

# Documentación del proyecto

Barrera Pérez Carlos Tonatihu  
Profesor: Eduardo Gutiérrez Aldana  
Administración de servicios en red  
Grupo: 4CM2

12 de diciembre de 2018

# Índice

<b>1. Descripción del proyecto</b>	<b>3</b>
1.1. Herramientas utilizadas . . . . .	3
<b>2. Desarrollo</b>	<b>3</b>
2.1. Servidor . . . . .	3
2.2. Enrutador . . . . .	4
2.3. Gestor . . . . .	4
2.3.1. Utilización CPU . . . . .	5
2.3.2. Retardo en respuesta del ping . . . . .	6
2.3.3. Tiempo de respuesta del servidor . . . . .	8
2.3.4. Memoria real . . . . .	9
2.3.5. Monitoreo de CPU y memoria . . . . .	10
2.3.6. Monitoreo de interfaces . . . . .	11

# 1. Descripción del proyecto

El proyecto consiste de una topología de red en la cual se presentan enrutadores, switches, servidores, gestor y clientes. Los enrutadores y servidor son monitoreados por el gestor con el objetivo de llevar un registro de los datos que estos presentan y poder detectar fallas en los equipos.

## 1.1. Herramientas utilizadas

El proyecto fue desarrollado en el ambiente de simulación *GNS3* que se encuentra integrado dentro del sistema operativo *Live Raizo - Linux for Virtual SysAdmin* en su versión *8.17.08.30p* (<https://sourceforge.net/projects/live-raizo/files/>) ya que es la ultima versión que incluye *Virtual Box* que se utilizo para trabajar con las maquinas virtuales que se trabaron.

Para poder trabajar de forma adecuada se adecuo una USB a modo de arranque con *Raizo* y para evitar la perdida del trabajo y guardar los datos que se encuentran en **/home** y **/opt** se creo una partición en el disco duro de la computadora de trabajo. Esta partición se formateo como **ext4** y se le etiqueto como **persistence** y se le agrego un archivo llamado **persistence.conf** con la siguiente información.

```
1 /home
2 /opt union
3 /root union
```

Es importante mencionar que este método de crear persistencia se puede realizar sobre una USB y tener una USB para el boot y otra para la persistencia o ambas características en una sola USB.

En el DVD que se entregó se encuentran las carpetas **home**, **opt** con las maquinas virtuales que se trabajaron y el archivo **persistence.conf**.

De manera general otras herramientas que se utilizaron fueron RCP100 y TinyCore como maquinas virtuales para los enrutadores, el gestor y el servidor.

# 2. Desarrollo

## 2.1. Servidor

El servidor web fue implementado en *Tiny Core* utilizando el paquete *Busybox HTTPD*, el codigo que se encarga de esto se encuentra en el archivo

`/opt/bootlocal.sh` para que cada vez que se encienda la computadora el servidor comience a trabajar. El código es el siguiente.

```
1 /usr/local/httpd/sbin/httpd -p 80 -h /home/tc/site
```

Lo que hace este código es activar el servicio de HTTP proporcionado por *Busybox* en el puerto 80 indicando que la página a mostrar (`index.html`) se encuentra en la carpeta `/home/tc/site`. Además, se debe de configurar una interfaz de la maquina virtual para que se pueda acceder al servidor.

## 2.2. Enrutador

Para el enrutador se utilizo RCP100, en cada uno de los enrutadores se configuraron las interfaces y el enrutamiento dinámico OSPF. Además de configurar el agente snmp necesario para llevar a cabo el monitoreo de interfaces, memoria y uso de CPU, esta configuración se realizo de la siguiente forma.

```
1 snmp-server contact TalesDeMileto
2 snmp-server location Topolovampo
3 snmp-server community soporte ro
```

## 2.3. Gestor

El gestor fue implementado en una maquina virtual con Tiny Core la cual es una distro de linux bastante liviana ya que tiene muy pocos paquetes preinstalados, por lo cual se agregaron diversos paquetes para que pueda funcionar de forma correcta, estos paquetes fueron.

- net-snmp. Necesario para utilizar el protocolo de monitoreo SNMP.
- swaks. Es un script que permite realizar pruebas con un servidor SMTP.
- bc. Es un lenguaje de calculadora utilizado para realizar operaciones matemáticas en bash
- curl. Es una herramienta para probar diferentes protocolos, en este caso se utilizo para realizar pruebas en el servidor http.
- rrdtool. Es una herramienta para trabajar con bases de datos circulares, con dicha herramienta se guarda la información que se consulta y ya que es circular no crece demasiado a pesar de que se genere una gran cantidad de datos.

- perl. Es necesario ya que el script de swaks lo utiliza para poder funcionar.

Todos los scripts que se utilizan para el monitoreo están elaborados en bash ya que es un lenguaje de scripting que se encuentra en muchas distribuciones de linux. Todos los scripts se ejecutan cada segundo como un demonio gracias al uso de crontab y para poder tener un control de todos los elementos que se gestionan se crea una carpeta por cada IP a monitorear y dentro de cada carpeta se crean los respectivos archivos de RRDtool para guardar datos.

### 2.3.1. Utilización CPU

El siguiente scripts es el encargado de determinar si se presenta una incidencia por exceso en el uso de CPU, los variables que se utilizan para cambiar su comportamiento son la lista de direcciones IP a monitorear, para el proyecto se especifico una por enrutador, el limite que se va a tolerar, la comunidad SNMP a la cual pertenece el agente, el directorio del archivo de incidencias en donde se registran los problemas que ocurren.

```

1  #!/bin/bash
2  # uno.sh
3  # A1 la utilizacion del CPU excede del 60%
4  # Lista de direcciones IP a monitorear separadas por un espacio
5  direcciones="10.0.1.154"
6  # limite a utilizar
7  LIMITE=60
8  # comunidad snmp
9  comunidad=public
10 # directorio donde se guardan los archivos que se generen
11 directorio=/home/tc/gestor
12
13 # Funcion de comparacion, retorna 1 si valor es mayor a limite
14 comparacion() {
15     local me=$(awk -v lim="$1" -v valor="$2" 'BEGIN { printf (
16         valor>lim?1:0) }')
17     echo "$me"
18 }
19
20 for direccion in $direcciones; do
21     # comando snmp y oid a consultar
22     usado=$(snmpget -v 1 -c $comunidad -OQv $direccion
23         1.3.6.1.2.1.25.3.3.1.2.768)
24
25     # No debe de ser mayor que el LIMITE por ciento
26     eva=$(comparacion $LIMITE $usado)

```

```

26     if [ $eva -eq 1 ]; then
27         # Si se excede se guarda una incidencia y se manda un
        correo electronico
28         fecha=$(date +%y/%m/%d-%H:%M:%S)
29         # Se registra la incidencia en el directorio y en el
        archivo incidencias
30         echo "A1;$direccion;utilizacion CPU;$fecha" >>
        $directorio/incidencias.log
31         #Codigo para enviar un correo
32         # perl swaks --to "destino@gestor.com" \
33         # --from "soporte@gestor.com" \
34         # -s 10.0.1.11:1025 \
35         # --data "Date: %DATE%\nTo: %TO_ADDRESS%\nFrom: %
        FROMADDRESS%
36         # \nSubject: A1; $direccion; utilizacion CPU
37         # \nX-Mailer: swaks v20181104 jetmore.org/john/code/swaks/
38         # \n%NEW_HEADERS%\n"
39     fi
40 done

```

La secuencia de pasos que sigue para realizar su trabajo es la siguiente.

1. Realizar la consulta del OID correspondiente al uso de CPU, en este caso el 1.3.6.1.2.1.25.3.3.1.2.768.
2. Compararlo con la función de comparación con el límite y en el caso de que sea mayor se obtendrá un uno, cero en caso contrario.
3. Si el resultado del paso anterior es uno se obtiene la fecha del sistema y se registra la incidencia en el archivo incidencias.log, en caso de contar con un servidor de correos se podría enviar un correo ya que se cuenta con el código necesario para hacerlo se utiliza *swaks*.

### 2.3.2. Retardo en respuesta del ping

Este script se encarga de hacer un ping a cada una de las IP que se pongan en la variable direcciones y se guarda el valor en una base de datos de RRDtool que esta configurada para llevar un registro del promedio del tiempo de respuesta a lo largo de una hora. Además, si se presenta un tiempo de respuesta mayor a 5 segundos (5000 milisegundos) se registra la incidencia en el archivo incidencias.log y en caso de contar con un servidor SMTP se podría enviar un correo electrónico.

```

1 #!/bin/bash
2 # tres.sh

```

```

3 # A3 Retardo en respuesta del ping mayor a 5 segundos      A3; IP
   ; retardo en ping
4 # Lista de direcciones IP a monitorear separadas por un espacio
5 direcciones="10.0.1.254"
6 # Tiempo limite en milisegundos
7 LIMITE=5000
8 # directorio donde se guardan los archivos que se generen
9 directorio=/home/tc/gestor
10
11 # retorna 1 si valor es mayor que limite
12 comparacion() {
13     local me=$(awk -v lim="$1" -v valor="$2" 'BEGIN { printf (
14         valor>lim?1:0) }')
15     echo "$me"
16 }
17 # variable=$(date +%y/%m/%d-%H:%M:%S)
18 for direccion in $direcciones; do
19     # Si no existe se crea un subdirectorio para la ip
20     # monitorear
21     subdirectorio="$directorio/$direccion"
22     if [ ! -d $subdirectorio ]; then
23         mkdir $subdirectorio
24     fi
25     base="$subdirectorio/ping.rrd"
26     # Si no existe se crea la base de rrdtool del ping
27     if [ ! -f $base ]; then
28         rrdtool create $base --step 60 \
29             DS:tiempo:GAUGE:120:U:U \
30             RRA:AVERAGE:0.5:1:60
31     fi
32     # Obtencion del tiempo de respuesta en ms
33     tiempo=$(ping $direccion -c 1 | awk -F'time=' '/time=/ {
34         print $2}' | awk -F' ' '{print $1}')
35     #echo $tiempo
36     rrdtool update $base N:$tiempo
37     eva=$(comparacion $LIMITE $tiempo)
38
39     # Se ejecuta si se supera el limite
40     if [ $eva -eq 1 ]; then
41         # Si se excede se guarda una incidencia y se manda un
42         # correo electronico
43         fecha=$(date +%y/%m/%d-%H:%M:%S)
44         # Se registra la incidencia en el directorio y en el
45         # archivo incidencias
46         echo "A3;$direccion;retardo en ping;$fecha" >>
47         $directorio/incidencias.log
48         #Codigo para enviar un correo
49         # perl swaks --to "destino@gestor.com" \

```

```

45 #      --from "soporte@gestor.com" \
46 #      -s 10.0.1.11:1025 \
47 #      --data "Date: %DATE%\nTo: %TO_ADDRESS%\nFrom: %
FROM_ADDRESS%
48 # \nSubject: A3; $direccion; retardo en ping
49 # \nX-Mailer: swaks v20181104 jetmore.org/john/code/swaks/
50 # \n%NEW_HEADERS%\n"
51 fi
52 done

```

### 2.3.3. Tiempo de respuesta del servidor

El siguiente script es similar al de tiempo de respuesta del ping, solo que en este se hace la consulta al servidor HTTP y se almacena en una base de RRDtool. Se registra la incidencia en el archivo incidencias.log si es que el tiempo de respuesta es mayor a 10 segundos.

```

1  #!/bin/bash
2  # cuatro.sh
3  # A4 El tiempo de respuesta del servidor web excede 10 segundos
4  # Lista de direcciones IP a monitorear separadas por un espacio
5  direcciones="10.0.10.1"
6  # Tiempo limite en segundos
7  LIMITE=10
8  # directorio donde se guardan los archivos que se generen
9  directorio=/home/tc/gestor
10
11 # Retorna 1 si valor es mayor a limite
12 comparacion() {
13     local me=$(awk -v lim="$1" -v valor="$2" 'BEGIN { printf (
14         valor>lim?1:0) }')
15     echo "$me"
16 }
17
18 for direccion in $direcciones; do
19     # Respuesta en segundos
20     tiempo=$(curl -s -w %{time_total} \n -o /dev/null $direccion
21 )
22     eva=$(comparacion $LIMITE $tiempo)
23
24     #echo $tiempo
25
26     if [ $eva -eq 1 ]; then
27         # Si se excede se guarda una incidencia y se manda un
28         correo electronico
29         fecha=$(date +%y/%m/%d-%H:%M:%S)
30         # Se registra la incidencia en el directorio y en el
31         archivo incidencias

```



```

28     echo "A4;$direccion;retardo http;$fecha" >> $directorio/
    incidencias.log
29     # Envio de correo
30     # perl swaks --to "destino@gestor.com" \
31     # --from "soporte@gestor.com" \
32     # -s 10.0.1.11:1025 \
33     # --data "Date: %DATE%\nTo: %TO-ADDRESS%\nFrom: %
    FROM-ADDRESS%
34     #\nSubject: A4; $direccion; retardo http
35     #\nX-Mailer: swaks v20181104 jetmore.org/john/code/swaks
    /
36     #\n%NEW-HEADERS%\n"
37 fi
38 done

```

#### 2.3.4. Memoria real

Este script revisa la memoria real de cada IP de enrutador que se encuentre en la lista de direcciones IP a monitorear y en el caso en el que la memoria disponible sea menor al 80 % se registra la incidencia. Se tienen las variables de comunidad, limite y del directorio para cambiar la funcionalidad del script de ser necesario.

Para poder obtener el porcentaje de memoria disponible se debe de obtener el valor de memoria disponible y el valor de total de memoria para a partir de estos valores calcular el porcentaje.

```

1  #!/bin/bash
2  # ocho.sh
3  # A8 la memoria real es menor de 80% del total
4  # Lista de direcciones IP a monitorear separadas por un espacio
5  direcciones="10.0.1.254"
6  # limite a utilizar
7  LIMITE=0.8
8  # comunidad snmp
9  comunidad=soporte
10 # directorio donde se guardan los archivos que se generen
11 directorio=/home/tc/gestor
12
13 # Funcion de comparacion, retorna 1 si valor es menor a limite
14 comparacion() {
15     local me=$(awk -v lim="$1" -v valor="$2" 'BEGIN { printf (
        valor<lim?1:0) }')
16     echo "$me"
17 }
18
19 for direccion in $direcciones; do
20     # Consulta de memoria disponible

```

```

21 disponible=$(snmpget -v 1 -c $comunidad -OQv $direccion
1.3.6.1.4.1.2021.4.6.0)
22 # Consulta de memoria total
23 total=$(snmpget -v 1 -c $comunidad -OQv $direccion
1.3.6.1.4.1.2021.4.5.0)
24 # Porcentaje de memoria disponible
25 porcentaje=$(echo "scale=2; $disponible/$total" | bc -l)
26 #echo "disponible $disponible total $total poercentaje
$porcentaje"
27
28 eva=$(comparacion $LIMITE $porcentaje)
29
30 if [[ eva -eq 1 ]]; then
31     # Si se excede se guarda una incidencia y se manda un
correo electronico
32     fecha=$(date +%y/%m/%d-%H:%M:%S)
33     # Se registra la incidencia en el directorio y en el
archivo incidencias
34     echo "A8;$direccion;memoria;$fecha" >> $directorio/
incidencias.log
35     #Codigo para enviar un correo
36     # perl swaks --to "destino@gestor.com" \
37     # --from "soporte@gestor.com" \
38     # -s 10.0.1.11:1025 \
39     # --data "Date: %DATE%\nTo: %TO_ADDRESS%\nFrom: %
FROM_ADDRESS%
40     # \nSubject: A8; $direccion; memoria
41     # \nX-Mailer: swaks v20181104 jetmore.org/john/code/swaks/
42     # \n%NEW_HEADERS%\n"
43     fi
44 done

```

### 2.3.5. Monitoreo de CPU y memoria

Este script realiza el monitoreo del CPU y la memoria para cada enrutador, guardando los datos recopilados en una base de datos de RRDtool. Al igual que en los scripts anteriores se pueden indicar varias direcciones IP a monitorear e indicar la comunidad SNMP. Este script solo guarda los datos recopilados, no realiza registro de incidencias ya que existe un script para CPU y para la memoria.

```

1 #!/bin/bash
2 # monitoreo.sh
3 # script para el monitoreo de varias IP
4 # Lista de direcciones IP a monitorerar separadas por un espacio
5 direcciones="10.0.1.254"
6 # directorio donde se guardan los archivos que se generen
7 directorio=/home/tc/gestor

```

```

8 # comunidad snmp
9 comunidad=soporte
10
11 # Poner un while aqui si es necesario
12 for direccion in $direcciones; do
13     # Si no existe se crea un subdirectorio para la ip
14     # monitorear
15     subdirectorio="$directorio/$direccion"
16     if [ ! -d $subdirectorio ]; then
17         mkdir $subdirectorio
18     fi
19     base="$subdirectorio/monitoreo.rrd"
20     if [ ! -f $base ]; then
21         rrdtool create $base --step 60 \
22             DS:usoCPU:GAUGE:120:U:U \
23             DS:usoMemoria:GAUGE:70:U:U \
24             RRA:AVERAGE:0.5:1:60
25     fi
26     # comandos snmp y oids a consultar
27     cpu_usado=$(snmpget -v 1 -c $comunidad -OQv $direccion
28     1.3.6.1.2.1.25.3.3.1.2.768)
29     memoria_disponible=$(snmpget -v 1 -c $comunidad -OQv
30     $direccion 1.3.6.1.4.1.2021.4.6.0)
31     memoria_total=$(snmpget -v 1 -c $comunidad -OQv $direccion
32     1.3.6.1.4.1.2021.4.5.0)
33     # porcentaje de memoria usado
34     porcentaje=$(echo "scale=2; (1-($memoria_disponible/
35     $memoria_total))*100" | bc -l)
36     #echo $memoria_total $memoria_disponible $porcentaje
37     rrdtool update $base N:$cpu_usado:$porcentaje
38 done

```

Para obtener el porcentaje de memoria utilizado se obtiene la memoria disponible y la total y hacer una operación para obtener el valor a guardar en la base de datos.

### 2.3.6. Monitoreo de interfaces

Ya que cada enrutador tiene cinco interfaces diferentes se crea una base de datos para cada interfaz en la que se guardan los siguientes datos.

1. ifInErrors Errores de entrada.
2. ifInUcastPkts El numero de paquetes unicast enviados.
3. ifInNUcastPkts EL número de paquetes no unicast enviados.
4. ifInOctets El número total de octetos recibidos en la interfaz.

5. ifOutOctets El número total de octetos transmitidos por la interfaz.

6. ifSpeed Es un estimado del ancho de banda en bits por segundo.

Esta scripts solo se encarga de guardar los datos recopilados con SNMP, no se realiza un registro de incidencias en esta parte.

```
1  #!/bin/bash
2  # interfaces.sh
3  # Direcciones ip a monitorear
4  direcciones="10.0.1.254"
5  # Directorio donde se guardan los archivos
6  directorio=/home/tc/gestor
7  # Comunidad snmp
8  comunidad=soporte
9  for direccion in $direcciones; do
10     # num_interfaces=$(snmpwalk -v 1 -c $comunidad $direccion
ifIndex | awk 'END{print NR}')
11     for i in 1 2 3 4 5; do
12         # Si no existe se crea un subdirectorio para la ip
monitorear
13         subdirectorio="$directorio/$direccion"
14         if [ ! -d $subdirectorio ]; then
15             mkdir $subdirectorio
16         fi
17         # Base de datos por interfaz
18         base="$subdirectorio/interfaz-$i.rrd"
19         # Si no existe se crea la base de rrdtool para cada
interfaz de ip a monitorear
20         if [ ! -f $base ]; then
21             rrdtool create $base --step 60 \
22             DS:ifInErrors:GAUGE:120:U:U \
23             DS:ifInUcastPkts:GAUGE:120:U:U \
24             DS:ifInNUcastPkts:GAUGE:120:U:U \
25             DS:ifInOctets:GAUGE:120:U:U \
26             DS:ifOutOctets:GAUGE:120:U:U \
27             DS:ifSpeed:GAUGE:120:U:U \
28             RRA:AVERAGE:0.5:1:60
29         fi
30         ifInErrors=$(snmpget -v 1 -c $comunidad -OQv localhost
1.3.6.1.2.1.2.2.1.14.$i)
31         ifInUcastPkts=$(snmpget -v 1 -c $comunidad -OQv
localhost 1.3.6.1.2.1.2.2.1.11.$i)
32         ifInNUcastPkts=$(snmpget -v 1 -c $comunidad -OQv
localhost 1.3.6.1.2.1.2.2.1.12.$i)
33         ifInOctets=$(snmpget -v 1 -c $comunidad -OQv localhost
1.3.6.1.2.1.2.2.1.10.$i)
34         ifOutOctets=$(snmpget -v 1 -c $comunidad -OQv localhost
1.3.6.1.2.1.2.2.1.16.$i)
```

```
35     ifSpeed=$(snmpget -v 1 -c $comunidad -OQv localhost  
1.3.6.1.2.1.2.2.1.5.$i)  
36     # echo $ifInErrors $ifInUcastpkts $ifInNUcastPkts  
$ifInOctets $ifOutOctets $ifSpeed  
37     rrdtool update $base N:$ifInErrors:$ifInUcastpkts:  
$ifInNUcastPkts:$ifInOctets:$ifOutOctets:$ifSpeed  
38     done  
39 done
```