido exponencial reflejado o en inglés Reflected Exponential Chirp *(reach)*. En este concepto se define el *chirp* como un flujo de $2k + 1$ paquetes capaz de probar una gran cantidad de frecuencias dentro de un intervalo (L, U), con mayor densidad en el centro. Esto significa que un solo *chirp* o flujo es suficiente para obtener una estimación del ancho de banda. La cantidad de flujos a utilizar depende del tiempo que se seleccione para realizar la medida.

La frecuencia con la que se envían los primeros k paquetes de un *chirp* va aumentando hasta probar los valores entre el ínfimo L del intervalo y el valor intermedio $H = \frac{L+U}{2}$. Para enviar el siguiente paquete se aumenta la frecuencia con el fin de probar el valor H y los últimos k van destinados a probar las frecuencias entre H y el supremo U. La forma en que la frecuencia va aumentando al ir probando los distintos valores, no es lineal, sino que va incrementando cada vez más lento desde que parte de cierto valor entre L y H hasta llegar a H y luego comienzan a disminuir cada vez más rápido hasta llegar a un cierto valor final menor que U. De esta forma la zona central del intervalo es la más densa en cuanto a pruebas a realizar, y es donde se obtiene la mayor exactitud, lo cual es acorde a la presunción de que la velocidad buscada probablemente se encuentre en torno al centro de dicho intervalo.

El mecanismo de espaciamiento entre frecuencias funciona de la siguiente manera. Primero se define el valor σ, la máxima precisión relativa con la que este programa puede realizar una estimación, y en consecuencia

$$S = \sigma \left(\frac{U - L}{2} \right) \tag{5.1}$$

es el error absoluto entorno al centro del intervalo.

Otro parámetro importante es γ, la velocidad con que va cambiando la densidad de frecuencias a probar. Con la meta de simplificar esta deducción, se introduce el parámetro extra

$$\Delta_x = S\gamma^{|x-1|} = \sigma \left(\frac{U - L}{2} \right) \gamma^{|x-1|} \tag{5.2}$$

Por más que el usuario es libre de modificar los parámetros recientemente descritos acorde a la situación, en este proyecto se usan los valores por defecto de $\sigma = 5\,\%$ y $\gamma = 1{,}2$.

De esta forma los valores a muestrear desde H hasta el límite superior del intervalo son: $H + \Delta_1$, $H + \Delta_1 + \Delta_2$, $H + \Delta_1 + \Delta_2 + \Delta_3$,... Y los valores desde el centro hasta el límite inferior son $H - \Delta_1$, $H - \Delta_1 - \Delta_2$, $H - \Delta_1 - \Delta_2 - \Delta_3$,..., lo cual permite observar que estos valores varían simétricamente en torno al centro del intervalo.

Por lo tanto las frecuencias de prueba para valores mayores a H se pueden escribir más formalmente como:

$$\begin{cases} R_j = H + \Delta_j, & si\ j = 1 \\ R_j = R_{j-1} + \Delta_j, & \forall\ j > 1 \end{cases} \tag{5.3}$$

Dado que se utilizan k valores para probar las frecuencias entre H y el máximo U, el valor R_k debe coincidir o ser cercano al valor más alto del intervalo, esto se puede escribir como:

$$U = R_k = H + \sum_{i=1}^{k} \Delta_i \ \rightarrow\ U - H = \sum_{i=1}^{k} \Delta_i \tag{5.4}$$

Luego sustituyendo los valores de Δ_i y H se obtiene:

$$U - \frac{L+U}{2} = \sigma \left(\frac{U-L}{2} \right) \sum_{i=1}^{k} \gamma^{|i-1|} \tag{5.5}$$

Lo cual se simplifica a:

$$\frac{1}{\sigma} = \sum_{i=1}^{k} \gamma^{|i-1|} \tag{5.6}$$

El valor de dicha suma se puede escribir como:

$$\sum_{i=1}^{k} \gamma^{|i-1|} = \frac{\gamma^{k+1} - 1}{\gamma - 1} \tag{5.7}$$

y por ende se tiene que

$$\gamma^{k+1} = \frac{\gamma - 1}{\sigma} + 1 \tag{5.8}$$

Luego despejando el valor k se llega a que:

$$k = log_{\gamma}(\frac{\gamma - 1}{\sigma} + 1) - 1 \tag{5.9}$$

Tomando en cuenta el hecho de que la frecuencias de prueba R_k nunca supera al supremo del intervalo, o sea U, la ecuación 5.6 puede reescribirse como

$$U \geqslant H + \sum_{i=1}^{k} \Delta_i \tag{5.10}$$

y por los pasos anteriormente realizados se llega a que:

$$k = \left\lfloor log_\gamma(\frac{\gamma - 1}{\sigma} + 1) - 1 \right\rfloor \tag{5.11}$$

Assolo prueba k frecuencias entre L y H, la frecuencia H y k frecuencias entre H y U, lo cual conforma un total de $2k + 1$ paquetes.

Con la meta de calcular los retardos, Assolo envía un paquete inicial al que no se le asocia ninguna velocidad de llegada, simplemente se utiliza como referencia de tiempo. Gracias a esto, dicha herramienta no depende de ningún reloj externo para calcular tiempos, ni de que los relojes de los distintos equipos estén sincronizados.

El número total de paquetes a enviar en un *chirp* es $2k+2$ o más precisamente:

$$2 \left\lfloor log_\gamma(\frac{\gamma - 1}{\sigma} + 1) - 1 \right\rfloor + 2 \tag{5.12}$$

Esta última ecuación permite evidenciar un detalle importante y es que la cantidad de paquetes a enviar en un *chirp* sólo depende de los parámetros γ y σ. De aquí se concluye que siendo estos parámetros constantes, también lo es la cantidad de paquetes que utiliza el programa para estimar una velocidad.

Asegurar que Assolo utiliza una cantidad fija o poco variable de paquetes para extraer una medición es una ventaja importante. Muchas de las herramientas del mismo tipo que se usan actualmente pueden variar la cantidad de paquetes a utilizar dependiendo de las circunstancias y así llegar a introducir una gran cantidad de tráfico en la red.

5.4.3. Ejemplo de Uso de Assolo

Este programa cuenta con tres partes, un transmisor, un receptor y un maestro. El transmisor es el encargado de enviar los paquetes, el receptor de recibirlos y dar una devolución de los retardos con los que han llegado y finalmente el maestro tiene el trabajo de dar la orden de inicio de la medida. El maestro puede estar ubicado en mismo equipo que el transmisor, receptor o cualquier otro punto de la red que tenga acceso a los mismos.

Con el fin de ilustrar el funcionamiento de Assolo a nivel práctico, se describe el proceso realizado para obtener una medida o conjunto con el mismo. Los pasos son los siguientes:

- *Seleccionar extremos de la red:* El primer paso es seleccionar los dos puntos de la red entre los cuales se desea estimar el ancho de banda disponible e instalar el programa en ambos. Dado que el código fuente del mismo está programado en el lenguaje C y se encuentra disponible en el sitio web [16], es posible descargarlo y compilarlo en una variedad de sistemas operativos Linux, tales como Ubuntu, CentOS, entre otros.

- *Activar transmisor y receptor:* El siguiente paso es activar el transmisor en un equipo y el receptor en otro. Para esto sólo es necesario ir a la carpeta de donde se instaló Assolo y correr los comandos *./assolo_snd* y *./assolo_rcv* en ambos equipos respectivamente. En este caso los mismos tienen las IP 10.64.17.13 y 10.64.17.14.

- *Activar maestro:* Finalmente se procede con activar el maestro, el mismo puede encontrarse en el mismo equipo que el transmisor o receptor, o en algún otro de la red en el cual por supuesto debe estar instalado Assolo. En este caso el mismo se encuentra en una PC de direción IP 10.64.17.10. Para activarlo es necesario ir la carpeta de instalación y correr el comando *./assolo_run* acompañado de los siguientes parámetros obligatorios, *-S* seguido de la IP del transmisor y *-R* seguido de la IP del receptor. Otros parámetros no obligatorios que también pueden ser introducidos son: *-t* acompañado del lapso de tiempo en segundos que se desee que dure la medida, de no utilizar este campo se utiliza el tiempo por defecto, *-U* que permite seleccionar por que puerto del transmisor salen los paquetes o *-p* para seleccionar el tamaño de los mismo. Por información acerca del resto de los parámetros del programa se puede visitar el sitio web [16].

  En este caso se da inicio a la medida mediante la siguiente línea

  ```
  ./assolo_run -S 10.64.17.13 -R 10.64.17.14 -t 20
  ```

- *Inicio de la medición:* Una vez ejecutado el maestro, el mismo envía una paquete TCP al receptor con puerto de origen 42270 y puerto de destino 7365, con el fin de informar al mismo del comienzo de la medida y de los parámetros con los cuales se realizará la misma, como se muestra en la figura 5.1 marcado con una flecha verde. Mientras que el puerto de origen es aleatorio, el puerto de destino es fijo y sólo puede modificarse cambiando el código fuente del programa. Es importante tener conocimiento de este puerto ya que el equipo con el receptor debe tener su firewall configurado para aceptar paquetes TCP provenientes del maestro por el mismo. La figura 5.2 muestra una captura de Wireshark [15] donde se marca el envío y respuesta del paquete recientemente descrito, con un uno.

- *Transmisor al receptor:* El receptor es quien se encarga de informar al transmisor del comienzo de la medida y los parámetros a utilizar en la misma. Para esto envía un paquete UDP al puerto de destino 8365 y con puerto de origen 57159 como se ve marcado en azul en la figura 5.1 y marcado con un dos en la captura de la figura 5.2. El puerto de origen es aleatorio mientras que el de destino es 8365 por defecto y puede ser configurado por el usuario mediante el parámetro *-U*. Es importante que el *firewall* del transmisor esté configurado para aceptar paquetes UDP del receptor por dicho puerto.

- *chirps:* Finalizado el diálogo y ya establecidos los parámetros de medición, el transmisor comienza a enviar una serie de paquetes UDP al receptor marcados en rojo en la figura 5.1 y con un tres en la figura 5.2. Estos paquetes

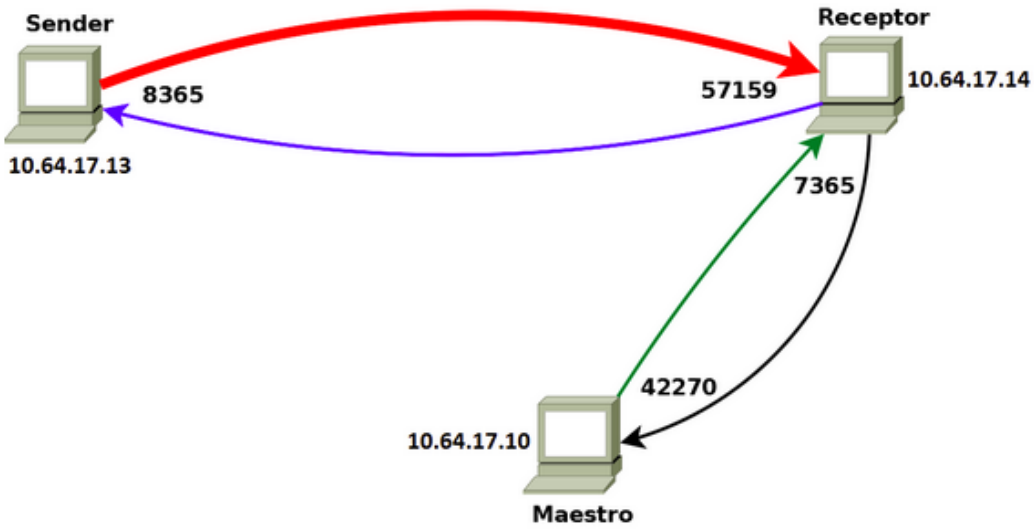


Figura 5.1: Diagrama de equipos, enlaces y puertos usados por Assolo.

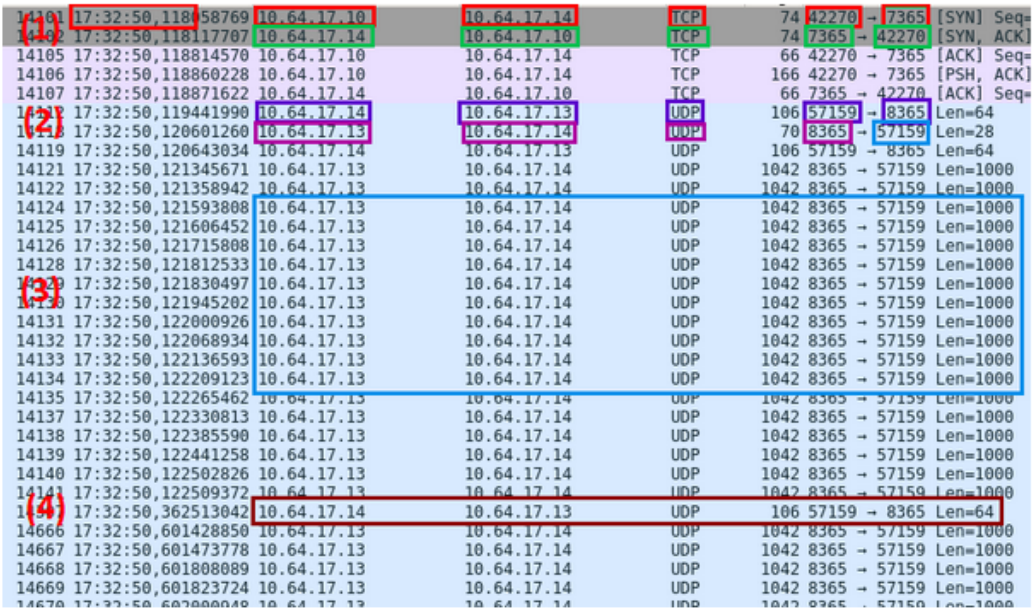


Figura 5.2: Captura de paquetes en Wireshark.

conforman los ya mencionados *chirps* los cuales son utilizados para efectuar la medida.

- *Mensajes de verificación:* En la figura 5.2 marcado con un cuatro puede verse como el receptor envía cada cierto tiempo paquetes de verificación al transmisor con la información pertinente acerca de los retardos con los que está recibiendo los paquetes para que el transmisor mejore la estimación de

la capacidad del canal en los próximos envíos.

- *Estimación del ancho de banda:* Dado que el encargado de procesar la información obtenida para así calcular el ancho de banda es el maestro, el receptor le envía un paquete TCP con la información colectada como se ve marcado con un uno en la figura 5.3. Estos paquetes son enviados desde el puerto de origen 7365 al puerto de destino 42270 como se ve en la figura 5.1.

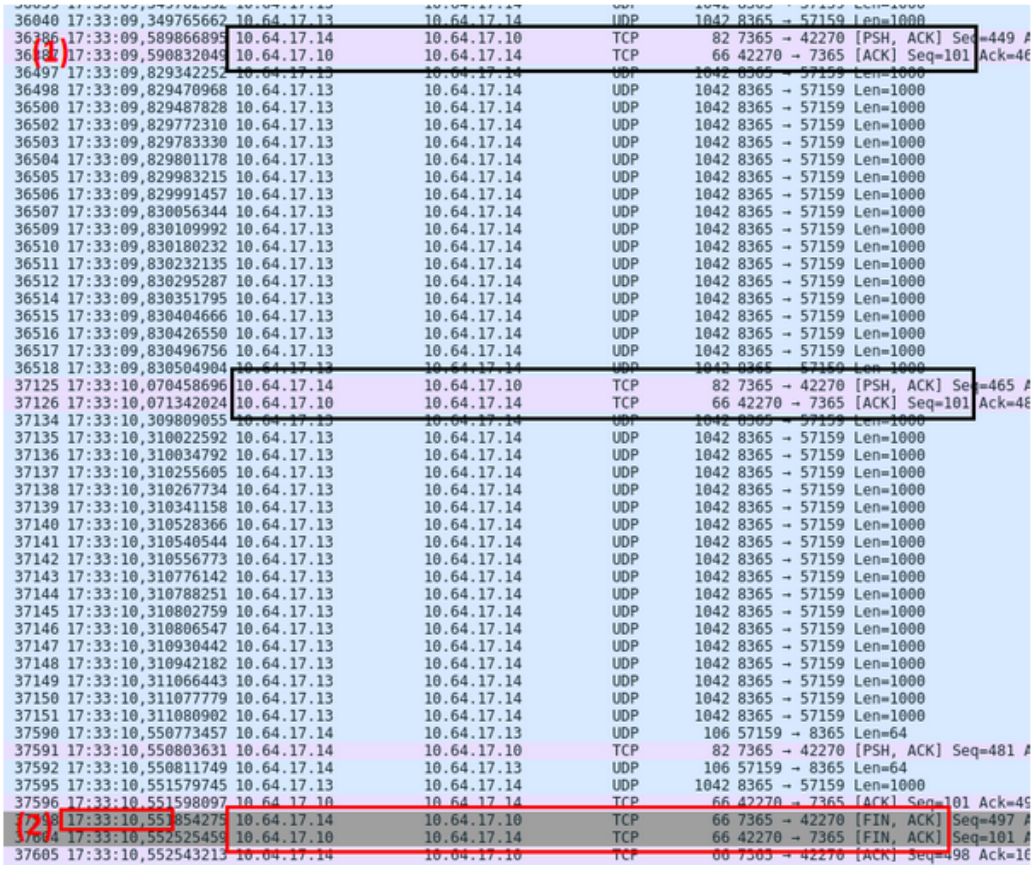


Figura 5.3: Captura de paquetes en Wireshark.

- *Fin de la medición:* La medición es concluida con un paquete que es enviado del receptor al maestro a través de los mismos puertos que en el *item* anterior como se puede ver más abajo en la figura. Hay que recordar que por cada *chirp* enviado Assolo logra hacer una medición, todos estos resultados quedan guardados en un archivo de texto en el mismo equipo que el maestro.

5.4.4. Funcionamiento del Módulo

El funcionamiento de este módulo se diferencia un poco al de los descritos anteriormente ya que para empezar no cuenta con un *script* corriendo en cada uno

de los equipos de los centros educativos, sino que los *scripts* desarrollados corren principalmente en el servidor central de Pando.

En el caso de esta plataforma las medidas son realizadas en ambas direcciones, usando el servidor central como transmisor y el centro educativo como receptor y viceversa. Para esto Assolo debe estar instalado tanto en los centros educativos como en el servidor central y los programas transmisor y receptor de Assolo están todo el tiempo corriendo en los centros educativos seleccionados esperando la orden del maestro para comenzar una medición. Con el fin de lograr que ambos elementos estén siempre corriendo y a la espera en el centro educativo se utiliza el sistema *cron* de Linux de la siguiente forma, agregando las siguientes líneas al archivo de configuración de *cron*, llamado *crontab*.

```
@reboot [ruta a donde se encuentra el programa transmisor de Assolo]
@reboot [ruta a donde se encuentra el programa receptor de Assolo]
```

De esta manera cada vez que se reinicie el equipo se comienzan a ejecutar ambos programas a la vez, quedando a la espera en caso de que se desee tomar alguna medición en cualquier momento. Como se ve más adelante, la ruta hacia los programas de Assolo depende por un lado de la carpeta que se haya seleccionado para instalar la plataforma y por el otro de la arquitectura del equipo. El programa Assolo ensambla una estructura de carpetas dentro del directorio donde se lo instaló, que es dependiente de dicha arquitectura. En la mayoría de los casos se trabajó con equipos Odroid-C2, pero en algunos otros se trabajó con otros tipos de sistemas, por lo tanto la ruta hacia el programa transmisor y receptor no es siempre fija. En consecuencia en lugar de escribir las rutas concretas en el archivo *crontab*, se escribió unas sentencias de código usando el comando *find* de Linux, las cuales encuentran automáticamente dichos programas dentro del directorio asignado y luego, en caso de encontrarlos, los ejecutan. Estas se ven de la siguiente manera.

```
@reboot snd=$(find [carpeta de instalación de la plataforma] -iname
assolo_snd); if [ ‘‘$snd’’ != ‘‘0’’ ]; then $snd; fi;

@reboot rcv=$(find [carpeta de instalación de la plataforma] -iname
assolo_rvc); if [ ‘‘$rcv’’ != ‘‘0’’ ]; then $rcv; fi;
```

Como se puede ver, estas líneas de código primero utilizan en comando *find* para encontrar la ruta a los programas transmisor y receptor de Assolo dentro de la carpeta donde se almacenan los archivos de la plataforma y guardan dicha ruta en una variable. Luego existe una sentencia *if* que comprueba que dicha ruta haya sido encontrada con éxito y si de ser así, ejecutan ambos programas.

Por otro lado, ambos programas también deben estar corriendo en el servidor central, pero en este caso en lugar de estar siempre corriendo, sólo lo están cuando se desee tomar la medida.

En este caso el encargado de agendar las mediciones a una frecuencia elegida

por el usuario y dar las órdenes de ejecución es el programa *cron* alojado en el servidor central y no Zabbix. Cabe aclarar que por más que no se use Zabbix para dar órdenes de ejecución, el mismo sigue siendo el encargado de almacenar los datos colectados.

Cuando llega el momento de ejecución el sistema *cron* activa un *script* programado en Python, que se denomina *moniceibal_assolo_uno*. Este *script* se encarga de leer dos archivos de configuración, uno conteniendo la lista de IPs contra las cuales se quiere realizar la medida y otro conteniendo el lapso de tiempo en segundos que dura cada una. Una vez leídos los archivos el *script* se encarga de activar el transmisor y el receptor de Assolo en el servidor central y luego disparar la medida utilizando el *script moniceibal_assolo_dos* que se describe más adelante en este capítulo.

Como se vio en la sección 5.4.2, el transmisor de Assolo necesita reservar un puerto específico para lanzar los paquetes, y ese puerto sólo puede ser usado para lanzar los mismos contra un solo receptor. Por lo tanto si se quiere enviar paquetes a dos receptores, se necesita reservar dos puertos diferentes. Por este motivo y a la vez para no sobrecargar la red, se eligió realizar las medidas en forma secuencial contra cada una de las IPs. En otras palabras, dada una lista, la plataforma lanza las medidas de subida y bajada para la primera IP de la lista, una vez que estas medidas hayan finalizado lanza la medida contra la segunda y así consecutivamente hasta llegar al final de la lista.

Un posible problema es que el usuario elija tomar las medidas cada un lapso de tiempo determinado, pero el mismo no sea suficiente para finalizar de tomar las medidas contra todas las IPs de lista. En esta situación puede ocurrir que cierta tanda de medidas no haya finalizado y el programa *cron* active el *script* nuevamente para que este lance las medidas nuevamente. Esto implica un problema ya que como se dijo antes el transmisor de Assolo necesita múltiples puertos para ejecutar múltiples medidas y en este caso sólo hay un puerto asignado. Como forma de prevenir este problema, el *moniceibal_assolo_uno* que es ejecutado por el sistema *cron*, revisa que no haya ningún proceso de Assolo corriendo en el servidor central antes de comenzar a disparar las medidas. En caso de encontrarse tal proceso activo, significa que la tanda anterior de medidas no ha finaliza de ejecutarse, entonces el *script* no realiza ninguna otra acción, quedando a la espera de que *cron* lo llame nuevamente.

El siguiente algoritmo ilustra el funcionamiento de *moniceibal_assolo_uno* recientemente descrito.

Algoritmo 4: Moniceibal Assolo Uno.

```
Consultar si hay algún proceso activo de Assolo;
si Hay algún proceso de activo entonces
    Finalizar la ejecución del script;
en otro caso
    Leer el tiempo que debe durar cada medición del archivo de
     configuración;
    Obtener lista de IPs contra las cuales ser realiza la medida del archivo
     de configuración;
    Activar transmisor de Assolo;
    Activar receptor de Assolo;
    para IP in Lista_de_IP hacer
        Ejecutar el Segundo script de Assolo
    fin
fin
```

El *script* denominado *moniceibal_assolo_dos* es el que concretamente se encarga de hacer correr al maestro de Assolo y luego enviar los datos a Zabbix. El mismo está programado en el lenguaje Bash y recibe como parámetros de entrada desde *moniceibal_assolo_uno*, la IP del servidor central, la dirección IP contra la cual se desea tomar las medidas y el lapso de tiempo que durará la misma.

En primer lugar, contando con que tanto el receptor como transmisor se encuentran activos y a la espera en el centro educativo y en el servidor central, este ejecuta el maestro que se encuentra en dicho servidor para poner a correr las medidas de subida y bajada simultáneamente. En otras palabras la medición que utiliza el servidor central como transmisor y el centro educativo como receptor y viceversa. El programa Assolo guarda los resultados colectados en un archivo de texto en el servidor central que utiliza como parte de su nombre el instante de tiempo en que se dio inicio a la medición, pero si se ejecutan dos medidas al mismo tiempo, se termina generando un solo archivo donde Assolo guarda todos los resultados mezclados, lo cual por supuesto resulta inútil. Como solución a este problema se decidió realizar las medidas de forma simultánea pero dando comienzo a una 2 s después que la otra. De esta forma el *script* puede leer los dos archivos por separado sin generar dudas de que dato pertenece a cada medida.

Como ya se dijo Assolo obtiene más de una estimación para el ancho de banda, de hecho obtiene tantas como *chirps* haya podido ejecutar. Por tal motivo es que el *script* debe leer los archivos generados, realizar un promedio de todas las medidas que se encuentran en un archivo y luego realizar un promedio de todas las medidas que se encuentran en el otro para luego enviar ambos datos a Zabbix. Por último es necesario borrar los archivos generados por Assolo para que no se acumulen en el equipo.

Puede darse el caso de que no sea posible tomar alguna medida y esto produce que los archivos se encuentren vacíos, en este caso se envía un mensaje de error a Zabbix explicando que no fue posible estimar el ancho de banda para cierta IP. El funcionamiento de *script* de describe brevemente en el algoritmo 5.

Algoritmo 5: Moniceibal Assolo Dos.

```
Ejecutar Assolo en Ambas Direcciones de Forma Simultánea;
si Archivos Generados por Assolo Se Encuentran Vacíos entonces
    Enviar Mensajes de Error al Servidor de Zabbix: No se Pudo Realizar
     la Medida para la IP Seleccionada;
en otro caso
    Leer Archivos Generados por Assolo y Realizar Promedio de las
     Medidas;
    Enviar Promedios a Zabbix;
fin
Borrar Archivos Generados por Assolo;
```

La figura 5.4 ilustra el funcionamiento del módulo de ancho de banda y la interacción entre *cron* y sus *scripts* recientemente descrita.

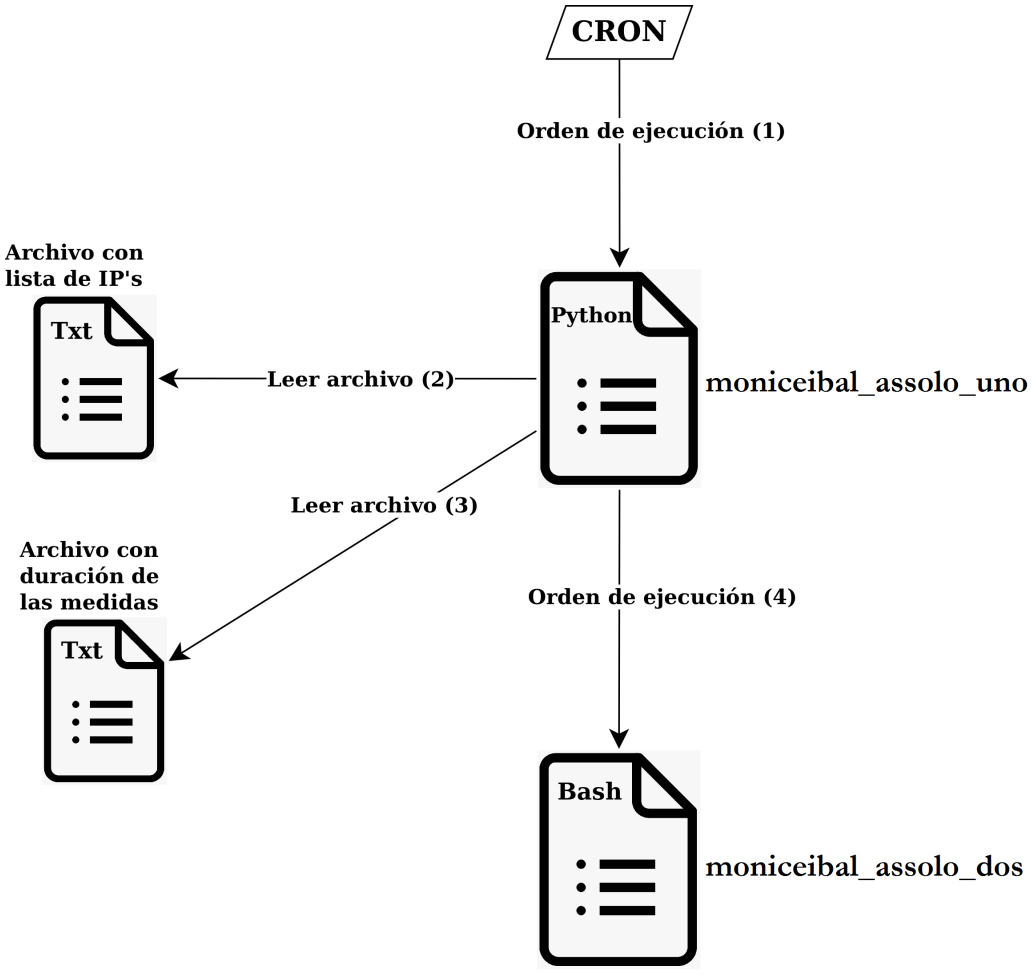


Figura 5.4: Diagrama de funcionamiento del módulo Ancho de Banda.

Capítulo 6

Plataforma de Monitoreo de Red

La plataforma cuenta con un conjunto de módulos recientemente descritos que recaban información de un grupo de equipos y luego la envían al servidor central de Plan Ceibal en Pando. La información es almacenada en dicho lugar durante cierto lapso de tiempo con el fin de poder ser examinada o visualizada.

El conjunto de equipos sobre los cuales se extrae información es variable ya que la plataforma permite agregar o eliminar equipos de la lista de *hosts* a los que se toman medidas. A la vez puede elegirse un determinado *host* para tomar sólo ciertas mediciones, no todos los módulos tienen por que correr en todos los equipos del grupo.

En un principio para que un módulo funcione en un equipo es necesario realizar ciertas modificaciones sobre el mismo, instalar o configurar programas, copiar archivos, etc, todas estas tareas son realizadas usando el programa Ansible desde el servidor central.

Una vez que un módulo esté listo para funcionar en cierto equipo debe de darse la orden de ejecución para que la medida sea tomada cada cierto período de tiempo. Esto es tarea de Zabbix ya que como se dijo antes, este programa es capaz de agendar frecuencias u horarios y dar la orden de ejecución a los *scripts* descritos en el capítulo anterior que se encuentran en los centros educativos. Por último como se mencionó en el capítulo anterior, cada *script* envía el dato colectado a través del comando *zabbix_sender*, el cual es recibido por un *trapper item* y almacenado en la base de datos. En general esta es la estructura de la mayor parte de los módulos como se muestra en la figura 6.1, excepto el de medida de ancho de banda, el cual se describe posteriormente.

6.1. Estructura

En esta sección se describen algunos de los aspectos más importantes para el funcionamiento de los módulos de Tiempo DHCP, Tiempo DNS, y Tiempo de Descarga de Sitio Web.

Por el lado del servidor de Zabbix deben existir dos *items* por módulo, un *item*

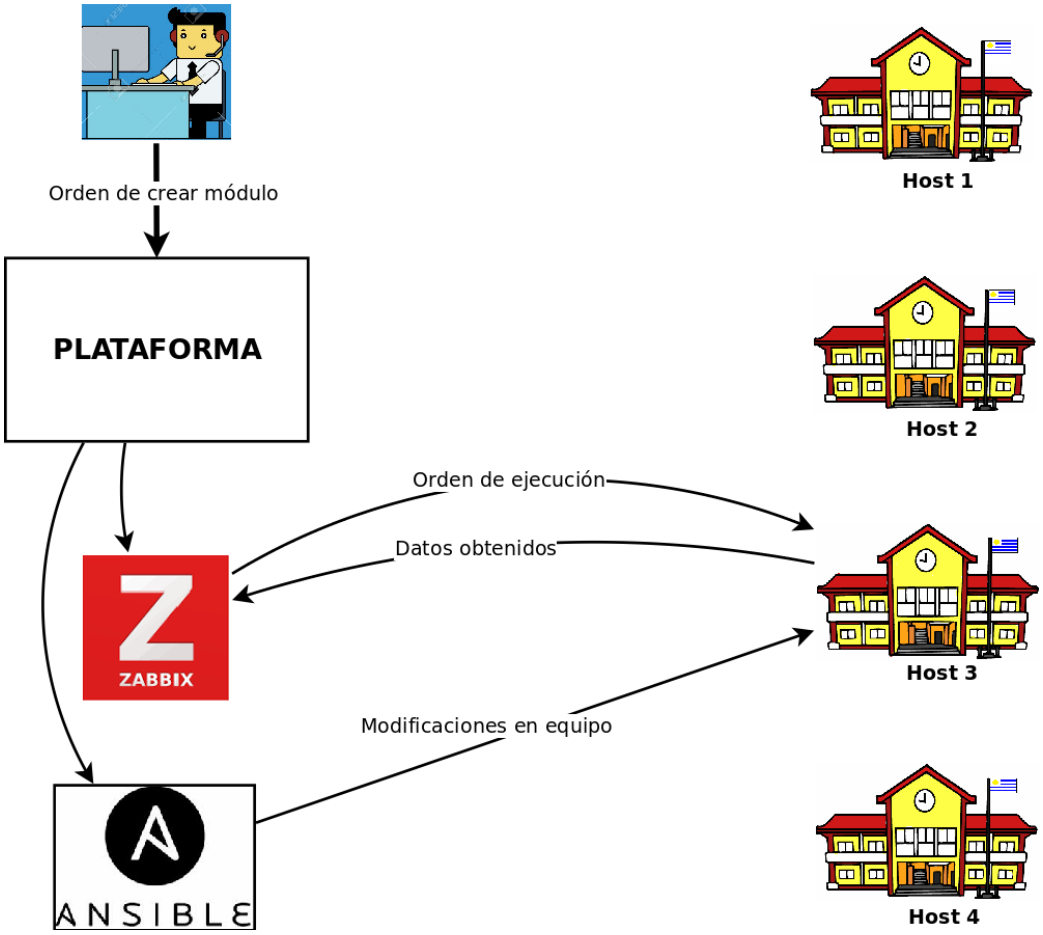


Figura 6.1: Diagrama de funcionamiento de la Plataforma.

del tipo Zabbix *agent* el cual es llamado *item* Launch ya que tiene la tarea de disparar la medida encargándole la tarea al usuario daemon mediante el comando *at*. El otro *item* es un *trapper item* que simplemente recibe la medida obtenida cuando el usuario daemon ejecuta el *script* en el centro educativo. Finalmente debe existir un tercer *item* llamado Errores que es común a todos los módulos y es el encargado de recibir todos los mensajes de error descritos en el capítulo anterior que puedan ocurrir durante la toma de las medidas.

Como se mencionó antes, existe una gran lista de medidas que Zabbix es capaz de realizar por defecto, pero si se desea extender esta lista mediante *scripts* desarrollados por los usuarios de Zabbix, también es posible hacerlo. Para esto es necesario agregar una línea llamada UserParameter al archivo de configuración del agente en el *host* donde corre dicho *script*, en este caso la línea luce de la siguiente forma:

`UserParameter= [`*`key`* `del nuevo tipo de` *`item`*`], echo [ruta hacia el` *`script`*`] [IP del servidor de Zabbix] [directorio donde se encuentran los archivos de la plataforma dentro del` *`host`*`] [ruta hacia el archivo de configuración del`

módulo] | *at now*

Observando dicha línea, luego de la ruta hacia el *script*, se encuentra la IP del servidor de Zabbix, el propósito de este parámetro es que el *script* sepa a qué dirección enviar los datos luego de tomar la medición. El siguiente parámetro es la ruta a donde se encuentran los archivos de la plataforma dentro del *host*, el mismo existe en caso de que el módulo necesite crear o leer archivos durante su ejecución. En particular el módulo de tiempo de descarga de sitio web necesita guardar los archivos descargados al ejecutar el comando *wget*, los cuales por supuesto son eliminados posteriormente. El último parámetro es la ruta hacia el archivo de configuración. En particular el *script* de tiempo DNS necesita leer este archivo para extraer la lista de *hostnames* para realizar las peticiones y medir los tiempos, y el *script* de tiempo de descarga de sitio web necesita leer ese archivo para saber qué sitio web descargar y con qué opciones del comando *wget*. Se aprecia que el módulo de tiempo DHCP no tiene archivo de configuración ya que el mismo no tiene parámetros configurables por lo tanto esa parte de la línea permanece en blanco para dicho módulo.

Para el funcionamiento del módulo es necesario realizar múltiples modificaciones en el equipo, la primera ya mencionada es modificar el archivo de configuración del agente de Zabbix para agregar la línea UserParameter. También es necesario copiar los *scripts* o archivo de configuración o instalar los programas ya mencionados, todas estas tareas deben ser realizadas por Ansible antes de crear los *items* en Zabbix y antes de que comiencen a tomarse las medidas.

Existen ciertas variables importantes para el funcionamiento de la plataforma las cuales son utilizadas por todos los *scripts*. Las mismas contienen información acerca de los directorios en los que opera la plataforma o las direcciones IPs de sus elementos más importantes. Puede que debido a algún cambio en la estructura del servidor central, el usuario decida modificar alguna de ellas. Todas estas variables se encuentran concentradas en un mismo archivo de Python llamado *locations*, el cual es importado por el resto de los *scripts*. De esta forma se logra que en caso de querer modificar alguna variable, sólo haya que modificar un archivo y no varios. Las variables disponibles en *locations* son las siguientes:

- *Usuario de Zabbix*
- *Contraseña de Zabbix*
- *IP de Zabbix*
- *Usuario de Ansible*
- *IP de Ansible*
- *IP de Plataforma*
- *Usuario de Plataforma*

- *Carpeta Plataforma* Carpeta alojada en el equipo donde corre la plataforma, que almacena todos los archivos necesarios para el funcionamiento de la misma.
- *Carpeta Ansible* Carpeta alojada en el equipo de Ansible a donde se le transmiten los archivos que el mismo debe distribuir.
- *Usuario Centros educativos* Es el usuario que usa Ansible para loguearse en los centros educativos y así poder realizar los cambios necesarios.
- *Carpeta Destino* Es una carpeta que está en cada equipo de cada centro educativo donde se alojan los archivos necesarios para el funcionamiento de la plataforma.

Finalmente la plataforma cuenta con una función para cada módulo que se encarga de realizar la gestiones necesarias para que la medida se comience a tomar en un equipo o grupo seleccionado por el usuario. La plataforma se comunica con Ansible para realizar todas las tareas relacionadas con la instalación, copia de archivo, etc. A su vez realiza diferentes chequeos de errores que buscan detectar distintos tipos de fallas. Por otro lado, también existe otra función que hace que la medida se deje de tomar en dicho equipo, todas estas funciones se describen a continuación.

6.2. Funciones

Cada una de las funciones que interactúan con el usuario implementadas dentro de la plataforma están constituidas por un *script* programado en el lenguaje Python.

A continuación se describen las referentes a los módulos de tiempo DNS, tiempo DHCP y tiempo de descarga de sitio web, dejando las referentes al módulo de ancho de banda para la siguiente sección:

- *adddhcp*: La función se encarga de crear el módulo tiempo DHCP en los equipos en los cuales aún no existe o realizar cambios en los que ya está funcionando. Sus parámetros de entrada son los siguientes:
  - *-f*: Este campo permite seleccionar o cambiar la frecuencia de medición, así como agendar pruebas para realizarlas en días concretos de forma no periódica. Su contenido debe ser introducido en la sintaxis de Zabbix, ya que dicho parámetro es pasado directamente a Zabbix para que el mismo se encargue de dar las órdenes de ejecución.

    Dado que este campo también es opcional, si no se introduce nada en su lugar o si se introduce una frecuencia con la sintaxis incorrecta, se utiliza el valor por defecto de una hora o en caso de que el módulo ya exista en el equipo se continúa usando el valor que se usó hasta el momento.

- *-force*: Antes de instalar el módulo en una de las IPs introducidas por el usuario, la plataforma comprueba si los *items* correspondientes no existen ya en Zabbix para tal IP. En el caso de existir los *items*, se asume que el módulo ya se encuentra funcionando en dicho equipo y se ahorra el trabajo de la instalación. No obstante, pueden darse distintas situaciones como por ejemplo que los *items* estén creados en Zabbix pero el equipo correspondiente a esa dirección IP haya sido recientemente remplazado y no tenga el módulo instalado. Otra situación posible es que el *script* del módulo sea actualizado y que tenga los *items* creados en Zabbix, de todas formas se quiera reinstalar el mismo.

  Por lo tanto al introducir la opción *-force*, el *script* procede con la instalación del módulo ya sea que los *items* existan o no en Zabbix.

A continuación se describen los tres parámetros que permiten introducir una IP o grupo. Ya que la función debe trabajar sí o sí sobre al menos un equipo, es obligatorio introducir una y sólo una de las siguientes opciones.

- *-u*: Permite introducir una lista de IPs separadas por espacios sobre las cuales se instala el módulo. En primer lugar la función se encarga de verificar que todas las IPs introducidas se encuentren registradas en el servidor de Zabbix, ya que si no están ahí no es posible que este les de órdenes de ejecución. En caso de que una de las IPs no esté registrada se detiene el programa, informando al usuario de lo ocurrido. Por otro lado, si una dirección se encuentra repetida, la función la toma en cuenta una sola vez.

  En caso de que el módulo ya se encuentre instalado en alguno de los equipos con dicha IP asignada, la plataforma se asegura de que el mismo se encuentre habilitado y de modificar su frecuencia de medición en caso de que se haya introducido una nueva.

- *-y*: En lugar de recibir una lista de IPs como el campo anterior, recibe la ruta de un archivo donde se encuentran guardadas tales direcciones. En este archivo las IPs deben estar escritas en una sola columna para que el programa pueda leerlas.

  Al igual que en la opción *-u*, la función se asegura de que todos los equipos estén registrados en Zabbix y de descartar IPs repetidas. En caso de que el módulo ya se encuentre instalado en alguno de los equipos con dicha IP asignada, la plataforma se asegura de que el mismo se encuentre habilitado y de ser necesario modifica su frecuencia.

- *-g*: En esta opción se recibe el nombre de un grupo de *hosts* de Zabbix con la meta de realizar la acción sobre todas las IP que se encuentran en ese grupo. En primer lugar la función se asegura que el grupo exista y siguiente procede a almacenar todas sus IPs en una variable.

  En el servidor de Zabbix no sólo hay registradas direcciones IPs de equipos Linux sobre los cuales se puede tomar medidas, también hay otros elementos, específicamente APs de Cisco a los cuales los registra

bajo las IPs 127.0.0.0 o 127.0.0.1, en consecuencia una vez obtenida la lista de IPs de cierto grupo, dichas IPs son filtradas, dado que sino se produce un error. En caso de que el módulo ya esté creado sobre alguna de las IPs, la función procede de la misma forma que en las opciones *-u* o *-y*.

A continuación se describe el funcionamiento de dicha función y sus bloques.

- *Recepción y Procesamiento de los Parámetros*: Esta comienza procesando los parámetros recibidos y asegurándose que todos se encuentren ingresados tal y como se menciona en la lista recientemente descrita. Para este proceso se elaboró el llamado contador de parámetros el cual cuenta cuántos de los parámetros introducidos por el usuario son efectivamente reconocidos por la función y serán utilizados. Si esta cantidad no coincide con la cantidad total de parámetros ingresada entonces el usuario cometió algún error al introducirlos. Es importante destacar que esta acción permite prevenir varios inconvenientes tales como errores de sintaxis por parte del usuario y es realizada al principio, antes que cualquier otro proceso con el fin de evitar que dichos errores se propaguen a lo largo de la ejecución del *script*.

  Como se dijo antes es obligatorio introducir uno y sólo uno de los parámetros *-u*, *-y* o *-g*, ya que el *script* debe trabajar sobre al menos una IP. Por lo tanto en caso de que no se haya introducido alguna opción de IP o si el contador de parámetros detecta algún error, la ejecución del *script* es detenida informando al usuario de lo ocurrió.

  Por último, antes de pasar a la siguiente etapa, se almacena en una variable en qué lugar de lista de parámetros de entrada fueron encontrados los parámetros *-f*, *-u*, *-y* o *-g*.

- *Autenticación en la API de Zabbix*: Hay que notar que este *script* necesita estar en contacto con el servidor de Zabbix durante su ejecución, ya sea para obtener información o para realizar cambios en el mismo. Como se dijo en el capítulo 3, esta comunicación se realiza a través de la API de Zabbix que funciona en base a mensajes *posts* con cuerpo en formato JSON. Por ende el primer paso es enviar un mensaje *post* con el fin de autenticarse en Zabbix. Este mensaje debe contener un nombre de usuario y contraseña y como respuesta recibe una clave la cual debe ser usada en cada mensaje *post* que se le envíe a Zabbix durante la ejecución del *script*.

- *Bloque -u*: En caso de que se haya introducido dicho parámetro a la entrada, el mismo debe estar seguido por una lista de IPs, y el trabajo de este bloque es almacenarlas en una variable, eliminando las que se encuentren repetidas.

  Es importante que todas estas IPs estén registradas en el servidor Zabbix. Si no lo están, no es posible que el mismo dé la orden de ejecución de los módulos en dichos equipos o almacene los datos recolectados.

De darse esta situación el *script* se detiene e informa al usuario de lo ocurrido.

A través del intercambio de mensajes JSON este bloque crea tres diccionarios de Python. El primero guarda los nombres de los *hosts*, el segundo guarda los números de identificación de los *host* y el tercero, los números de interfaz con el que está registrado cada *host* también en Zabbix.

- *Bloque -y*: Dicho parámetro debe estar seguido por la ruta a un archivo de texto que contenga la lista de IPs en una sola columna. En primer lugar el bloque verifica que este archivo exista y sea accesible y siguiente conforma una lista donde cada entrada se corresponde con una fila de la columna.

  Puede ocurrir que el contenido de alguna entrada de la lista no sea realmente una IP o no sea una IP registrada en Zabbix, por lo tanto, de la misma forma que en el caso anterior se verifica mediante mensajes *post* que todas las entradas de la lista sean IPs propiamente registradas.

  Por último en este caso también se crean los tres diccionarios creados en el bloque *-u*.

- *Bloque -g*: En este caso el parámetro se encuentra seguido no por una lista de IPs sino por el nombre de un grupo de Zabbix, por lo tanto es trabajo del bloque intercambiar mensajes *post* con Zabbix con el fin de extraer la lista de IPs que ese grupo contiene. Una vez conformada la lista se crean los tres diccionarios descritos en el bloque *-u* y como ya se explicó se eliminan las IPs 127.0.0.0 o 127.0.0.1.

- *Inventario de Ansible:* Como ya se dijo antes, Ansible es el encargado de realizar las modificaciones necesarias en los equipos en los cuales se toman las medidas. Dicho programa realiza los cambios deseados mediante conexiones *ssh*, por ende para realizar dichos cambios existen dos factores importantes. El primero es que el equipo desde el cual se ejecuta Ansible pueda conectarse por *ssh* mediante clave compartida a los equipos de los centros educativos, lo cual conlleva a un requisito previo para poder instalar la plataforma. De no ser así tiene que introducirse a mano una contraseña por cada equipo de la lista o tienen que almacenarse las contraseñas dentro del *script*, lo cual no es recomendable por motivos de seguridad. Un segundo factor es que para poder realizar una conexión, Ansible necesita que tal IP esté registrada dentro de su inventario. Esta segunda tarea es realizada por la plataforma, toma la lista de IPs ingresada por el usuario y las escribe en el inventario de dicho programa.

  Como se dijo anteriormente, la plataforma y Ansible se encuentran en dos equipos distintos, por lo tanto para poder modificar el inventario o darle órdenes a Ansible es necesario establecer una conexión *ssh* entre la computadora en que está corriendo el *script* y Ansible. Ya que el *script* se encuentra programado en Python, se utiliza el paquete Paramiko ya

mencionado anteriormente.

- *Crear items*: El siguiente bloque cuenta con dos tareas importantes, la primera es realizar todos los cambios necesarios en los equipos remotos para poder ejecutar los módulos y por el otro lado crear los *items* en Zabbix.

  Para empezar se debe contemplar la posibilidad de que el módulo ya esté creado y funcionando en el equipo. Para esto se verifica si el *item* launch ya se encuentra creado en el servidor de Zabbix. De ser así se considera que módulo ya se encuentra en el equipo y se ahorra el proceso de instalación. Dado que existe la posibilidad de que tal *item* esté creado, pero desactivado, se lo activa por si acaso. Si se da que el *item* aún no existe para dicho *host* o que el usuario ha introducido el parámetro *-force*, se da paso a Ansible para que ejecute todas las tareas referentes a la creación del módulo en el equipo remoto. Para esto el primer paso es transmitir el *playbook* que se utiliza para crear *items* al equipo donde está instalado Ansible, lo cual se logra usando el comando *sftp* de Paramiko que sirve para transmitir archivos. Por último se ejecuta el *playbook* el cual realiza las siguientes tareas:

  - *Instalar dnsutils*: Este paquete contiene el comando *dig*, el mismo se utiliza para realizar consultas DNS.
  - *Instalar wget*: Dicho comando se utiliza para descargar sitios web de forma recursiva.
  - *Instalar at*: A este comando se le puede encargar la ejecución de tareas a un horario definido por el usuario.
  - *Cambiar el tiempo máximo de ejecución de Zabbix a 30 s:* Si bien como ya se mencionó Zabbix da 30 s a la ejecución de un *item* del tipo Zabbix *agent*, este parámetro a veces se encuentra configurado por defecto a tiempos menores como 3 s. También es cierto que en este caso el *item* no toma ninguna medida de por sí, sino que se la encarga al usuario daemon mediante comando *at* y por lo tanto 3 s o menos debe ser suficiente para realizar dicho encargo. Sin embargo este valor es convertido en 30 s por precaución, ya que si en ese momento el CPU del equipo se encuentra sobrecargado puede que dicha tarea tome más tiempo.
  - *Crear directorio de trabajo:* Este parámetro es configurable por el usuario y es el lugar donde se alojan todos los archivos relacionados con la plataforma dentro del *host*. En caso de que el mismo ya exista, se lo mantiene sin modificaciones.
  - *Copiar script del módulo*: Se trata del *script* principal que al ejecutarse obtiene la medida deseada, el mismo se copia al directorio de trabajo. En caso de que ya exista un archivo con el mismo nombre en la carpeta, el mismo se sobrescribe.
  - *Crear UserParameter*: Se crea la línea UserParameter referente al *item* dhcp_launch en el archivo de configuración del agente de

Zabbix, la cual luce de la siguiente manera:

```
UserParameter= dhcp_launch, echo [ruta hacia el script] [IP
del servidor de Zabbix] [directorio donde se encuentran los
archivos de la plataforma dentro del host] | at now
```

En el caso de los *items dhcp_trapper* y Errores, no es necesario crear esta línea, ya que como se explicó antes, los *trapper items* no necesitan UserParameters. En caso de que la línea por algún motivo ya se encuentre ahí, el programa la sobrescribe para evitar que haya dos UserParameter con el mismo Key. Si esto sucede, ocurre un error en el archivo de configuración y se pierde la conexión con el servidor central.

- *Cambiar permisos de script para ejecutarlo*: El programa se encarga de hacer que el *script* del módulo sea ejecutable por cualquier usuario, incluyendo los usuarios zabbix y daemon.
- *Crear grupo Moniceibal y agregar los usuarios que lo conformarán*: Como ya se dijo existe una carpeta en cada *host*, la cual almacena todos los archivos vinculados al funcionamiento de la plataforma. Cada módulo no sólo debe usar archivos que se encuentren dentro de ella, sino que en ocasiones necesitará guardar archivos que haya generado y luego borrarlos.
  Para realizar todas estas acciones un usuario necesita tener permisos sobre dicha carpeta, el primer usuario que debe tener tales permisos es el usuario administrador que utilice Ansible para crearla. El mismo no sólo se encarga de crear la carpeta, sino que puede tener que realizar modificaciones sobre ella en el futuro, por ejemplo si se agregan nuevos módulos. Por otro lado, están los usuarios zabbix y daemon que necesitan acceso al contenido de la carpeta y posiblemente realicen modificaciones sobre la misma. Para que los tres usuarios puedan realizar todas estas tareas, es necesario que los tres sean los dueños de la carpeta. En Linux, la forma de que un directorio pertenezca a más de un usuario es primero crear un grupo de usuarios y luego hacer que ese grupo sea el poseedor de la misma. Por ende se crea el grupo Moniceibal, se agrega los tres usuarios al mismo y se lo convierte en el dueño de la carpeta.
- *Reiniciar agente de Zabbix*: Por último es necesario reiniciar el agente de Zabbix para que todos los cambios realizados cobren efecto.

- *Crear item dhcp_trapper*: En primer lugar se verifica si este *item* del tipo *trapper* existe o no. Si no existe se lo crea, y si ya existe se lo activa en caso de que pueda llegar a estar desactivado.
- *Crear item Errores*: Este también es un *item* de tipo *trapper*, igual que en el último caso, se verifica su existencia, si no está creado se lo crea y

si ya existe se lo activa por si acaso. Debido a que este *item* es común a todos los módulos, si ya hay otros módulos de la plataforma corriendo en dicho equipo, entonces el *item* ya está creado.

- *Bloque -f*: En caso de que el usuario haya introducido el parámetro *-f*, el mismo debe estar seguido por la frecuencia a la que se desea tomar la medida. Dado que dicha frecuencia es enviada directamente a Zabbix mediante la API, la misma debe estar en el formato de Zabbix. Como ya se dijo, el lenguaje que usa Zabbix para tomar medidas es muy rico y da lugar a una amplia gama de posibilidades. En consecuencia su sintaxis es un tanto compleja para ser chequeada directamente desde el *script*, por ende se deja que Zabbix mismo sea el encargado de verificarla. La frecuencia es enviada a Zabbix tal como la ingresó el usuario, si la misma es correcta entonces es adoptada por el módulo y sino es el mismo Zabbix quien la desecha. Si la misma es descartada, el módulo se mantiene con la frecuencia por defecto de una hora o la frecuencia que tuviera anteriormente en caso de ya existiera.

- *adddns*: Esta función se encarga de crear el módulo tiempo DNS en los equipos en los cuales aún no existe o realizar cambios en los que ya está funcionando. A diferencia del módulo anterior, este contiene un archivo de configuración el cual contiene todas los *hostnames* que se desea utilizar para tomar la medida. Por lo tanto esta función contiene todos los parámetros de la función anterior, más el nuevo parámetro denominado *-s* que se describe a continuación:

  - *-s*: Esta opción permite introducir una lista de *hostnames* separados por espacios sobre las que se toma la medida de tiempo de DNS. Dicha lista es guardada en el archivo de configuración del módulo en cada equipo donde el mismo se encuentre corriendo.

    Dado que este parámetro es opcional, de no introducirlo se utiliza la lista por defecto o en caso de que el módulo ya exista en el equipo se continúa usando la lista ya existente.

  En cuanto al funcionamiento interno de la función, esta recorre todos los mismos sectores que la anterior, con la adición de un nuevo bloque que se encuentra justo antes del bloque crear *item* y se describe a continuación.

  - *Bloque -s*: Como ya se mencionó esta opción debe ser precedida por una lista de *hostnames* con las cuales se toma la medida. En primer lugar se verifica que se haya introducido por lo menos un *hostnames* después del parámetro *-s*. Es importante comprobar que cada una de los *hostnames* introducidos realmente corresponda a un sitio web, para esto se realiza una petición DNS desde el *script* y en caso de encontrar un error se detiene el *script* y se informa al usuario.

    Una vez revisada la lista de *hostnames* se genera el archivo de configuración el cual consta de una sola columna con todas los *hostnames*. El

mismo debe ser transmitido al equipo donde reside Ansible para que este pueda enviarlo a cada uno de los equipos donde se desea instalar el módulo. De esta forma cuando se ejecute el módulo, es capaz de leer el archivo de configuración con el fin de extraer los *hostnames* y tomar las medidas.

En caso de que el usuario no haya introducido el parámetro *-s* y por lo tanto ningún *hostname* y el módulo aún no exista en el equipo seleccionado, el *playbook* crear *item* se encarga de transmitir un archivo de configuración con una lista de *hostnames* por defecto para que la medida pueda ser tomada de todos formas.

La última diferencia respecto a la función anterior es que la línea UserParameter creada en el archivo de configuración del *host* tiene por supuesto un *key* diferente y la ruta a un archivo de configuración, lo cual no era necesario para el último módulo. La misma luce de la siguiente manera:

`UserParameter= dns_launch, echo [ruta hacia el script] [IP del servidor de Zabbix] [directorio donde se encuentran los archivos de la plataforma dentro del host] [ruta hacia el archivo de configuración del módulo] | at now`

- *addwget*: Esta función se encarga de crear el módulo de tiempo de descarga de sitio web y tiene un aspecto muy similar a la anterior con la diferencia de que su parámetro y bloque *-s* cumplen funciones un poco diferentes. Los mismos se describen a continuación:

  - *-s*: Esta opción permite introducir el *hostname* que se quiere descargar para realizar la medida de tiempo junto con la opción del comando *wget* que se desea utiliza para realizar la descarga.
  - *Bloque -s*: Este parámetro debe estar seguido por un *hostname* y una opción del comando *wget*. Con el fin de verificar que el *hostname* realmente corresponde a un sitio web y que la sintaxis de la opción seleccionada es correcta, el *script* intenta realizar la descarga. Si la misma no logra realizarse, se informa al usuario de lo ocurrido y se finaliza la ejecución del *script*. En caso contrario se crea un archivo de configuración conteniendo la información introducida por el usuario el cual es luego transmitido a los equipos seleccionados usando Ansible. En caso de que el usuario no introduzca el parámetro *-s* y el módulo aún no esté creado en el equipo, el *script* transmite un archivo de configuración por defecto para que la medida pueda ser tomada de todas formas.

A continuación se describen tres funciones que permiten detener la ejecución de los tres módulos recientemente descritos. Los tres *scripts* son exactamente iguales, sólo difieren en su nombre y módulo a deshabilitar, naturalmente sus nombres son *deldhcp*, *deldns* y *delwget*.

Las mismas son capaces de recibir una IP o lista, de la misma forma que las funciones anteriormente descritas, o sea a través de los parámetros *-u*, *-y* y *-g*, de los cuales se debe introducir uno y sólo uno.

Como estas funciones sólo sirven para deshabilitar, no aceptan los parámetros *-f* o *-s* que están vinculados a cambiar la frecuencia u opciones del módulo.

A continuación se brinda una descripción de su estructura, sus primeros cinco bloques funcionan de la misma forma que se describió en las funciones anteriores, estos son:

- *Recepción y Procesamiento de los Parámetros*
- *Autenticación en la API de Zabbix*
- *Bloque -u*
- *Bloque -y*
- *Bloque -g*

Para que un módulo deje de tomar medidas alcanza con desactivar su *item* Launch ya que éste es el que da la orden de ejecución. Por lo tanto una vez deshabilitado tal *item*, no se toman más medidas y se mantienen las ya obtenidas en la base de datos. Dado que todo el contenido del módulo instalado en los equipos sigue estando ahí, en caso de que se quiera volver a colectar dichas medidas, sólo es necesario volver a activar el *item* Launch, lo cual puede hacerse mediante las funciones *adddhcp*, *addddns* o *addwget*.

Antes de desactivar dicho *item* es necesario verificar que el mismo realmente existe en el equipo, en caso de que no exista, se procede a desactivar el módulo en el resto de las IPs ingresadas.

6.3. Módulo Ancho de Banda

Este módulo se caracteriza por tener un funcionamiento diferente al de los descritos anteriormente, como ya se mencionó antes, no cuenta con un *script* desarrollado durante el proyecto que corra en cada *host*. Lo único que corre en cada uno de dichos *hosts* es el programa Assolo que se utiliza para tomar las medidas.

Las funciones correspondientes al módulo de ancho de banda son diferentes a las funciones anteriores en su estructura y tienen otra interacción entre la plataforma, Zabbix, Ansible y los *hosts* como se indica en la figura 6.2. En ella se puede ver que Ansible debe realizar ciertas modificaciones en los equipos remotos tales como instalar Assolo y ponerlo a correr como indica la flecha tres. Por otro lado, las medidas de ancho de banda de subida y bajada son disparadas directamente desde la plataforma contra los centros educativos mediante el sistema *cron* como indican las flechas cuatro y cinco. Por último, es la plataforma misma quien se encarga de enviarle los datos colectados a Zabbix como se ve en la flecha seis y no los centros educativos.

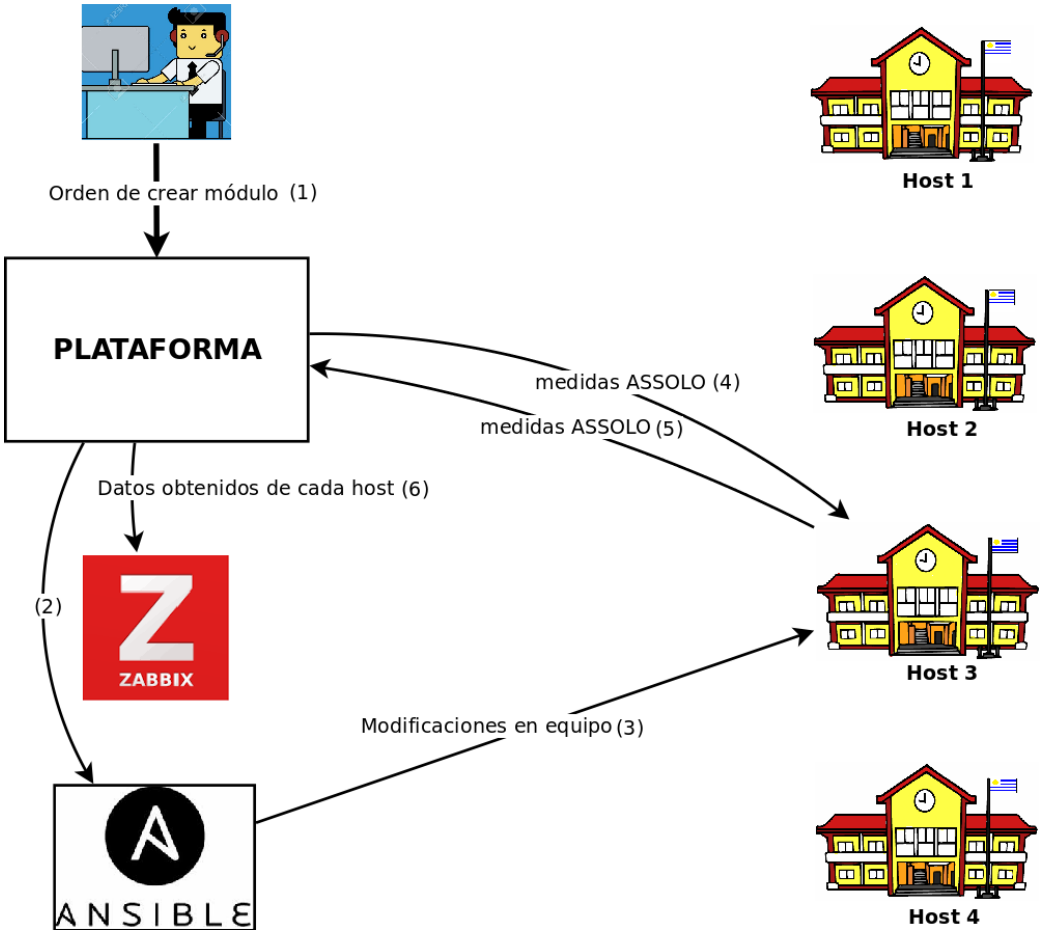


Figura 6.2: Diagrama de funcionamiento del módulo ASSOLO.

En este caso el usuario puede utilizar las funciones que se describen a continuación para agregar nuevas IPs a la lista de equipos donde se quiere tomar medidas, cambiar la frecuencia de ejecución y el tiempo que duran las medidas. Una vez agregada una IP a la lista, la plataforma crea los *items* necesarios para recibir los datos en Zabbix y da la orden a Ansible para que se encargue de instalar el programa Assolo en dicho equipo. Seguidamente se describen las dos funciones con las que se puede gestionar este módulo.

- *addassolo*: Es la función que se encarga de crear el módulo Assolo en los equipos en los cuales aún no está funcionando o realizar cambios en lo que ya está funcionando. La función consta de cinco parámetros que se describen a continuación.
  - *-t*: Esta opción permite establecer cuanto tiempo va a durar la estimación del ancho de banda por parte del programa Assolo para cada una de las IP. Su valor debe introducirse en segundos.
  - *-f*: Esta opción indica con qué frecuencia se realiza la medida para toda

la lista de IPs. Dado que en este caso se utiliza *cron* para ejecutar las mediciones, tal frecuencia debe ser introducida entre comillas y en el formato de *cron*.

Al igual que en las demás funciones, hay tres maneras en que es posible introducir una IP o grupo, o sea mediante los parámetros *-u*, *-y* y *-g*, cuyo funcionamiento es el mismo que en las funciones anteriores.

Seguidamente se describe la estructura de la función y sus bloques. Sus primeros cinco bloques son los siguientes y tiene el mismo funcionamiento que el descrito en las funciones anteriores:

- *Recepción y Procesamiento de los Parámetros*
- *Autenticación en la API de Zabbix*
- *Bloque -u*
- *Bloque -y*
- *Bloque -g*

Una vez extraída la lista de IPs contra las cuales se toma la medida, se las agrega a un archivo de configuración ubicado en el servidor central.

Los bloques siguientes son propios de esta función, por lo tanto su funcionamiento no se encuentra descrito en las funciones anteriores.

- *Bloque -t*: Después de dicho parámetro se espera que el usuario introduzca una cantidad de tiempo en segundos, que indica el lapso de tiempo que dura cada medición. Es necesario revisar que tal parámetro sea efectivamente un número entero para poder guardarlo en un archivo de configuración, de lo contrario se informa al usuario de lo ocurrido. Es importante notar que este archivo no es transmitido mediante Ansible, sino que se mantiene en la plataforma dentro del servidor central.
- *Crear item*: En primer lugar se inicia una conexión *ssh* a través del paquete Paramiko con el equipo que contiene Ansible como se describió anteriormente. Esta conexión es primero que nada usada para agregar al inventario de Ansible las IPs que aún no se encuentran ahí.

  Como en casos anteriores es posible que el módulo ya esté funcionando en algunas de la IP ingresadas, esto se verifica fácilmente mediante la API de Zabbix. Si el mismo ya se encuentra corriendo en cierta IP, entonces el servidor central debe tener creados los *trapper item*, Ancho de banda Subida y Ancho de banda bajada. Si dichos *items* no están ahí, se asume que el módulo aún no ha sido instalado en dicho equipo y se da paso a Ansible para que instale Assolo en cada equipo contra el cual se desea tomar la medida. Para dicha instalación el mismo debe realizar cierto conjunto de pasos que se describen en el siguiente *playbook*:

- *Instalar build-essential, gcc, g++, make, manpages-dev y libpcap-dev:* Se trata de un conjunto de herramientas necesarios para poder compilar y ejecutar programas, incluyendo Assolo.
- *Instalar cron:* En este caso es necesario instalar *cron* para que el mismo pueda activar el receptor y transmisor de Assolo al reiniciarse cada equipo. De esta forma ambos programas están siempre corriendo.
- *Habilitar cron:* Una vez instalado *cron* es necesario habilitarlo para que el mismo se ejecute al reiniciar cada equipo.
- *Crear Directorio de trabajo:* Se crea un directorio de trabajo donde se instala Assolo, este directorio es el mismo donde corren todos los otros archivos que forman parte de la plataforma. Si hay algún otro módulo ya corriendo en tal equipo, dicho directorio ya está creado.
- *Copiar Assolo:* Se transmite todo el código fuente de Assolo para poder compilarlo y ejecutarlo.
- *Descomprimir Assolo:* El código fuente del mismo se transmite en un archivo comprimido del tipo *.tar*, por lo tanto es necesario descomprimirlo para poder compilarlo.
- *Compilar Assolo:* Luego se compila dicho programa.
- *Habilitar Puertos para la IP del Servidor Central:* Como ya se dijo este programa necesita utilizar dos puertos para comunicarse con el otro equipo, el primero para que el transmisor pueda enviar paquetes y el otro para el receptor pueda recibirlos y enviar su respuesta. Por lo tanto es necesario modificar el firewall del equipo para que el mismo le permita comunicarse con el servidor central a través de dichos puertos.
- *Agregar Líneas a crontab:* Como se discutió en el capítulo anterior, el transmisor y receptor de Assolo deben estar todo el tiempo corriendo en cada equipo a la espera de que se desee tomar una medida. Esto se logra mediante el sistema *cron*, indicándole que comience a ejecutar dichos programas cada vez que se reinicia la computadora. Para que dicho programa realice esta acción es necesario modificar su archivo de configuración llamado *crontab*, agregándole unas líneas que le indiquen la tarea a realizar.
  Ansible se encarga de verificar que estas líneas estén escritas y las agrega en caso de no estarlo.
- *Dar permisos para ejecutar el Transmisor y Receptor de Assolo:* Para que *cron* sea capaz de ejecutar dichos programas, es necesario hacer que estos archivos sean ejecutables para el mismo.
- *Ejecutar el Transmisor y Receptor de Assolo:* Si bien *cron* se encarga de hacer correr dichos programas en el próximo reinicio de la computadora, puede que se quiera tomar una medida antes. Por lo tanto Ansible también se encarga de hacer correr dichos

programas al momento de instalar el módulo.

Una vez finalizada la ejecución de Ansible se procede a crear los *items* ancho de banda subida y ancho de banda bajada en Zabbix.

- *Item Errores:* En este punto se verifica que el *item* Errores encargado de recibir los errores de los distintos módulos esté creado para cada IP y si no lo está se lo crea.
- *Bloque -f:* Por último, en caso de que el usuario haya introducido una nueva frecuencia de ejecución, debe reescribirse la misma en *crontab*. Teniendo en cuenta que en este módulo el encargado de agendar las frecuencias es el sistema *cron*, el usuario debe introducir las mismas en su sintaxis. En primer lugar se borra la línea ya existente con el fin de que no quede duplicada y por último se procede a escribir la nueva. En caso de que la línea no haya sido introducida con la sintaxis correcta, se avisa al usuario de lo ocurrido y se finaliza la ejecución del *script*.

- *delassolo*: La función se encarga de deshabilitar el módulo de ancho de banda para uno o más equipos de la red Ceibal y/o detener totalmente su ejecución.

  Sus parámetros son los siguientes.

  - *-stop*: Al introducir este parámetro el módulo deja ejecutarse totalmente.
  - *-u*,*-y* y *-g*: Estos parámetros funcionan de la misma forma que en los módulos anteriores, permiten introducir una lista de IPs mediante línea de comandos, dar la ruta de un archivo de texto donde se encuentran escritas las IPs o seleccionar todas las IPs pertenecientes a cierto grupo de Zabbix.

  A continuación se describe el funcionamiento interno de la función. Los primeros cinco funcionan de la misma forma que en las funciones anteriores y son:

  - *Recepción y Procesamiento de los Parámetros*
  - *Autenticación en la API de Zabbix*
  - *Bloque -u*
  - *Bloque -y*
  - *Bloque -g*

  Los próximos bloques son característicos de esta función en particular.

  - *Eliminar IP de la lista:* Este bloque toma las IPs introducidas por el usuario y las borra del archivo que almacena la lista de IPs con las que se realiza la medida. Si alguna IP introducida no está en la lista, el programa la ignora y procede a eliminar la siguiente.

- *Bloque -stop:* En caso de que el usuario haya introducido este parámetro la función debe detener totalmente la ejecución del módulo. Dado que en este caso el encargado de dar la orden de ejecutar las medidas es el sistema *cron*, la forma de conseguir esto es modificar su archivo de configuración borrando la línea referente a este módulo.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 7

Aplicación Practica de la Plataforma de Monitoreo de Red en Laboratorio

En este capítulo se expone y describe cada paso de las pruebas realizadas para comprobar que la plataforma en efecto funciona como se describió en los capítulos anteriores. En primer lugar se describe el ambiente de prueba utilizado, posteriormente se ve un ejemplo del funcionamiento adecuado de cada módulo y luego se indujeron errores con el fin de demostrar que la misma responde a dichas fallas como fue previsto.

7.1. Ambiente de Prueba

El ambiente de prueba utilizado para esta tarea consiste en primer lugar de tres máquinas virtuales, las cuales residen en Plan Ceibal y cumplen el rol de servidor central. En la primera se instaló Zabbix, en la segunda Ansible y en la tercera llamada Monitoreo es donde se encuentran todos los programas desarrollados durante el proyecto, necesarios para poner en funcionamiento la plataforma y con los cuales tiene contacto el usuario. Estas máquinas virtuales se comunican a través de Internet con dos equipos Odroid-C2 los cuales cumplen el rol de los centros educativos. Los detalles de cada equipo se dan a continuación:

Sus características son:

- *Ansible:*
  - *Modelo:* Intel(R) Xeon(R) CPU E5-2609 v4 @ 1.70GHz
  - *Sistema operativo:* CentOS 7.7.1908
  - *CPU:* Intel(R) Xeon(R) CPU E5-2609 v4 @ 1.70GHz
  - *RAM:* 6 GB
  - *Conectividad:* 1 Gigabit Ethernet
  - *Almacenamiento:* Disco duro virtual 90 GB
- *Zabbix:*

  - *Modelo:* Intel(R) Xeon(R) CPU E5-2609 v4 @ 1.70GHz
  - *Sistema operativo:* CentOS 7.7.1908
  - *CPU:* Intel(R) Xeon(R) CPU E5-2609 v4 @ 1.70GHz
  - *RAM:* 20 GB
  - *Conectividad:* 1 Gigabit Ethernet
  - *Almacenamiento:* Disco duro virtual 800 GB
- *Monitoreo:*
  - *Modelo:* Intel(R) Xeon(R) CPU E5-2609 v4 @ 1.70GHz
  - *Sistema operativo:* CentOS 7.7.1908
  - *CPU:* Intel(R) Xeon(R) CPU E5-2609 v4 @ 1.70GHz
  - *RAM:* 6 GB
  - *Conectividad:* 1 Gigabit Ethernet
  - *Almacenamiento:* Disco duro virtual 70 GB
- *Dos equipos Odroid-C2:*
  - *Modelo:* Odroid-C2
  - *Sistema operativo:* Ubuntu Mate
  - *CPU:* Amlogic ARM® Cortex®-A53(ARMv8) 1.5 GHz Quad Core
  - *RAM:* 2 GB
  - *Conectividad:* Gigabit Ethernet
  - *Almacenamiento:* Tarjeta SD de 14 GB

7.2. Acondicionamiento

En esta sección se mencionan los cambios a realizar en los equipos del servidor central para poder correr cada una de las funciones implementadas.

- *Equipo Monitoreo:* Fue necesario instalar los paquetes Paramiko y Psutils de Python, el comando *at*, habilitar el sistema *cron*, crear la carpeta /root/moniceibal y finalmente transmitir todos los archivos necesarios para el funcionamiento de la plataforma a dicho directorio.
- *Equipo Ansible:* Para este equipo se realizaron dos cambios, se creó el directorio /root/moniceibal, que es donde se almacenan los archivos que el mismo debe distribuir y se creó una clave compartida con cada uno de los equipos a monitorizar, para que el mismo pueda conectarse sin contraseña.
- *Equipo Zabbix:* En este caso sólo se necesitó registrar ambos equipos Odroid-C2 que se utilizaron para la toma de medidas en la lista de *hosts* de Zabbix. Los nombres de dichos *hosts* son odroid_black y odroid_black_2.

7.3. Tiempo DHCP

7.3.1. Crear Módulo

A continuación se describen los pasos realizados para crear el módulo de tiempo DHCP en dos equipos usando la función *adddhcp.py* y luego se muestran las capturas de pantalla que comprueban su correcto funcionamiento. El primer paso consiste en situarse en la carpeta /root/moniceibal del equipo Monitoreo y ejecutar el siguiente comando:

```
./adddhcp.py -u 167.57.37.17 167.57.202.31 -f 30
```

La dirección 167.57.37.17 corresponde al *host* odroid_black y la 167.57.202.31 al *host* odroid_black_2. Ha de notarse que se usó la opción *-u* como método para introducir una lista de IPs, aunque también se realizaron los mismos experimentos con las otras dos formas de introducir IPs. Dichos resultados no son mostrados aquí ya que pasada la etapa de introducción de parámetros los resultados son exactamente los mismos.

Una vez finalizada la ejecución del *script* debe verificarse que cada uno de los cambios realizados por el mismo hayan surtido efecto, lo cual se muestra a continuación.

- *Agregar IPs a inventario de Ansible:* Una vez procesados los parámetros, el *script* procede a establecer una conexión *ssh* con el equipo Ansible y registrar las IPs introducidas en su inventario con el nombre de usuario correspondiente. La figura 7.1 muestra una captura de pantalla de dicho archivo donde se señalan ambas IPs con la meta de probar que esta tarea se realizó con éxito.

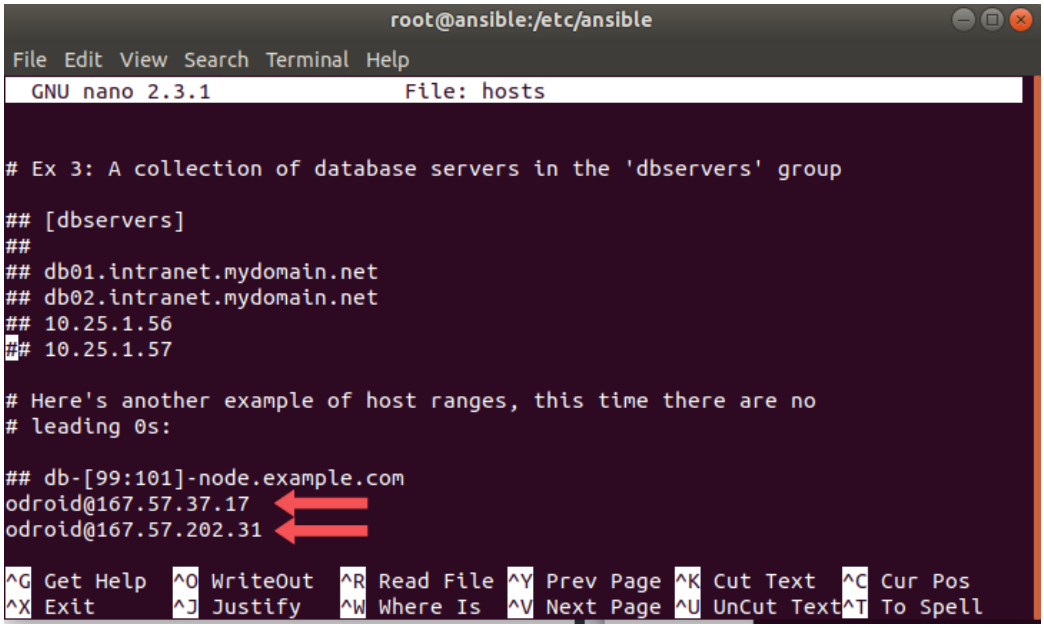


Figura 7.1: IPs agregadas al inventario de Ansible para el módulo de tiempo DHCP.

- *Carpeta de trabajo en el equipo de Ansible:* Una vez agregadas las IPs al inventario, el *script* debe transferir todos los archivos necesarios a la carpeta de trabajo de Ansible. En este caso los archivos son: el *playbook crear_item.yml* encargado de realizar todos los cambios necesarios en los equipos remotos para poder correr los módulos y el *script* dhcp_time que es el encargado de tomar las medidas y debe ser transmitido a cada *host*. En la figura 7.2 se puede ver como se utiliza el comando ls de Linux para demostrar que dichos archivos en efecto se encuentran en dicha carpeta.

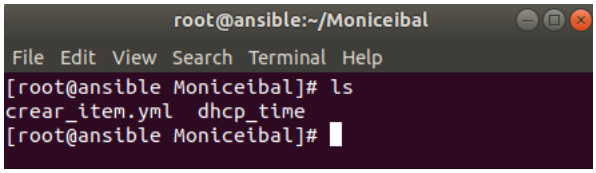


Figura 7.2: Archivos en la carpeta /root/moniceibal del equipo Ansible para el módulo de tiempo DHCP.

- *playbook de Ansible:* Luego es necesario comprobar que Ansible logró ejecutar todas las tareas encomendadas mediante el *playbook crear_item.yml* sin problemas y por ende logró realizar todos los cambios necesarios en los *hosts*. Cuando a Ansible se le da una lista de tareas a realizar, el mismo las ejecuta en el orden especificado en el archivo yaml. Además imprime en consola si se concretó correctamente o no las tareas especificadas. En la figura 7.3 puede observarse cómo el mismo fue capaz de realizar todas las acciones deseadas,

indicando *failed = 0*, es decir fallas igual cero, señalado con una flecha de color rojo. También es posible visualizar que por cada tarea realizada imprime en consola el mensaje *ok*.

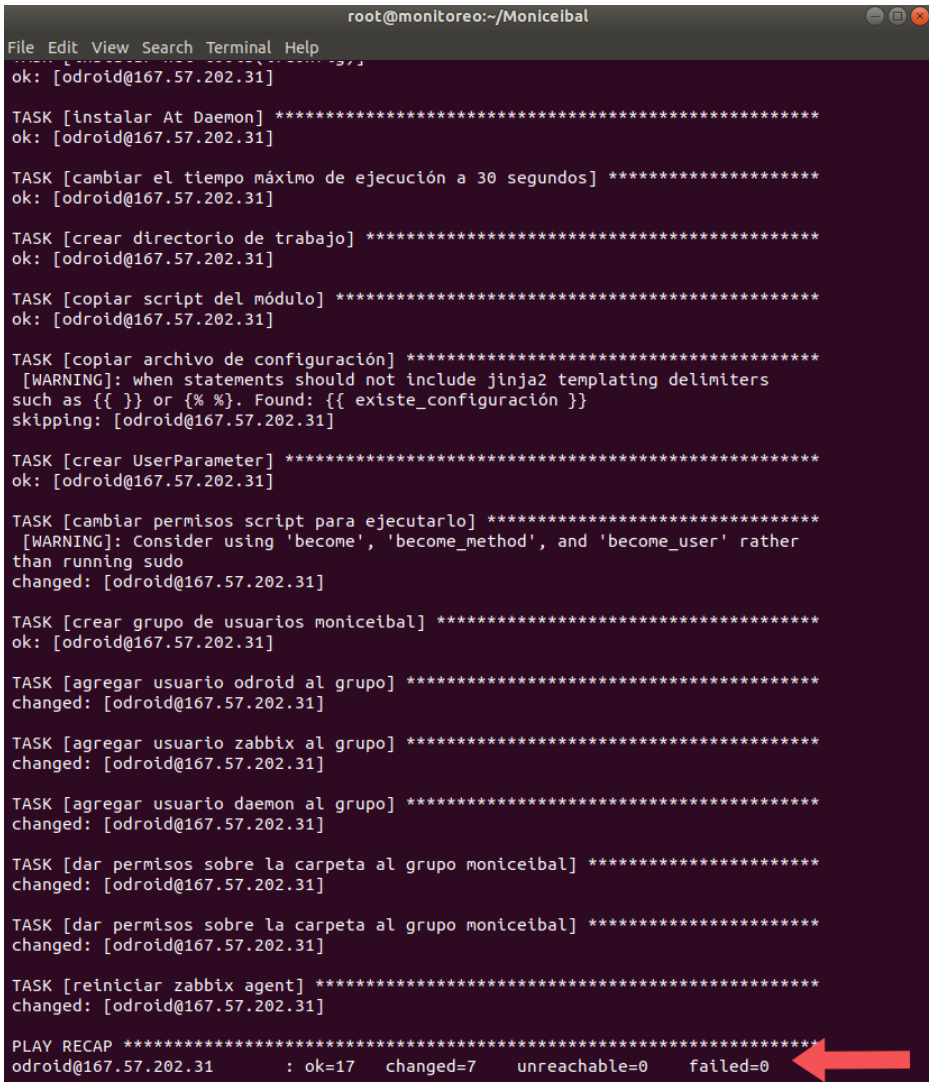


Figura 7.3: Ejecución de *playbook* de Ansible para el módulo de tiempo DHCP.

- *Programas instalados en los hosts:* Si la ejecución del *playbook* de Ansible es exitosa entonces debe poder comprobarse que los programas que el mismo instala en cada *host* se encuentran instalados. Para comprobar esto se realiza una conexión *ssh* con el *host* y se utiliza el comando *APT list –installed* el cual lista los programas instalados en el equipo, a la vez también se utiliza el comando *grep* con la meta de no ver todos los programas instalados, sino sólo los que interesan en este situación. La figura 7.4 muestra el resultado de dicho proceso.

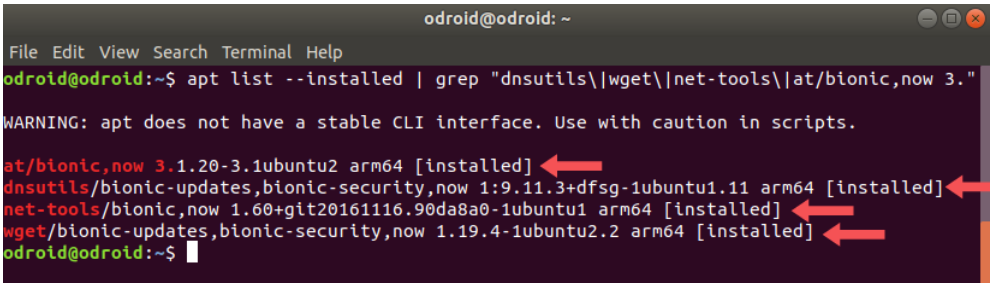


Figura 7.4: Programas instalados en el *host* odroid_black.

- *Archivo de configuración del agente de Zabbix:* El siguiente cambio que Ansible debe realizar en los *hosts* es modificar el archivo de configuración del agente Zabbix. En primer lugar el tiempo límite que Zabbix da a los comandos para ejecutarse debe extenderse a 30 s. Luego se debe agregar la línea UserParameter al archivo de configuración con la información del nuevo módulo. La figura 7.5 muestra una captura de pantalla del archivo, la cual comprueba que dichos cambios en efecto han tomado lugar, dichos cambios están señalados con flechas rojas.

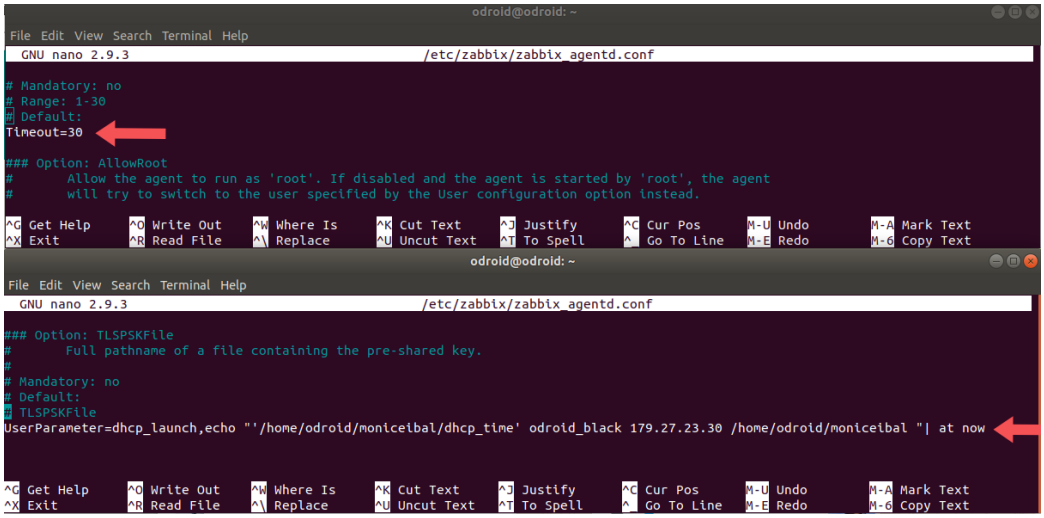


Figura 7.5: Archivo de configuración del agente de Zabbix en el *host* odroid_black para el módulo de tiempo DHCP.

- *Directorio moniceibal en el host:* Por otro lado, debe crearse el directorio moniceibal con el *script* del módulo dhcp_time dentro en cada *host*. La figura 7.6 muestra la existencia de la carpeta y su contenido.

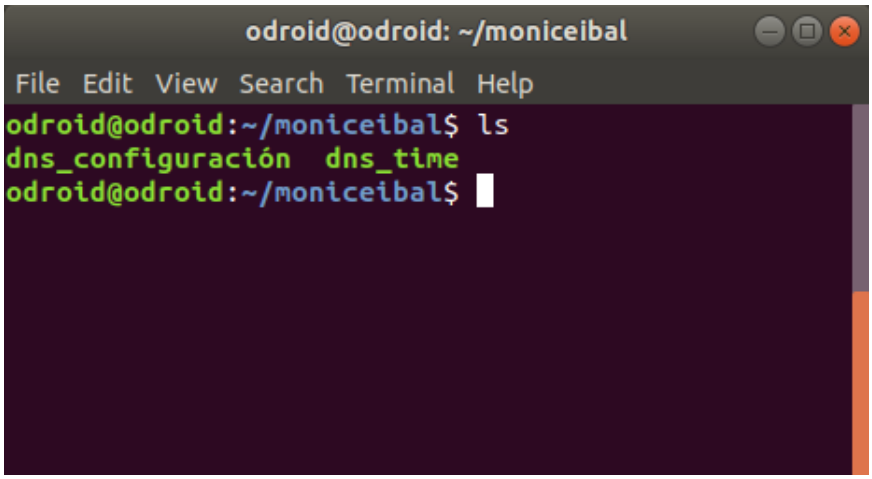


Figura 7.6: Contenido de la carpeta moniceibal en el *host* odroid_black para el módulo de tiempo DHCP.

- *Propietarios del directorio moniceibal:* Como se dijo en figura 6.2, el directorio moniceibal dentro del *host* debe tener tres propietarios. Ser propietario de una carpeta en Ubuntu es útil para poder realizar los cambios necesarios en ella sin problemas. El primer usuario que debe ser acreedor de la carpeta es el que utiliza Ansible para loguearse en el equipo y guardar archivos en la misma, en este caso se trata del usuario odroid. Los otros dos usuarios que también necesitan realizar cambios en la carpeta son zabbix y daemon, esto se debe a que los módulos al ejecutarse pueden necesitar crear archivos y luego borrarlos. Para lograr que más de un usuario sea dueño de una carpeta en Ubuntu es necesario primero crear un grupo de usuarios y luego hacer que la carpeta sea propiedad de ese grupo. En la figura 7.7 puede verse cómo se usa el comando *ls -ld* el cual permite saber a qué grupo pertenece el directorio, el nombre del grupo aparece subrayado en rojo.

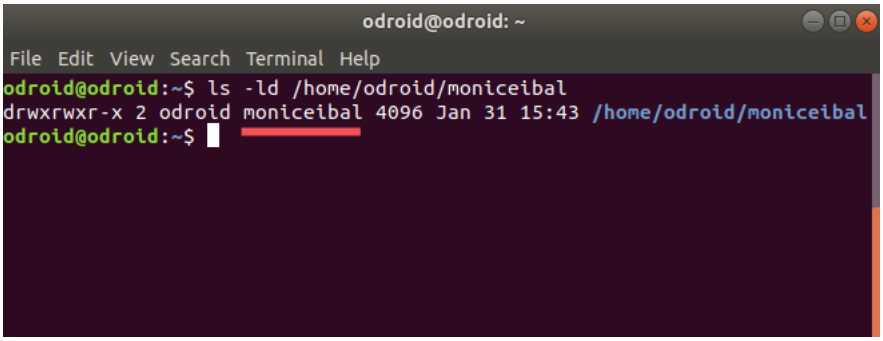


Figura 7.7: Grupo propietario de la carpeta moniceibal en el *host* odroid_black.

En la figura 7.8 se utilizó el comando *getent group* para verificar que los integrantes del grupo fueran en efecto los usuarios, odroid, zabbix y daemon.

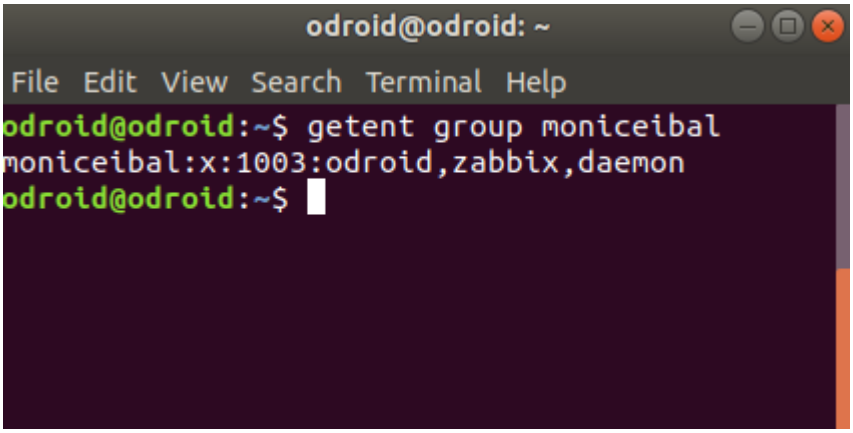


Figura 7.8: Integrantes del grupo Moniceibal en el *host* odroid_black.

- *Items Creados en el Servidor de Zabbix:* La figura 7.9 muestra cómo los tres *items* necesarios para el funcionamiento del módulo han sido creados en el servidor de Zabbix para el *host* nombrado odroid_black. El *item* dhcp_launch es el encargado de dar las órdenes de ejecución y es quien debe almacenar la frecuencia con la que se toma la medida. Se puede ver subrayado en rojo cómo esta frecuencia es en efecto 30 s, tal y como se introdujo al comienzo desde la línea de comandos. También se puede ver cómo se creó el *item dhcp_trapper* que es el encargado de recibir los datos y el *item* Errores el cual recibe los errores en caso de haberlos.

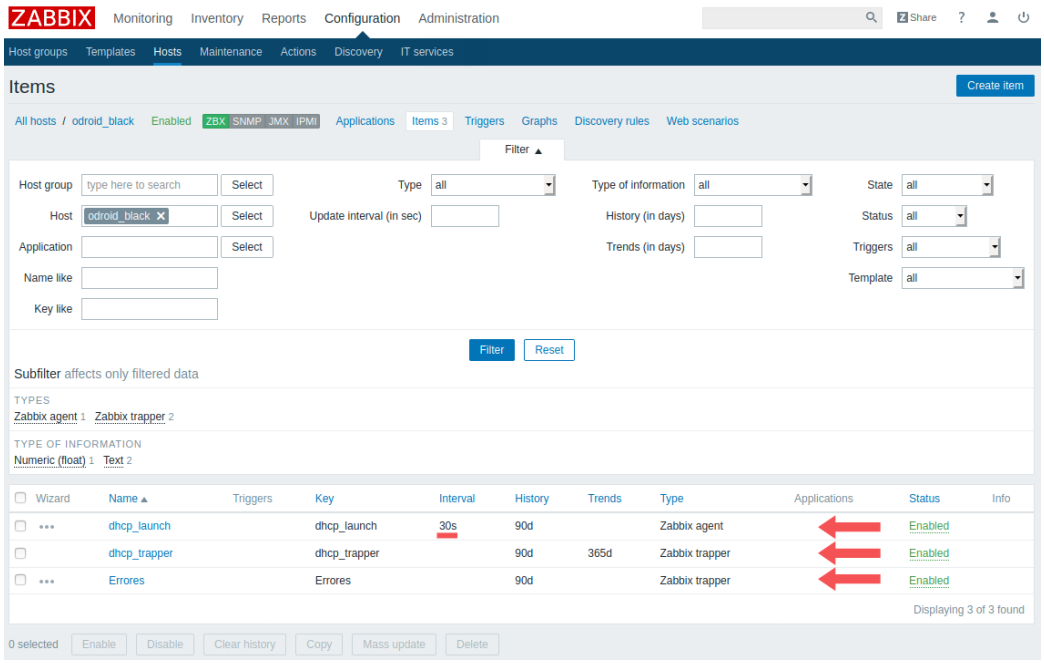


Figura 7.9: Items creados en el *host* odroid_black para el módulo de tiempo DHCP.

Por último la figura 7.10 incluye una captura de los módulos efectivamente funcionando en cada *host* y los datos que estos obtienen en el *item dhcp_trapper*.

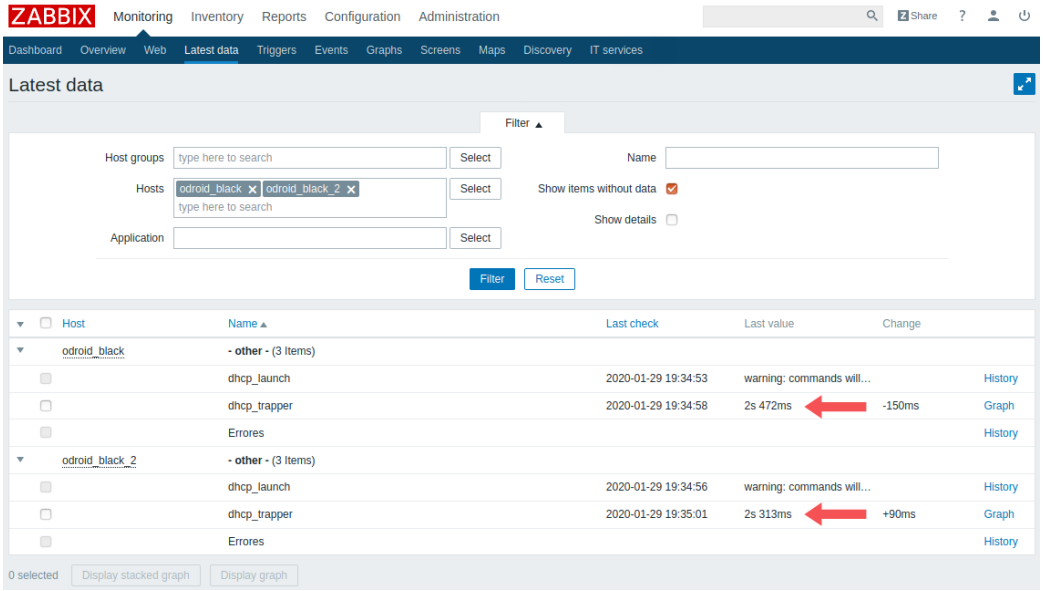


Figura 7.10: Módulo DHCP funcionando.

7.3.2. Deshabilitar Módulo

Con esta prueba se busca validar que la función *deldhcp.py* encargada de deshabilitar el módulo de medición de tiempo DHCP realmente cumple su trabajo. Como se mencionó anteriormente lo que hace esta función mediante el uso de la API de Zabbix es deshabilitar el *item* que da la orden de ejecución al módulo DHCP, en este caso ese *item* es dhcp_launch.

Para probarlo nos situamos en la carpeta /root/moniceibal del equipo Monitoreo y ejecutamos el siguiente comando:

```
./deldhcp.py -u 167.57.37.17 167.57.202.31
```

En la figura 7.11 podemos ver como el *item* dhcp_launch fue en efecto deshabilitado. Es importante recordar que esta función no borra el *script* dhcp_time ni su archivo de configuración, sólo deshabilita el *item*.

Ha de destacarse que la validación de las funciones encargadas de deshabilitar los módulos de tiempo DNS y tiempo de descarga de sitio web son omitidas en este documento ya que las mismas tienen exactamente el mismo código fuente que esta función con la única diferencia de que el *key* del *item* a deshabilitar es diferente.

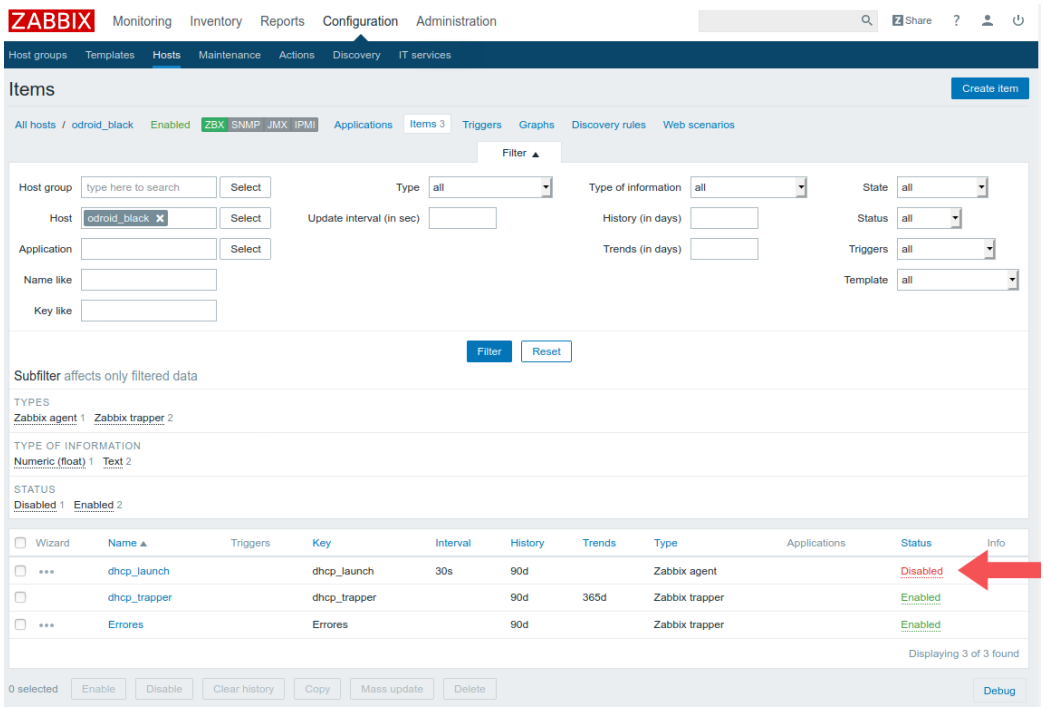


Figura 7.11: Módulo Tiempo DHCP deshabilitado.

7.3.3. Errores inducidos

Cuando se le pide una IP a un servidor DHCP puede que por varios motivos el mismo no responda, ya sea porque no le queden IPs disponibles o el servicio esté deshabilitado, entre otras cosas. Es por esto que el *script* desarrollado en este módulo tiene programada una cota de 20 s para esperar una respuesta, pasado este tiempo el mismo finaliza su ejecución y manda un mensaje de error al *item* Errores de Zabbix.

Con el fin de verificar que esto realmente se cumple se realizaron dos experimentos distintos, primero se deshabilitó el servicio DHCP del *router* al cual se encontraba conectado el equipo y luego se lo configuró para que ya no tuviera más IPs libres que otorgar. En ambos casos se cumplió que pasaron 20 s, no se obtuvo respuesta y por lo tanto el módulo envió el mensaje correspondiente al servidor de Zabbix como se lo puede ver en la figura 7.12.

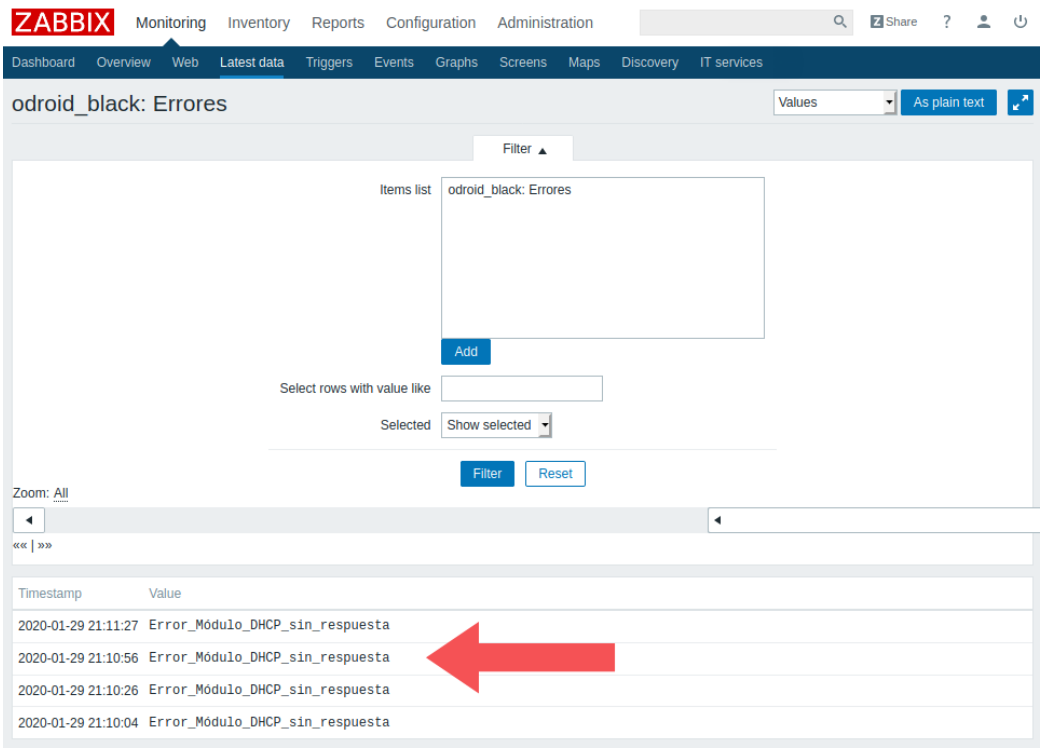


Figura 7.12: Mensaje de error del módulo de tiempo DHCP en Zabbix.

7.4. Tiempo DNS

7.4.1. Crear Módulo

En esta sección se crea el módulo tiempo DNS en dos equipos con el fin de comprobar el funcionamiento de la función *adddns.py* y se exponen capturas de pantalla que validan el funcionamiento de la misma. Para esto con la terminal situada en la carpeta /root/moniceibal se ejecuta el siguiente comando:

```
./adddns.py -u 167.60.246.100 167.60.224.231 -s www.google.com
www.yahoo.com www.youtube.com www.twitter.com -f 30
```

En este caso la dirección 167.60.246.100 corresponde al *host* odroid_black y la 167.60.224.231 al *host* odroid_black_2. Al igual que en el módulo anterior, sólo se exponen los resultados del experimento realizado ingresando las IPs mediante la opción *-u* ya que los otros resultados son análogos.

A continuación se explica cada paso realizado para comprobar que dicha función fue capaz de cumplir con la tarea de crear el módulo en los equipos.

- *Agregar IPs a inventario de Ansible:* El primer paso es verificar que las IPs ingresadas fueron efectivamente agregadas al inventario de Ansible. Dado que la parte del código que realiza esta tarea es exactamente igual a la de

la función del módulo anterior, la captura de pantalla que comprueba que dichas IPs fueron agregadas luce como en la figura 7.1.

- *Carpeta de trabajo en el equipo de Ansible:* En este paso se verifica que los archivos necesarios hayan sido efectivamente transmitidos al equipo de Ansible. Estos archivos son: el *playbook configuración_dns.yml*, el cual es utilizado para transmitir el archivo de configuración a los *hosts*, el archivo en cuestión configuración_dns, el *playbook* crear_item.yml encargado de crear el módulo y el *script* dns_time encargado de tomar las medidas. En la figura 7.13 se puede ver que los archivos realmente se encuentran en el equipo Ansible.

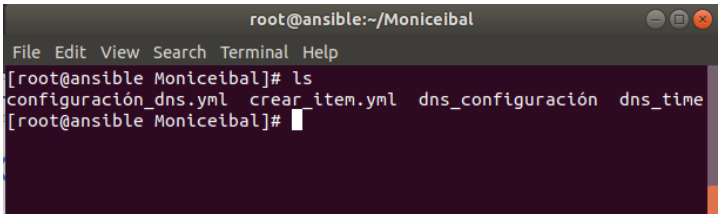


Figura 7.13: Archivos en la carpeta /root/moniceibal del equipo Ansible para el módulo de tiempo DNS.

- *Archivo de configuración:* Es importante comprobar que el archivo de configuración (configuración_dns), el cual es transmitido a todos los *hosts*, realmente contiene los *hostnames* introducidos mediante la línea de comandos. La figura 7.14 muestra una captura del contenido de dicho archivo.

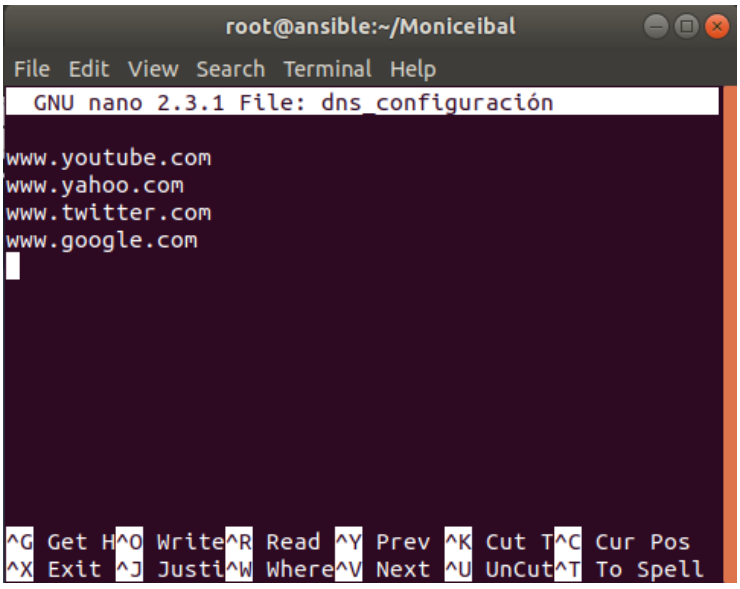


Figura 7.14: Archivos de Configuración en la carpeta /root/Moniceibal del equipo Ansible para el módulo de tiempo DNS.

- *playbook de Ansible:* La siguiente captura de pantalla muestra cómo Ansible logró ejecutar el *playbook crear_item.yml* sin ningún error y pudo realizar todos los cambios requeridos.

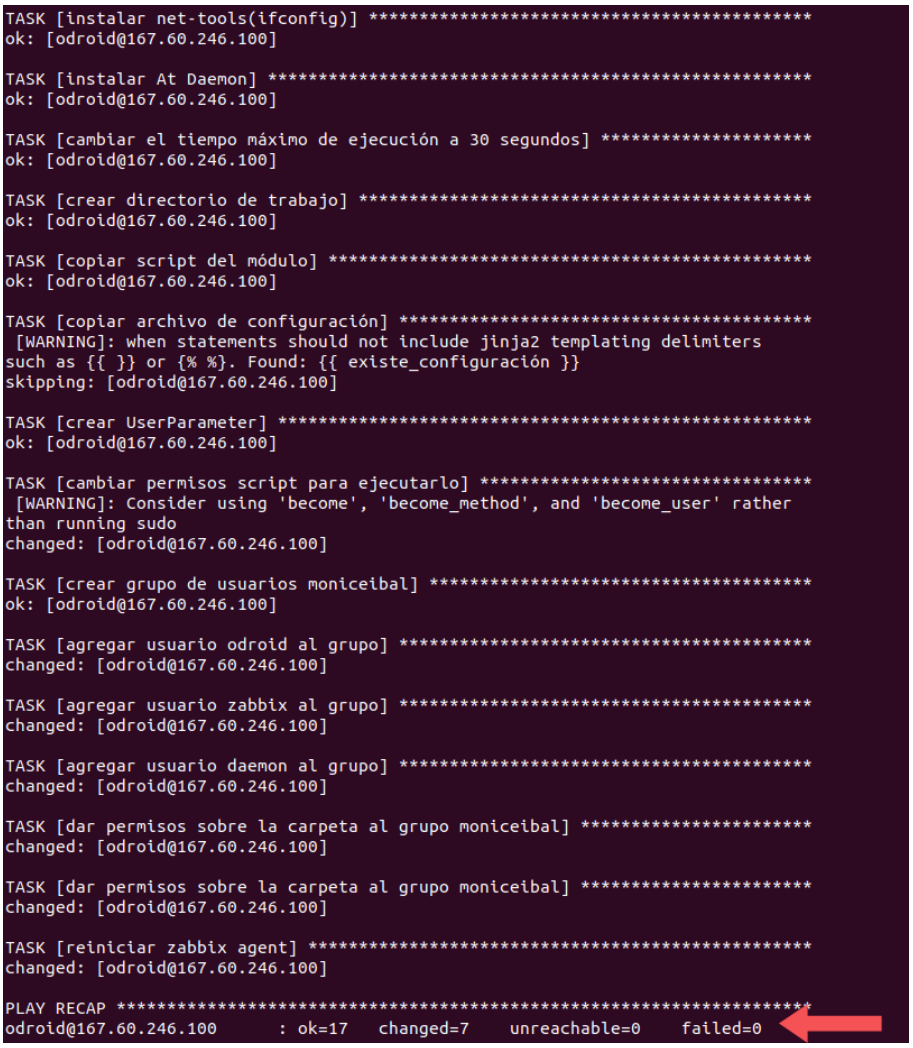


Figura 7.15: Ejecución de *playbook* de Ansible para el módulo de tiempo DNS.

- *Programas instalados en los hosts:* Dado que los programas a instalar por este *playbook* son los mismos que en el módulo anterior, la captura de pantalla que comprueba que los mismos han sido instalados luce de la misma forma que en la figura 7.5.

- *Archivo de configuración del agente de Zabbix:* Los cambios a realizar en el archivo de configuración del agente de Zabbix en el *host* son los mismos que en el módulo anterior, extender el tiempo límite de ejecución para los *items* y crear el nuevo UserParameter. Este resultado se muestra en la figura 7.16.

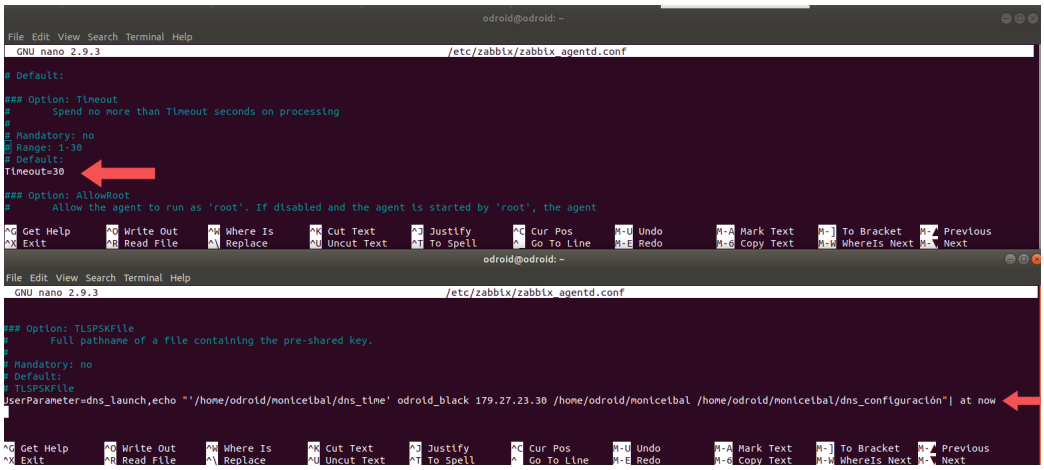


Figura 7.16: Archivo de configuración del agente de Zabbix para el módulo de tiempo DNS.

- *Directorio moniceibal en el host:* Por otro lado, debe crearse el directorio moniceibal en cada *host* con los archivos dentro de la carpeta. Estos archivos son el *script* encargado de tomar la medida y el archivo de configuración del cual se leen los parámetros necesarios para tomarlas. La figura 7.17 muestra la existencia de la carpeta y su contenido.

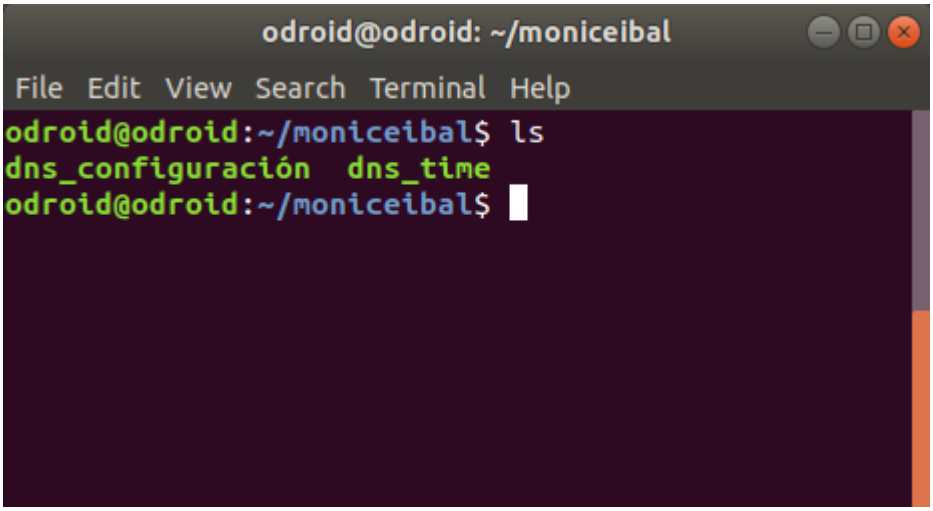


Figura 7.17: Contenido de la carpeta moniceibal en el *host* odroid_black para el módulo de tiempo DNS.

- *Propietarios del directorio moniceibal:* Al igual que en el módulo anterior, el propietario del directorio moniceibal en los *hosts* debe ser el grupo Moniceibal, constituido por los mismos usuarios. La capturas de pantalla que comprueban que este cambio fue logrado lucen de la misma forma que en el módulo anterior en la figura 7.7 y figura 7.8.
- *Items Creados en el Servidor de Zabbix:* En cuanto al servidor de Zabbix se tienen que crear tres *items*, dns_launch, encargado de dar la orden de ejecución, el *item dns_trapper* encargado de recibir la medida y el *item* Errores encargado de recibir los errores. La figura 7.18 muestra cómo los mismos fueron efectivamente creados y con la frecuencia introducida.

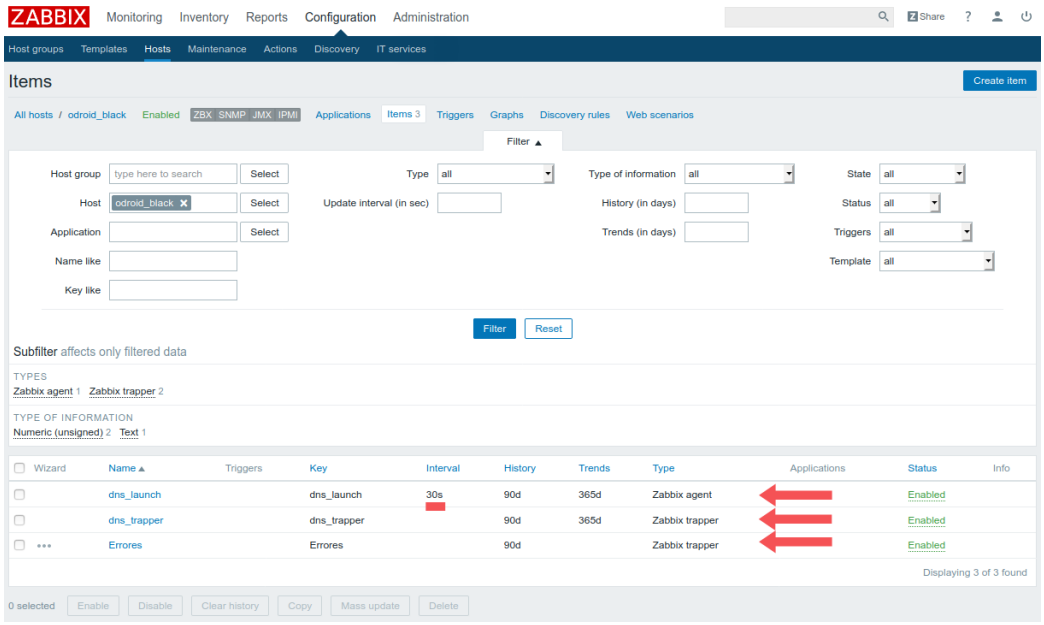


Figura 7.18: Items creados en el *host* odroid_black para el módulo de tiempo DNS.

Por último la figura 7.19 incluye una captura de los módulos efectivamente funcionando en cada *host* y los datos que estos obtienen en el *item dns_trapper*.

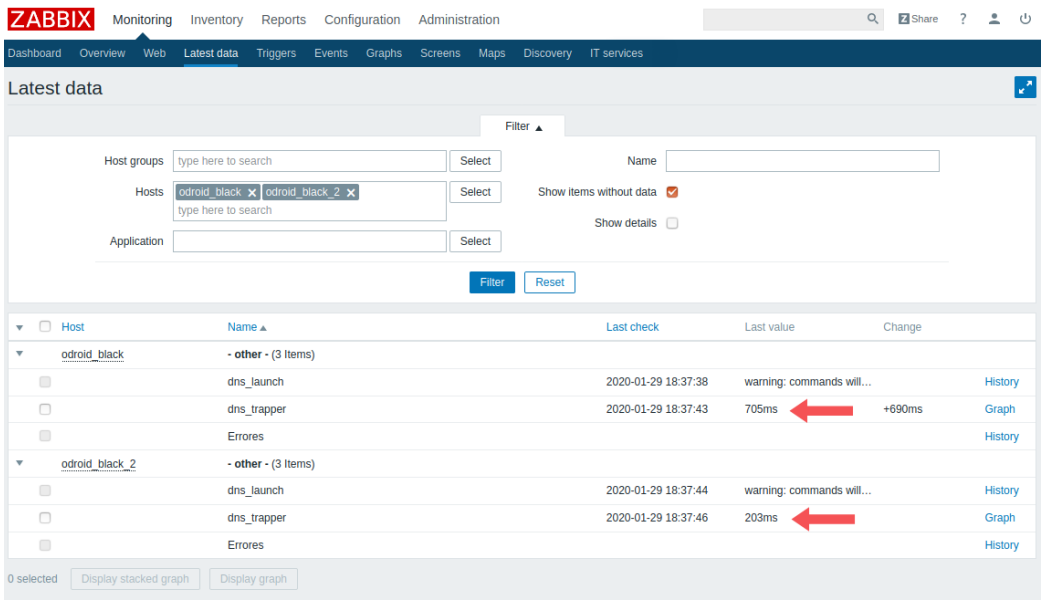


Figura 7.19: Módulos tiempo DNS funcionando.

7.4.2. Errores inducidos

Cuando el usuario introduce un nuevo *hostname* para realizar la medida de tiempo DNS, la plataforma comprueba que dicha dirección realmente corresponda a un sitio web. Aún puede ocurrir que tiempo después de introducida y comprobada la misma, el sitio web deje de existir. En este caso el módulo debe enviar un mensaje al *item* errores de Zabbix explicando lo ocurrido. Para comprobar el funcionamiento del módulo en esta situación se modificó manualmente el archivo de configuración del mismo en uno de los *hosts* remplazando uno de los *hostnames* por la dirección inexistente www.adsfadsfadsf.com como se puede ver en la figura 7.20.

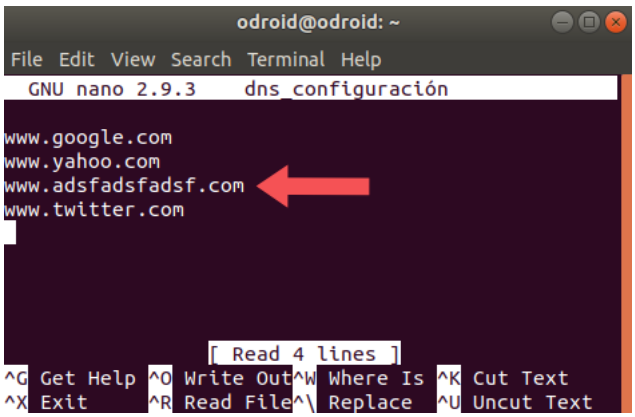


Figura 7.20: Error inducido en el archivo de configuración de DNS.

Dado que dicho *hostname* se encuentra en el tercer lugar de la lista, el módulo envió un mensaje al *item* Errores de Zabbix explicando que el *hostname* número tres no existe. Esto se puede comprobar en la figura 7.21.

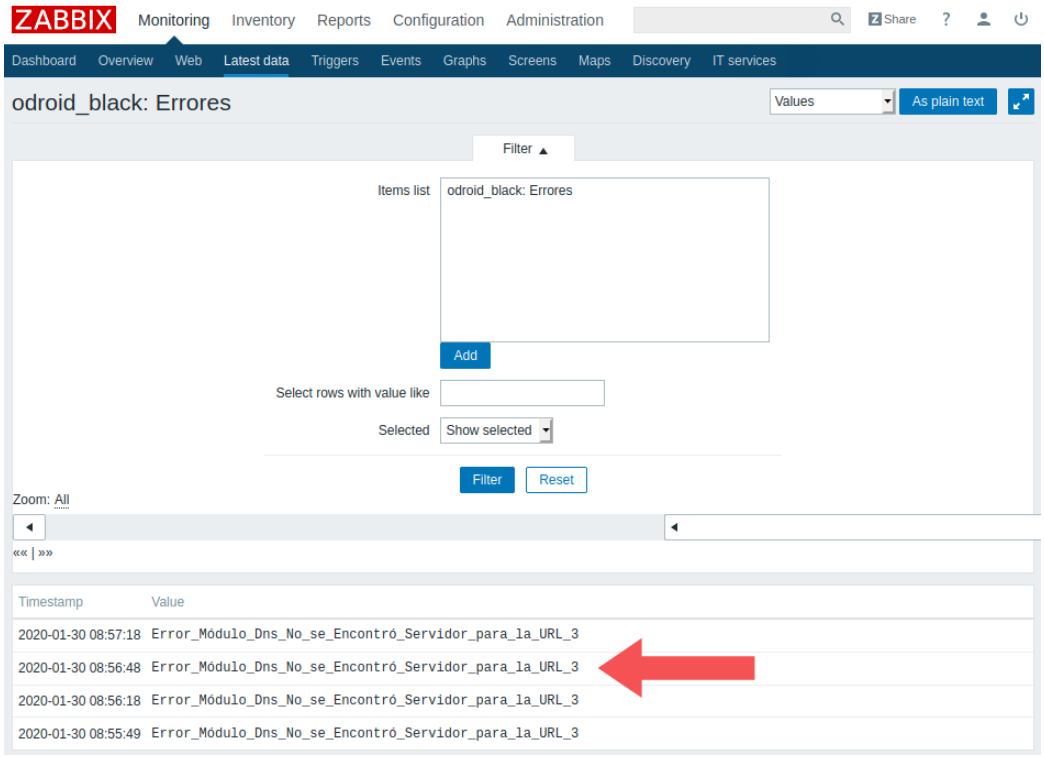


Figura 7.21: Mensaje de error del módulo de tiempo DNS en Zabbix.

7.5. Tiempo de Carga de Sitio Web

7.5.1. Crear Módulo

Para comprobar el funcionamiento de esta función el primer paso es situarse en la carpeta /root/moniceibal del equipo Monitoreo y luego ejecutar el siguiente comando:

```
./addwget -u 167.60.246.100 167.60.224.231 -s www.google.com -m -f 30
```

Donde la dirección 167.60.246.100 corresponde al *host* odroid_black y la 167.60.224.231 al odroid_black_2. El proceso de creación de este módulo es esencialmente igual al del módulo tiempo DNS con la salvedad de que su archivo de configuración es diferente. Por lo tanto se muestra en la figura 7.22 que dicho archivo fue copiado con éxito al equipo de Ansible, que contiene el *hostname* y la opción de *wget* ingresada. Luego se pasa directamente a demostrar que los *items* han sido creados en el servidor de Zabbix y están funcionando correctamente.

Otra diferencia es que el UserParameter creado en el archivo de configuración de los agentes debe tener un *key* diferente, salvo esto la línea debe lucir exactamente igual.

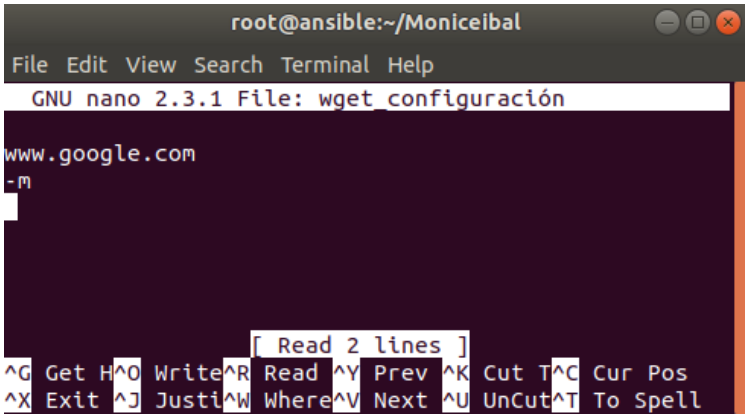


Figura 7.22: Archivos de configuración en la carpeta /root/moniceibal del equipo Ansible para el módulo de tiempo de descarga de sitio web.

- *Items Creados en el Servidor de Zabbix:* Al igual que en los módulos anteriores deben crearse tres *items* en Zabbix, el *item* wget_launch, el *item* *wget_trapper* y el *item* Errores, los cuales cumplen las mismas funciones que en los casos anteriores y se pueden ver la figura 7.23.

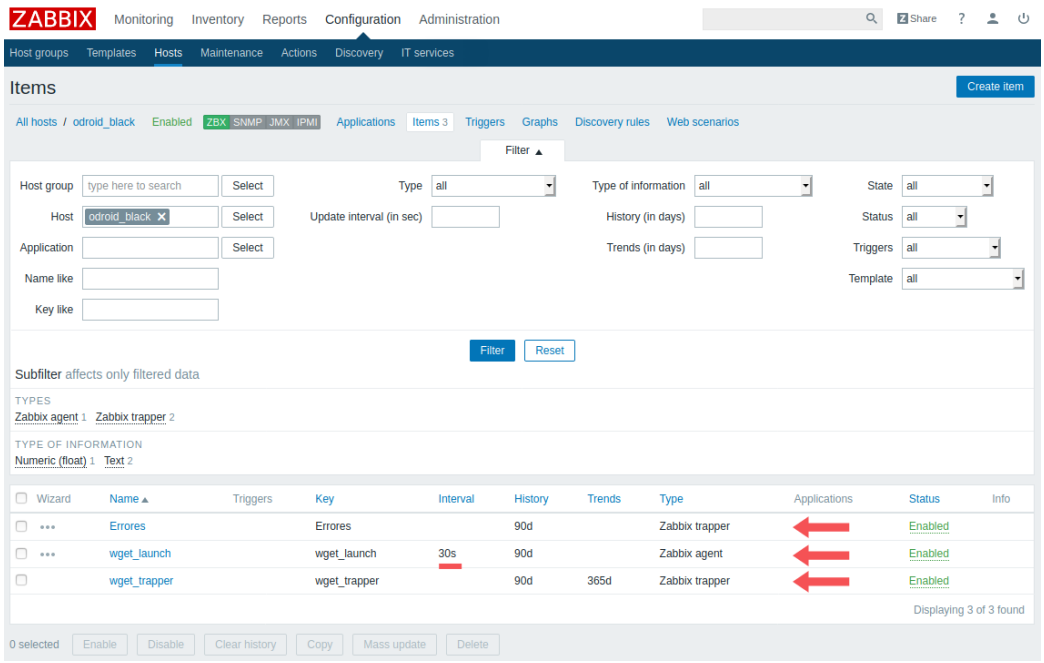


Figura 7.23: *items* creados en el *host* odroid_black para el módulo de tiempo de descarga de sitio web.

Finalmente la figura 7.24 muestra el módulo en funcionamiento para los dos *hosts*, como se puede ver el *item* *wget_trapper* es quien recibe los datos recolectados.

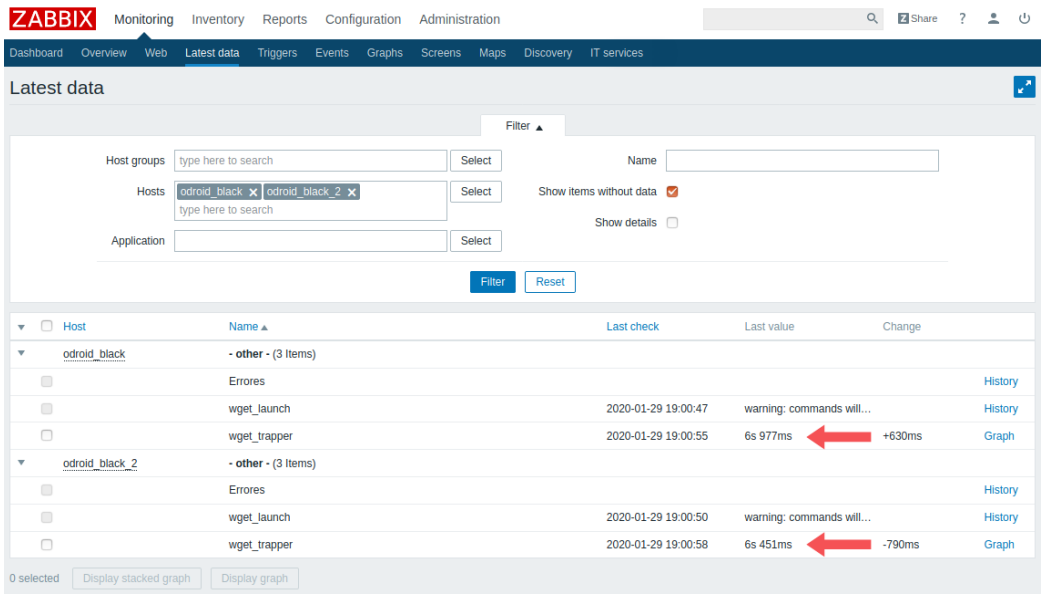


Figura 7.24: Módulo de tiempo de descarga de sitio web funcionando.

7.5.2. Errores inducidos

Al igual que en módulo anterior, la plataforma verificará que el *hostname* con que el usuario desea tomar la medida exista al momento de la introducción de la misma, pero esto no quita que pasado el tiempo dicho sitio web deje de existir. Para comprobar la respuesta del módulo ante este problema, se modificó el archivo de configuración del mismo en un *host*, introduciendo el *hostname* inexistente www.adsfadsfadsf.com como muestra la figura 7.25. En consecuencia el *item* Errores de Zabbix recibió el mensaje mostrado en la figura 7.26.

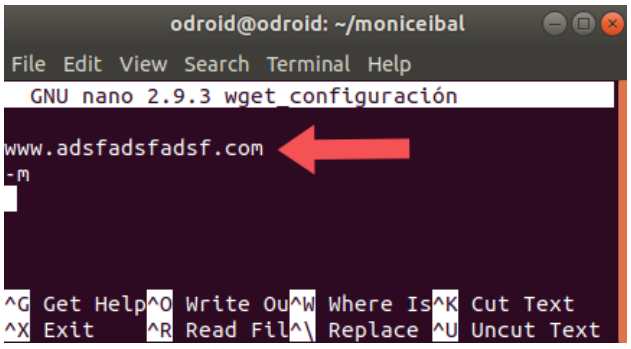


Figura 7.25: Archivo de configuración del módulo tiempo de descarga del sitio web, modificado para inducir errores.

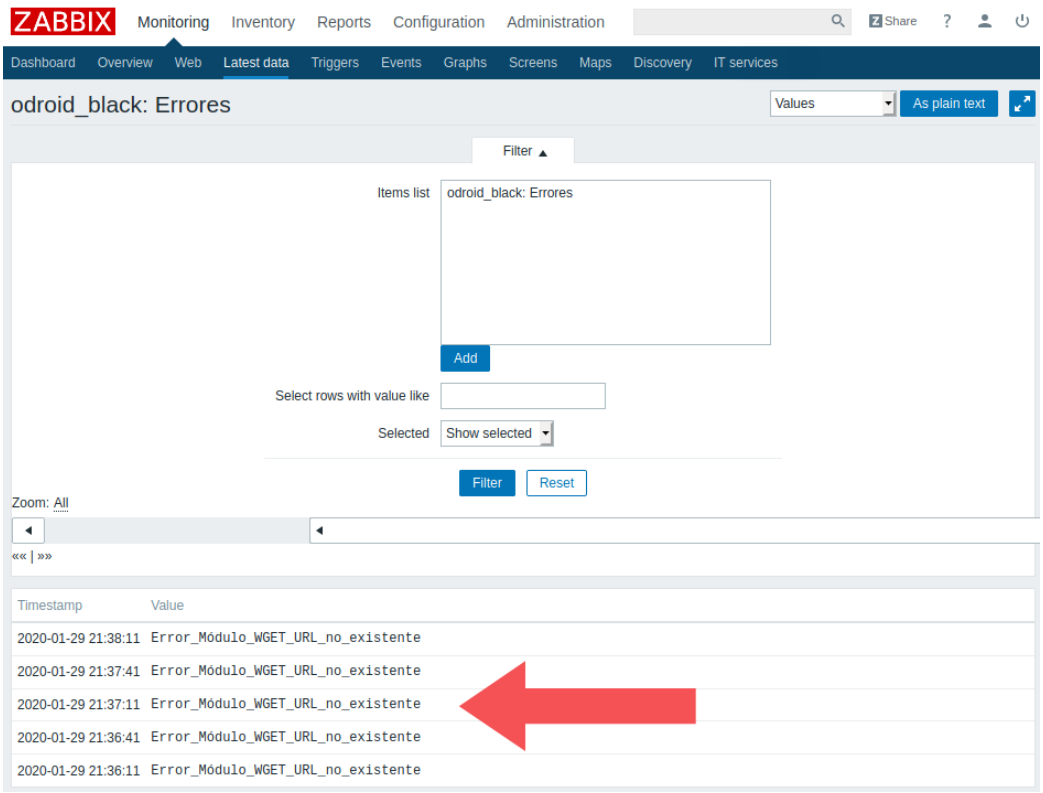


Figura 7.26: Mensaje de error de *hostname* que no existe, del módulo de tiempo de descarga de sitio web en Zabbix.

El otro error posible en este módulo es que por varios motivos la descarga continúe por tiempo indefinido o exageradamente largo, ya sea por algún problema en el servidor o por la velocidad a la que se está realizando la misma. Para alertar al sistema de este tipo de situaciones, el *script* impone un tiempo máximo de descarga de 5 minutos, pasado este lapso se envía un mensaje de error al servidor de Zabbix.

Con el fin de comprobar esto, se limitó manualmente la velocidad de descarga para que esta excediera los 5 minutos, en consecuencia el servidor de Zabbix recibió el mensaje correspondiente como se muestra en la figura 7.27.

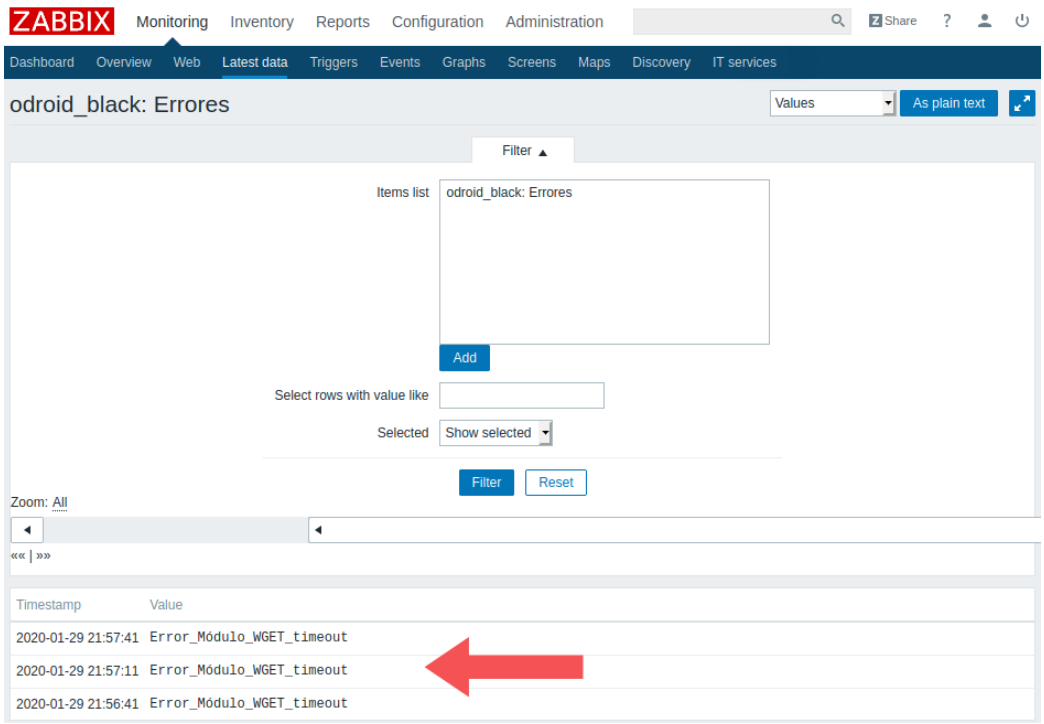


Figura 7.27: Mensaje de error de *timeout* del módulo de tiempo de descarga de sitio web en Zabbix.

7.6. Ancho de Banda

7.6.1. Crear Módulo

Al igual que para todos los módulos anteriores el primer paso es acondicionar los equipos Ansible, Zabbix y Monitoreo, luego situarse en la carpeta /root/moniceibal del equipo Monitoreo y ejecutar el siguiente comando:

```
./addassolo.py -u 167.57.33.41 190.134.232.247 -f ''*/10 * * * *'' -t 20
```

Donde la IP 167.57.33.41 corresponde al *host* odroid_black, la IP 190.134.232.247 corresponde al *host* odroid_black_2, la frecuencia "*/10 * * * *" corresponde a una frecuencia de medición de cada 10 minutos y el número 20 corresponde al lapso de tiempo que dura cada medida.

- *Crear archivos de configuración:* Después de procesados los parámetros el programa crea dos archivos en el computador Monitoreo, que almacenan información introducida por el usuario. El primero se puede ver en la figura 7.28, este contiene la lista de IPs contra las cuales se tomarán las medidas. El segundo, visible en la figura 7.29 contiene el tiempo que tomará cada medida.

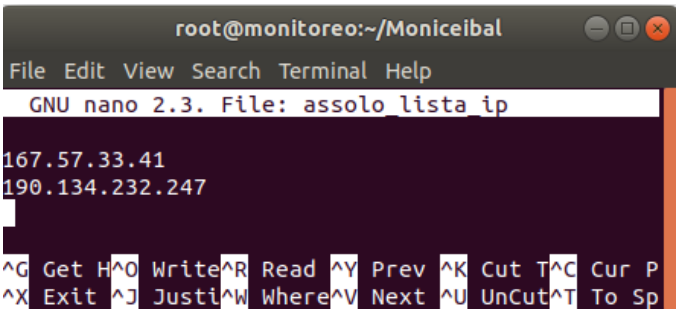


Figura 7.28: Archivo de configuración con la lista de IPs.

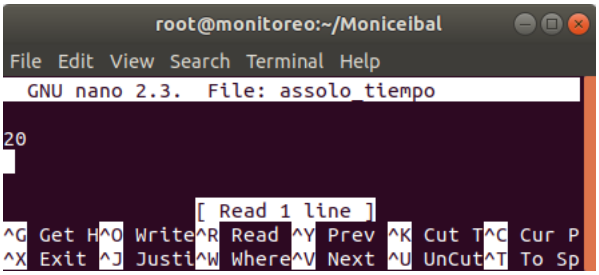


Figura 7.29: Archivo de configuración con el tiempo de medición.

- *Assolo en crontab de Monitoreo:* Otro cambio que debe realizarse en Monitoreo es agregar a su archivo *crontab* una línea que incluya la nueva frecuencia de medición. Como se puede ver en la figura 7.30, la operación se realizó con éxito.

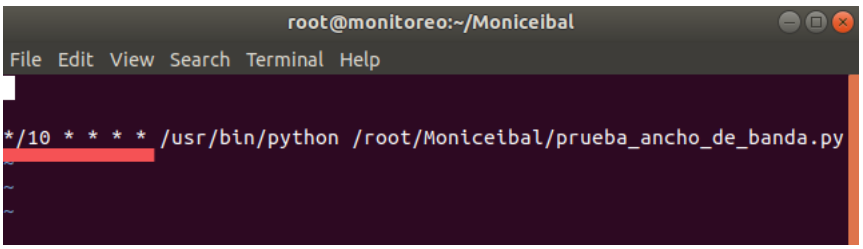


Figura 7.30: Archivo de configuración de *cron*.

- *Agregar IPs a inventario de Ansible:* Como en los casos anteriores, el *script* se conecta con la computadora Ansible y modifica su inventario agregando las nuevas IP, la figura 7.31 muestra el resultado de dicha operación.

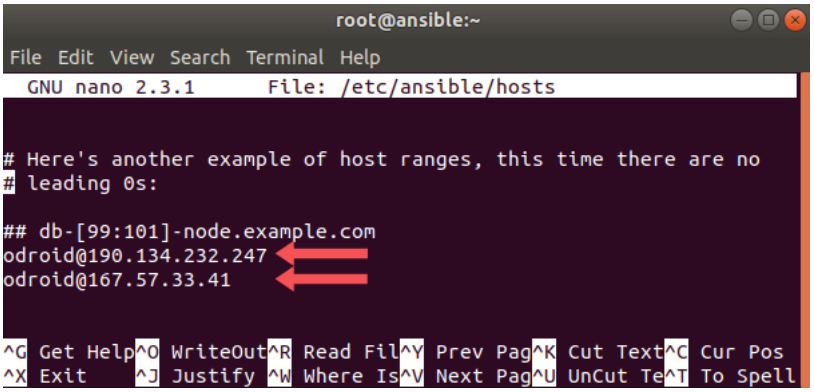


Figura 7.31: Inventario de Ansible.

- *Carpeta de trabajo en el equipo de Ansible:* En este caso lo único que Ansible debe distribuir a todos los *hosts* es el programa Assolo. Por lo tanto lo único que el equipo Monitoreo debe transmitir a Ansible es el instalador de Assolo y el *playbook* que se encarga de realizar todas las modificaciones necesarias en los equipos remotos. En la figura 7.32 se puede observar cómo ambos archivos pudieron en efecto ser enviados al equipo Ansible.

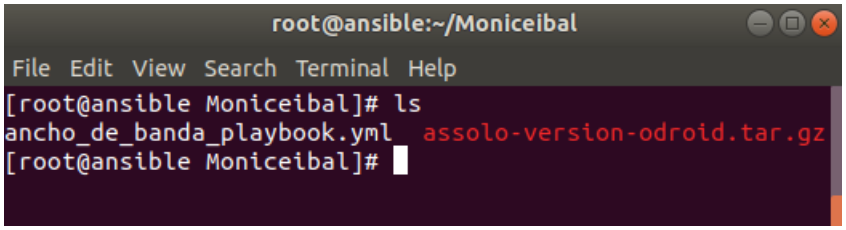


Figura 7.32: Inventario de Ansible.

- *playbook de Ansible:* El *playbook* recientemente mostrado debe ser ejecutado por Ansible de forma que este logre hacer todos los cambios necesarios en los *hosts* seleccionados. En la parte inferior de la figura 7.33 se puede verificar cómo el mismo logró realizar todas las tareas requeridos sin ninguna falla.

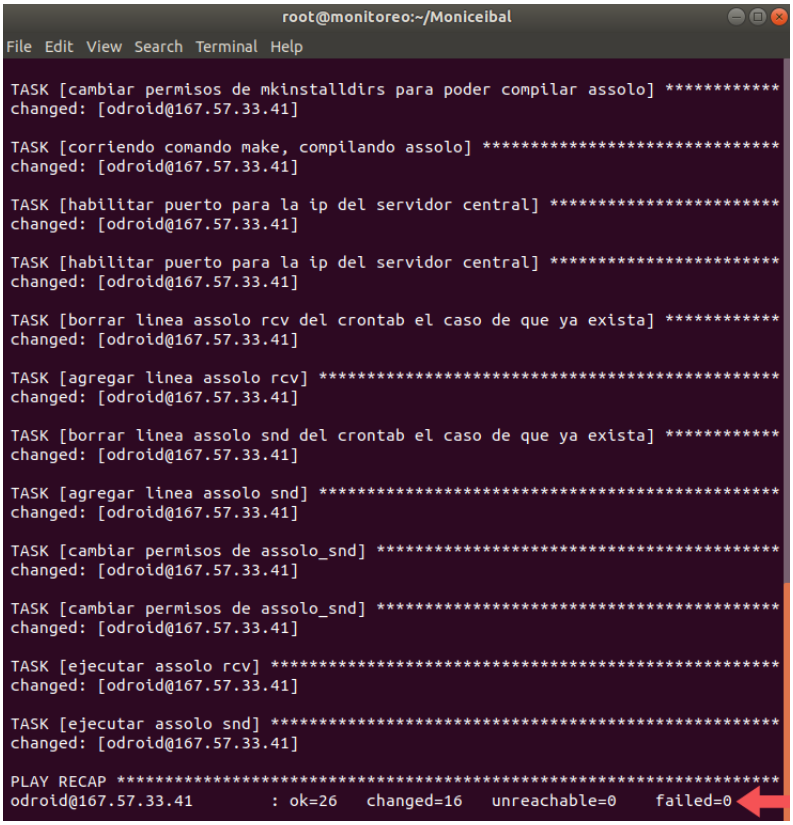


Figura 7.33: Ejecucion de *playbook* de Ansible.

- *Procesos de Assolo Corriendo:* Una de las tareas del *playbook* de Ansible era dejar corriendo los procesos *assolo_snd* y *assolo_rcv* en cada *host*. La figura 7.34 muestra que dichos procesos se encuentran efectivamente corriendo en el *host*. Esto además implica que el programa Assolo fue instalado con éxito.

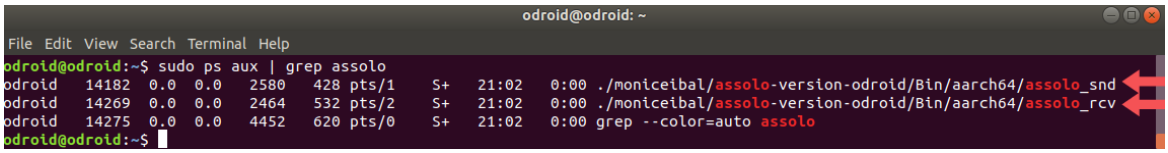


Figura 7.34: Procesos de Assolo corriendo.

- *Assolo en crontab del host:* Como se dijo antes, Ansible no sólo debe hacer que los procesos *assolo_snd* y *assolo_rcv* queden corriendo en el *host*, sino que debe lograr que dichos procesos comiencen a ejecutarse cada vez que el equipo se reinicia en caso de que el mismo se apague por accidente. Para esto se modifica el archivo *crontab* agregándole dos líneas que le indican que debe ejecutar dichos programas cada vez que se encienda el equipo como se muestra en la figura 7.35.

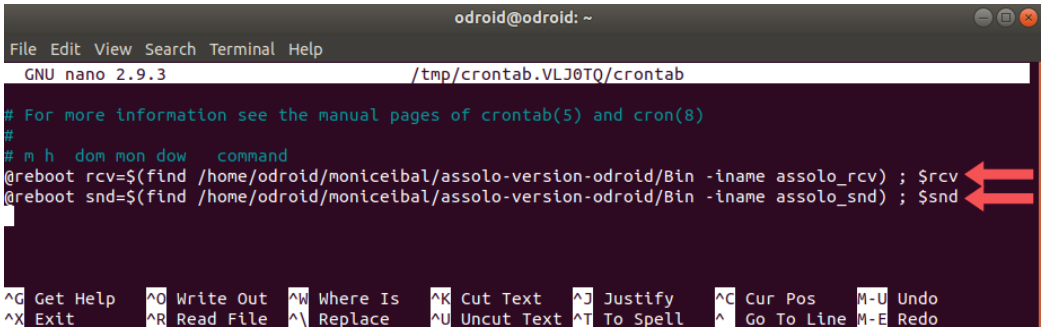


Figura 7.35: Assolo en *crontab* de los *hosts*.

- *Creación de items en Zabbix:* Debe verificarse que los tres *trapper items* necesarios hayan sido creados en Zabbix, Ancho_de_banda_subida, Ancho_de_banda_bajada y Errores. Esto se puede constatar observando la figura 7.36.

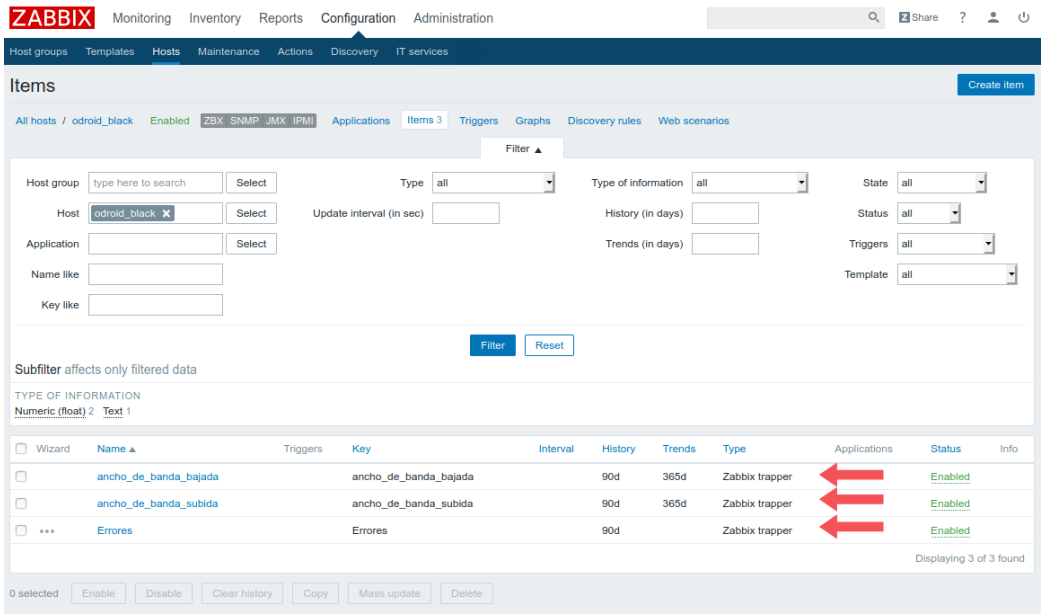


Figura 7.36: Items de Ancho de Banda creados.

Por último se muestra una captura del módulo recibiendo efectivamente las medidas.

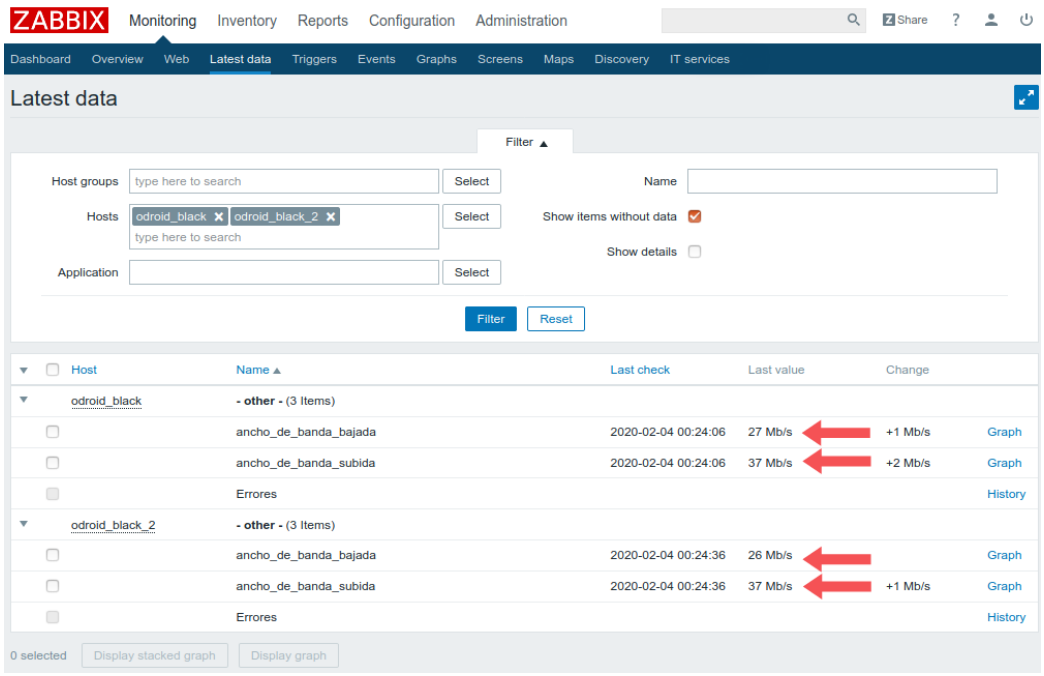


Figura 7.37: Módulo de Ancho de Banda funcionando.

7.6.2. Deshabilitar Módulo

Existen dos tareas que puede cumplir la función *delassolo.py*, la primera es eliminar una IP de la lista de equipos contra los cuales se esta tomando la medida. En este caso se borra la IP 190.134.232.247 que había sido inicialmente introducida. Para esto se ejecuta la línea de código:

```
./delassolo.py -u 190.134.232.247
```

Una vez ejecutado el *script* se puede verificar en la figura 7.38 que dicha IP ya no se encuentra en el archivo que contiene la lista de IPs contra las cuales se toma las medida. De esta forma el módulo continúa activo pero sólo con la IP que aún se encuentra en el archivo.

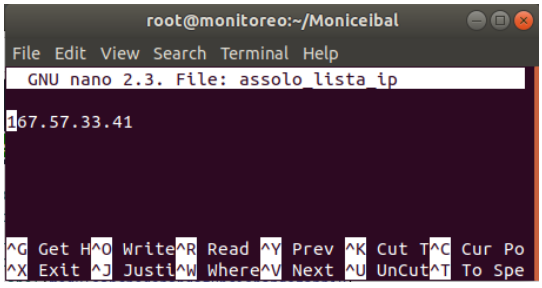


Figura 7.38: IP eliminada de la lista del módulo de medición de ancho de banda.

El otro objetivo que cumple esta función es detener completamente el funcionamiento del módulo mediante el parámetro *-stop*, lo cual se logra eliminando la línea del archivo *crontab* encargada de ejecutar los *scripts*. Por lo tanto se introduce el siguiente texto en la terminal:

```
./delassolo.py -stop
```

y luego puede verificarse en la figura 7.39 cómo dicha línea ha sido eliminada con éxito.

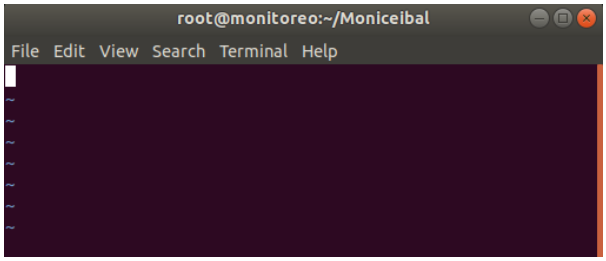


Figura 7.39: Línea eliminada de *crontab* para el módulo de ancho de banda.

7.6.3. Errores inducidos

Existen varias causas por las cuales la medida de ancho de banda no pueda ser tomada, tales como pérdida de conexión con el *host*, o que el mismo se apague por algún motivo, entre otras. En este caso cuando el *script* intente abrir el archivo que contiene las medidas tomadas por Assolo, el mismo se encontrará vacío y por ende enviará un mensaje de error al *item* errores de Zabbix. Para lograr inducir este error en el módulo, uno de los *host* fue desconectado y en consecuencia el mensaje de error correspondiente fue recibido por Zabbix como se muestra en la figura 7.40.

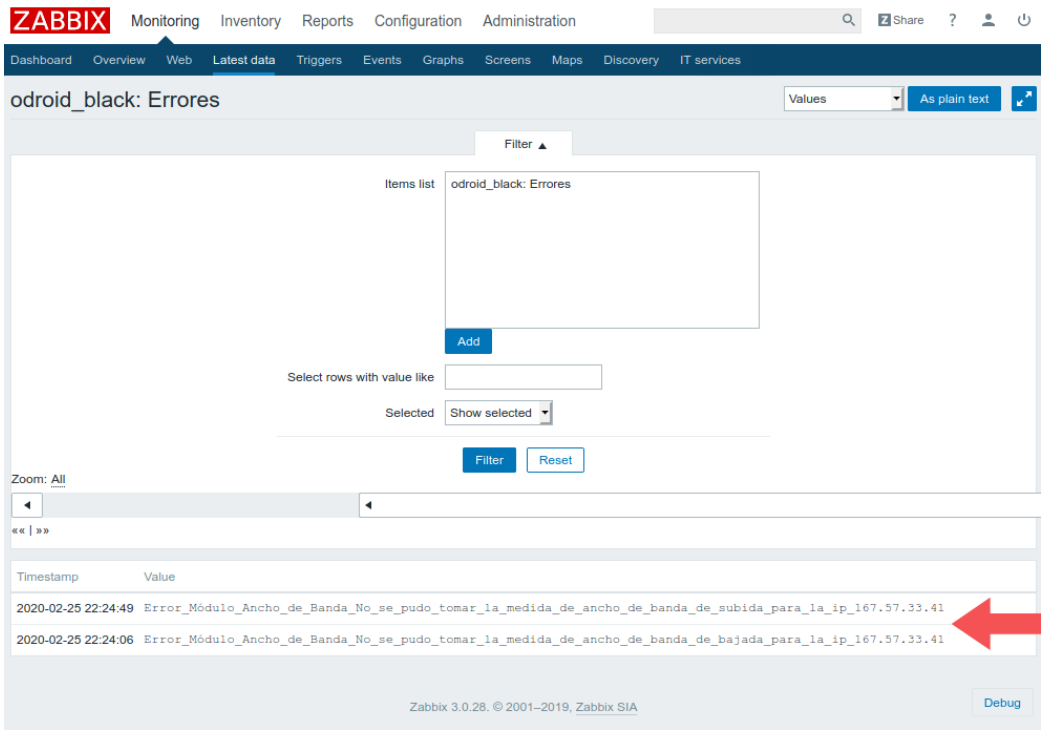


Figura 7.40: Mensaje de error proveniente del módulo de medida de ancho de banda.

Capítulo 8

Verificación de las Medidas

A lo largo de este capítulo se exhiben medidas tomadas con Assolo dentro de un ambiente de prueba controlado, con el fin de comprobar su exactitud y su monitoreo "liviano". Luego, se muestran medidas tomadas por los módulos dentro de un ambiente de prueba proporcionado por Plan Ceibal, cuya estructura es muy similar a la de un centro educativo, con el fin de comprobar que los valores obtenidos se ajustan a lo esperado.

8.1. Verificación de exactitud Assolo

En la mayoría de los módulos las medidas son tomadas con los comandos que normalmente se utilizan para realizar dichas tareas y que se han vuelto un estándar del sistema operativo Linux. Por otro lado, a la hora medir el ancho de banda existe una amplia variedad de herramientas muy utilizadas con distintas características y distintos niveles de exactitud. Por lo tanto en esta sección se exhiben medidas tomadas con Assolo dentro de un ambiente controlado con el fin de evaluar la calidad de las mismas.

Para esto se utilizó el ambiente de prueba visible en la figura 8.1 que consiste de los siguientes componentes:

- Un *router* desconectado de Internet para que ningún tráfico entrante o saliente pueda afectar los resultados de la medida y conectado vía dos cables RJ45 al *switch* y a una Raspberry Pi.
  - *Modelo:* ZTE ZXHN F660
  - *Conectividad:* 1 Gigabit Ethernet
- Un *switch* de capa de red conectado al *router* y a una Raspberry Pi
  - *Modelo:* ENH916P-NWY-E
  - *Conectividad:* 100 Mbps

- Dos equipos Raspberry Pi, uno conectado al *switch* y otro conectado al *router*.
    - *Modelo:* Raspberry Pi 3 B+
    - *Sistema operativo:* Ubuntu Mate
    - *CPU:* Broadcom BCM2837B0, CorteC-A53 (ARMv8) SoC de 64 bits a 1,4 GHz
    - *RAM:* 1 GB
    - *Conectividad:* Ethernet 300 Mbps
    - *Almacenamiento:* Tarjeta SD de 32 GB

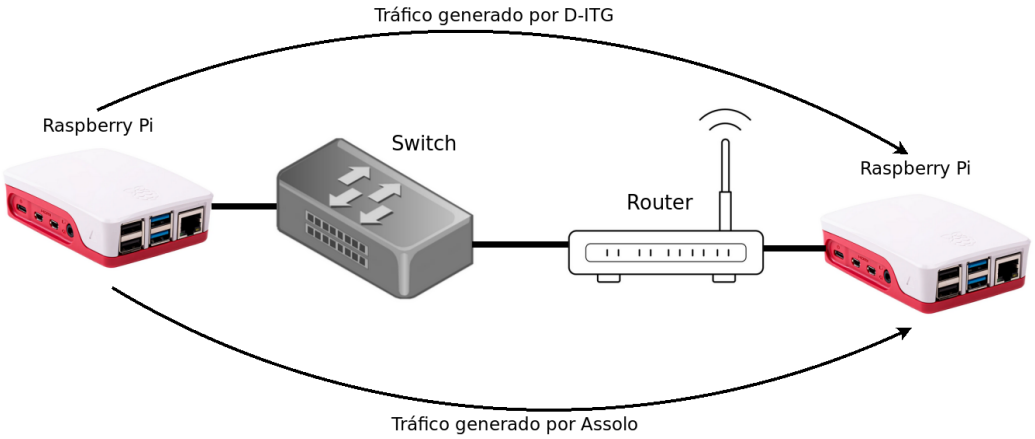


Figura 8.1: Ambiente de prueba para evaluar Assolo.

Para realizar este experimento se utilizaron los programas Assolo y D-ITG [25]. Éste último consiste en un generador de tráfico, el mismo debe ser instalado en dos puntos de la red entre los cuales se quiere generar tráfico, uno actuando como transmisor y el otro como receptor.

Como se puede ver en la figura 8.1, el equipo Raspberry Pi de la izquierda es utilizado para correr los transmisores de Assolo y D-ITG y el de la derecha para los receptores.

El experimento se basa en la premisa de que la suma del tráfico generado por D-ITG y el ancho de banda disponible debe ser igual a la capacidad total del canal. Dado que se trata de una red de un solo camino y el componente con menor capacidad es el *switch* de 100 Mbps, esta debería ser la capacidad del canal. Por lo tanto se asume que el ancho de banda disponible real en Mbps, está dado por la expresión:

$$A = 100 - T \tag{8.1}$$

Donde A representa el ancho de banda disponible real y T el tráfico generador por D-ITG. De aquí se deduce que el error relativo obtenido con Assolo en cada medida es:

$$E = \frac{|A - M|}{|A|} \tag{8.2}$$

Donde E representa el error relativo y M el ancho de banda disponible medido por Assolo. Dado que D-ITG permite seleccionar la velocidad del tráfico a enviar se hicieron experimentos de una duración de 60 s y distintos niveles de tráfico, los resultados se muestran a continuación.

La gráfica de la figura 8.2 muestra el ancho de banda disponible en función del tráfico generado y la tabla 8.1 exhibe los resultados de las medidas junto a su error relativo.

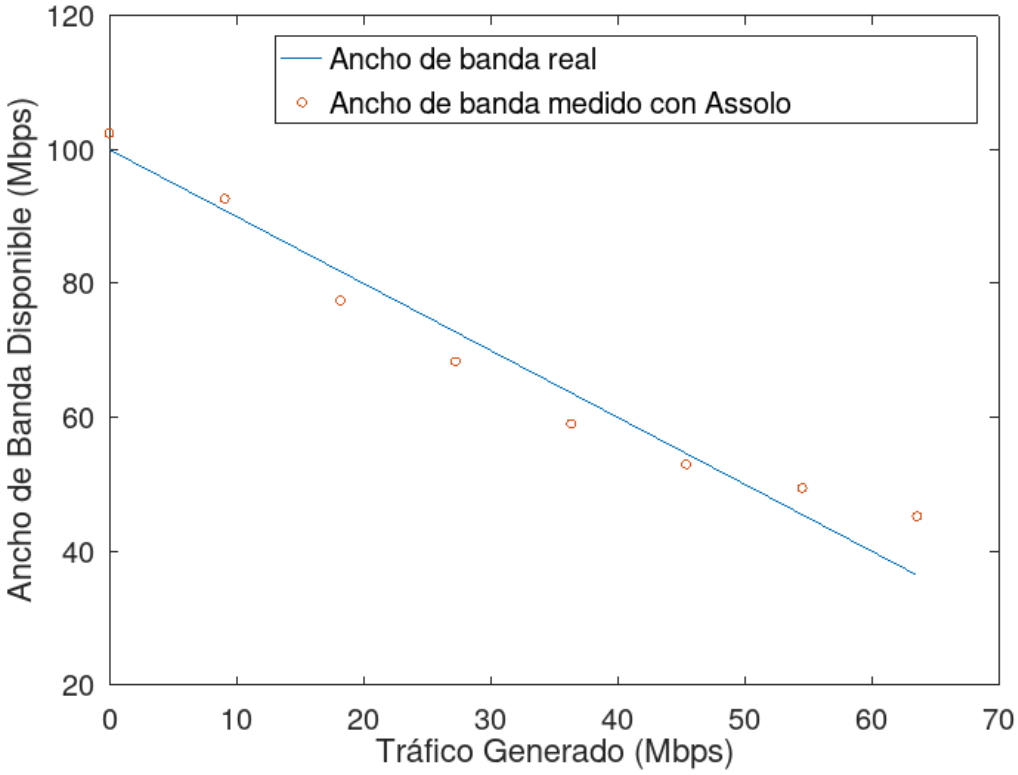


Figura 8.2: Ancho de banda disponible medido con Assolo en función del tráfico generado.

Tráfico generado (Mbps)	Ancho de banda disponible real (Mbps)	Ancho de banda disponible estimado por Assolo (Mbps)	Error relativo (%)
0	100.00	102.418	2.4180
9.094	90.906	92.643	1.9108
18.175	81.825	77.461	5.3333
27.253	72.747	68.323	6.0814
36.340	63.660	59.048	7.2447
45.399	54.601	53.016	2.9029
54.524	45.476	49.435	8.7057
63.574	36.426	45.228	24.1641

Tabla 8.1: Medidas obtenidas con Assolo junto a su error relativo.

Al observar la gráfica de la figura 8.2 se puede apreciar claramente cómo Assolo refleja la reducción del ancho de banda disponible al aumentar el tráfico inducido. No obstante, es cierto que para algunos casos hay una estimación por encima del ancho de banda disponible acorde al modelo teórico y en otros casos por debajo.

En la tabla 8.1 se puede apreciar cómo el error en general crece al aumentar el tráfico inducido, siendo muy pequeño para niveles bajos de tráfico. De todas formas para la mayoría de los casos este error se mantiene dentro de un margen menor al 10 % lo cual representa una estimación aceptable.

8.2. Verificación del Monitoreo "liviano" de Assolo

En el mismo ambiente de prueba que en la sección 8.1 se realizó una medida de ancho de banda utilizando Assolo, con el fin de verificar que el mismo realiza un monitoreo "liviano" sin generar niveles de tráficos altos.

El procedimiento consiste en sumar el tamaño de cada paquete enviado durante la medición, obteniendo la cantidad total de bits transmitidos durante la misma y luego dividir dicha cantidad entre el tiempo total de duración de la prueba, en este caso 20 s.

La figura 8.3 muestra una serie de paquetes enviados desde el transmisor al receptor de Assolo durante el experimento. En total fueron enviados una cantidad de aproximadamente 800 paquetes de 8000 bits cada uno. Por lo tanto la cantidad de tráfico generada por Assolo que circuló durante estos 20 s es

$$E = \frac{800 \times 8000 bits}{20 s} = 320 kbps \tag{8.3}$$

Dicho número coincide con la cantidad de tráfico que asegura la publicación [17]. Esta medida puede compararse por ejemplo con los anchos de banda disponibles obtenidos en las pruebas de la Subsección 8.3.4, los cuales son de aproximadamente 11 y 26.7 Mbps, o sea dos órdenes superiores de magnitud. En tal

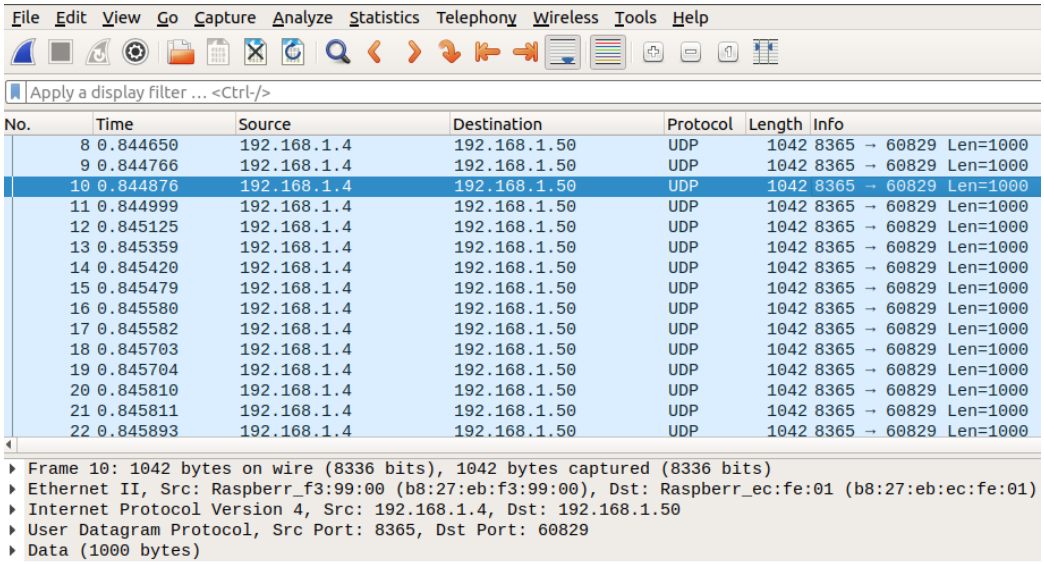


Figura 8.3: Captura de Wireshark de paquetes enviados por Assolo.

caso dicha medida representa una interferencia prácticamente despreciable.

Hay que destacar que como dice [17], la cantidad de tráfico por unidad de tiempo generado en las mediciones por este programa es constante y no depende de factores tales como la duración de la prueba y el ancho de banda disponible.

8.3. Medidas realizadas en la maqueta de centro educativo

Durante el desarrollo del proyecto. Plan Ceibal ofreció el uso de una maqueta que simula la estructura de un centro educativo con el fin de poner los módulos a prueba en la misma.

La parte de la maqueta utilizada durante este experimento consta de una Odroid-C2 igual a la descrita en la sección 7.1 y conectada a Internet mediante un *router* de borde el cual utiliza la misma configuración que los *routers* de los centros educativos. Por lo tanto la Odroid-C2 se conectó mediante Internet al ambiente de prueba que funciona como servidor central ya utilizado en la sección 7.1. El funcionamiento de este sistema se encuentra representado en la figura 8.4

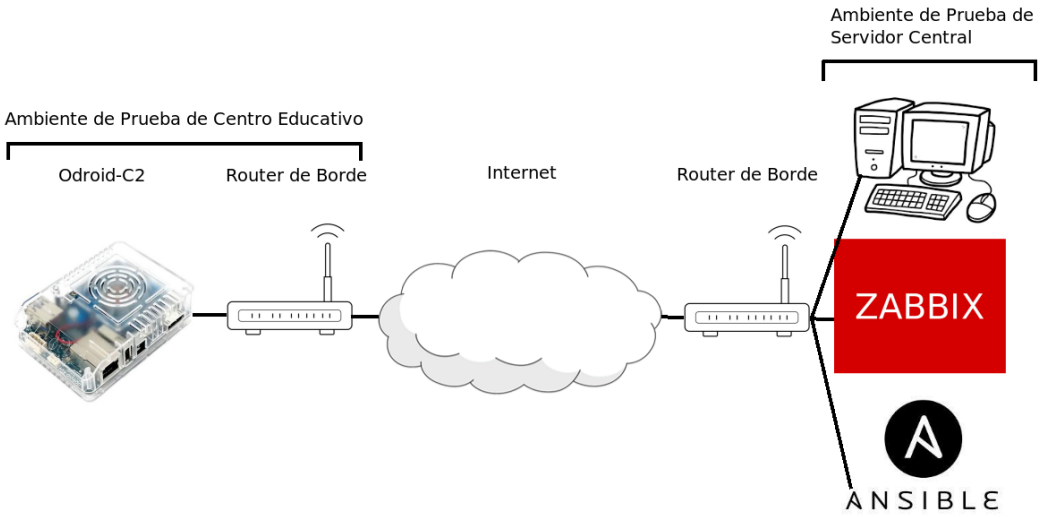


Figura 8.4: Ambientes de prueba utilizados para la validación de las medidas.

Para realizar las medidas se eligió un frecuencia de medición de 5 minutos, durante un período de una semana.

A continuación se muestran capturas de Zabbix con las medidas obtenidas. Para cada uno de los módulos se exhiben dos capturas, una mostrando las medidas en el transcurso de la semana y otra mostrando las medidas en el transcurso del último día con el fin de poder visualizar los datos con mayor exactitud.

En las gráficas se pueden apreciar tres curvas:

- *Curva verde:* Esta curva aparece en todas las gráficas y representa el valor de la medida promediado cada una hora.
- *Curva roja:* La misma sólo aparece en las gráficas de una semana de largo y representa sólo los máximos ocurridos durante un período de una hora.
- *Curva verde claro:* La misma también sólo aparece en las gráficas de una semana de duración y representa sólo los mínimos ocurridos durante un período de una hora.

8.3.1. Tiempo DHCP:

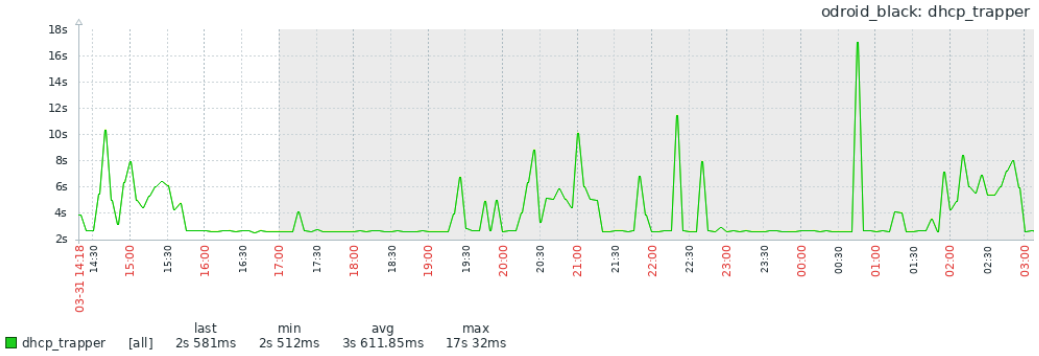


Figura 8.5: Medidas de tiempo DHCP tomadas a lo largo de aproximadamente 12 horas durante el día con una periodicidad de 5 minutos.

En la figura 8.5 se puede ver cómo el servidor DHCP fue capaz de funcionar durante doce horas sin errores. Su tiempo de respuesta media es de 3.6 s, lo que constituye un valor aceptable para tal servidor. No obstante también se observa que este tiempo puede ser variable, existiendo un pico de 17 s al final del gráfico. Esto se corresponde con un instante en el cual el servidor se tomó un poco más de lo habitual en responder.

A continuación la figura 8.6 muestra el comportamiento del mismo servidor pero a lo largo de una semana, en este caso la media del tiempo de respuesta es muy similar al de la captura anterior. Por otro lado, se pueden visualizar cuatro puntos en el gráfico donde se indica un tiempo de cero, estos se corresponden con momentos en los cuales el servidor no fue capaz de responder, lo cual puede deberse a varias causas tales como, momentos en que la red se encontraba sobrecargada o quizás instantes donde el mismo se encontraba fuera de servicio. En cada uno de estos instantes la plataforma se encargó de enviar un mensaje al *item* Errores de Zabbix advirtiendo de la situación.

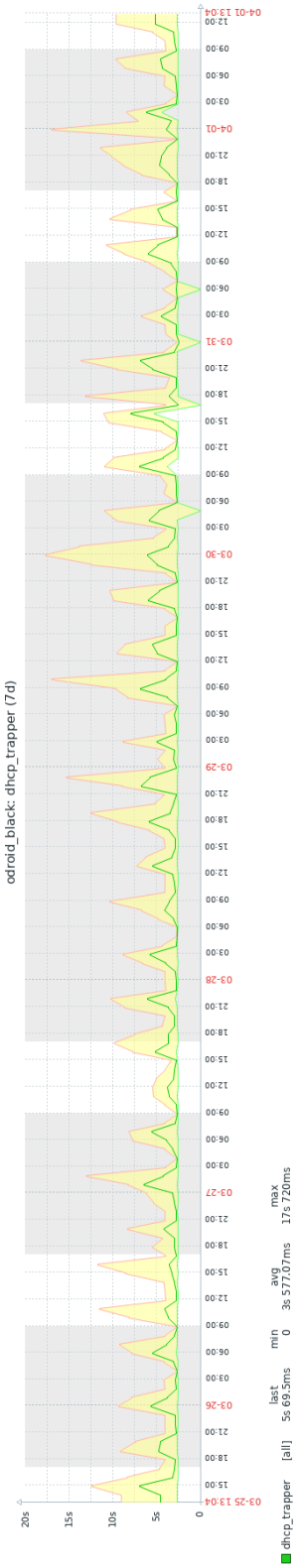


Figura 8.6: Medidas de tiempo DHCP tomadas a lo largo de una semana con una periodicidad de 5 minutos.

8.3.2. Tiempo DNS:

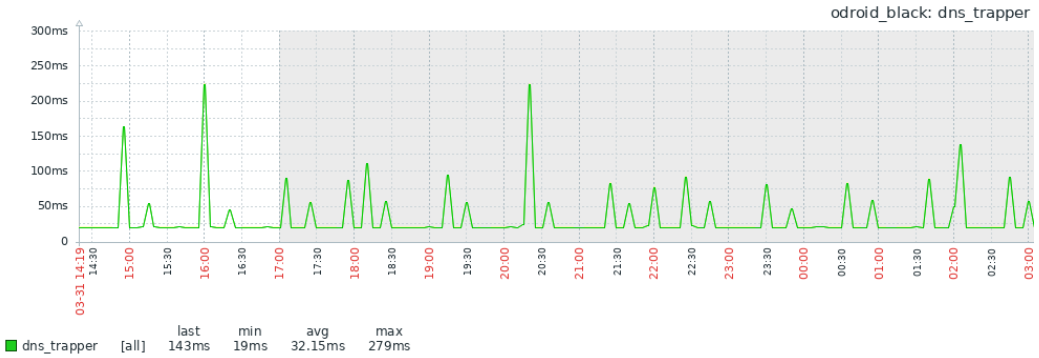


Figura 8.7: Medidas de tiempo DNS tomadas a lo largo de aproximadamente 12 horas con una periodicidad de 5 minutos.

En la figura 8.7 se puede ver cómo se obtuvieron las IPs correspondientes a los *hostnames* pasados como parámetros durante doce horas sin errores. La media del tiempo de respuesta fue de 32.15 ms. Estos tiempos pueden ser variables, llegando a un tiempo máximo de 279 ms. Este es un comportamiento esperable ya que la velocidad de la red cambia constantemente por causas independientes de los equipos involucrados en las medidas.

La figura 8.8 muestra los valores obtenidos pero a lo largo de una semana. En este caso la media del tiempo de respuesta es muy similar al de la captura anterior, al igual que lo son los tiempos máximo y mínimo. Se puede apreciar valores muy estables al paso del tiempo.

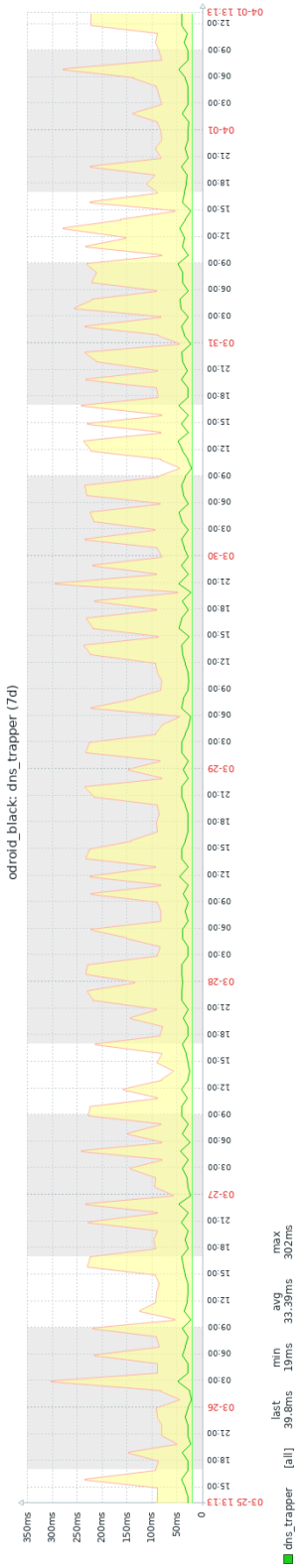


Figura 8.8: Medidas de tiempo DNS tomadas a lo largo de una semana con una periodicidad de 5 minutos.

8.3.3. Tiempo de descarga de sitio web:

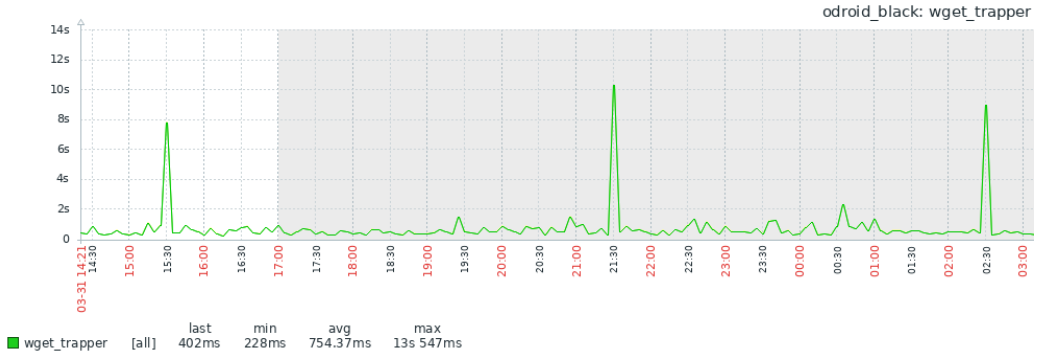


Figura 8.9: Medidas de tiempo de descarga de sitio web tomadas a lo largo de 12 horas con una periodicidad de 5 minutos.

En este caso la figura 8.9 muestra el módulo de tiempo de carga de sitio web funcionando por 12 horas de corrido, midiendo el tiempo de carga del sitio www.fing.edu.uy. Analizando el gráfico se puede observar que en ningún momento surgió un error. Eso se comprueba viendo que en todo momento se relevaron medidas distintas de cero. En caso de que hubiera un error, aparecería un tiempo de medición igual a cero, lo cual no sucedió y si sucediera se informaría mediante el *item* errores.

La media de tiempo de descarga para este sitio fue de 754 ms, se puede notar que hubo tres picos, donde el mayor arrojo una medición de 13.5 s. Este tipo de pico entran dentro de lo esperado, pues es normal que una red experimente oscilaciones en su velocidad repercutiendo directamente en la velocidad de descarga. Fuera de eso se comprueba que con este módulo se obtienen valores razonables.

La figura 8.10 muestra el comportamiento del mismo servidor pero a lo largo de una semana, en este caso la media del tiempo de respuesta se asemeja al de la captura anterior. En cuanto se refiere a los picos en el tiempo de demora, en este caso se obtuvo uno mayor, del orden de 18 s, más allá de esto, no se notó ningún tiempo abrupto en comparación con los picos anteriores. Por lo tanto se puede concluir que en general la red se comportó de forma similar a lo largo de toda la semana.

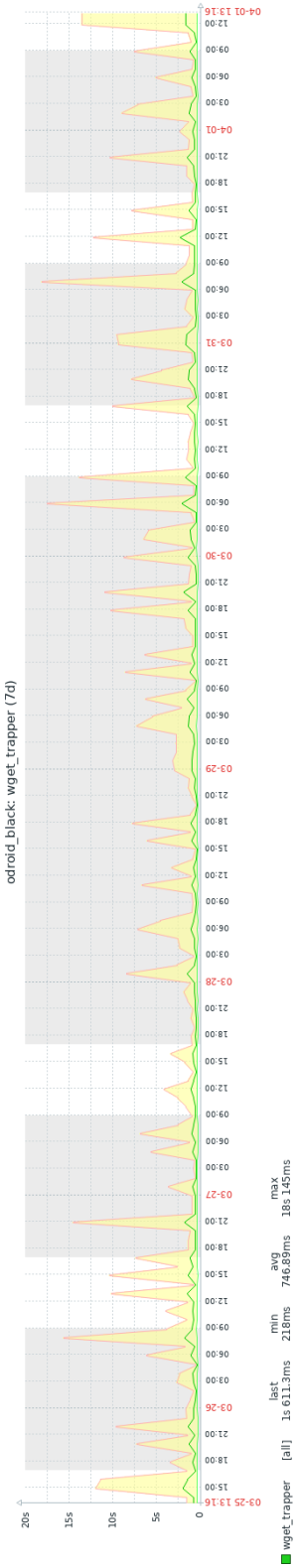


Figura 8.10: Medidas de tiempo de descarga de sitio web tomadas a lo largo de una semana con una periodicidad de 5 minutos.

8.3.4. Ancho de banda de subida:

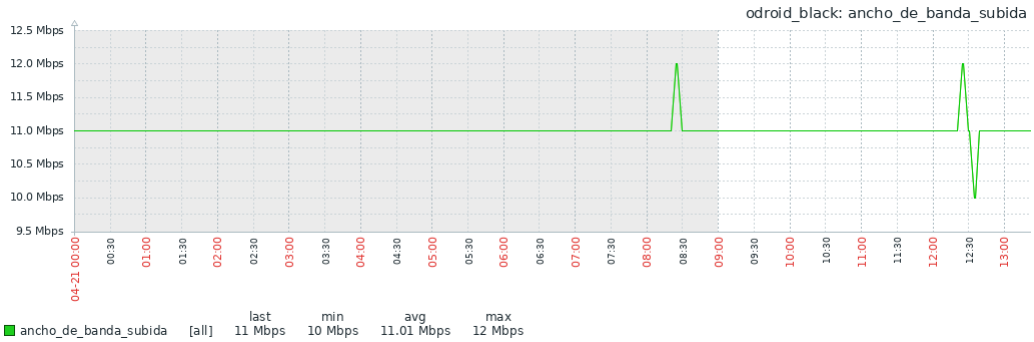


Figura 8.11: Medidas de ancho de banda de subida tomadas a lo largo de aproximadamente 12 horas con una periodicidad de 5 minutos.

Para los gráficos de la figura 8.11 y la figura 8.12 se puede observar que el ancho de banda de subida medido tiene una media de aproximadamente 11 Mbps. En el ambiente de prueba utilizado, el servicio de Internet contratado debe proveer un ancho de banda de subida de 10 Mbps.

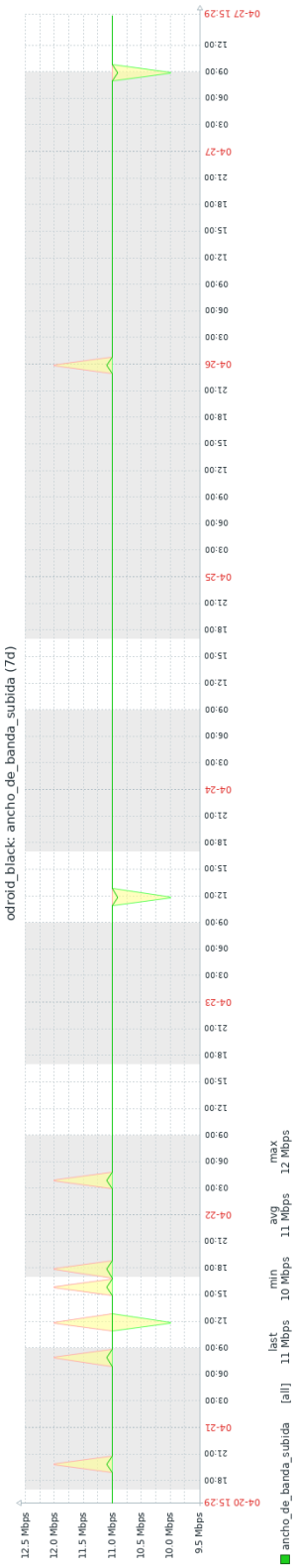


Figura 8.12: Medidas de ancho de banda de subida tomadas a lo largo de una semana con una periodicidad de 5 minutos.

8.3.5. Ancho de banda de bajada:

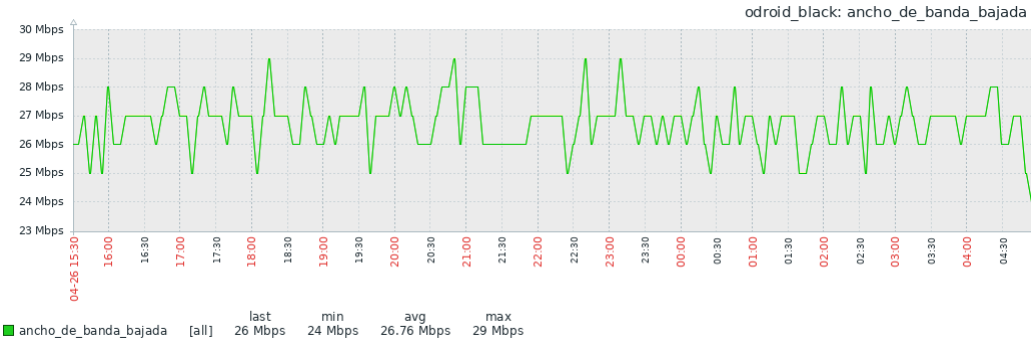


Figura 8.13: Medidas de ancho de banda de bajada tomadas a lo largo de aproximadamente 12 horas con una periodicidad de 5 minutos.

En los gráficos de la figura 8.13 y la figura 8.14, la media del ancho de banda disponible es de aproximadamente 26.7 Mbps y varía entre 24 y 30 Mbps. El ancho de banda de bajada contratado para el ambiente de prueba es de 30 Mbps. Teniendo en cuenta que el ancho de banda disponible puede bajar por causa del tráfico introducido por otros procesos o usuarios y que las medidas realizadas por Assolo poseen cierto margen de error, se puede decir que la media obtenida es coherente con lo que se esperaba.

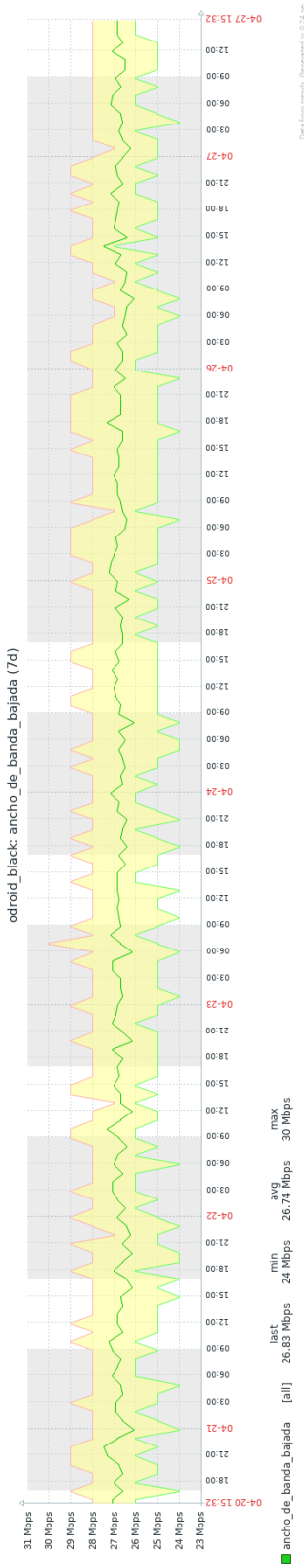


Figura 8.14: Medidas de ancho de banda de bajada tomadas a lo largo de una semana con una periodicidad de 5 minutos.

Capítulo 9

Conclusiones

Actualmente debido al rápido crecimiento de las redes, se ha vuelto de gran importancia realizar una buena gestión y monitoreo de las mismas. Por lo tanto existe una constante búsqueda de mejoras en lo que se refiere a estos aspectos mencionados anteriormente.

Durante el desarrollo del proyecto se profundizó en el uso de distintas herramientas con el objetivo de poder determinar cuáles serían usadas, brindando en este documento comparaciones sobre varias opciones que existen en la actualidad. Destacándose entre estas herramientas: Ansible, Zabbix y Assolo, herramientas que fueron de gran ayuda durante el desarrollo de éste proyecto.

Este trabajo plantea la realización de módulos de medición, los cuales permitieran obtener datos sobre distintos parámetros de una red, no sólo limitándose su aplicación a la red Ceibal, sino a otras redes con características similares.

En particular se logró crear módulos que evalúan aspectos tales como el desempeño del servicio de DNS, DHCP, tiempo de descarga de sitio web y ancho de banda, lo cual contribuye como una mejora al monitoreo que Plan Ceibal ya realiza sobre sus redes.

En lo que respecta a ancho de banda es importante destacar que se pudo lograr de forma exitosa construir un módulo que midiera el ancho de banda de forma poco invasiva, generando un monitoreo "liviano", lo cual abre la posibilidad de poder tomar medidas en horas pico sin afectar la calidad de experiencia de los usuarios.

El proyecto llevó un proceso de diseño considerable en el cuál se debió evaluar y probar distintas alternativas, evaluar cuáles se adaptaban más a la red de Ceibal y a los requisitos del cliente. Se debe destacar que la plataforma desarrollada logró cumplir con las expectativas del cliente, dando como resultado una herramienta que permite recabar datos de importancia para Plan Ceibal y logrando integrar herramientas ya utilizadas. A su vez se priorizó el uso de herramientas gratuitas de forma tal que se minimizaron los costos pero que a su vez se tuviera un correcto rendimiento. Por otro lado, se logró obtener un diseño escalable, de forma de poder desarrollar nuevas funcionalidades en un futuro, tales como nuevos módulos u optimizar el funcionamiento de la misma.

Por último se concluye que se logró de forma exitosa cumplir todos los objetivos planteados en la sección 1.2.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 10

Trabajo Futuro

- *Creación de nuevos módulos:* Como trabajo futuro existe en principio la posibilidad de expandir la plataforma agregando nuevos módulos de medición. Para la creación de los mismos puede utilizarse como ejemplo el trabajo realizado durante este proyecto, tanto la elaboración de los *scripts* que toman las medidas como la integración de distintas herramientas informáticas.

  Por ejemplo en el futuro se podrían crear módulos que permitan medir aspectos tales como calidad de experiencia en cuanto a aspectos más concretos. Estos aspectos podrían ser aplicaciones específicas, plataformas educativas, calidad de audio y video o videollamadas, entre otros.

- *Modificación de parámetros:* Para funcionar, los módulos actualmente utilizan ciertos parámetros, algunos fácilmente modificables y otros embebidos en el código fuente. A lo largo del tiempo, los usuarios de la plataforma podrían lograr, en base a la experiencia, establecer valores óptimos para dichos parámetros que permitan un mejor aprovechamiento de la plataforma y sus recursos.

- *Alarmas:* Si bien la plataforma permite visualizar los resultados obtenidos en todo momento, esta no posee alarmas que alerten al usuario de la ocurrencia de ningún evento concreto. Por lo tanto otro punto que podría ser trabajado en el futuro es por ejemplo la elaboración de alarmas que advirtieran al usuario que ciertas medidas han alcanzado cierto valor o que han estado obteniendo el mismo resultado durante cierto tiempo.

- *Interfaz gráfica:* Durante la elaboración de este proyecto el foco principal estuvo puesto en la elaboración de módulos que realmente reflejen el estado de salud de la red, la posibilidad de que los mismos sean fácilmente distribuibles entre los distintos *hosts* y la interconexión entre las distintas herramientas de *software* a utilizar. Por ende un aspecto que fue dejado de lado, es el desarrollo de una interfaz gráfica que permita una interacción más amigable con el usuario. Dicho aspecto podría ser incorporado a la plataforma en un futuro.

- *Base de datos:* La información de qué módulos se encuentran instalados en cada *host* se puede obtener desde Zabbix, pero para saber qué configuración exacta tiene cada módulo habría que verlo desde los propios equipos en los centros educativos o mediante conexiones *ssh* a cada uno. Para eso, un trabajo a futuro que mejoraría la visualización del contenido de cada *host* en forma eficiente, es crear una base de datos con información relevante acerca de qué módulos y con qué configuración se encuentran instalados en cada equipo.

Apéndice A

Equipos Mini-PC frente a otras alternativas del mercado

En los últimos años estos equipos se han posicionado en el mercado fuertemente, logrando prestaciones similares a la de equipos convencionales de escritorio. Entre sus mayores fortalezas se destacan su versatilidad, potencia y su relación costo calidad.
Se consideró importante dar un panorama general de algunos equipos que se encuentran en plaza similares en prestaciones a la Odroid-C2, modelo de Mini-PC utilizado en este proyecto.

Estos equipos compiten con equipos tales como las *notebooks* o PCs de escritorio en ciertos aspectos que se mencionarán a continuación.

Tamaño: Dependiendo del modelo, puede ser tan pequeño como un libro, a veces incluso más pequeño como el caso de la Mini Stick T6 con un tamaño de 110x39x17 mm. Su tamaño reducido permite guardar la PC donde es esencialmente invisible y ahorrar una cantidad significativa de espacio. Lo cual en algunos escenarios puede ser muy beneficioso.

Versatilidad: Una Mini-PC puede tener la potencia de una PC de escritorio estándar y dado su tamaño puede ser conectada a la pantalla que se elija fácilmente de forma que la misma funcione como un ordenador PC .

Costo: Este es uno de sus puntos fuertes, ya que pueden conseguir prestaciones similares a los de una PC convencional por una suma menor a 100 dólares. Sin embargo existe una gama de estos equipos que cuestan una cantidad de dinero mucho mayor, como el caso de la Mac Mini 1 que actualmente se puede comprar en Amazon por un costo de 729 dólares, como es lógico estos equipos presentan prestaciones mucho mejores.

Consumo de Energía: Estos equipos son diseñados para consumir poca energía, además pueden estar encendidos por largos períodos de tiempo.

A.0.1. Comparación entre equipos Mini-PC

En este caso se mencionará las principales características de 4 modelos que en la actualidad tienen una amplia adopción en el mercado. Se pretende dar un pa-

norama general de los mismos, nombrando sus principales características y dando comparativas entre ellos.

Estos equipos son: Odroid-C2 [31], Odroid xu4 [30], Raspberry Pi 3 B+ [32], Raspberry Pi 4 [33].

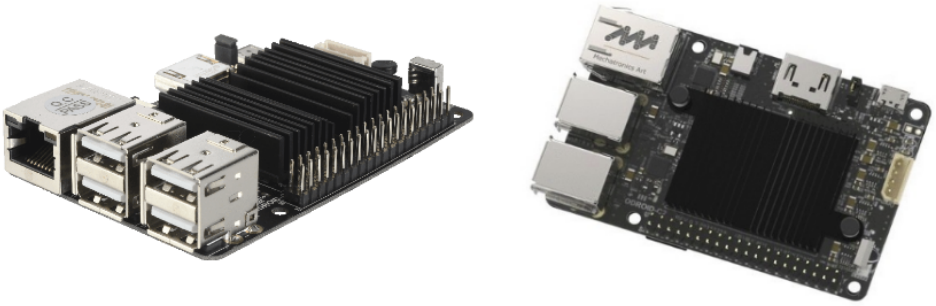


Figura A.1: Odroid-C2

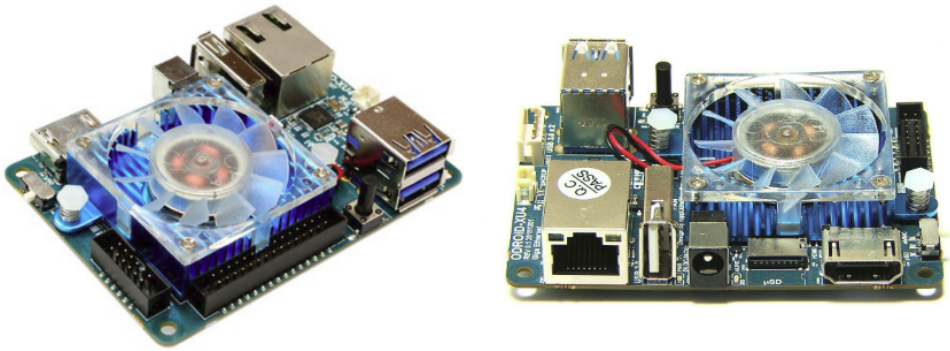


Figura A.2: Odroid Xu4

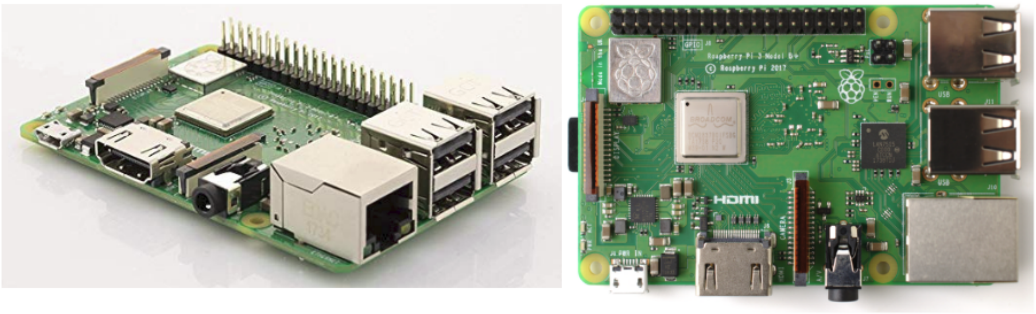


Figura A.3: Raspberry Pi 3 B+

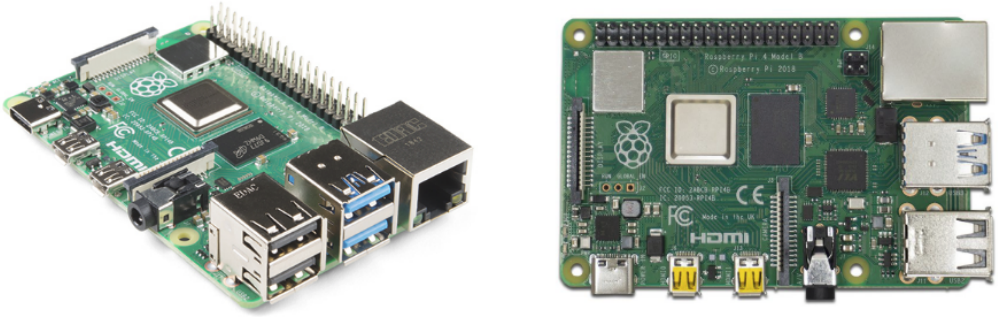


Figura A.4: Raspberry Pi 4

A.1. Tarjeta de red

En la actualidad lograr altas velocidades en la transferencia de tráfico se ha vuelto indispensable para diversas tareas. Por lo que un punto en el que se concentran estos equipos, es en su tarjeta de red, buscando poder lograr prestaciones que les permitan obtener velocidades del orden de los Mbps o Gbps en algunos casos.

La Raspberry Pi 3 B+ posee una tarjeta de red que permite lograr velocidades máximas de 300 Mbps. Mientras que la Odroid-C2 supera ese límite llegando a 1000 Mbps. Luego tanto la Odroid xu4 como Raspberry Pi 4 poseen una interfaz Ethernet de un Gbps. Podemos notar como estos dos últimos modelos superan ampliamente a los dos modelos mencionados en un principio. Todos los modelos poseen una única interfaz de red.

A.2. Periféricos

Tanto la Raspberry Pi 3 B+ como la Odroid-C2 poseen 4 puertos USB 2.0. Por otro lado la Odroid xu4 posee 2 puertos USB 3.0 y uno 2.0, mientras que la Raspberry Pi 4 posee 2 USB 2.0 y 2 USB 3.0. En cuestión de puertos periféricos todos los modelos mencionados tienen aproximadamente la misma cantidad y con especificaciones similares, siendo superados los modelos Odroid-C2 y Raspberry Pi 3 B+ por la Odroid xu4 y Raspberry Pi 4. La diferencia más apreciable en cuestión de periféricos es que la Raspberry Pi 4 no incluye entrada HDMI sino que la cambia por dos entradas micro HD. Estos equipos no poseen Wi-Fi pero es posible agregarlo, comprando una extensión que no tiene un costo mayor a 6 dólares.

A.3. Temperatura

Una característica que impacta directamente en su rendimiento es su temperatura de funcionamiento, estos equipos a la hora de su compra en general no incluyen ni disipadores ni ventiladores, a excepción de las Odroid-C2 las cuales lo traen incluido. Por lo tanto a la hora de comprar una de estas Mini-PC, se debe tener en cuenta que estos elementos de disipación de calor en muchas de estas no están incluidos en el precio. Su uso no es mandatorio, sin embargo tanto la comunidad de Raspberry Pi 3 B+ como la que está detrás de las Odroid-C2 recomiendan su uso. El precio de un disipador o un ventilador actualmente ronda en el precio de los 5 dólares, lo cual hace que sea accesible su compra. En otros casos se puede conseguir *pack* en los cuales se pueden incluir estos elementos además de otros y conseguir un mejor precio.

A.4. Precio

En cuestión de precios las Raspberry Pi 3 B+ y Odroid-C2 son similares, a continuación se listará el precio que saldría si se quisiera comprar alguno de los

equipos discutidos, incluyendo una carcasa, un disipador o ventilador y la alimentación, en general estos equipos no incluyen este último. Además se incluye una tabla con las principales características de cada equipo.

	ODROID-C2	ODROID XU4	RASPBERRY PI 3 B+	RASPBERRY PI 4
CPU	Amlogic S905 SoC 4 x ARM Cortex-A53 1.5GHz 64bit ARMv8 Architecture @28nm	Samsung Exynos5422 ARM® Cortex™ A15 Quad 2.0GHz/Cortex™ A7 Quad 1.4GHz	Broadcom BCM2837B0 , Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
RAM	2GB 32bit DDR3 912MHz	2 Gbyte LPDDR3 RAM PoP 750Mhz	1GB LPDDR2 SDRAM	1GB, 2GB or 4GB LPDDR4-3200 SDRAM (dependiendo del modelo)
Ethernet / LAN	10 / 100 / 1000 Mbit/s	10/100/1000Mbps	10 / 100 Mbit/s	Gigabit Ethernet
Almacenamiento	Micro-SD o eMMC5.0 storage option	eMMC module socket eMMC 5.0 Flash Storage MicroSD Card Slot (up to 128GByte)	Micro-SD No eMMC storage option	Micro-SD card slot
Size	85 x 56 mm * 20 mm approx.	83 x 58 x 20 mm approx.	85 x 56 mm x 20 mm approx.	similar al de raspbeery pi 3
Compra de Mini-PC				
Heat sink	incluido	incluido	U$S 3.99	U$S 6.99
Carcasa	U$S 4.50	U$S 5.40	U$S 6.49	U$S 9.35
Alimentación	U$S 5	U$S 5	U$S 7.99	U$S 7.99
Equipo	U$S 46	U$S 49	U$S 38	U$S 42
Total	U$S 55.5	U$S 59.4	U$S 56.47	U$S 66.33

Figura A.5: Información Mini-PC, observación: los precios pueden variar con el tiempo

A.5. Conclusión

Los equipos discutidos en esta sección demuestran tener un buen rendimiento y con presentaciones que permiten realizar una gran gama de proyectos. Se concluye que en cuanto a prestaciones y rendimiento las Odroid-C2 presentan mejores prestaciones por un precio un poco mayor al de las Raspberry Pi 3 B+. En cuanto a estos últimos equipos las ventajas que más presentan frente a las Odroid-C2 y otras Mini-PC de gama similar es la gran comunidad que hay detrás de estos equipos lo que le da a los mismos un gran soporte.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice B

playbooks Ansible

En este capítulo se mostrarán los distintos *playbooks* de Ansible que son utilizados por los *scripts* con el fin de realizar los cambios necesarios en los equipos remotos. En estos *playbooks* todas las sentencias que se encuentran entre llaves de la forma { } son variables cuyo contenido es pasado desde el *script* que los está ejecutando.

El primer *playbook* se utiliza en caso de que el módulo de tiempo DNS ya se encuentre creado en cierto *host* pero de todas formas el usuario quiera cambiar su archivo de configuración sin tener que instalar el módulo nuevamente. En este caso el *playbook* distribuye el nuevo archivo de configuración a todos los equipos deseados sin volver a instalar el módulo.

```
- \textit{hosts}: "{{ variable_host | default('web') }}"
  tasks:
  - name: crear directorio de trabajo
    file:
     path: "{{ carpeta_destino_odroid }}"
     state: directory
  - name: copiar archivo de configuraci n dns
    copy:
     force: yes
     src: "{{ directorio_archivo_configuraci n }}"
     dest: "{{ carpeta_destino_odroid }}"
```

El siguiente *playbook* cumple la misma función que el anterior pero para el módulo de tiempo de descarga de sitio web.

```
- \textit{hosts}: "{{ variable_host | default('web') }}"
  tasks:
  - name: crear directorio de trabajo
    file:
     path: "{{ carpeta_destino_odroid }}"
     state: directory
```

```
7     - name: copiar archivo de configuraci n dns
8       copy:
9        force: yes
10       src: "{{ directorio_archivo_configuraci n }}"
11       dest: "{{ carpeta_destino_odroid }}"
```

El próximo *playbook* se utiliza para crear los *items* de tiempo DHCP, tiempo DNS y tiempo de descarga de sitio web en los equipos en donde aún no se ha instalado.

```
1  - \textit{hosts}: "{{ variable_host | default('web') }}"
2    become: true
3    tasks:
4    - name: instalar dnsutils
5      apt:
6       name: dnsutils
7    - name: instalar wget
8      apt:
9       name: wget
10   - name: instalar net-tools(ifconfig)
11     apt:
12      name: net-tools
13   - name: instalar at
14     apt:
15      name: at
16   - name: cambiar el tiempo m ximo de ejecuci n a 30
        segundos
17     lineinfile:
18      path: /etc/zabbix/zabbix_agentd.conf
19      regexp: Timeout=
20      line: Timeout=30
21   - name: crear directorio de trabajo
22     file:
23      path: "{{ carpeta_destino_odroid }}"
24      state: directory
25   - name: copiar \textit{script} del m dulo
26     copy:
27      force: yes
28      src: "{{ directorio_script }}"
29      dest: "{{ carpeta_destino_odroid }}"
30   - name: copiar archivo de configuraci n
31     when: "{{ existe_configuraci n }}"
32     copy:
33      force: yes
34      src: "{{ directorio_archivo_configuraci n }}"
```

```
35        dest: "{{ carpeta_destino_odroid }}"
36      - name: crear UserParameter
37        lineinfile:
38         path: /etc/zabbix/zabbix_agentd.conf
39         regexp: "{{ regenerar }}"
40         line: UserParameter={{ key }},echo "'{{ \textit{
              script} }}' {{ nombrehost }} {{ ip_zabbix }} {{
              carpeta_destino_odroid }} {{
              archivo_configuraci n }}"| at now
41      - name: cambiar permisos \textit{script} para
           ejecutarlo
42        shell: "{{ activar_script }}"
43      - name: crear grupo de usuarios moniceibal
44        group:
45          name: moniceibal
46          state: present
47      - name: agregar usuario odroid al grupo
48        shell: sudo usermod -a -G moniceibal odroid
49      - name: agregar usuario zabbix al grupo
50        shell: sudo usermod -a -G moniceibal zabbix
51      - name: agregar usuario daemon al grupo
52        shell: sudo usermod -a -G moniceibal daemon
53      - name: dar permisos sobre la carpeta al grupo
           moniceibal
54        shell: sudo chgrp -R moniceibal {{
             carpeta_destino_odroid }}
55      - name: dar permisos sobre la carpeta al grupo
           moniceibal
56        shell: sudo chmod -R 775 {{ carpeta_destino_odroid
             }}
57      - name: reiniciar zabbix \textit{agent}
58        service:
59         state: restarted
60         name: zabbix-agent
```

Este último *playbook* se utiliza para crear el módulo de medida de ancho de banda en los equipos en donde aún no se ha instalado.

```
1   - \textit{hosts}: "{{ variable_host | default('web') }}"
2     become: true
3     tasks:
4     - name: instalar build-essential lo cual incluye gcc,
         g++ y make entre otros
5       apt:
6        name: build-essential
```

```
7     - name: instalar build-essential lo cual incluye gcc,
          g++ y make entre otros
8       apt:
9         name: gcc
10    - name: instalar manpages-dev
11      apt:
12        name: manpages-dev
13    - name: instalar libpcap-dev
14      apt:
15        name: libpcap-dev
16    - name: instalar cron
17      apt:
18        name: cron
19    - name: habilitar cron
20      shell: sudo systemctl enable cron
21    - name: instalar at
22      apt:
23        name: at
24    - name: cambiar el tiempo maximo de ejecucion a 30
          segundos
25      lineinfile:
26        path: /etc/zabbix/zabbix_agentd.conf
27        regexp: Timeout=
28        line: Timeout=30
29    - name: crear directorio de trabajo
30      file:
31        path: "{{ carpeta_destino_odroid }}"
32        state: directory
33    - name: copiar assolo
34      copy:
35        src: "{{ directorio_assolo_instalador }}"
36        dest: "{{ carpeta_destino_odroid }}"
37    - name: descomprimir assolo
38      shell: ( cd "{{ carpeta_destino_odroid }}" ; sudo
            tar -xzf assolo-version-odroid.tar.gz )
39    - name: cambiar permisos de assolo configure para
          poder ejecutarlo
40      file:
41        path: "{{
              cambiar_permisos_de_assolo_configure_para_poder_ejecutarlo
              }}"
42        mode: a+x
43    - name: ejecutar configure de assolo para poder
          compilar assolo
44      shell: "{{
```

```
        ejecutar_configure_de_assolo_para_poder_compilar_assolo
         }}"
45    - name: cambiar permisos de mkinstalldirs para poder
       compilar assolo
46      file:
47       path: "{{
           cambiar_permisos_de_mkinstalldirs_para_poder_compilar_assolo
            }}"
48       mode: a+x
49    - name: corriendo comando make, compilando assolo
50      shell: "{{ corriendo_comando_make_compilando_assolo
          }}"
51    - name: habilitar puerto para la ip del servidor
       central
52      shell: sudo ufw allow from 192.168.1.7 to any port
          8365
53    - name: habilitar puerto para la ip del servidor
       central
54      shell: sudo ufw allow from 192.168.1.7 to any port
          7365
55    - name: borrar linea assolo rcv del crontab el caso de
        que ya exista
56      shell: crontab -u odroid -l | grep -v '{{
          borrar_linea_assolo_rcv_del_crontab_el_caso_de_que_ya_exista
           }}'  | crontab -u odroid -
57    - name: agregar linea assolo rcv
58      shell: (crontab -u odroid -l; echo '{{
          agregar_linea_assolo_rcv }}' ) | crontab -u
          odroid -
59    - name: borrar linea assolo snd del crontab el caso de
        que ya exista
60      shell: crontab -u odroid -l | grep -v '{{
          borrar_linea_assolo_snd_del_crontab_el_caso_de_que_ya_exista
           }}'  | crontab -u odroid -
61    - name: agregar linea assolo snd
62      shell: (crontab -u odroid -l; echo '{{
          agregar_linea_assolo_snd }}' ) | crontab -u
          odroid -
63    - name: cambiar permisos de assolo_snd
64      shell: "{{ cambiar_permisos_de_assolo_snd }}"
65    - name: cambiar permisos de assolo_snd
66      shell: "{{ cambiar_permisos_de_assolo_rcv }}"
67    - name: ejecutar assolo rcv
68      shell: echo '{{ ejecutar_assolo_rcv }}' | at now
69    - name: ejecutar assolo snd
```

```
70      shell: echo '{{ ejecutar_assolo_snd }}' | at now
```

Apéndice C

Manual de Usuario

C.1. Introducción

El siguiente documento pretende guiar al usuario en el trabajo de conocer y usar la plataforma Moniceibal.

En particular esta sección se dedica a explicar en qué tipo de entorno o red podría llegar a funcionar ya que si bien esta plataforma fue desarrollada con el objetivo primordial de funcionar en Plan Ceibal puede adaptarse a otras redes.

La plataforma permite tomar mediciones útiles para evaluar el estado de salud de la red en base a distintos módulos de medición, véase capítulo 5. Estos pueden ser instalados y gestionados en cada *host* de forma remota desde un servidor central al cual luego son enviadas las medidas. La lista de módulos es:

- *Tiempo DHCP:* tiempo que toma obtener una IP otorgada por el servidor DHCP.
- *Tiempo DNS:* tiempo promedio que toma al servidor DNS en realizar una consulta.
- *Tiempo de descarga de un sitio web:* tiempo que toma en acceder al contenido parcial o total de un sitio web seleccionado por el administrador.
- *Ancho de banda:* capacidad de los enlaces entre el servidor central y cada uno de los equipos de la red.

En un principio tanto los equipos que se encargan de realizar la tarea de monitoreo como los que son monitorizados deben correr un sistema operativo Linux como por ejemplo Ubuntu o CentOs y tener una interfaz de red con una IP asignada de forma estática.

Debe existir un punto de la red que sea capaz de acceder a todos los otros nodos mediante conexiones *ssh* y un usuario con los privilegios suficientes para realizar los cambios deseados en dichos equipos. Dichos cambios incluyen instalación de programas, edición y creación de archivos y carpetas y por último habilitación de

puertos. De ser posible las conexiones deben realizarse mediante intercambio de clave pública con el fin de no tener que almacenar contraseñas en los *scripts*.

El nodo principal, que es desde donde se realiza el monitoreo, puede estar compuesto por uno o más equipos. En el mismo deben estar instalados los servidores de Zabbix y Ansible.

La lista entera de equipos que se desea monitorizar debe estar registrada en el servidor de Zabbix. A la vez cada equipo que se utilice para tomar mediciones debe tener instalado el agente de Zabbix y debe estar programado para poder responder al servidor central. Por mas información acerca de los conceptos de servidor y agente de Zabbix, puede dirigirse al capítulo 4.

En cuanto a las conexiones que se desea realizar, es necesario que los equipos tengan ciertos puertos habilitados en su *firewall* para poder comunicarse. En primer lugar se necesita un puerto para realizar conexiones *ssh*, en segundo lugar es necesario un puerto para que el servidor de Zabbix pueda comunicarse con los agentes el cual debe estar habilitado en todos los equipos a monitorizar. Por último, Assolo, la herramienta de medición de ancho de banda que se utiliza necesita de dos puertos los cuales también deben estar habilitados en todos los equipos de la red. En todos los casos dichos puertos son asignados por defectos pero pueden ser modificados.

C.2. Requerimientos Previos

C.2.1. Estructura

Moniceibal fue desarrollada para ser instalada en una red administrada por un servidor central, el cual tenga acceso directo a cada nodo de la red vía *ssh* mediante clave compartida.

Esta plataforma asume que el servidor central consta de tres equipos, la primera donde reside el programa Zabbix, la segunda donde se instala Ansible y la última, denominada Monitoreo, que es donde corren todos los *scripts* capaces de gestionar los módulos de medición.

C.2.2. Programas y Sistemas Operativos

Servidor Central

El servidor debe constar de tres equipos Linux con procesadores de arquitecturas x86_x64 que se describen a continuación:

- *Zabbix:* Este equipo debe tener instalado el servidor de Zabbix versión 3.0.28. Dicho servidor debe tener registrada en su lista la totalidad de *host* sobre los cuales se quiere correr los módulos con el fin de poder contactarse con ellos y darles órdenes de tomar las mediciones. En caso que se ingrese la IP de algún equipo que no se encuentre registrado en Zabbix, la ejecución de cualquiera de las funciones de la plataforma dará error.

- *Ansible:* Este equipo debe contar con los siguientes programas instalados:
  - *Python:* versión 2.7.5
  - *ssh:* Este equipo debe contar con el programa *openssh-server* y ser capaz de conectarse con todos los *hosts* de la red mediante clave compartida.
  - *Ansible:* versión 2.4.2.0.
- *Monitoreo:* Este equipo debe tener instalado:
  - *Python:* versión 2.7.5 con sus paquetes Paramiko, DNS Lookup y Psutils.
  - *Paquete Dns-utils*
  - *Paquete Net-tools*
  - *Programa wget*
  - *Programa at*
  - *Programa zabbix-sender*
  - *Programa Assolo*
  - *Programa cron habilitado*
  - *Openssh-server:* Este equipo debe contar con el programa *OpenSSH-Server* y debe ser capaz de conectarse con el equipo Ansible vía *ssh* mediante clave compartida.

host:

Los nodos de la red donde se desee tomar medias deben correr sistemas operativos Linux y tener dos programas instalados. El primero es *Openssh-Server* para que Ansible pueda conectarse con los mismos y hacer los cambios deseados, el segundo es el agente de Zabbix, configurado para poder responder a las órdenes del servidor central.

C.3. Instalación y Configuración:

Para instalar la plataforma deben realizarse las siguientes acciones en cada equipo del servidor central.

- *Ansible:* Será necesario crear una carpeta a donde el equipo Monitero enviará los *playbooks*, véase apéndice B y archivos a distribuir a los *hosts*.
- *Monitoreo:*
  - Deberá crearse una carpeta donde residirán todos los archivos de la plataforma los cuales pueden ser descargados de GitLab [34] y se listan a continuación.

- *adddhcp.py*
- *adddns.py*
- *addwget.py*
- *addassolo.py*
- *deldhcp.py*
- *deldns.py*
- *delwget.py*
- *delassolo.py*
- *adddhcp.py*
- *locations.py*
- *prueba_ancho_de_banda.py*
- *configuracion_dns.yml*
- *configuracion_wget.yml*
- *ancho_de_banda_playbook.yml*
- *crear_item.yml*
- *dns_configuración*
- *wget_configuración*
- *assolo_lista_ip*
- *assolo_tiempo*
- *ancho_de_banda*
- *dhcp_time*
- *dns_time*
- *wget_time*
- *assolo-0.9a:* Dentro de esta carpeta deberá estar instalado el programa Assolo.

- Aparte de los puertos necesarios para realizar conexiones *ssh*, el programa Assolo también necesita tener los puertos 7365 y 8365 habilitados en su *firewall* para poder recibir paquetes UDP y TCP.
- Dentro de la carpeta situada en el equipo monitoreo se deberá modificar el archivo locations.py. En este archivo se define un grupo de variables compartidas por todas las funciones de la plataforma.

  Cada función necesita tener cierta información para poder realizar las tareas requeridas por el usuario, entre esta información se encuentra la ruta al directorio de donde ir a buscar los módulos a distribuir, el usuario y clave de Zabbix, entre otros. Los dos últimos parámetros son necesarios para poder loguearse en la Zabbix API, utilizada por la plataforma para desempeñar diferentes tareas. El hecho de tener un archivo centralizado simplifica algunas tareas para el usuario, por ejemplo si cambiara el directorio donde se encuentras los módulos alojados, sólo debe cambiar la variable correspondiente a esta información en el

archivo locations.py. De otra forma debería ingresar está información para cada función que utilice este parámetro.

El archivo se compone por las siguientes variables:

user_zabbix = Usuario de Zabbix.
password= Contraseña de Zabbix.
ip_zabbix = IP privada donde corre Zabbix.
ip_zabbix_zender= IP pública a la que cada *host* debe reportar sus medidas.
ip_ansible = IP donde corre Ansible.
ip_plataforma = IP donde corre plataforma.
usuario_plataforma = Usuario donde estará la plataforma.
carpeta_plataforma = Ruta del directorio donde se encuentran los archivos a utilizar por la plataforma.
carpeta_ansible = Ruta de la carpeta situada en el equipo donde se aloja Ansible, en la misma se copian los archivos desde la plataforma y que luego serán distribuidos a los diferentes *host*.
carpeta_destino_odroid = Ruta de carpeta donde se alojaran los módulos y archivos de configuración.
usuario_host = Nombre de usuario del equipo *host*, por defecto es Odroid-C2.

C.4. Funciones Básicas

La plataforma cuenta con un conjunto de funciones básicas programadas en Python capaces de instalar o deshabilitar los módulos en los *hosts* seleccionados por el administrador. Estas funciones se encargan de realizar todos los cambios necesarios en cada *host* usando Ansible y crean lo *items* necesarios en Zabbix para el funcionamiento de la plataforma.

C.4.1. Módulo DHCP

El módulo DHCP permite medir el tiempo en segundos que demora el servidor DHCP en responder a una solicitud de una nueva IP. Para instalar o deshabilitar este módulo existen las siguientes funciones:

- **adddhcp.py:**
  - Descripción:
    adddhcp.py crea o modifica el módulo DHCP para el conjunto de IPs seleccionadas. Sus parámetros son los siguientes:
    - *-f*: Consiste de un parámetro opcional el cual indica la frecuencia con la cual se tomará la medición. Debe ser introducido en la

sintaxis de Zabbix ya que su valor sera pasado directamente a dicho programa.

Existen tres parámetros capaces de introducir la lista de IPs de equipos sobres los cuales crear los módulos. Dado que las funciones deben funcionar si o si sobre alguna de estas IPs, es necesario introducir uno y solo uno de dichos parámetros.

- *-u*: Seguido de este parámetro se ingresa directamente la lista de IPs sobre la linea de comando. Es importante que dichos *hosts* estén registrados en el servidor de Zabbix, de lo contrario la función mostrara un mensaje de error.
- *-y*: Se ingresa la ruta hacia un archivo de texto que contenga la lista de IPs solicitada, en dicho archivo la lista de IPs debe estar ingresada en una sola columna, en caso contrario la plataforma dará error. Al igual que en la opción anterior los *hosts* introducidos deberán estar registrados en Zabbix.
- *-g*: Se escribe el nombre de un grupo de *host* de Zabbix para que la función misma se encargue de extraer las IPs contenidas en el mismo y instalar el módulo en ellas.
- *-force*: Se utiliza esta opción para forzar a la plataforma a distribuir los módulos independientemente de si están creados los *items* en Zabbix. Es importante incluir esta funcionalidad dado que si en un momento dado se cambia un equipo por otro, la plataforma al ver que los *items* están creados (cambia el equipo no la IP, Zabbix asocia los *items* a la IP no al equipo) no distribuiría los módulos de medición. Lo cuál sería un error por lo tanto se incluye esta opción que ignora el hecho de que estén creados los *items* y distribuye los archivos. Por otro lado *-force* también da la posibilidad de poder actualizar los módulos.

Ejemplos de uso de la función:

- `adddhcp.py -u 192.0.2.2 192.0.2.3 -f 30:` Crea el módulo en los equipos de IP 192.0.2.2 y 192.0.2.3 con una frecuencia de medición de 30 segundos.
- `adddhcp.py -y [ruta al archivo] -f 30:` Crea el módulo en los equipos cuya IP se encuentre dentro del archivo seleccionado con una frecuencia de medición de 30 segundos.
- `adddhcp.py -g [nombre de grupo de Zabbix] -f 30:` Crea el módulo en los equipos que se encuentren dentro del grupo de Zabbix seleccionado con una frecuencia de medición de 30 segundos.

- **deldhcp.py:**

  - Descripción:
    Esta opción se encarga de deshabilitar el *item* que dan la orden de

ejecución del módulo en Zabbix sin borrar ningún archivo o *item*. De esta manera la medida cesa de tomarse pero el módulo sigue existiendo y conservando todos los datos adquiridos. Sus parámetros son *-u*, *-y* y *-g*, estos funcionan de la misma forma que en la función anterior permitiendo ingresar una IP o lista. Dado que la función debe sí o sí trabajar sobre una IP, debe introducirse uno y sólo uno de dichos parámetros.

Ejemplo de uso de la función:

- `deldhcp.py -u 192.0.2.2 192.0.2.3:` Deshabilita el módulo en los equipos con IP 192.0.2.2 y 192.0.2.3.

C.4.2. Módulo DNS

El módulo de tiempo DNS permite medir el tiempo promedio de respuesta del servidor DNS en base a un número arbitrario de *hostnames* introducidos por el administrador. Las funciones que instalan o desinstalan este módulo son las siguientes:

- **adddns.py:**
  - Descripción:
    adddns.py crea o modifica el módulo tiempo DNS para el conjunto de IPs seleccionadas. Contiene los mismos parámetros que el módulo *adddhcp.py* más un nuevo parámetro que se describe a continuación.
    - *-s:* El mismo es seguido por una lista de *hostnames* introducidos directamente desde la linea de comandos con los cuales el administrador desea tomar la medida de tiempo DNS.

  Ejemplo de uso de la función:

  - `adddns.py -u 192.0.2.2 192.0.2.3 -s www.google.com www.yahoo.com -f 30`: Crea el módulo en los equipos con IP 192.0.2.2 y 192.0.2.3 para que los mismos realicen la medida en base a los *hostnames* introducidos cada 30 segundos. La medida será el resultado de realizar una consulta DNS para cada *hostname* y luego promediar los tiempos de respuesta.

- **deldns.py:**
  - Descripción:
    Esta cumple la misma función que *deldhcp.py* pero para el módulo de tiempo DNS, sus parámetros de entrada son exactamente los mismos, así como su funcionamiento interno, la única diferencia es el *key* del *item* que deshabilitan.

  Ejemplo de uso de la función:

- `deldns.py -y [ruta al archivo]`: Deshabilita el módulo para todos los equipos cuya IP se encuentra ingresada dentro del archivo.

C.4.3. Módulo de Tiempo de Descarga de Sitio web

Este módulo realiza la descarga de un sitio web seleccionado por el administrador mediante el comando *wget* y cualquiera de sus opciones, con el fin de obtener una estimativa de los tiempo de acceso a la web para los usuarios. En este caso se cuenta con las siguientes funciones:

- **addwget.py:**

  - Descripción:
    Esta crea o modifica el módulo tiempo de descarga de sitio web para el conjunto de IPs seleccionadas. Contiene los mismos parámetros que las funciones *adddns.py* salvo por el funcionamiento del parámetro *-s:* el cual sera descrito a continuación.

    - *-s:* A continuación de este parámetro debe ingresarse un *hostname* y la opción del comando *wget* con la cual se quiere realizar la descarga. Esta función acepta todas las opciones del comando *wget*, esto significa que entre otras cosas es posible seleccionar el nivel de recursividad y de profundidad de descargar del sitio web seleccionado.

  Ejemplo de uso de la función:

  - `addwget.py -u 192.0.2.2 192.0.2.3 -s www.google.com -m -f 30`: Crea el módulo en los equipos con IP 192.0.2.2 y 192.0.2.3 para que los mismos realicen la medida en base a los *hostnames* y las opciones introducidas cada 30 segundos. La opción -m indica descargar la totalidad del sitio web, por lo tanto la medida sera el resultado de descargar totalmente el sitio www.google.com.

- **delwget.py:**

  - Descripción:
    El funcionamiento de esta función es exactamente igual al de las funciones *deldhcp.py* y *deldns.py* con la única diferencia de que se encargan de deshabilitar el módulo de tiempo de descarga de sitio web.

  Ejemplo de uso de la función:

  - `delwget.py -g [nombre de grupo de Zabbix]`: Deshabilita el módulo para todos los equipos pertenecientes al grupo de Zabbix seleccionado.

C.4.4. Módulo Medida de Ancho de Banda:

El mismo se encarga de medir la capacidad del enlace entre cada *host* seleccionado y el servidor central. Las funciones que gestionan este módulo son las siguientes:

- **addassolo.py:**

  - Descripción:
    Se encarga de crear el módulo de tiempo de ancho de banda para los equipos seleccionados. Los parámetros que le permiten ingresar una IP son los mismos que en los módulos anteriores, con la diferencia de que aquí no es obligatorio ingresar ninguno de ellos, pues dicha función puede ser usada para por ejemplo solo modificar la frecuencia de ejecución o el lapso de tiempo que toma cada medida. El resto de los parámetros son diferentes y se describen a continuación.
    - *-t parámetros*: es el valor del tiempo que va a durar cada medida, es opcional y su valor es un entero en segundos. Por defecto es 3600 segundos (una hora).
    - *-f parámetros*: es el valor de la frecuencia a la cual se repetirán las pruebas en Zabbix. Es opcional, su valor tiene que estar entre comillas en formato de *crontab* y si se escribe de manera incorrecta dará un error. Esto se debe a que en este caso Zabbix no es el encargado de dar las órdenes de ejecución, sino que esta es tarea del programa *cron*.

  Ejemplo de uso de la función:

  - *`addassolo.py -u 192.0.2.2 192.0.2.3 -t 20 -f ''*/10 * * * *''`*:
    De esta forma las mediciones que se comenzarán a tomar para las IPs 192.0.2.2 y 192.0.2.3. Cada medición durará 20 segundos y se realizarán cada 10 minutos cada una. Ha de destacarse que la frecuencia fue introducida entre comillas, esto se debe a que el carácter asterisco tiene un significado especial para la terminal de Linux y de no introducir las comillas la frecuencia no sera interpretada como tal.

- **delassolo.py:**

  - Descripción:
    A diferencia de las funciones deshabilitadoras anteriores, ésta tiene dos funcionalidades. La primera al igual que antes es simplemente deshabilitar la toma de medidas para las IP ingresadas. La segunda consiste en ingresar el parámetro *-stop* el cual detiene completamente el funcionamiento del módulo para todas las direcciones IPs de la lista.

  Ejemplos de uso de la función:

- `delassolo.py -g [nombre de grupo de Zabbix]`: Deshabilita el módulo para todos los equipos pertenecientes al grupo de Zabbix seleccionado.
- `delassolo.py -stop`: Deshabilita el funcionamiento del módulo borrando la linea de *crontab* que se encarga de dar la orden de ejecución de las medidas. Por otro lado, aunque ya no se tomen más medidas, la lista de IP contra las cuales debería tomarse permanece intacta.

C.5. Errores de Ejecución de las Funciones

Al ejecutar las funciones recientemente descritas es posible que el usuario cometa errores al introducir sus parámetros. En caso de que la función detecte un error, la misma detendrá su funcionamiento e informará al usuario del error ocurrido.

- **adddhcp**
  - "Más de una vez pasado parámetro *-f*": dicho parámetro puede escribirse como máximo una vez.
  - "Más de una vez pasado parámetro *-u*": dicho parámetro puede escribirse como máximo una vez.
  - "Más de una vez pasado parámetro *-y*": dicho parámetro puede escribirse como máximo una vez.
  - "Más de una vez pasado parámetro *-g*": dicho parámetro puede escribirse como máximo una vez.
  - "Más de una vez pasado parámetro *-force*": dicho parámetro puede escribirse como máximo una vez.
  - "Más de una opción de grupo IP seleccionada": cada vez que se ejecuta la función, sólo puede introducirse uno solo de los parámetros *-u*, *-y* o *-g*.
  - "No hay grupo de IP seleccionada": cada vez que se ejecuta la función, debe introducirse por lo menos uno de los parámetros *-u*, *-y* o *-g*.
  - "Una de las IP pasada no es válida": todas las IPs ingresadas deben estar registradas en el servidor de Zabbix.
  - "No se ingresaron IPs": se ingresó el parámetro *-u* sin ninguna IP a continuación.
  - "Una de las IP pasada en la lista no es válida": Se escribió el parámetro *-y* y el nombre de un archivo, una de las IPs contenidas en el archivo no se encontraba registrada en Zabbix.
  - "Error en lectura de archivo": Se escribió el parámetro *-y* y el nombre de un archivo, el mismo no se lo pudo leer.
  - "No existe este grupo en Zabbix": Se ingresó el nombre de un grupo no existente en Zabbix.

- "No se ingresó grupo de Zabbix": Se escribe el parámetro *-g* a continuación se debe de pasar el nombre del grupo
- "Verifique los parámetros introducidos": se escribieron parámetros inválidos, no reconocidos por la función.
- "La frecuencia no se introdujo correctamente, se usará el valor por defecto de una hora": la frecuencia no se escribió en el formato correcto (formato Zabbix).

- **adddns**
  Esta función contiene los mismos mensajes de error que la función anterior más los siguientes:
  - "Más de una vez pasado parámetro *-s*": dicho parámetro puede escribirse como máximo una vez.
  - "No se introdujo sitio web después del parámetro *-s*": Se escribió el parámetro *-s* sin ningún *hostname* a continuación.
  - "El *hostname* (nombre dirección web) no corresponde a ningún sitio web": Se introdujo el parámetro *-s* y un sitio web no valido a continuación.

- **addwget**
  Esta función contiene los mismos mensajes de error que la función *adddhcp* más los siguientes:
  - "más de una vez pasado parámetro *-s*": ídem a función *adddns*.
  - "El comando *wget* o el *hostname* o las opciones introducidas no son correctas": El *hostname* u opción pasada después del parámetro *-s* no son validas.

- **addassolo**
  - "más de un parámetro *-t*": dicho parámetro puede escribirse como máximo una vez.
  - "más de una vez pasado parámetro *-g*": ídem a función *adddhcp*.
  - "más de una vez pasado parámetro *-u*": ídem a función *adddhcp*.
  - "más de una vez pasado parámetro *-y*": ídem a función *adddhcp*.
  - "más de una vez pasado parámetro *-force*": ídem a función *adddhcp*.
  - "más de una opción de grupo IP seleccionada": ídem a función *adddhcp*.
  - "no hay grupo de IP seleccionada": ídem a función *adddhcp*.
  - "una de las IP pasada no es válida": ídem a función *adddhcp*.
  - "No se ingresaron IPs": ídem a función *adddhcp*.
  - "Una de las IP pasada en la lista no es válida": ídem a función *adddhcp*.

- "Error en lectura de archivo": ídem a función *adddhcp*.
- "Una de las IP pasada no es válida": ídem a función *adddhcp*.
- "No existe este grupo en Zabbix": ídem a función *adddhcp*.
- "No se ingresó grupo de zabbix": ídem a función *adddhcp*.
- "Verifique los parámetros introducidos" ídem a función adddhcp.
- "La sintaxis de *crontab* no es correcta por lo tanto no se tomarán las medidas": la frecuencia no fue introducida en una frecuencia reconocida por *crontab*.

- **deldns, deldhcp y delwget**
  Estas funciones contienen todos los mensajes de error que contiene *adddhcp.py* a excepción del mensaje "La frecuencia no se introdujo correctamente, se usará el valor por defecto de una hora".

- **delassolo**
  Esta función contienen todos los mensajes de error que contienen las funciones *deldns*, *deldhcp* y *delwget* más el mensaje:

  - "Más de una vez pasado parámetro *-stop*": el parámetro debe de ser escrito sólo una vez.

C.6. Errores de Ejecución de los Módulos

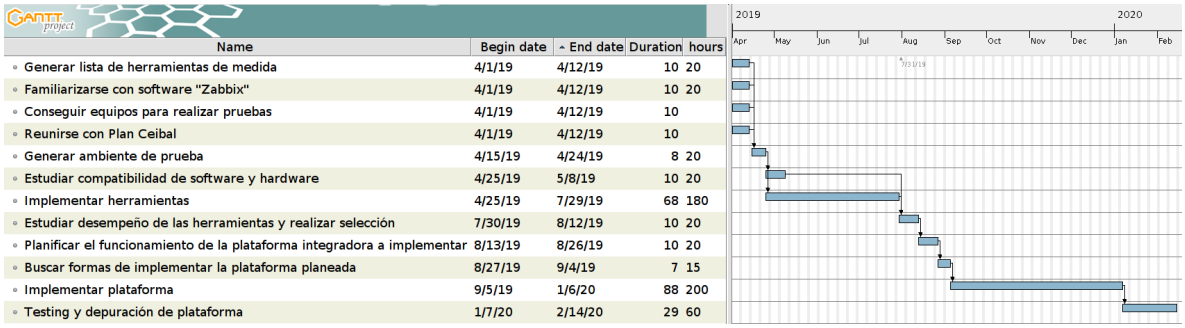
Existe un *item* en Zabbix llamado Errores que es común a todos los módulos y se encarga de recibir mensajes avisando de errores ocurridos mientras se ejecutan los módulos.

- Error_Módulo_DHCP_sin_respuesta: El módulo de tiempo DHCP no recibió respuesta al pedido de IP.
- Error_Módulo_Dns_No_se_Encontró_Servidor_para_el_*hostname*_[número del *hostname*]: Al realizar la consulta DNS no se encontró servidor que aloja el *hostname* mencionado.
- Error_Módulo_WGET_*hostname*_no_existente: El *hostname* utilizado para realizar la medida de tiempo de descarga de sitio web no fue encontrada.
- Error_Módulo_WGET_timeout: La descarga realizada en el módulo de tiempo de descarga de sitio web excedió el límite de tiempo de cinco minutos.
- Error_Módulo_Ancho_de_Banda_No_se_pudo_tomar_la_medida_de_ancho _de_banda_de_subida_para_la_ip_[IP]: La medida ancho de banda de subida no se pudo realizar para la IP en cuestión.
- Error_Módulo_Ancho_de_Banda_No_se_pudo_tomar_la_medida_de_ancho _de_banda_de_bajada_para_la_ip_[IP]: La medida ancho de banda de bajada no se pudo realizar para la IP en cuestión.

Apéndice D

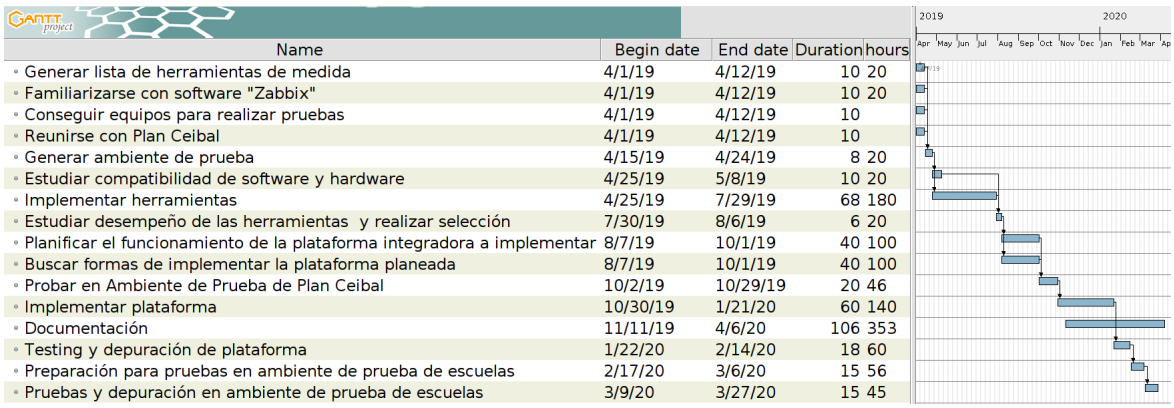
Gestión de proyecto

La figura D.1 muestra el cronograma ideado al comienzo del proyecto y la figura D.2 muestra el cronograma con las fechas en que realmente se llevaron a cabo las tareas en formato de Gantt Project [27]. A continuación se explican las diferencias entre uno y el otro.



Name	Begin date	End date	Duration	hours
Generar lista de herramientas de medida	4/1/19	4/12/19	10	20
Familiarizarse con software "Zabbix"	4/1/19	4/12/19	10	20
Conseguir equipos para realizar pruebas	4/1/19	4/12/19	10	
Reunirse con Plan Ceibal	4/1/19	4/12/19	10	
Generar ambiente de prueba	4/15/19	4/24/19	8	20
Estudiar compatibilidad de software y hardware	4/25/19	5/8/19	10	20
Implementar herramientas	4/25/19	7/29/19	68	180
Estudiar desempeño de las herramientas y realizar selección	7/30/19	8/12/19	10	20
Planificar el funcionamiento de la plataforma integradora a implementar	8/13/19	8/26/19	10	20
Buscar formas de implementar la plataforma planeada	8/27/19	9/4/19	7	15
Implementar plataforma	9/5/19	1/6/20	88	200
Testing y depuración de plataforma	1/7/20	2/14/20	29	60

Figura D.1: Cronograma Inicial



Name	Begin date	End date	Duration	hours
Generar lista de herramientas de medida	4/1/19	4/12/19	10	20
Familiarizarse con software "Zabbix"	4/1/19	4/12/19	10	20
Conseguir equipos para realizar pruebas	4/1/19	4/12/19	10	
Reunirse con Plan Ceibal	4/1/19	4/12/19	10	
Generar ambiente de prueba	4/15/19	4/24/19	8	20
Estudiar compatibilidad de software y hardware	4/25/19	5/8/19	10	20
Implementar herramientas	4/25/19	7/29/19	68	180
Estudiar desempeño de las herramientas y realizar selección	7/30/19	8/6/19	6	20
Planificar el funcionamiento de la plataforma integradora a implementar	8/7/19	10/1/19	40	100
Buscar formas de implementar la plataforma planeada	8/7/19	10/1/19	40	100
Probar en Ambiente de Prueba de Plan Ceibal	10/2/19	10/29/19	20	46
Implementar plataforma	10/30/19	1/21/20	60	140
Documentación	11/11/19	4/6/20	106	353
Testing y depuración de plataforma	1/22/20	2/14/20	18	60
Preparación para pruebas en ambiente de prueba de escuelas	2/17/20	3/6/20	15	56
Pruebas y depuración en ambiente de prueba de escuelas	3/9/20	3/27/20	15	45

Figura D.2: Cronograma Final

El primer cambio visible en los diagramas fue debido a que la etapa de "Estudiar desempeño de las herramientas y realizar selección" logró realizarse en un

tiempo menor al previsto, en principio se supuso que podría tomar 10 días pero terminó llevando 6 días.

Luego se puede ver cómo inicialmente se estimó que la etapa de "Planificar el funcionamiento de la plataforma integradora a implementar" iba a llevar 10 días pero terminó llevando 40 días, esto se debió a que integrar los diferentes componentes del sistema resultó una tarea más compleja de lo previsto dado que algunos componentes no estaban particularmente diseñados para poder trabajar en conjunto.

Lo mismo ocurrió con la etapa "Buscar formas de implementar la plataforma planeada" ya que también, todo lo que se planeó, resultó mucho más complejo de implementar en la práctica debido a nuevas complicaciones que surgieron.

A continuación se puede ver que aparece una nueva etapa "Probar en Ambiente de Prueba de Plan Ceibal" que no estaba en el diagrama original, esto se debe a que Plan Ceibal ofreció un ambiente de prueba más similar al entorno real en el cual debería funcionar la plataforma, por esto es que se decidió tomar un lapso de 20 días para probar parte del código ya desarrollado en la misma. Luego puede verse cómo las etapas "Implementar Plataforma" y "Testing y depuración de plataforma" terminaron llevando menos tiempo del previsto, esto sólo se debió a una sobre estimación inicial de las mismas.

A continuación se agregaron dos etapas nuevas "Preparación para prueba en ambiente de prueba de escuelas" y "Prueba y depuración en ambiente de prueba de escuelas", esto se debió a la oferta por parte de Plan Ceibal de probar los módulos en un ambiente de prueba que simula el entorno de una escuela.

Por último se agregó la etapa de "Documentación" al diagrama, la cual no había sido tenida en cuenta en la realización del diagrama inicial.

Referencias

[1] Sitio web de Plan Ceibal con información sobre su infraestructura [Online] Available: https://www.ceibal.edu.uy/es/articulo/ceibal-en-cifras

[2] Rihards Olups "Zabbix Network Monitoring - Second Edition" August 10, 2016

[3] Jim Kurose - Keith Ross "Computer Networking: A Top-Down Approach" Fifth Edition, Pearson 2010

[4] Goldman, George (17 January 2007). "The network graphing solution Cacti was designed to provide more ease of use than RRDtool and more flexibility than MRTG". ISP-Planet. Retrieved 16 March 2012.

[5] Documentación de Ansible [Online] Available: https://docs.ansible.com/ansible/latest/index.html

[6] RFC 1157 "A Simple Network Management Protocol" [Online] Available: https://tools.ietf.org/html/rfc1157

[7] RFC 1034 "DOMAIN NAMES - CONCEPTS AND FACILITIES" [Online] Available: https://tools.ietf.org/html/rfc1034

[8] RFC 1034 "DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION" [Online] Available: https://tools.ietf.org/html/rfc1035

[9] RFC 2131 "Dynamic Host Configuration Protocol" [Online] Available: https://tools.ietf.org/html/rfc2131

[10] "Nagios - DHCP Monitoring With Nagios" [Online] Available: https://www.nagios.com/solutions/dhcp-monitoring/

[11] "Nagios - DNS Monitoring" [Online] Available: https://www.nagios.com/solutions/dns-monitoring/

[12] Sitio web con información del sistema Cron [Online] Available: https://www.linuxtotal.com.mx/?cont=info_admon_006

[13] "Linux wget command" [Online] Available: https://www.computerhope.com/unix/wget.htm

[14] Deepak Chahal, Latika Kharb, Deepanshu Choudhary - "Performance Analytics of Network Monitoring Tools" International Journalof Innovative Technology and Exploring Engineering (IJITEE) June, 2019 [Online] Available: https://www.ijitee.org/wp-content/uploads/papers/v8i8/H7081068819.pdf

[15] Sitio web de Wireshark [Online] Available: http://www.wireshark.org

[16] Sitio web de Assolo [Online] Available: http://netlab-mn.unipv.it/assolo/

[17] E.Goldoni, G. Rossi, and A. Torelli, "Assolo, a new method for available bandwidth estimation" in Proc. IARIA International Conference on Internet Monitoring and Protection (ICIMP 2009), May 2009, pp. 130–136. [Online] Available: https://www.researchgate.net/publication/224506567_Assolo_a_New_Method_for_Available_Bandwidth_Estimation

[18] Lao, Li & Dovrolis, Constantine & Sanadidi, M.Y.. (2006). The Probe Gap Model can underestimate the available bandwidth of multihop paths. Computer Communication Review. 36. 29-34. 10.1145/1163593.1163599. [Online] Available: https://www.researchgate.net/publication/220195337_The_Probe_Gap_Model_can_underestimate_the_available_bandwidth_of_multihop_paths

[19] Sitio web y documentación de Paramiko . [Online] Available: http://www.paramiko.org/

[20] Sitio web y documentación de Pchar . [Online] Available: https://www.kitchenlab.org/www/bmah/Software/pchar/

[21] Q. Wang and L. Cheng. FEAT: Improving accuracy in end-to-end available bandwidth measurement. In Global Telecommunications Conference GLOBECOM, Nov. 2006.

[22] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cot-trell. pathchirp: Efficient available bandwidth estimation for network paths. In Proceedings of 4th PAM Workshop, San Diego, CA, USA, April 2003.

[23] Sitio web y documentación de Cygwin. [Online] Available: https://www.cygwin.com/

[24] Documentación de CFEngine [Online] Available: https://docs.cfengine.com/docs/3.12/reference.html

[25] A. Botta, A. Dainotti, A. Pescapè, .ª tool for the generation of realistic network workload for emerging networking scenarios", Computer Networks (Elsevier), 2012, Volume 56, Issue 15, pp 3531-3547 [Online] Available: http://wpage.unina.it/a.botta/pub/COMNET_WORKLOAD.pdf

[26] Performance Analytics of Network Monitoring Tools [Online] Available: https://www.ijitee.org/wp-content/uploads/papers/v8i8/H7081068819.pdf

[27] Sitio web y documentación de Gantt Project [Online] Available: https://www.ganttproject.biz/

[28] Odroid Magazine [Online] Available: https://magazine.odroid.com/wp-content/uploads/ODROID-Magazine-201401.pdf

[29] Odroid Comparation Raspbeery 3 B [Online] Available: https://www.jeffgeerling.com/blog/2016/review-odroid-c2-compared-raspberry-pi-3-and-orange-pi-plus

[30] Manual Odroid Xu4 [Online] Available: https://magazine.odroid.com/odroid-xu4

[31] Manual Odroid C2 [Online] Available: https://magazine.odroid.com/odroid-c2

[32] Raspberry 3 B+ [Online] Available: https://www.xataka.com/ordenadores/raspberry-pi-3-model-b-analisis-mas-potencia-y-mejor-wifi-para-un-minipc-que-sigue-asombrando

[33] Raspberry 3 B+ [Online] Available: https://www.xataka.com/ordenadores/raspberry-pi-4-analisis-caracteristicas-precio-especificaciones

[34] Sitio web de Gitlab [Online] Available: https://about.gitlab.com/

Esta página ha sido intencionalmente dejada en blanco.

Índice de tablas

Esta página ha sido intencionalmente dejada en blanco.

Índice de figuras

Esta es la última página.
Compilado el miércoles 24 junio, 2020.
http://iie.fing.edu.uy/