

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

**PRÁCTICA 2: ADMINISTRACIÓN DE
RENDIMIENDO DE SERVICIOS EN RED**

ADMINISTRACIÓN DE SERVICIOS EN RED

EQUIPO 5 (Primer Examen), EQUIPO 9 (Segundo Examen):

+ HERNÁNDEZ PINEDA MIGUEL ANGEL

+ MONROY MARTOS ELIOTH

+ ZÚÑIGA HERNÁNDEZ CARLOS

PROFRA. TANIBET PÉREZ DE LOS SANTOS MONDRAGÓN

9 de Abril 2019

Índice general

1. Introducción	1
1.1. FCAPS	1
1.2. Línea base	2
1.3. Mínimos cuadrados	3
1.4. Holt Winters	3
2. Desarrollo y Resultados	5
2.1. Examen 1.- Administración de fallas con mínimos cuadrados	5
2.1.1. Ejercicio 1: Definición de umbrales	5
2.1.2. Ejercicio 2: Mínimos Cuadrados	6
2.1.3. Ejercicio 3: Envío de Correos	7
2.1.4. Ejercicio 4: Lectura Completa	8
2.2. Examen 2.- Administración de fallas usando series de datos no lineales mediante Holt Winters	9
2.2.1. Evidencia 1	9
2.2.2. Evidencia 2	10
2.2.3. Evidencia 3	11
3. Cuestionario	12
4. Códigos	17
4.1. Examen 1	17
4.2. Examen 2	39
5. Conclusiones	45
Referencias Bibliográficas	46

Capítulo 1

Introducción

La administración de redes se dedica a establecer las actividades que deben realizarse sobre una red informática con el fin de brindar los diferentes servicios de manera eficiente y eficaz, garantizando la disponibilidad y calidad que el usuario final espera. En la administración de redes es muy usado el protocolo SNMP o protocolo simple de Administración de red, que facilita el intercambio de paquetes de información de administración entre los diferentes dispositivos de la red. La administración de las redes, también consiste en tener un constante monitoreo de como se comportan las mismas, esto con la finalidad de poder prevenir malos funcionamientos y en caso de que estos se presentasen, poder actuar de forma correcta. La administración de fallos, es una parte fundamental de la administración de servicios, algunos conceptos básicos son presentados a continuación.

1.1. FCAPS

FCAPS es un marco de gestión de redes que fue creado por la Organización Internacional de Normalización, mejor conocida como ISO (por sus siglas en inglés). Las siglas FCAPS significan:

- F (Fault): Fallas.
- C (Configuration): Configuración.
- A (Accounting): Administración.
- P (Performance): Desempeño.
- S (Security): Seguridad.

En esta práctica, nos enfocaremos en la F, es decir la Administración de Fallas. Una falla, se puede definir como un evento que muestra un problema en la red. El objetivo de la administración de fallas es el de detectar, aislar, corregir y registrar fallas que ocurren en la red.

La administración de fallas también incluye el análisis de tendencias para predecir

errores, de modo que la red siempre brinde el servicio que pretende o para lo que fue diseñada.

Para gestionar las fallas, se requiere de un sistema para monitorear la red y que pueda generar alarmas o notificaciones. Un sistema básico de gestión de alarmas proporciona una lista de alarmas basadas en la topología de la red. Una vez que se activan las alarmas o los usuarios de la red enfrentan un problema, se requiere un sistema de tickets para administrar la carga de trabajo y las prioridades asociadas con las fallas.

Una alarma solo muestra el síntoma de un problema, por lo que adicionalmente, se requiere de un sistema que brinde la solución de problemas para recopilar más información sobre las posibles causas que llevan a la falla [5].

1.2. Línea base

Una línea de base o línea de fondo, consiste en un proceso que permite estudiar la red con base en intervalos regulares, esto con el objetivo de asegurarse de que la red se encuentra trabajando acorde a lo que fue diseñada.

La línea de base sirve para obtener información de importancia como:

- Información de la salud del hardware y software de la red
- Determinar los usos de los recursos de la red
- Tomar decisiones precisas sobre los umbrales de alarmas de la red
- Identificar los problemas o fallas de la red
- Predecir problemas o fallas futuros

A grandes rasgos, consiste en definir un umbral, o bien 3 (Ready, Set, Go), a partir de los cuales en el momento en que el estado actual de la red graficada cruce las líneas previamente definidas, se identificará como una falla [6].



Figura 1.1: Representación gráfica de línea base

1.3. Mínimos cuadrados

El método de mínimos cuadrados permite obtener una línea del mejor ajuste de forma más precisa.

Los pasos para este método son [7]:

- Calcular la media de los valores x así como la media de los valores y
- Calcular la pendiente de la línea de mejor ajuste
- Calcular la intercepción en y de la línea usando la fórmula de la recta.
- Utilizar la pendiente m y la intercepción en y para formar la ecuación de la recta.

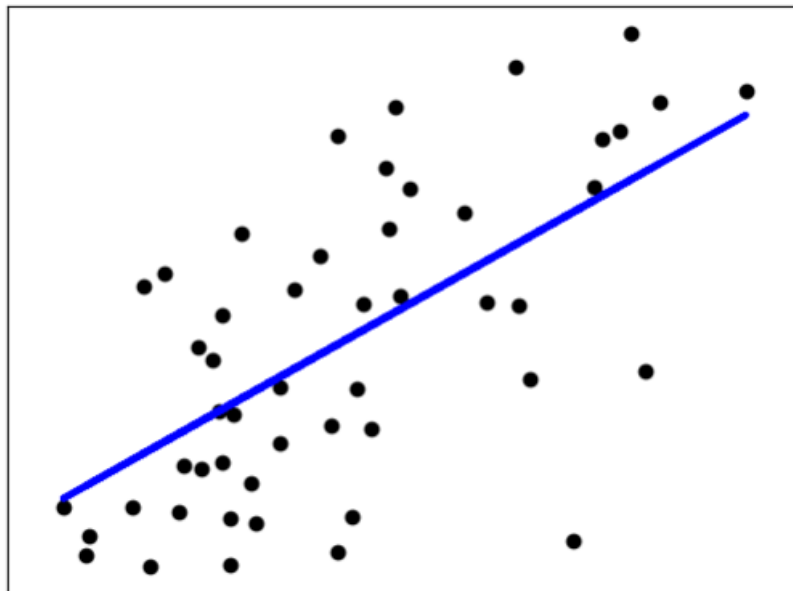


Figura 1.2: Representación gráfica de mínimos cuadrados

1.4. Holt Winters

El método de Holt-Winters consiste en un método que ofrece un pronóstico de triple exponente suavizante. Este método tiene la ventaja de poder adaptarse con mayor facilidad a medida que obtiene nueva información real.

Este método considera nivel, tendencia y estacional de una determinada serie de tiempos.

Holt-Winters se aplica solamente cuando la serie de datos es estacional, es decir, es periódica.

Por medio de este método se puede detectar y hacer predicciones de comportamiento anómalo[8].

Dos conceptos muy importantes para Holt-Winters son:

- Valor observado: Se trata de aquellos valores que ya conocemos.
- Valor esperado: Se trata de aquellos valores que resultan de una predicción con base a los valores observados.

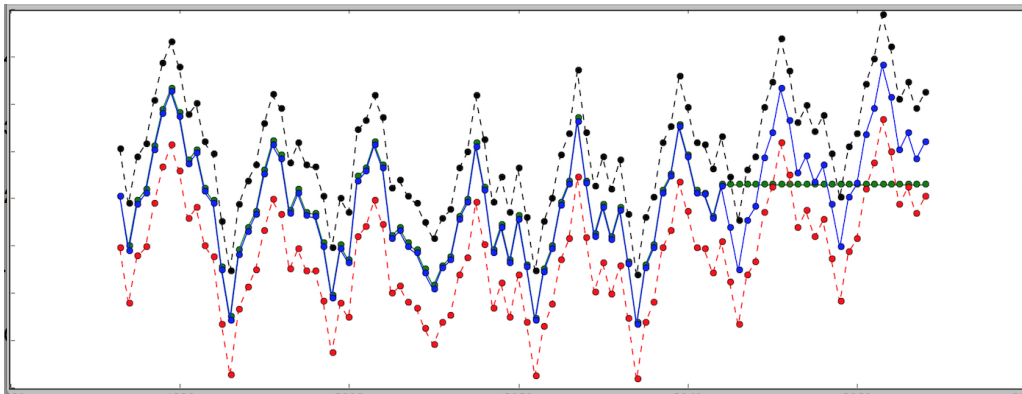


Figura 1.3: Representación gráfica de Holt Winters

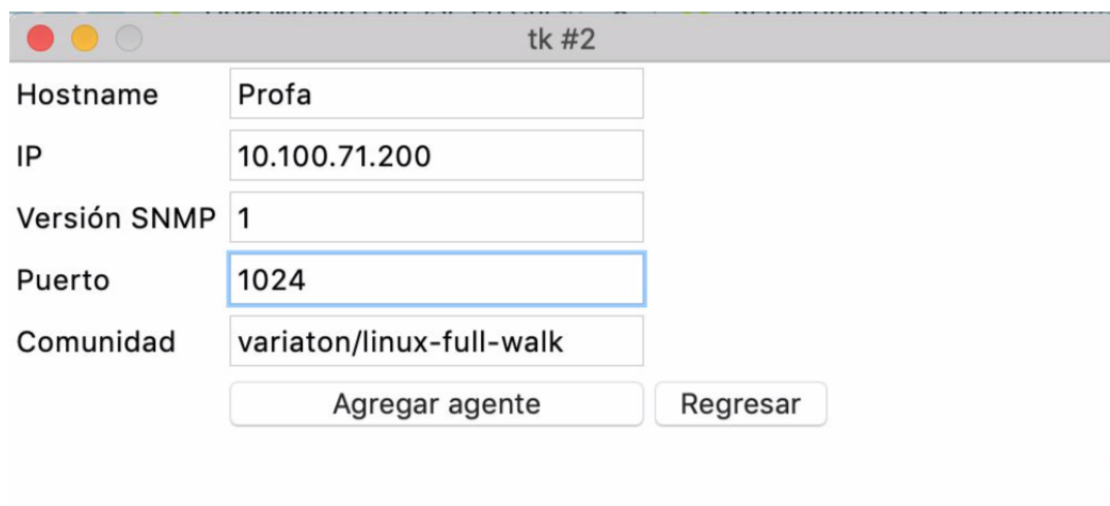
Capítulo 2

Desarrollo y Resultados

2.1. Examen 1.- Administración de fallas con mínimos cuadrados

2.1.1. Ejercicio 1: Definición de umbrales

Para llevar a cabo la definición de umbrales, realizamos el método propuesto por CISCO; consiste en hacer una lectura de datos por un periodo de tiempo (en este caso fueron 10 minutos), almacenar los datos obtenidos y generar una tabla de dispersión y con base en ella definir el umbral GO (umbral mayor) y a partir de este definir los umbrales inferiores. En este caso el agente que íbamos a monitorear se nos fue proporcionado por la profesora, el cual tenía los siguientes valores.



Hostname	Profra
IP	10.100.71.200
Versión SNMP	1
Puerto	1024
Comunidad	variaton/linux-full-walk

Agregar agente Regresar

Figura 2.1: Agente

Una vez que fue ejecutado el programa y la lectura de los datos de uso de CPU se llevó a cabo en el plazo indicado, se obtuvo la siguiente tabla.

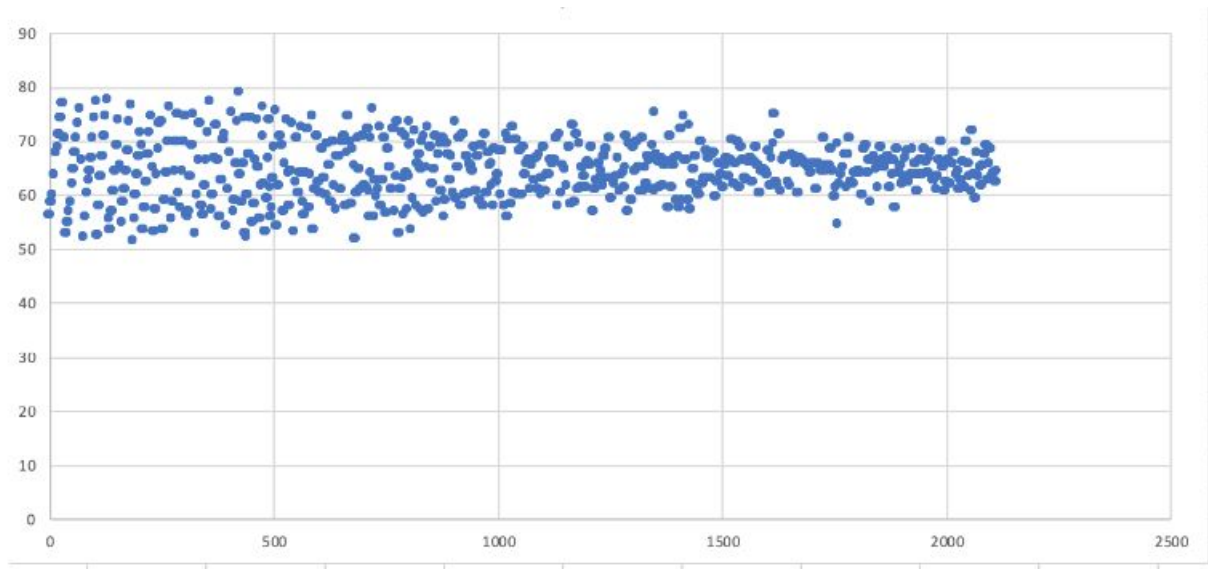


Figura 2.2: Tabla de Dispersión

2.1.2. Ejercicio 2: Mínimos Cuadrados

Una vez obtenidos los umbrales por medio de la tabla de dispersión se definieron dentro de nuestro sistema con el fin de poder hacer una predicción por medio del método de mínimos cuadrados, además en la gráfica se indica cuando ocurrirá un error y los colores varían cada que uno de los umbrales es rebasado; la línea de mejor ajuste se muestra de color amarillo.

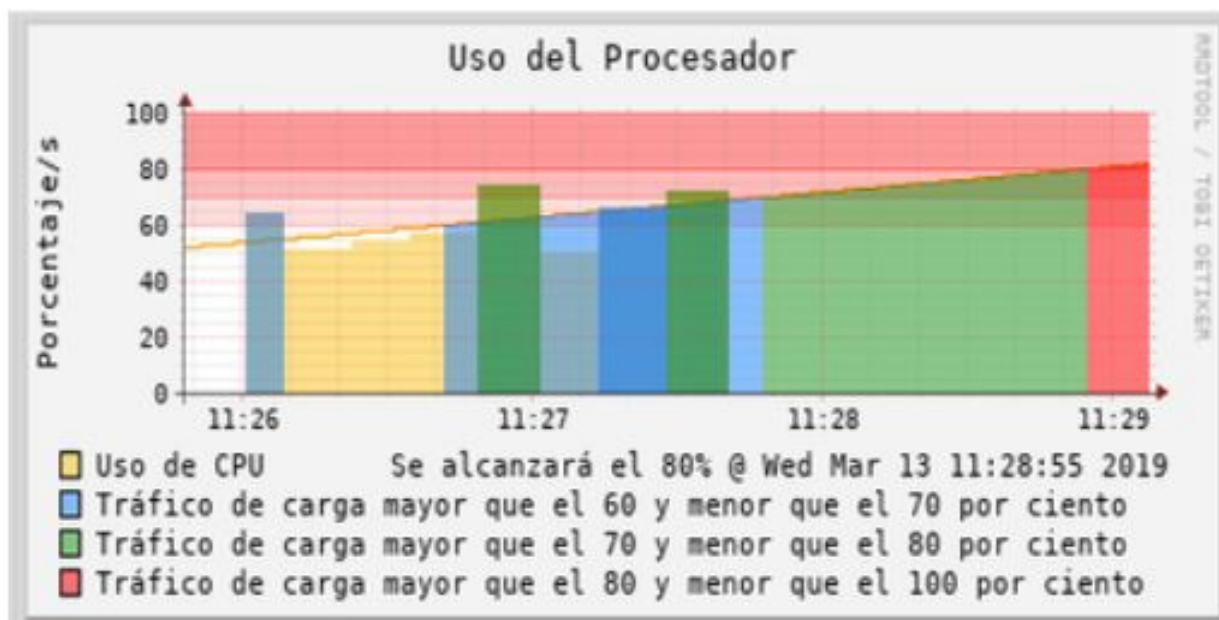


Figura 2.3: Lectura del procesador

En la gráfica se puede observar que el umbral GO es del 80%. También, en las descripciones de la gráfica se encuentra el tiempo en el que el uso del CPU va a alcanzar ese umbral. Este tiempo es: 13 de marzo del 2019 a las 11:28:55. Para poder llevar a cabo este ejercicio se hizo uso del código siguiente.


```
def graficarCPU(self, grupo, old1, archivo, header, numero):
    umbral = int(self.umbralCPU)
    ultimo = rrdtool.last(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".rrd")
    primero = rrdtool.first(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".rrd")
    consulta = consultav25MWP(self.agente.comunidad, self.agente.ip, int(self.agente.version), numero, grupo, old1, int(self.agente.puerto))
    if consulta!="":
        valor = "N:"+" "+str(consulta)
        rrdtool.update(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".rrd", valor)
        rrdtool.dump(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".rrd", self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".xml")
        ret = rrdtool.graphv2(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".png",
            "--start",str(ultimo-100),
            "--end",str(ultimo-100),
            "--width",str(ultimo+200),
            "--vertical-label=Porcentaje/s",
            "--title="+header,
            "--lower-limit", "0",
            "--upper-limit", "100",
            "DEF:usage_"+self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".rrd:valor1:AVERAGE",
            "CDEF:umbral_"+str(umbral-20)+"_1=usage,"+str(umbral-20)+"_GT,usage,"+str(umbral-10)+"_LT,E0,usage,0,TF",
            "CDEF:umbral_"+str(umbral-10)+"_1=usage,"+str(umbral-10)+"_GT,usage,"+str(umbral)+"_LT,E0,usage,0,IF",
            "CDEF:umbral_"+str(umbral)+"_1=usage,"+str(umbral)+"_GT,usage,"+str(umbral+100-umbral)+"_LT,E0,usage,0,IF",
            "AREA:usage#FF000077:Uso de CPU",
            "VDEF:usage_LSL:LSLOPE",
            "VDEF:busage_LSL:LSLINT",
            "CDEF:avg_usage_POP,n,COUNT,"+b,+",
            "CDEF:umbral_"+str(umbral-20)+"_avg,"+str(umbral-20)+"_LIMIT",
            "CDEF:umbral_"+str(umbral-10)+"_avg,"+str(umbral-10)+"_LIMIT",
            "CDEF:umbral_"+str(umbral)+"_avg,"+str(umbral)+"_LIMIT",
            "VDEF:minumbral_"+str(umbral)+"_umbral_"+str(umbral)+"_FIRST",
            "VDEF:lastusage_LAST",
            "PRINT:last:%6.2lf %5",
            "GPRINT:minumbral_"+str(umbral)+"_": Se alcanzará el "+str(umbral)+"% @ %c: %c: strftime",
            "TIME:avg#FF0000",
            "LINE2:"+str(umbral-20),
            "AREA:10#FF000022::STACK",
            "AREA:10#FF000044::STACK",
            "AREA:"+str(100-umbral)+"#FF000066::STACK",
            "AREA:umbral_"+str(umbral-20)+"#0077FF77:Tráfico de carga mayor que el "+str(umbral-20)+" y menor que el "+str(umbral-10)+" por ciento",
            "AREA:umbral_"+str(umbral-10)+"#00808077:Tráfico de carga mayor que el "+str(umbral-10)+" y menor que el "+str(umbral)+" por ciento",
            "AREA:umbral_"+str(umbral)+"#00000080:Tráfico de carga mayor que el "+str(umbral)+" y menor que el "+str(umbral+100-umbral)+" por ciento",
            "AREA:umbral_"+str(umbral-10)+"_1#00808077",
            "AREA:umbral_"+str(umbral)+"_1#FF000088"}
        valor=float(ret["print(0)"])
        if valor<float(umbral):
            if not self.notificacionCPU:
                self.notificacionCPU=True
                self.noti.enviarCorreo(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".png")
                self.log.escribirLog(0)
            else:
                self.notificacionCPU=False
```

Figura 2.4: Mínimos Cuadrados

2.1.3. Ejercicio 3: Envío de Correos

Este ejercicio consistió en enviar un correo a la dirección de la profesora **tani-bet.escom@gmail.com** en el momento en que se presentara una falla con fecha y hora del momento en el que ocurrió.

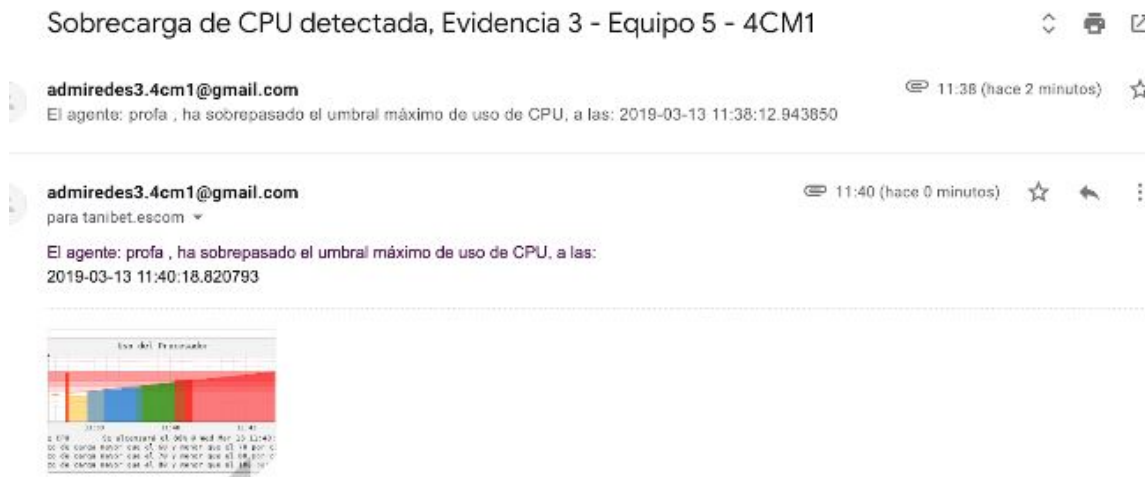


Figura 2.5: Correo Enviado

Para llevar a cabo este proceso se utilizó el siguiente código que nos permite crear el contenido de un correo y enviarlo a la dirección especificada con la información definida.

```

class Notificador():
    def __init__(self, agente):
        self.remitente = "admiredes3.4CM1@gmail.com"
        self.destinatario = "tanibet.escom@gmail.com"
        self.servidor = 'smtp.gmail.com: 587'
        self.contra = '#Equipo5'
        self.agente=agente
    def enviarCorreo(self, tipo, imagen=""):
        asunto, texto=self.obtenerContenido(tipo)
        hora=self.obtenerHora()
        texto="El agente: "+self.agente+" "+texto+hora
        #Definimos contenido del correo
        mensaje = MIMEMultipart()
        mensaje['Subject'] = asunto
        mensaje['From'] = self.remitente
        mensaje['To'] = self.destinatario
        mensaje.attach(MIMEText(texto, "plain"))
        #Verificamos si hay que enviar una imagen
        if imagen!="":
            with open(imagen, 'rb') as f:
                img = MIMEImage(f.read())
                f.close()
                mensaje.attach(img)
        #Enviamos el correo
        servidor_correo = smtplib.SMTP(self.servidor)
        servidor_correo.starttls()
        servidor_correo.login(self.remitente, self.contra)
        servidor_correo.sendmail(self.remitente, self.destinatario, mensaje.as_string())
        servidor_correo.quit()
    def obtenerContenido(self, tipo):
        with open("mensajes.txt") as f:
            for i, linea in enumerate(f):
                if i==tipo:
                    return linea.split(";")
                else:
                    return ["", ""]
    def obtenerHora(self):
        return str(datetime.datetime.now())

```

Figura 2.6: Envío de Correos

2.1.4. Ejercicio 4: Lectura Completa

Para el ejercicio final, se tenía que llevar a cabo la lectura de los siguientes 3 parámetros: Uso de CPU, RAM y Disco Duro, además se tenían que mostrar las gráficas elaboradas en la práctica 1 y de igual manera, era necesario enviar el correo de notificación de que un error había ocurrido.

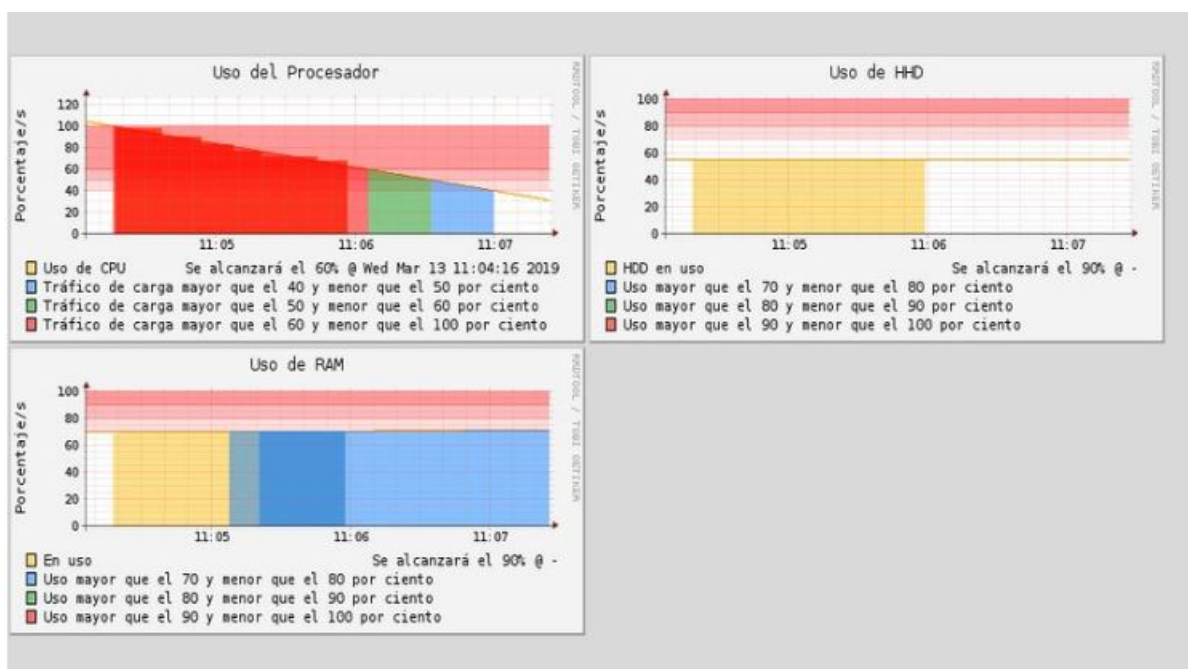


Figura 2.7: Lectura de RAM, HDD y CPU

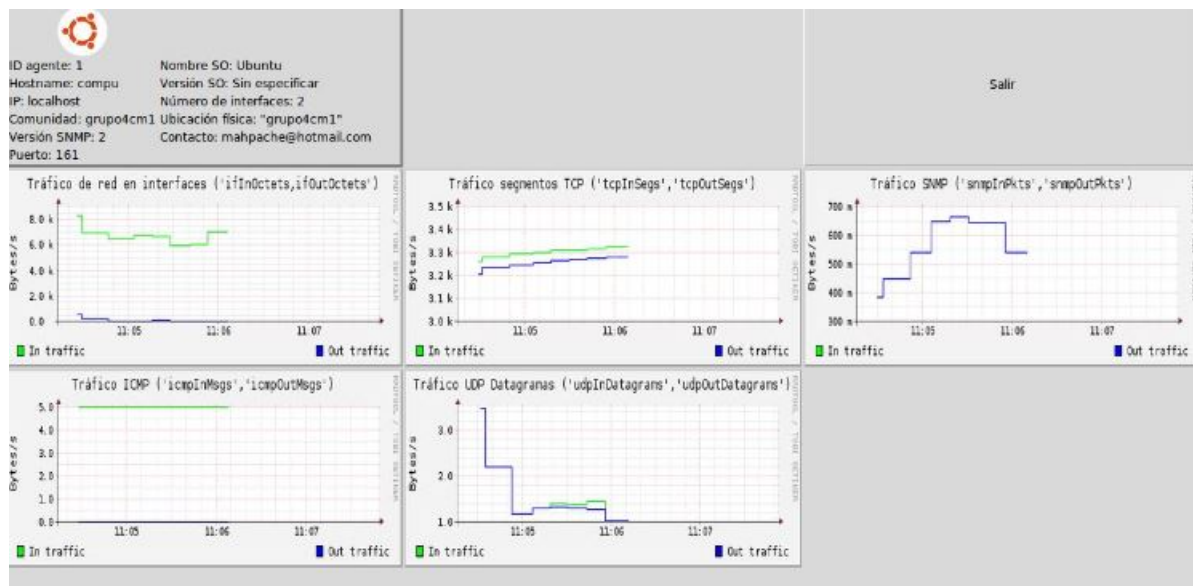


Figura 2.8: Lectura de datos (Práctica 1)

2.2. Examen 2.- Administración de fallas usando series de datos no lineales mediante Holt Winters

2.2.1. Evidencia 1

Nos fue proporcionada una base de datos round robin (predict.rrd), el cual contenía las mediciones hechas para el objeto de la mib “ifInOctets”. Esto con la finalidad de que nosotros determináramos si los valores de alpha, beta y gamma eran los adecuado para realizar el monitoreo o podían mejorar.

Para esto, lo primero que realizamos, fue graficar el archivo rrd con la intención de obtener información de esta gráfica. La gráfica obtenida puede observarse en la figura 2.9.

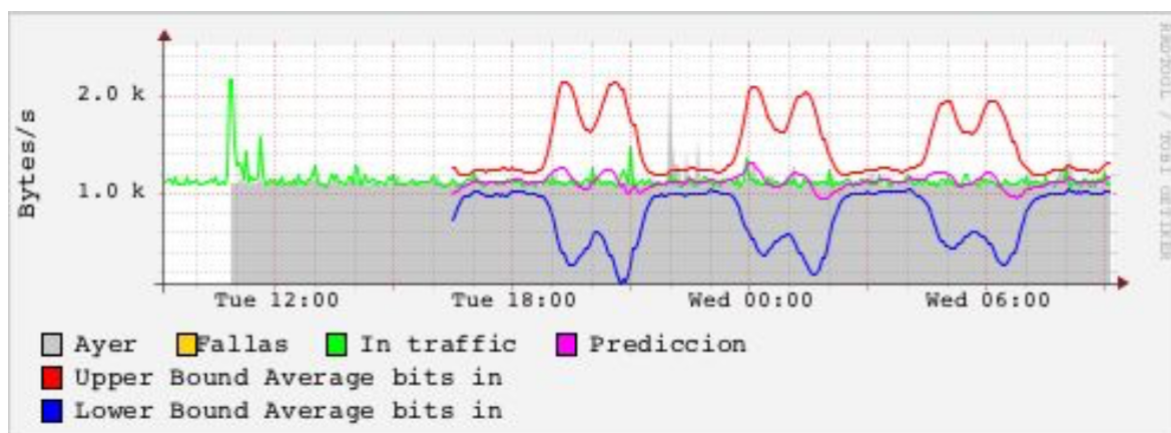


Figura 2.9: Gráfica obtenida del archivo rrd

Como se puede observar, la gráfica obtenida muestra el tráfico de entrada de color verde, los límites superior e inferior de color rojo y azul respectivamente, y

la predicción de holt-winters de color rojo. Así como también el histórico de datos leídos de color gris.

Posterior al análisis de la gráfica, se determino que los valores de alpha, beta y gamma se mantuvieran como los mismos, ya que la predicción era bastante cercana a los valores reales.

Cabe señalar, que los valores de alpha, beta y gamma eran conocidos, gracias al “dump” hecho al archivo rrd, el cual genera un archivo xml desde el cual se pueden consultar estos valores.

En caso de que hubiera sido necesario modificar el valor de alguno de estos valores, se puede usar la función “tune”, con el cual se pueden modificar estos parámetros. Esto se puede observar en la figura 2.10.

```
final=False
fecha_inicio=fecha_final=""
noti=Notificador("MacOs")
log=Logger("MacOs")
rrdtool.tune("predict.rrd","--alpha","0.1")
rrdtool.tune("predict.rrd","--beta","0.0035")
rrdtool.tune("predict.rrd","--gamma","0.1")
archivo_rrd="predict.rrd"
while 1:
    consulta=consultav2SNMP("variation/virtualtable","10.100.71.200",1,1,'IF-MIB','ifInOctets',1024)
    #consulta=consultav2SNMP("variation/virtualtable","10.100.71.200",1,1,"HOST-RESOURCES-MIB","hrProcessorLoad",161)
    valor = "N:" + str(consulta)+" ":" "+ str(consulta)
```

Figura 2.10: Uso de la función tune

2.2.2. Evidencia 2

Para la siguiente parte del examen, se nos pidió detectar una falla (comportamiento anormal que sobrepasas uno de los límites) y mostrar la gráfica correspondiente donde se observe como se coloea de color rojo el area donde se muestra la falla detectada.

En la figura 2.11

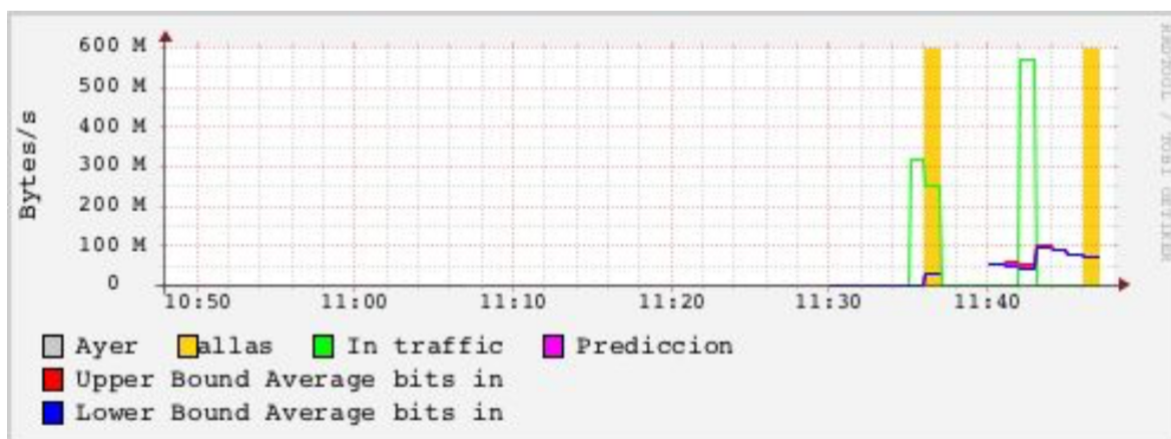


Figura 2.11: Gráfica que muestra como se detecta un fallo

Además, se nos solicito, que se mostrará el histórico de las mediciones hechas

en un tiempo anterior al actual, esto puede observarse en la figura 2.12 con el área marcada de color gris.

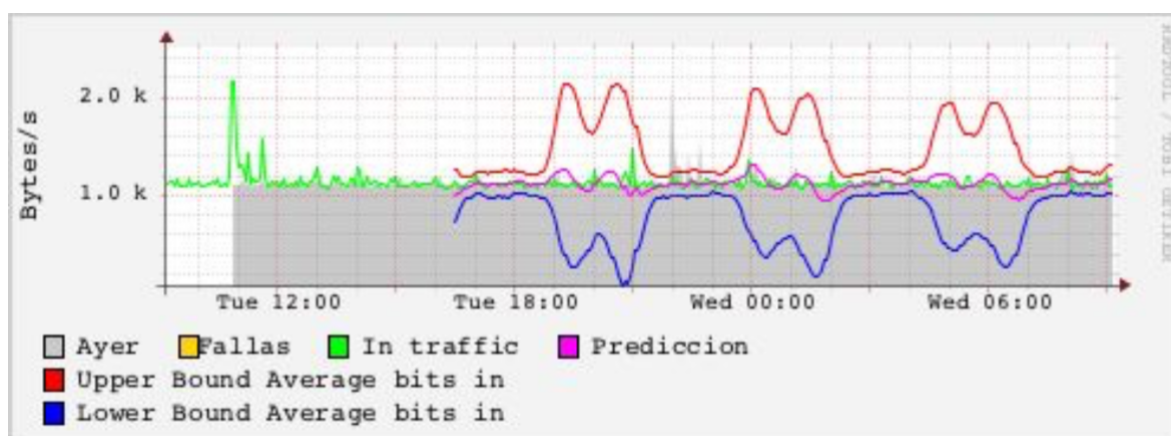


Figura 2.12: Gráfica que muestra el historial coloreado.

2.2.3. Evidencia 3

Finalmente, fue necesario enviar una notificación al administrador, vía correo electrónico, cuando se presentaría alguna falla, estas notificaciones tenían que enviarse, al iniciar una falla y al culminar, en estas notificaciones, se incluye la información del agente donde se presentó el problema, la fecha y hora, y además una imagen que muestra la gráfica que es generada en tiempo real del rendimiento del objeto mib.

En la figura 2.13 se muestra uno de los correos que fue enviado.

Detección de Falla 4CM1 Equipo9



admiredes3.4cm1@gmail.com

11:47 (hace 7 minutos)



para tanibet.escom

El agente: Compu Profa, se ha salido del rango definido para iflnOctets, a las:
Wed Mar 27 11:47:00 2019



Responder

Reenviar

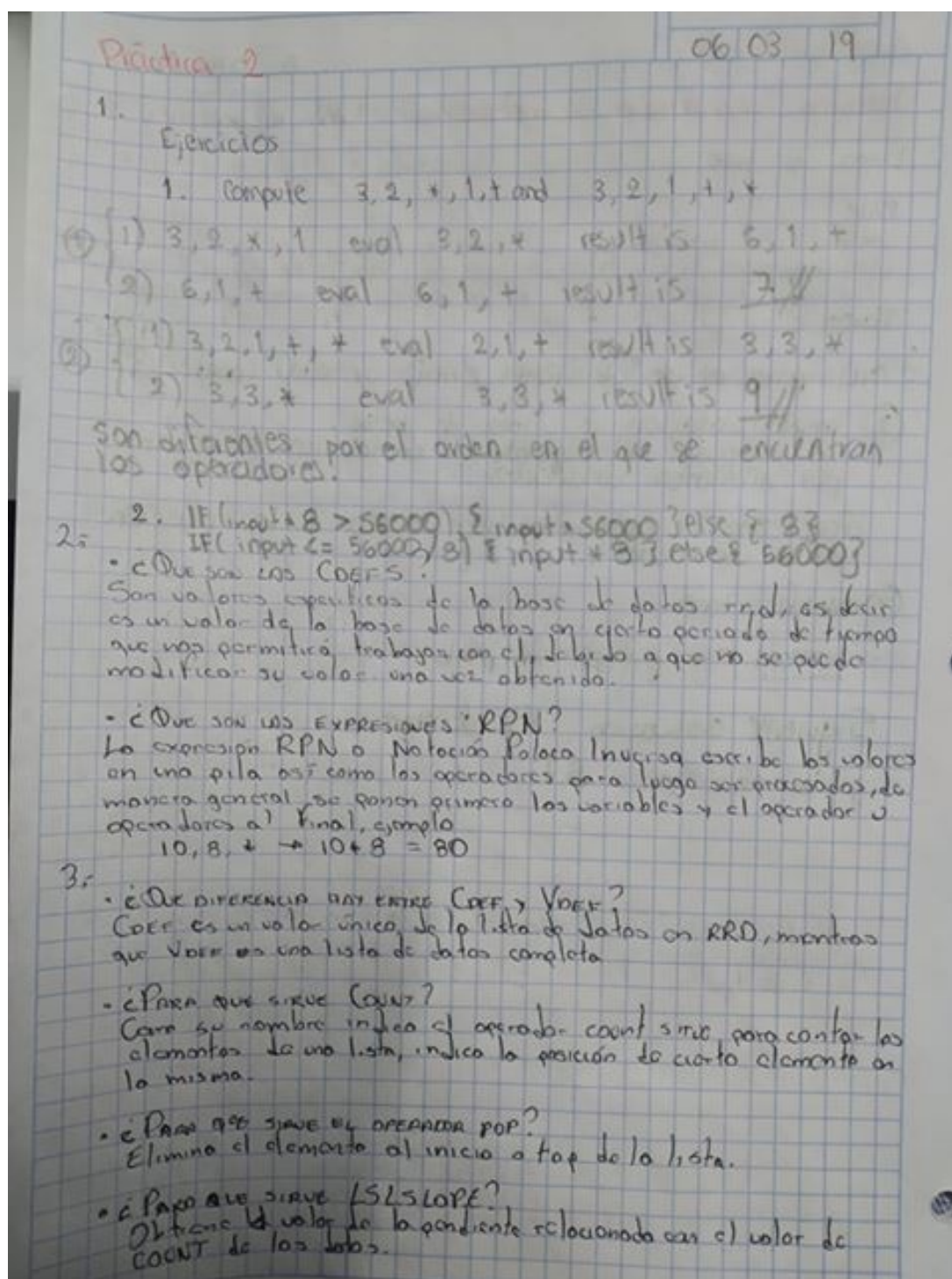
Figura 2.13: Notificación enviada al detectarse una falla.

Capítulo 3

Cuestionario

1. Leer la documentación “rpn tutorial” de rrdtool
2. Leer la documentación “cdef tutorial” de rrdtool
 - ¿Qué son las CDEFs?
 - ¿Qué son las expresiones RPN?
3. Leer la documentación “rrdgraph_rpn” de rrdtool
 - ¿Qué diferencia hay entre CDEF y VDEF?
 - ¿Para qué sirve el operador COUNT?
 - ¿Para qué sirve el operador POP?
 - ¿Para qué sirve LSLSLOPE?
 - ¿Para qué sirve LSLINT?
4. Ejercicio 1 – Sello 1 – Resolver el cuestionario
5. Ejercicio 2 – Sello 2 - Resolver los siguientes ejercicios a mano.
 - CDEF:bits=inoctets,8,* (inoctets = [1,2,3,4])
 - CDEF:avg=in,out,+,2,/ (in = [1,2,3,4], out = [5,6,7,8])
 - CDEF:bigger=x,y,MAX (x = [1,2,3,4], y = [5,6,7,8])
 - CDEF:bigger=x,y,GT,x,y,IF (x = [1,8,3,10], y = [5,6,7,8])
 - CDEF:unzero=x,UN,0,x,IF (x = [UN,8,UN,10])
 - CDEF:avg2=v1,v2,v3,3,AVG (v1 = [1,8,3,10], v2 = [5,6,7,8], v3= [1,2,3,4])
6. Ejercicio 3 – Sello 3 - Usar SNMP para monitorizar dos valores. Usar rrdtool para crear dos colecciones (DS) almacenar la información usando SNMP. Usar operadores booleanos, aritméticos y comparación de valores para graficar la información recopilada.

En las siguientes fotos, se pueden apreciar las repuestas y el desarrollo de cada una de las preguntas y ejercicios anteriores.



¿Por qué sirve LSCINT?

Es el valor inicial de la pendiente, en la mayoría de los casos es el primer valor de la lista de datos.

o CDEF: bits = in, at, +, 8, * (in = [1, 2, 3, 4])

1) bits = 1, 8, + eval 1, 8, + result 8 //

2) bits = 2, 8, + eval 2, 8, + result 16 //

3) bits = 3, 8, + eval 3, 8, + result 24 //

4) bits = 4, 8, + eval 4, 8, + result 32 //

o CDEF: avg = in, at, +, 2, / (in = [1, 2, 3, 4], at = [5, 6, 7, 8])

* bits = [8, 16, 24, 32] //

1) 1, 5, +, 2, / eval 1, 5, + result is 6, 2, /

2) 6, 2, / eval 6, 2, / result is 3

1) 2, 6, +, 2, / eval 2, 6, + result is 8, 2, /

2) 8, 2, / eval 8, 2, / result is 4

1) 3, 7, +, 2, / eval 3, 7, + result is 10, 2, /

2) 10, 2, / eval 10, 2, / result is 5

1) 4, 8, +, 2, / eval 4, 8, + result is 12, 2, /

2) 12, 2, / eval 12, 2, / result is 6 //

* avg = [3, 4, 5, 2] //

o CDEF: bigger = x, y, MAX (x = [1, 2, 3, 4], y = [5, 6, 7, 8])

1) 1, 5, MAX eval 1, 5, MAX result is 5

2) 2, 6, MAX eval 2, 6, MAX result is 6

3) 3, 7, MAX eval 3, 7, MAX result is 7

4) 4, 8, MAX eval 4, 8, MAX result is 8

* bigger = [5, 6, 7, 8] //

6 03 19

o CDEF: bigger = x, y, GT, x, y, IF (x = [1, 6, 3, 10], y = [5, 8, 7, 8])

1) 1, 5, GT, 1, 5, IF eval 1, 5, GT, result is 0, 1, 5, IF

2) 0, 1, 5, IF eval 0, 1, 5, IF result is 5 //

1) 8, 6, GT, 8, 6, IF eval 8, 6, GT result is 1, 8, 6, IF

2) 1, 8, 6, IF eval 1, 8, 6, IF result is 8 //

1) 3, 7, GT, 3, 7, IF eval 3, 7, GT result is 0, 3, 7, IF

2) 0, 3, 7, IF eval 0, 3, 7, IF result is 7 //

1) 10, 8, GT, 10, 8, IF eval 10, 8, GT result is 1, 10, 8, IF

2) 1, 10, 8, IF eval 1, 10, 8, IF result is 10 //

* bigger = [5, 8, 7, 10] //

o CDEF: unzero = x, UN, 0, x, IF (x = [UN, 8, UN, 10])

1) UN, UN, 0, UN, IF eval UN, UN, 0, UN, IF result is 0 //

2) 8, UN, 0, 8, IF eval 8, UN, 0, 8, IF result is 8 //

3) UN, UN, 0, UN, IF result is 0 //

4) 10, UN, 0, 10, IF result is 10 //

* un zero = [0, 8, 0, 10] //

6 03 19

o CDEF: avg2 = v1, v2, v3, 3, AVG (v1 = [1, 8, 3, 10],
v2 = [5, 6, 7, 8], v3 = [1, 2, 3, 4])

1) 1, 6, 1, 3, AVG result is 2.5

2) 8, 6, 2, 3, AVG result is 4.75

3) 3, 2, 3, 3, AVG result is 4

4) 6.25

M. en C. Tanibet Pérez
Santos V. Manzanilla
6/03/19

M. en C. Tanibet Pérez
Santos V. Manzanilla
6/03/19

M. en C. Tanibet Pérez
Santos V. Manzanilla
6/03/19

2 personas

* avg2 = [2.5, 4.75, 4, 6.25]

Capítulo 4

Códigos

4.1. Examen 1

A continuación, se muestran los códigos elaborados para la realización del examen.

El archivo con el que se inicia la ejecución del programa, es con main.py, el cual solo crea una instancia del objeto Gestor, el cual se encuentra dentro del archivo gestor.py. Esto para ajustarnos al desarrollo orientado a objetos.

main.py:

```
1 from gestor import Gestor
2
3 gestor=Gestor()
```

El archivo gestor.py, es el que maneja toda la interfaz principal.

gestor.py:

```
1 #https://www.tutorialspoint.com/python/python_gui_programming.htm
2 from tkinter import *
3 from agregar_agente import AgregarAgente
4 from agente import obtenerAgentes, obtenerInfoPrincipalAgente, eliminar
5 from monitor import Monitor
6 from functools import partial
7
8 class Gestor():
9     """Clase principal del programa"""
10    def __init__(self):
11        #Creamos la ventana y sus medidas
12        self.top=Tk()
13        self.top.geometry("800x600")
14        self.top.resizable(0,0)
15        #Definimos los elementos de la misma y a cada uno le hacemos un pack
16        self.b = Button(self.top, text="Agregar agente", command=self.agregarAgente)
17        self.b.grid(row=0, column=3, sticky=W+E+N+S)
18        self.obtenerAgentes()
19        #Final
20        self.top.mainloop()
21    def agregarAgente(self):
22        #Destruimos la ventana actual y traemos a AgregarAgente
23        self.top.destroy()
24        self.agente=AgregarAgente()
25    def monitorearAgente(self, id_agente):
26        monitor=Monitor(id_agente)
27        monitor.start()
```

```

28
29 def eliminarAgente(self, id_agente):
30     self.top.destroy()
31     eliminar(id_agente)
32
33 def obtenerAgentes(self):
34     ids_antes=obtenerAgentes()
35     if len(ids_antes)>0:
36         Label(self.top, text="Total de agentes registrados: "+ str(len(ids_antes
37         ))) .grid(row=0, sticky=W)
38     else:
39         Label(self.top, text="No hay agentes registrados").grid(row=0, sticky=W)
40     for id_agente in ids_antes:
41         info=obtenerInfoPrincipalAgente(id_agente)
42         frame=Frame(self.top, width = 600, height = 200, relief = 'raised',
43         borderwidth=2)
44         frame.grid(row = int(id_agente), column = 0, columnspan=6, sticky=W+E+N+S
45         )
46         Label(frame, text="ID agente: "+id_agente).grid(row=1, column=0, sticky=W)
47         Label(frame, text="Hostname: "+info[0]).grid(row=2, column=0, sticky=W)
48         Label(frame, text="IP: "+info[1]).grid(row=3, column=0, sticky=W)
49         if info[2]==0:
50             Label(frame, text="Estado de conexión: Desconectado").grid(row=4, column
51             =0, sticky=W)
52             Button(frame, text="Monitorear", state=DISABLED, command=partial(self.
53             monitorearAgente, id_agente)).grid(row=2, column=3, sticky=E)
54         else:
55             Label(frame, text="Estado de conexión: Conectado").grid(row=4, column=0,
56             sticky=W)
57             Button(frame, text="Monitorear", command=partial(self.monitorearAgente,
58             id_agente)).grid(row=2, column=3, sticky=E)
59             #Label(frame, text="").grid(row=1, column=0, sticky=W)
60             Button(frame, text="Eliminar", command=partial(self.eliminarAgente,
61             id_agente)).grid(row=4, column=3, sticky=E)

```

Este archivo, maneja todo lo referente a los agentes.

agente.py:

```

1 import json
2 import os.path
3 from SNMP import getInfo, consultav2SNMP, consultav3SNMP
4 def obtenerAgentes():
5     print("Voy a obtener los agentes")
6     with open('agentes.json', 'r') as f:
7         if os.path.getsize('agentes.json') > 0:
8             #print("Existe al menos un agente")
9             data=json.load(f)
10            return list(data.keys())
11        else:
12            pass
13            #print("No hay agentes registrados")
14    return []
15 def obtenerInfoPrincipalAgente(id):
16     data={}
17     info=[]
18     with open('agentes.json', 'r') as f:
19         if os.path.getsize('agentes.json') > 0:
20             data=json.load(f)
21             #0->hostname
22             info.append(data[id]["hostname"])
23             #1->ip
24             info.append(data[id]["ip"])

```

```

25 #2-> estado conexión
26 res=consultav2SNMP( data[id]["comunidad"], data[id]["ip"], int( data[id]["version"]
27 ),0, 'SNMPv2-MIB', 'sysName', int( data[id]["puerto"]))
28 if res=="":
29     info.append(0)
30 else:
31     info.append(1)
32 #info.append(consultav2SNMP( data[id]["comunidad"], data[id]["ip"], int( data[id]
33 ["version"],0, 'SNMPv2-MIB', 'sysUpTime', int( data[id]["puerto"])))
34 #info.append(consultav3SNMP( data[id]["comunidad"], data[id]["ip"], int( data[id]
35 ["version"], '1.3.6.1.2.1.1.3', int( data[id]["puerto"])))
36 info.append( data[id]["comunidad"])
37 info.append( data[id]["version"])
38 info.append( data[id]["puerto"])
39 return info
40
41 def obtenerInfoAgente(id):
42     data={}
43     info=[]
44     with open('agentes.json','r') as f:
45         if os.path.getsize('agentes.json') > 0:
46             data=json.load(f)
47
48     aux=getInfo( data[id]["comunidad"], data[id]["ip"], int( data[id]["version"] ), int(
49 data[id]["puerto"]))
50 #obtener nombre versión y logo SO, número de interfaces de red, tiempo de
51 actividad desde último reinicio,
52 #ubicación física e información de contacto del administrador
53 #0->nombre so
54 if "Ubuntu" in aux[1]:
55     info.append("Ubuntu")
56     info.append("ubuntu.png")
57 elif "Windows" in aux[1]:
58     info.append("Windows")
59     info.append("windows.png")
60 elif "Darwin" in aux[1]:
61     info.append("MacOs")
62     info.append("macos.png")
63 else:
64     info.append("Linux")
65     info.append("linux.png")
66 continuar=0
67 info.append("Sin especificar")
68 for a in aux[1].split():
69     if continuar==1:
70         info[2]=a
71         break
72     if "Version" in a:
73         continuar=1
74 info.append(aux[2])
75 #info.append( int( aux[3])/100)
76 info.append(aux[3])
77 info.append(aux[4])
78 #print (info)
79 return info
80
81 def eliminar(id_agente):
82     data={}
83     with open('agentes.json','r+') as f:
84         if os.path.getsize('agentes.json') > 0:
85             data=json.load(f)

```



```

81     data.pop(id_agente, None)
82     #print(data)
83     f.seek(0)
84 with open('agentes.json', 'w') as f:
85     json.dump(data, f, sort_keys=True, indent=4)
86 from gestor import Gestor
87 gestor=Gestor()
88
89 class Agente():
90     def __init__(self, id_agente):
91         self.id_agente=id_agente
92         self.hostname, self.ip, self.conexion, self.comunidad, self.version, self.
puerto = obtenerInfoPrincipalAgente(id_agente)
93         self.nombre_so, self.logo_so, self.version_so, self.num_interfaces, self.
ubicacion, self.contacto=obtenerInfoAgente(id_agente)
94         #Falta obtener tiempo de reinicio

```

Este archivo, permite el registro de un nuevo agente.
agregar_agente.py:

```

1 from tkinter import *
2 from tkinter import messagebox
3 import json
4 import os.path
5 #from main import Main
6
7 class AgregarAgente():
8     """docstring for Agente"""
9     def __init__(self):
10         #Creamos la ventana y sus medidas
11         self.data={}
12         self.top=Tk()
13         self.top.geometry("500x500")
14         self.top.resizable(0,0)
15         #Definimos los elementos de la misma y a cada uno los acomodamos en el grid
16         self.b = Button(self.top, text="Regresar", command=self.cancelar)
17         self.b.grid(row=10, column=2, sticky=W+E+N+S)
18         self.b = Button(self.top, text="Agregar agente", command=self.crearAgente)
19         self.b.grid(row=10, column=1, sticky=W+E+N+S)
20
21         Label(self.top, text="Hostname").grid(row=1, sticky=W)
22         self.hostname=Entry(self.top)
23         self.hostname.grid(row=1, column=1)
24         Label(self.top, text="IP").grid(row=2, sticky=W)
25         self.ip=Entry(self.top)
26         self.ip.grid(row=2, column=1)
27         Label(self.top, text="Versión SNMP").grid(row=3, sticky=W)
28         self.version=Entry(self.top)
29         self.version.grid(row=3, column=1)
30         Label(self.top, text="Puerto").grid(row=4, sticky=W)
31         self.puerto=Entry(self.top)
32         self.puerto.insert(0, '161')
33         self.puerto.grid(row=4, column=1)
34         Label(self.top, text="Comunidad").grid(row=5, sticky=W)
35         self.comunidad=Entry(self.top)
36         self.comunidad.grid(row=5, column=1)
37         #Final
38         self.top.mainloop()
39         #Regresar a la pantalla anterior
40     def cancelar(self):
41         self.top.destroy()
42         #El import lo puse aquí por que si no marca un error extraño

```

```

43 from gestor import Gestor
44 self.gestor=Gestor()
45 #Crear un agente nuevo
46 def crearAgente(self):
47     agente_id=1
48     with open('agentes.json','r+') as f:
49         if os.path.getsize('agentes.json') > 0:
50             #print("No estoy vacío")
51             self.data = json.load(f)
52             f.seek(0)
53             llaves=list(self.data.keys())
54             lista = [int(x) for x in llaves]
55             lista.sort()
56             agente_id=int(llaves[len(lista)-1])+1
57             nuevo_agente={
58                 'hostname': self.hostname.get(),
59                 'ip': self.ip.get(),
60                 'version': self.version.get(),
61                 'puerto': self.puerto.get(),
62                 'comunidad': self.comunidad.get()
63             }
64             self.data[str(agente_id)]=nuevo_agente
65             json.dump(self.data,f,sort_keys=True,indent=4)
66     messagebox.showinfo("Éxito", "Agente registrado exitosamente")

```

Ejemplo de como se almacena la información sobre los agentes, para la persistencia. agentes.json:

```

1 {
2     "1": {
3         "comunidad": "public",
4         "hostname": "MacOs",
5         "ip": "localhost",
6         "puerto": "161",
7         "version": "1"
8     },
9     "2": {
10        "comunidad": "grupo4CM1",
11        "hostname": "Ubuntu1",
12        "ip": "localhost",
13        "puerto": "161",
14        "version": "1"
15    },
16    "3": {
17        "comunidad": "public",
18        "hostname": "Mac2",
19        "ip": "localhost",
20        "puerto": "161",
21        "version": "1"
22    },
23    "4": {
24        "comunidad": "variation/linux-full-walk",
25        "hostname": "Profa",
26        "ip": "10.100.71.200",
27        "puerto": "1024",
28        "version": "1"
29    }
30 }

```

Permite monitorear a un agente y además realiza la consultas correspondientes para mostrar las gráficas.

monitor.py:

```

1 from tkinter import *
2 from PIL import Image, ImageTk
3 import threading, time, calendar
4 from agente import Agente
5 from SNMP import *
6 from notificador import Notificador
7 from logger import Logger
8 import rrdtool
9 class Monitor(threading.Thread):
10     """docstring for Agente"""
11     def __init__(self, id_agente):
12         threading.Thread.__init__(self)
13         self.agente=Agente(id_agente)
14         self.data={}
15         self.top=Toplevel()
16         self.top.geometry("2100x800")
17         self.top.resizable(0,0)
18         self.b = Button(self.top, text="Salir", command=self.salir)
19         self.b.grid(row=1, column=2, sticky=W+E+N+S)
20         self.frame=Frame(self.top, width = 550, height = 150, relief = 'raised',
borderwidth=3)
21         self.frame.grid(row = 1, column = 0, columnspan=1, sticky=W+E+N+S)
22         self.continuar=True
23         self.image=[""]*8
24         self.photo=[""]*8
25         self.label=[""]*8
26         self.umbralCPU, self.umbralRAM, self.umbralHDD=self.obtenerUmbrales()
27         self.notificacionCPU=self.notificacionRAM=self.notificacionHDD=False
28         self.noti=Notificador(self.agente.hostname)
29         self.log=Logger(self.agente.hostname)
30         self.mostrarInfo()
31         self.crearRRDs()
32     def run(self):
33         while self.continuar:
34             #Interfaces
35             self.graficarRRD('IF-MIB','ifInOctets','ifOutOctets',"interfaz","Tráfico
de red en interfaces ('ifInOctets','ifOutOctets')",2)
36             #ICMP ->Deprecated
37             self.graficarRRD('IP-MIB','icmpInMsgs','icmpOutMsgs',"icmp","Tráfico ICMP
('icmpInMsgs','icmpOutMsgs')",0)
38             #TCP
39             self.graficarRRD('TCP-MIB','tcpInSegs','tcpOutSegs',"tcp","Tráfico
segmentos TCP ('tcpInSegs','tcpOutSegs')",0)
40             #SNMP
41             self.graficarRRD('SNMPv2-MIB','snmpInPkts','snmpOutPkts',"snmp", "Tráfico
SNMP ('snmpInPkts','snmpOutPkts')",0)
42             #####
43             #LOS ULTIMOS VALORES PARA CPU, RAM Y HDD, VARIAN PARA WINDOWS Y LINUX
44             #PARA WINDOWS LOS VALORES SON: 6,3,1
45             #PARA LINUX SON: 196608,1,36
46             #####
47             if self.agente.nombre_so=="Ubuntu":
48                 #UDP
49                 self.graficarRRD('UDP-MIB','udpInDatagrams','udpOutDatagrams',"udp","
Tráfico UDP Datagramas ('udpInDatagrams','udpOutDatagrams')",0)
50                 #CPU
51                 self.graficarCPU("HOST-RESOURCES-MIB","hrProcessorLoad","cpu","Uso del
Procesador",196608)
52                 #RAM
53                 self.graficarRAM("HOST-RESOURCES-MIB","hrStorageSize","hrStorageUsed",

```



```

hrStorageAllocationUnits","ram", "Uso de RAM",1)
#HDD
self.graficarHDD("HOST-RESOURCES-MIB","hrStorageSize","hrStorageUsed","
hrStorageAllocationUnits","hdd", "Uso de HDD",36)
elif self.agente.nombre_so=="Windows":
#UDP
self.graficarRRD('UDP-MIB','udpInDatagrams','udpOutDatagrams',"udp","
Tráfico UDP Datagramas ('udpInDatagrams','udpOutDatagrams')",0)
#CPU
self.graficarCPU("HOST-RESOURCES-MIB","hrProcessorLoad","cpu","Uso del
Procesador",6)
#RAM
self.graficarRAM("HOST-RESOURCES-MIB","hrStorageSize","hrStorageUsed","
hrStorageAllocationUnits","ram", "Uso de RAM",3)
#HDD
self.graficarHDD("HOST-RESOURCES-MIB","hrStorageSize","hrStorageUsed","
hrStorageAllocationUnits","hdd", "Uso de HDD",1)
elif self.agente.nombre_so=="MacOs":
#UDP
self.graficarRRD('UDP-MIB','udpInErrors','udpOutDatagrams',"udp","
Tráfico UDP Datagramas ('udpInDatagrams','udpOutDatagrams')",0)
#CPU
self.graficarCPU("HOST-RESOURCES-MIB","hrProcessorLoad","cpu","Uso del
Procesador",196608)
#RAM
self.graficarRAM("HOST-RESOURCES-MIB","hrStorageSize","hrStorageUsed","
hrStorageAllocationUnits","ram", "Uso de RAM",1)
#HDD->el 31 es tentativo
self.graficarHDD("HOST-RESOURCES-MIB","hrStorageSize","hrStorageUsed","
hrStorageAllocationUnits","hdd", "Uso de HDD",31)
elif self.agente.nombre_so=="Linux":
#UDP
self.graficarRRD('UDP-MIB','udpInDatagrams','udpOutDatagrams',"udp","
Tráfico UDP Datagramas ('udpInDatagrams','udpOutDatagrams')",0)
#CPU
self.graficarCPU("HOST-RESOURCES-MIB","hrProcessorLoad","cpu","Uso del
Procesador",1281)
#RAM
self.graficarRAM("HOST-RESOURCES-MIB","hrStorageSize","hrStorageUsed","
hrStorageAllocationUnits","ram", "Uso de RAM",1)
#HDD
self.graficarHDD("HOST-RESOURCES-MIB","hrStorageSize","hrStorageUsed","
hrStorageAllocationUnits","hdd", "Uso de HDD",31)

#Ahora actualizamos las imágenes
self.actualizarImagen(0,self.agente.hostname+"_"+self.agente.id_agente+"
_interfaz.png",2,0)

self.actualizarImagen(1,self.agente.hostname+"_"+self.agente.id_agente+"
_icmp.png",3,0)

self.actualizarImagen(2,self.agente.hostname+"_"+self.agente.id_agente+"
_tcp.png",2,1)

self.actualizarImagen(3,self.agente.hostname+"_"+self.agente.id_agente+"
_udp.png",3,1)

self.actualizarImagen(4,self.agente.hostname+"_"+self.agente.id_agente+"
_snmp.png",2,2)

self.actualizarImagen(5,self.agente.hostname+"_"+self.agente.id_agente+"

```

```

        _cpu.png",2,3)
96
97         self.actualizarImagen(6,self.agente.hostname+"_"+self.agente.id_agente+"
        _ram.png",3,3)
98
99         self.actualizarImagen(7,self.agente.hostname+"_"+self.agente.id_agente+"
        _hdd.png",2,4)
100
101         time.sleep(5)
102         self.top.destroy()
103     def salir(self):
104         self.continuar=False
105     def obtenerUmbrales(self):
106         umbrales=[]
107         with open("umbrales.txt") as f:
108             for i, linea in enumerate(f):
109                 umbrales.append(linea.split(":")[1])
110         return umbrales
111     def mostrarInfo(self):
112         imagen1=Image.open(self.agente.logo_so).resize((50,50),Image.ANTIALIAS)
113         photo1=ImageTk.PhotoImage(imagen1)
114         label1=Label(self.frame,image=photo1)
115         label1.image=photo1
116         label1.grid(row=1, column=1, sticky=W+E+N+S)
117         Label(self.frame, text="ID agente: "+self.agente.id_agente).grid(row=2,
        column=1, sticky=W)
118         Label(self.frame, text="Hostname: "+self.agente.hostname).grid(row=3, column
        =1, sticky=W)
119         Label(self.frame, text="IP: "+self.agente.ip).grid(row=4, column=1, sticky=W
        )
120         Label(self.frame, text="Comunidad: "+self.agente.comunidad).grid(row=5,
        column=1, sticky=W)
121         Label(self.frame, text="Versión SNMP: "+str((int(self.agente.version))+1)).
        grid(row=6, column=1, sticky=W)
122         Label(self.frame, text="Puerto: "+self.agente.puerto).grid(row=7, column=1,
        sticky=W)
123         Label(self.frame, text="Nombre SO: "+self.agente.nombre_so).grid(row=2,
        column=2, sticky=W)
124         Label(self.frame, text="Versión SO: "+self.agente.version_so).grid(row=3,
        column=2, sticky=W)
125         Label(self.frame, text="Número de interfaces: "+self.agente.num_interfaces).
        grid(row=4, column=2, sticky=W)
126         Label(self.frame, text="Ubicación física: "+self.agente.ubicacion).grid(row
        =5, column=2, sticky=W)
127         Label(self.frame, text="Contacto: "+self.agente.contacto).grid(row=6, column
        =2, sticky=W)
128     def crearRRDs(self):
129         #Para tr{afico de red:
130         crearRRDDos(self.agente.hostname+"_"+self.agente.id_agente+"_interfaz.rrd","
        N","1","60","1","1","100","100","COUNTER")
131         #Para tr{afico de IP
132         crearRRDDos(self.agente.hostname+"_"+self.agente.id_agente+"_icmp.rrd","N","
        1","60","1","1","100","100","GAUGE")
133         #Para tr{afico de TCP segmentos
134         crearRRDDos(self.agente.hostname+"_"+self.agente.id_agente+"_tcp.rrd","N","1
        ","60","1","1","100","100","GAUGE")
135         #Para tr{afico de Datagramas UDP
136         crearRRDDos(self.agente.hostname+"_"+self.agente.id_agente+"_udp.rrd","N","1
        ","60","1","1","100","100","COUNTER")
137         #Para tr{afico de SNMP
138         crearRRDDos(self.agente.hostname+"_"+self.agente.id_agente+"_snmp.rrd","N","

```

```

139 1","60","1","1","100","100","COUNTER")
140 #Para CPU
141 crearRRDUno(self.agente.hostname+"_"+self.agente.id_agente+"_cpu.rrd","N","1
142 ","60","1","1","100","100","GAUGE")
143 #Para Ram
144 crearRRDTres(self.agente.hostname+"_"+self.agente.id_agente+"_ram.rrd","N","
145 1","60","1","1","1","100","100","100","GAUGE")
146 #Para HDD
147 crearRRDTres(self.agente.hostname+"_"+self.agente.id_agente+"_hdd.rrd","N","
148 1","60","1","1","1","100","100","100","GAUGE")
149
150 def graficarRRD(self, grupo, oid1, oid2, archivo, header, numero):
151     ultimo=rrdtool.last(self.agente.hostname+"_"+self.agente.id_agente+"_"+
152     archivo+".rrd")
153     consulta=consultaSNMP(self.agente.comunidad, self.agente.ip, int(self.agente.
154     version), numero, grupo, oid1, oid2, int(self.agente.puerto))
155     if consulta[0]!="" or consulta[1]!="":
156         valor = "N:" + str(consulta[0]) + ':' + str(consulta[1])
157         #print (valor)
158         rrdtool.update(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+
159         ".rrd", valor)
160         rrdtool.dump(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".
161         rrd", self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".xml")
162         ret = rrdtool.graph(self.agente.hostname+"_"+self.agente.id_agente+"_"+
163         archivo+".png",
164                             "--start", str(ultimo-100),
165                             "--end", str(ultimo+100),
166                             "--vertical-label=Bytes/s",
167                             "--title="+header,
168                             "DEF:in="+self.agente.hostname+"_"+self.agente.
169                             id_agente+"_"+archivo+".rrd:in:AVERAGE",
170                             "DEF:out="+self.agente.hostname+"_"+self.agente.
171                             id_agente+"_"+archivo+".rrd:out:AVERAGE",
172                             "LINE1:in#00FF00:In traffic",
173                             "LINE1:out#0000FF:Out traffic")
174
175     def graficarCPU(self, grupo, oid1, archivo, header, numero):
176         umbral=int(self.umbralCPU)
177         ultimo=rrdtool.last(self.agente.hostname+"_"+self.agente.id_agente+"_"+
178         archivo+".rrd")
179         primero=rrdtool.first(self.agente.hostname+"_"+self.agente.id_agente+"_"+
180         archivo+".rrd")
181         consulta=consultav2SNMP(self.agente.comunidad, self.agente.ip, int(self.agente
182         .version), numero, grupo, oid1, int(self.agente.puerto))
183         if consulta!="":
184             valor = "N:" + str(consulta)
185             rrdtool.update(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+
186             ".rrd", valor)
187             rrdtool.dump(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".
188             rrd", self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".xml")
189             ret = rrdtool.graphv(self.agente.hostname+"_"+self.agente.id_agente+"_"+
190             archivo+".png",
191                                 "--start", str(ultimo-100),
192                                 "--end", str(ultimo+100),
193                                 #"--end", str(ultimo+200),
194                                 "--vertical-label=Porcentaje/s",
195                                 "--title="+header,
196                                 "--lower-limit", "0",
197                                 "--upper-limit", "100",
198                                 "DEF:usage="+self.agente.hostname+"_"+self.agente.
199                                 id_agente+"_"+archivo+".rrd:valor1:AVERAGE",
200                                 "CDEF:umbral"+str(umbral-20)+"_l=usage,"+str(umbral-20)

```

```

+","GT,usage,"+str(umbral-10)+","LT,EQ,usage,0,IF",
182         "CDEF:umbral"+str(umbral-10)+"_1=usage,"+str(umbral-10)+","GT,
usage,"+str(umbral)+","LT,EQ,usage,0,IF",
183         "CDEF:umbral"+str(umbral)+"_1=usage,"+str(umbral)+","GT,usage,"+
str(umbral+(100-umbral))+","LT,EQ,usage,0,IF",
184         "AREA:usage#FFBB0077:Uso de CPU",
185         "VDEF:m=usage,LSLSLOPE",
186         "VDEF:b=usage,LSLINT",
187         "CDEF:avg=usage,POP,m,COUNT,*,b,+",
188         "CDEF:umbral"+str(umbral-20)+"=avg,"+str(umbral-20)+"_"+str(
umbral-10)+","LIMIT",
189         "CDEF:umbral"+str(umbral-10)+"=avg,"+str(umbral-10)+"_"+str(
umbral)+","LIMIT",
190         "CDEF:umbral"+str(umbral)+"=avg,"+str(umbral)+"_"+str(umbral
+(100-umbral))+","LIMIT",
191         "VDEF:minUmbral"+str(umbral)+"=umbral"+str(umbral)+","FIRST",
192         "VDEF:last=usage,LAST",
193         "PRINT:last:%6.2lf %S",
194         "GPRINT:minUmbral"+str(umbral)+": Se alcanzará el "+str(umbral)
+"% @ %c :strftime",
195         "LINE1:avg#FF9F00",
196         "LINE2:"+str(umbral-20),
197         "AREA:10#FF000022::STACK",
198         "AREA:10#FF000044::STACK",
199         "AREA:"+str(100-umbral)+"#FF000066::STACK",
200         "AREA:umbral"+str(umbral-20)+"#0077FF77:Tráfico de carga
mayor que el "+str(umbral-20)+" y menor que el "+str(umbral-10)+" por ciento
",
201         "AREA:umbral"+str(umbral-10)+"#00880077:Tráfico de carga mayor
que el "+str(umbral-10)+" y menor que el "+str(umbral)+" por ciento",
202         "AREA:umbral"+str(umbral)+"#FF000088:Tráfico de carga mayor que
el "+str(umbral)+" y menor que el "+str(umbral+(100-umbral))+","por ciento",
203         "AREA:umbral"+str(umbral-20)+"_1#0077FF77",
204         "AREA:umbral"+str(umbral-10)+"_1#00880077",
205         "AREA:umbral"+str(umbral)+"_1#FF000088")
206     valor=float(ret["print[0]"])
207     if valor>float(umbral):
208         if not self.notificacionCPU:
209             self.notificacionCPU=True
210             self.noti.enviarCorreo(0,self.agente.hostname+"_"+self.agente.
id_agente+"_"+archivo+".png")
211             self.log.escribirLog(0)
212         else:
213             self.notificacionCPU=False
214
215     def graficarHDD(self, grupo, oid1, oid2, oid3, archivo, header, numero):
216         umbral=int(self.umbralHDD)
217         ultimo=rrdtool.last(self.agente.hostname+"_"+self.agente.id_agente+"_"+
archivo+".rrd")
218         primero=rrdtool.first(self.agente.hostname+"_"+self.agente.id_agente+"_"+
archivo+".rrd")
219         consulta=consultav5SNMP(self.agente.comunidad,self.agente.ip,int(self.agente
.version),numero,grupo,oid1,oid2,oid3,int(self.agente.puerto))
220         if consulta[0]!="" or consulta[1]!="" or consulta[2]!="":
221             valor = "N:" + str(consulta[0]) + ':' + str(consulta[1]) + ':' + str(
consulta[2])
222             rrdtool.update(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+
".rrd", valor)
223             rrdtool.dump(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".
rrd",self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".xml")
224             ret = rrdtool.graphv(self.agente.hostname+"_"+self.agente.id_agente+"_"+

```

```

archivo+".png",
    "--start",str(ultimo-100),
    "--end",str(ultimo+100),
    "--vertical-label=Porcentaje/s",
    "--title="+header,
    "--lower-limit","0",
    "--upper-limit","100",
    "DEF:val1="+self.agente.hostname+"_"+self.agente.
id_agente+"_"+archivo+".rrd:size:AVERAGE",
    "DEF:val2="+self.agente.hostname+"_"+self.agente.
id_agente+"_"+archivo+".rrd:used:AVERAGE",
    "DEF:val3="+self.agente.hostname+"_"+self.agente.
id_agente+"_"+archivo+".rrd:all:AVERAGE",
    "CDEF:totalHDD=val1,val3,*",
    "CDEF:usoHDDporcentaje=val2,val3,*,100,*,totalHDD,/ ",
    "CDEF:usoHDD=val2,val3,*",
    "CDEF:umbral"+str(umbral-20)+"_l=usoHDDporcentaje,"+str(umbral
-20)+",GT,usoHDDporcentaje,"+str(umbral-10)+"_l=usoHDDporcentaje,0,IF",
    "CDEF:umbral"+str(umbral-10)+"_l=usoHDDporcentaje,"+str(umbral
-10)+",GT,usoHDDporcentaje,"+str(umbral)+"_l=usoHDDporcentaje,0,IF",
    "CDEF:umbral"+str(umbral)+"_l=usoHDDporcentaje,"+str(umbral)+"
GT,usoHDDporcentaje,"+str(umbral+(100-umbral))+"_l=usoHDDporcentaje,0,IF
",
    "AREA:usoHDDporcentaje#FFBB0077:HDD en uso",
    "VDEF:m=usoHDDporcentaje,LSLSLOPE",
    "VDEF:b=usoHDDporcentaje,LSLINT",
    "CDEF:avg=usoHDDporcentaje,POP,m,COUNT,*,b,+",
    "CDEF:umbral"+str(umbral-20)+"=avg,"+str(umbral-20)+"",
+str(umbral-10)+"_l=LIMIT",
    "CDEF:umbral"+str(umbral-10)+"=avg,"+str(umbral-10)+"",+str(
umbral)+"_l=LIMIT",
    "CDEF:umbral"+str(umbral)+"=avg,"+str(umbral)+"",+str(umbral
+(100-umbral))+"_l=LIMIT",
    "VDEF:minUmbral"+str(umbral)+"=umbral"+str(umbral)+"_l=FIRST",
    "VDEF:last=usoHDDporcentaje,LAST",
    "PRINT:last:%6.2lf %S",
    "GPRINT:minUmbral"+str(umbral)+"_l: Se alcanzará el "+str(umbral)
+"%@ %c :strtime",
    "LINE1:avg#FF9F00",
    "LINE3:"+str(umbral-20),
    "AREA:10#FF000022::STACK",
    "AREA:10#FF000044::STACK",
    "AREA:"+str(100-umbral)+"_l#FF000066::STACK",
    "AREA:umbral"+str(umbral-20)+"_l#0077FF77:Uso mayor que el "+str(
umbral-20)+" y menor que el "+str(umbral-10)+" por ciento",
    "AREA:umbral"+str(umbral-10)+"_l#00880077:Uso mayor que el "+str(
umbral-10)+" y menor que el "+str(umbral)+" por ciento",
    "AREA:umbral"+str(umbral)+"_l#FF000088:Uso mayor que el "+str(
umbral)+" y menor que el "+str(umbral+(100-umbral))+" por ciento",
    "AREA:umbral"+str(umbral-20)+"_l#0077FF77",
    "AREA:umbral"+str(umbral-10)+"_l#00880077",
    "AREA:umbral"+str(umbral)+"_l#FF000088")
    valor=float(ret["print[0]"])
    if valor>float(umbral):
        if not self.notificacionHDD:
            self.notificacionHDD=True
            self.noti.enviarCorreo(2,self.agente.hostname+"_"+self.agente.
id_agente+"_"+archivo+".png")
            self.log.escribirLog(2)
        else:
            self.notificacionHDD=False

```

```

270
271 def graficarRAM(self, grupo, oid1, oid2, oid3, archivo, header, numero):
272     umbral=int(self.umbralRAM)
273     ultimo=rrdtool.last(self.agente.hostname+"_"+self.agente.id_agente+"_"+
274     archivo+".rrd")
275     primero=rrdtool.first(self.agente.hostname+"_"+self.agente.id_agente+"_"+
276     archivo+".rrd")
277     consulta=consultav5SNMP(self.agente.comunidad, self.agente.ip, int(self.agente
278     .version), numero, grupo, oid1, oid2, oid3, int(self.agente.puerto))
279     if consulta[0]!="" or consulta[1]!="" or consulta[2]!="":
280         valor = "N:" + str(consulta[0]) + ':' + str(consulta[1]) + ':' + str(
281         consulta[2])
282         rrdtool.update(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+
283         ".rrd", valor)
284         rrdtool.dump(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".
285         rrd", self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".xml")
286         ret = rrdtool.graphv(self.agente.hostname+"_"+self.agente.id_agente+"_"+
287         archivo+".png",
288             "--start", str(ultimo-100),
289             "--end", str(ultimo+100),
290             "--vertical-label=Porcentaje/s",
291             "--title="+header,
292             "--lower-limit", "0",
293             "--upper-limit", "100",
294             "DEF: val1="+self.agente.hostname+"_"+self.agente.
295             id_agente+"_"+archivo+".rrd:size:AVERAGE",
296             "DEF: val2="+self.agente.hostname+"_"+self.agente.
297             id_agente+"_"+archivo+".rrd:used:AVERAGE",
298             "DEF: val3="+self.agente.hostname+"_"+self.agente.
299             id_agente+"_"+archivo+".rrd:all:AVERAGE",
300             "CDEF: totalRAM=val1, val3, *",
301             "CDEF: usoRAMporcentaje=val2, val3, *, 100, *, totalRAM, /",
302             "CDEF: usoRAM=val2, val3, *",
303             "CDEF: umbral"+str(umbral-20)+"_1=usoRAMporcentaje,"+str(umbral
304             -20)+" ,GT, usoRAMporcentaje,"+str(umbral-10)+" ,LT,EQ, usoRAMporcentaje ,0 ,IF",
305             "CDEF: umbral"+str(umbral-10)+"_1=usoRAMporcentaje,"+str(umbral
306             -10)+" ,GT, usoRAMporcentaje,"+str(umbral)+" ,LT,EQ, usoRAMporcentaje ,0 ,IF",
307             "CDEF: umbral"+str(umbral)+"_1=usoRAMporcentaje,"+str(umbral)+" ,
308             GT, usoRAMporcentaje,"+str(umbral+(100-umbral))+" ,LT,EQ, usoRAMporcentaje ,0 ,IF
309             ",
310             "AREA: usoRAMporcentaje#FFBB0077:En uso",
311             "VDEF:m=usoRAMporcentaje,LSLSLOPE",
312             "VDEF:b=usoRAMporcentaje,LSLINT",
313             "CDEF: avg=usoRAMporcentaje ,POP,m,COUNT,* ,b,+ ",
314             "CDEF: umbral"+str(umbral-20)+"=avg,"+str(umbral-20)+" ,"+str(
315             umbral-10)+" ,LIMIT",
316             "CDEF: umbral"+str(umbral-10)+"=avg,"+str(umbral-10)+" ,"+str(
317             umbral)+" ,LIMIT",
318             "CDEF: umbral"+str(umbral)+"=avg,"+str(umbral)+" ,"+str(umbral
319             +(100-umbral))+" ,LIMIT",
320             "VDEF: minUmbral"+str(umbral)+"=umbral"+str(umbral)+" ,FIRST",
321             "VDEF: last=usoRAMporcentaje, LAST",
322             "PRINT: last:%6.2lf %S",
323             "GPRINT: minUmbral"+str(umbral)+" : Se alcanzará el "+str(umbral)
324             +"%@ %c :strftime",
325             "LINE1: avg#FF9F00",
326             "LINE3:"+str(umbral-20),
327             "AREA:10#FF000022::STACK",
328             "AREA:10#FF000044::STACK",
329             "AREA:"+str(100-umbral)+"#FF000066::STACK",
330             "AREA: umbral"+str(umbral-20)+"#0077FF77:Uso mayor que el "+str(

```



```

313     umbral-20)+" y menor que el "+str(umbral-10)+" por ciento",
        "AREA:umbral"+str(umbral-10)+"#00880077:Usó mayor que el "+str(
314     umbral-10)+" y menor que el "+str(umbral)+" por ciento",
        "AREA:umbral"+str(umbral)+"#FF000088:Usó mayor que el "+str(
315     umbral)+" y menor que el "+str(umbral+(100-umbral))+" por ciento",
        "AREA:umbral"+str(umbral-20)+"_1#0077FF77",
316     "AREA:umbral"+str(umbral-10)+"_1#00880077",
317     "AREA:umbral"+str(umbral)+"_1#FF000088")
318     valor=float(ret["print[0]"])
319     if valor>float(umbral):
320         if not self.notificacionRAM:
321             self.notificacionRAM=True
322             self.noti.enviarCorreo(1,self.agente.hostname+"_"+self.agente.
id_agente+"_"+archivo+".png")
323             self.log.escribirLog(1)
324         else:
325             self.notificacionRAM=False
326
327     def actualizarImagen(self,index,archivo,fila,columna):
328         self.image[index]=Image.open(archivo).resize((400,200),Image.ANTIALIAS)
329         self.photo[index]=ImageTk.PhotoImage(self.image[index])
330         self.label[index]=Label(self.top,image=self.photo[index])
331         self.label[index].image=self.photo[index]
332         self.label[index].grid(row=fila,column=columna,sticky=W)

```

Archivo en donde se encuentran todas las consultas necesarias para el correspondiente monitoreo.

SNMP.py:

```

1 from pysnmp.hlapi import *
2 import rrdtool
3 import time
4 #Funcion para obtener las interfaces
5 def getInterfaces(comunidad,host,version,puerto):
6     resultado=""
7     errorIndication,errorStatus,errorIndex,varBinds = next(getCmd(SnmpEngine(),
CommunityData(comunidad,mpModel=version),UdpTransportTarget((host,puerto)),
ContextData(),
8     ObjectType(ObjectIdentity('IF-MIB','ifNumber',0).addAsn1MibSource('file:///usr/share/snmp','http://mibs.snmplabs.com/asn1/@mib@'))))
9
10    if errorIndication:
11        print(errorIndication)
12    elif errorStatus:
13        print('%s at %s' %(errorStatus.prettyPrint(),
14            errorIndex and varBinds[int(errorIndex)-1][0] or '?')
15    )
16    else:
17        for varBind in varBinds:
18            VarB=('='.join([x.prettyPrint() for x in varBind]))
19            resultado=VarB.partition('=')[2]
20    return resultado
21
22 #Funcion para obtener el estatus de una interfaz
23 def getStatus(comunidad,host,version,interfaz,puerto):
24     errorIndication,errorStatus,errorIndex,varBinds = next(getCmd(SnmpEngine(),
CommunityData(comunidad,mpModel=version),UdpTransportTarget((host,puerto)),
ContextData(),
25     ObjectType(ObjectIdentity('IF-MIB','ifAdminStatus',interfaz).
addAsn1MibSource('file:///usr/share/snmp','http://mibs.snmplabs.com/asn1/
@mib@'))))

```

```

25
26 if errorIndication:
27     print(errorIndication)
28 elif errorStatus:
29     print('%s at %s' % (errorStatus.prettyPrint(),
30                         errorIndex and varBinds[int(errorIndex) - 1][0] or '?')
31     ))
32 else:
33     for varBind in varBinds:
34         VarB=(' = '.join([x.prettyPrint() for x in varBind]))
35         resultado=VarB.partition(' = ')[2]
36     return resultado
37
38 #Funcion para obtener informacion del estado del dispositivo
39 def getInfo(comunidad, host, version, puerto):
40     resultado=[]
41     errorIndication, errorStatus, errorIndex, varBinds = next(getCmd(SnmpEngine(),
42                             CommunityData(comunidad, mpModel=version),
43                             UdpTransportTarget((host, puerto)),
44                             ContextData(),
45                             ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysName', 0).addAsn1MibSource('file:///usr/share/snmp', 'http://mibs.snmplabs.com/asn1/@mib@')),
46                             ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysDescr', 0).addAsn1MibSource('file:///usr/share/snmp', 'http://mibs.snmplabs.com/asn1/@mib@')),
47                             ObjectType(ObjectIdentity('IF-MIB', 'ifNumber', 0).addAsn1MibSource('file:///usr/share/snmp', 'http://mibs.snmplabs.com/asn1/@mib@')),
48                             #ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysUpTime', 0).addAsn1MibSource('file:///usr/share/snmp', 'http://mibs.snmplabs.com/asn1/@mib@')),
49                             ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysLocation', 0).addAsn1MibSource('file:///usr/share/snmp', 'http://mibs.snmplabs.com/asn1/@mib@')),
50                             ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysContact', 0).addAsn1MibSource('file:///usr/share/snmp', 'http://mibs.snmplabs.com/asn1/@mib@'))))
51
52 if errorIndication:
53     print(errorIndication)
54 elif errorStatus:
55     print('%s at %s' % (errorStatus.prettyPrint(),
56                         errorIndex and varBinds[int(errorIndex) - 1][0] or '?')
57     ))
58 else:
59     for varBind in varBinds:
60         VarB=(' = '.join([x.prettyPrint() for x in varBind]))
61         resultado.append(VarB.partition(' = ')[2])
62     return resultado
63
64 def getInfo2(comunidad, host, version, puerto, oids):
65     resultado=[]
66     for element in oids:
67         errorIndication, errorStatus, errorIndex, varBinds = next(getCmd(SnmpEngine(),
68                             CommunityData(comunidad, mpModel=version),
69                             UdpTransportTarget((host, puerto)),
70                             ContextData(),
71                             ObjectType(ObjectIdentity(element))))
72         if errorIndication:
73             print(errorIndication)
74         elif errorStatus:
75             print('%s at %s' % (errorStatus.prettyPrint(),
76                                 errorIndex and varBinds[int(errorIndex) - 1][0] or '?')
77             ))
78         else:
79             for varBind in varBinds:
80                 VarB=(' = '.join([x.prettyPrint() for x in varBind]))
81                 resultado.append(VarB.partition(' = ')[2])

```



```

73     return resultado
74
75 #Funcion para hacer consultas version facil (entrada y salida)
76 def consultaSNMP(comunidad, host, version, interfaz, grupo, objeto1, objeto2, puerto):
77     resultado=[]
78     errorIndication, errorStatus, errorIndex, varBinds = next(getCmd(SnmpEngine(),
79         CommunityData(comunidad, mpModel=version), UdpTransportTarget((host, puerto)),
80         ContextData(),
81         ObjectType(ObjectIdentity(grupo, objeto1, interfaz).addAsn1MibSource('file:///usr/share/snmp', 'http://mibs.snmplabs.com/asn1/@mib@')),
82         ObjectType(ObjectIdentity(grupo, objeto2, interfaz).addAsn1MibSource('file:///usr/share/snmp', 'http://mibs.snmplabs.com/asn1/@mib@'))))
83
84     if errorIndication:
85         print(errorIndication)
86         resultado.append("")
87         resultado.append("")
88     elif errorStatus:
89         print('%s at %s' % (errorStatus.prettyPrint(),
90             errorIndex and varBinds[int(errorIndex) - 1][0] or '?'))
91     else:
92         for varBind in varBinds:
93             VarB=(' = '.join([x.prettyPrint() for x in varBind]))
94             resultado.append(VarB.partition(' = ')[2])
95     return resultado
96
97 #Funcion para hacer consultas a un solo objeto con referencias al nombre
98 def consultav2SNMP(comunidad, host, version, interfaz, grupo, objeto, puerto):
99     resultado=""
100     errorIndication, errorStatus, errorIndex, varBinds = next(getCmd(SnmpEngine(),
101         CommunityData(comunidad, mpModel=version), UdpTransportTarget((host, puerto)),
102         ContextData(),
103         ObjectType(ObjectIdentity(grupo, objeto, interfaz).addAsn1MibSource('file:///usr/share/snmp', 'http://mibs.snmplabs.com/asn1/@mib@'))))
104
105     if errorIndication:
106         print(errorIndication)
107     elif errorStatus:
108         print('%s at %s' % (errorStatus.prettyPrint(),
109             errorIndex and varBinds[int(errorIndex) - 1][0] or '?'))
110     else:
111         for varBind in varBinds:
112             VarB=(' = '.join([x.prettyPrint() for x in varBind]))
113             resultado=VarB.partition(' = ')[2]
114     return resultado
115
116 #Funcion para hacer consultas de un objeto con el OID
117 def consultav3SNMP(comunidad, host, version, oid, puerto):
118     resultado=""
119     errorIndication, errorStatus, errorIndex, varBinds = next(getCmd(SnmpEngine(),
120         CommunityData(comunidad, mpModel=version), UdpTransportTarget((host, puerto)),
121         ContextData(),
122         ObjectType(ObjectIdentity(oid))))
123
124     if errorIndication:
125         print(errorIndication)
126     elif errorStatus:
127         print('%s at %s' % (errorStatus.prettyPrint(),
128             errorIndex and varBinds[int(errorIndex) - 1][0] or '?'))

```

```

    ))
123 else:
124     for varBind in varBinds:
125         VarB=(' = '.join([x.prettyPrint() for x in varBind]))
126         resultado=VarB.partition(' = ')[2]
127     return resultado
128
129 #Funcion para hacer consultas de dos objetos con OID
130 def consultav4SNMP(comunidad, host, version, oid1, oid2, puerto):
131     resultado=[]
132     errorIndication, errorStatus, errorIndex, varBinds = next(getCmd(SnmpEngine(),
133         CommunityData(comunidad, mpModel=version), UdpTransportTarget((host, puerto)),
134         ContextData(),
135         ObjectType(ObjectIdentity(oid1)),
136         ObjectType(ObjectIdentity(oid2))))
137
138     if errorIndication:
139         print(errorIndication)
140     elif errorStatus:
141         print('%s at %s' % (errorStatus.prettyPrint(),
142             errorIndex and varBinds[int(errorIndex) - 1][0] or '?'))
143     else:
144         for varBind in varBinds:
145             VarB=(' = '.join([x.prettyPrint() for x in varBind]))
146             resultado.append(VarB.partition(' = ')[2])
147     return resultado
148
149 #Funcion para ram y HDD
150 def consultav5SNMP(comunidad, host, version, interfaz, grupo, objeto1, objeto2, objeto3,
151     , puerto):
152     resultado=[]
153     errorIndication, errorStatus, errorIndex, varBinds = next(getCmd(SnmpEngine(),
154         CommunityData(comunidad, mpModel=version), UdpTransportTarget((host, puerto)),
155         ContextData(),
156         ObjectType(ObjectIdentity(grupo, objeto1, interfaz).addAsn1MibSource('file:///
157     usr/share/snmp', 'http://mibs.snmplabs.com/asn1/@mib@')),
158         ObjectType(ObjectIdentity(grupo, objeto2, interfaz).addAsn1MibSource('file:///
159     usr/share/snmp', 'http://mibs.snmplabs.com/asn1/@mib@')),
160         ObjectType(ObjectIdentity(grupo, objeto3, interfaz).addAsn1MibSource('file:///
161     usr/share/snmp', 'http://mibs.snmplabs.com/asn1/@mib@'))))
162
163     if errorIndication:
164         print(errorIndication)
165         resultado.append("")
166         resultado.append("")
167     elif errorStatus:
168         print('%s at %s' % (errorStatus.prettyPrint(),
169             errorIndex and varBinds[int(errorIndex) - 1][0] or '?'))
170     else:
171         for varBind in varBinds:
172             VarB=(' = '.join([x.prettyPrint() for x in varBind]))
173             resultado.append(VarB.partition(' = ')[2])
174     return resultado
175
176 #Funcion para crear el archivo .rrd con un valor
177 def crearRRDUno(nombre, inicio, step, tiempo, steps1, steps2, row1, row2, tipo):
178     ret=rrdtool.create(nombre, '--start', inicio, '--step', step,
179         "DS:valor1:" + tipo + ":" + tiempo + ":U:U",
180         "RRA:AVERAGE:0.5:" + steps1 + ":" + row1,

```

```

173     "RRA:AVERAGE:0.5:" + steps2 + ":" + row2)
174     if ret:
175         print(rrdtool.error())
176
177 #Funcion para crear el archivo .rrd con dos valores
178 def crearRRDDos(nombre, inicio, step, tiempo, steps1, steps2, row1, row2, tipo):
179     ret=rrdtool.create(nombre, '--start', inicio, '--step', step,
180         "DS:in:" + tipo + ":" + tiempo + ":U:U",
181         "DS:out:" + tipo + ":" + tiempo + ":U:U",
182         "RRA:AVERAGE:0.5:" + steps1 + ":" + row1,
183         "RRA:AVERAGE:0.5:" + steps2 + ":" + row2)
184     if ret:
185         print(rrdtool.error())
186
187 def crearRRDTres(nombre, inicio, step, tiempo, steps1, steps2, steps3, row1, row2, row3,
188     tipo):
189     ret=rrdtool.create(nombre, '--start', inicio, '--step', step,
190         "DS:size:" + tipo + ":" + tiempo + ":U:U",
191         "DS:used:" + tipo + ":" + tiempo + ":U:U",
192         "DS:all:" + tipo + ":" + tiempo + ":U:U",
193         "RRA:AVERAGE:0.5:" + steps1 + ":" + row1,
194         "RRA:AVERAGE:0.5:" + steps2 + ":" + row2,
195         "RRA:AVERAGE:0.5:" + steps3 + ":" + row3)
196     if ret:
197         print(rrdtool.error())

```

Posteriormente, para la detección de umbrales, se uso el siguiente código.
detector_umbrales.py:

```

1 from agente import Agente
2 from SNMP import *
3 import rrdtool
4 import time
5 class DetectorUmbrales():
6     def __init__(self, id_agente, tiempo, intervalo):
7         self.agente=Agente(id_agente)
8         self.umbralCPU=self.umbralRAM=self.umbralHDD=90
9         self.tiempo_inicio = float(0)
10        self.tiempo=tiempo
11        self.intervalo=intervalo
12        self.crearRDDs()
13    def crearRDDs(self):
14        #Para CPU
15        crearRRDUno(self.agente.hostname+"_"+self.agente.id_agente+"_cpu_umbral.rrd",
16            "N","1",str(self.tiempo*60),"1","1",str(self.tiempo*60),str(self.tiempo*60),
17            "GAUGE")
18        #Para Ram
19        crearRRDTres(self.agente.hostname+"_"+self.agente.id_agente+"_ram_umbral.rrd",
20            "N","1",str(self.tiempo*60),"1","1","1",str(self.tiempo*60),str(self.
21            tiempo*60),str(self.tiempo*60),"GAUGE")
22        #Para HDD
23        crearRRDTres(self.agente.hostname+"_"+self.agente.id_agente+"_hdd_umbral.rrd",
24            "N","1",str(self.tiempo*60),"1","1","1",str(self.tiempo*60),str(self.
25            tiempo*60),str(self.tiempo*60),"GAUGE")
26        with open("umbral_CPU.txt","w") as f:
27            pass
28        with open("umbral_RAM.txt","w") as f:
29            pass
30        with open("umbral_HDD.txt","w") as f:
31            pass
32    def obtenerUmbrales(self):
33        self.tiempo_inicio=time.time()

```

```

28 while float((time.time()-self.tiempo_inicio))<(float(self.tiempo*60)):
29     if self.agente.nombre_so=="Ubuntu":
30         #CPU
31         self.graficarCPU("HOST-RESOURCES-MIB","hrProcessorLoad","cpu_umbral","
Uso del Procesador",196608)
32         #RAM
33         self.graficarRAM("HOST-RESOURCES-MIB","hrStorageSize","hrStorageUsed","
hrStorageAllocationUnits","ram_umbral", "Uso de RAM",1)
34         #HDD
35         self.graficarHDD("HOST-RESOURCES-MIB","hrStorageSize","hrStorageUsed","
hrStorageAllocationUnits","hdd_umbral", "Uso de HDD",36)
36     elif self.agente.nombre_so=="Windows":
37         #CPU
38         self.graficarCPU("HOST-RESOURCES-MIB","hrProcessorLoad","cpu_umbral","
Uso del Procesador",6)
39         #RAM
40         self.graficarRAM("HOST-RESOURCES-MIB","hrStorageSize","hrStorageUsed","
hrStorageAllocationUnits","ram_umbral", "Uso de RAM",3)
41         #HDD
42         self.graficarHDD("HOST-RESOURCES-MIB","hrStorageSize","hrStorageUsed","
hrStorageAllocationUnits","hdd_umbral", "Uso de HDD",1)
43     elif self.agente.nombre_so=="MacOs":
44         #CPU
45         self.graficarCPU("HOST-RESOURCES-MIB","hrProcessorLoad","cpu_umbral","
Uso del Procesador",196608)
46         #RAM
47         self.graficarRAM("HOST-RESOURCES-MIB","hrStorageSize","hrStorageUsed","
hrStorageAllocationUnits","ram_umbral", "Uso de RAM",1)
48         #HDD->el 31 es tentativo
49         self.graficarHDD("HOST-RESOURCES-MIB","hrStorageSize","hrStorageUsed","
hrStorageAllocationUnits","hdd_umbral", "Uso de HDD",31)
50     elif self.agente.nombre_so=="Linux":
51         #CPU
52         self.graficarCPU("HOST-RESOURCES-MIB","hrProcessorLoad","cpu_umbral","
Uso del Procesador",1281)
53         #RAM
54         self.graficarRAM("HOST-RESOURCES-MIB","hrStorageSize","hrStorageUsed","
hrStorageAllocationUnits","ram_umbral", "Uso de RAM",1)
55         #HDD
56         self.graficarHDD("HOST-RESOURCES-MIB","hrStorageSize","hrStorageUsed","
hrStorageAllocationUnits","hdd_umbral", "Uso de HDD",31)
57         time.sleep(self.intervalo)
58 def graficarCPU(self, grupo, oid1, archivo, header, numero):
59     umbral=int(self.umbralCPU)
60     ultimo=rrdtool.last(self.agente.hostname+"_"+self.agente.id_agente+"_"+
archivo+".rrd")
61     primero=rrdtool.first(self.agente.hostname+"_"+self.agente.id_agente+"_"+
archivo+".rrd")
62     consulta=consultav2SNMP(self.agente.comunidad, self.agente.ip, int(self.agente
.version), numero, grupo, oid1, int(self.agente.puerto))
63     if consulta!="":
64         valor = "N:" + str(consulta)
65         rrdtool.update(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+
".rrd", valor)
66         rrdtool.dump(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".
rrd", self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".xml")
67         ret = rrdtool.graphv(self.agente.hostname+"_"+self.agente.id_agente+"_"+
archivo+".png",
68                               "--start", str(ultimo-100),
69                               "--end", "+100",
70                               #"--end", str(ultimo+200),

```

```

71         "--vertical-label=Porcentaje/s",
72         "--title="+header,
73         "--lower-limit","0",
74         "--upper-limit","100",
75         "DEF:usage="+self.agente.hostname+"_"+self.agente.
id_agente+"_"+archivo+".rrd:valor1:AVERAGE",
76         "CDEF:umbral"+str(umbral-20)+"_l=usage,"+str(umbral-20)
+" ,GT,usage,"+str(umbral-10)+" ,LT,EQ,usage,0,IF",
77         "CDEF:umbral"+str(umbral-10)+"_l=usage,"+str(umbral-10)+" ,GT,
usage,"+str(umbral)+" ,LT,EQ,usage,0,IF",
78         "CDEF:umbral"+str(umbral)+"_l=usage,"+str(umbral)+" ,GT,usage,"+
str(umbral+(100-umbral))+" ,LT,EQ,usage,0,IF",
79         "AREA:usage#FFBB0077:Uso de CPU",
80         "VDEF:m=usage,LSLSLOPE",
81         "VDEF:b=usage,LSLINT",
82         "CDEF:avg=usage,POP,m,COUNT,*,b,+ ",
83         "CDEF:umbral"+str(umbral-20)+"=avg,"+str(umbral-20)+" ,"+str(
umbral-10)+" ,LIMIT",
84         "CDEF:umbral"+str(umbral-10)+"=avg,"+str(umbral-10)+" ,"+str(
umbral)+" ,LIMIT",
85         "CDEF:umbral"+str(umbral)+"=avg,"+str(umbral)+" ,"+str(umbral
+(100-umbral))+" ,LIMIT",
86         "VDEF:minUmbral"+str(umbral)+"=umbral"+str(umbral)+" ,FIRST",
87         "VDEF:last=usage,LAST",
88         "PRINT:last:%6.2lf %S",
89         "GPRINT:minUmbral"+str(umbral)+" : Se alcanzará el "+str(umbral)
+"% @ %c : strftime",
90         "LINE1:avg#FF9F00",
91         "LINE2:"+str(umbral-20),
92         "AREA:10#FF000022::STACK",
93         "AREA:10#FF000044::STACK",
94         "AREA:"+str(100-umbral)+"#FF000066::STACK",
95         "AREA:umbral"+str(umbral-20)+"#0077FF77:Tráfico de carga
mayor que el "+str(umbral-20)+" y menor que el "+str(umbral-10)+" por ciento
",
96         "AREA:umbral"+str(umbral-10)+"#00880077:Tráfico de carga mayor
que el "+str(umbral-10)+" y menor que el "+str(umbral)+" por ciento",
97         "AREA:umbral"+str(umbral)+"#FF000088:Tráfico de carga mayor que
el "+str(umbral)+" y menor que el "+str(umbral+(100-umbral))+" por ciento",
98         "AREA:umbral"+str(umbral-20)+"_l#0077FF77",
99         "AREA:umbral"+str(umbral-10)+"_l#00880077",
100        "AREA:umbral"+str(umbral)+"_l#FF000088")
101        valor=float(ret["print[0]"])
102        with open("umbral_CPU.txt","a") as f:
103            f.write(str(valor)+"\n")
104            f.flush()
105    def graficarHDD(self, grupo, oid1, oid2, oid3, archivo, header, numero):
106        umbral=int(self.umbralHDD)
107        ultimo=rrdtool.last(self.agente.hostname+"_"+self.agente.id_agente+"_"+
archivo+".rrd")
108        primero=rrdtool.first(self.agente.hostname+"_"+self.agente.id_agente+"_"+
archivo+".rrd")
109        consulta=consultav5SNMP(self.agente.comunidad, self.agente.ip, int(self.agente
.version), numero, grupo, oid1, oid2, oid3, int(self.agente.puerto))
110        if consulta[0]!="" or consulta[1]!="" or consulta[2]!="":
111            valor = "N:" + str(consulta[0]) + ':' + str(consulta[1]) + ':' + str(
consulta[2])
112            rrdtool.update(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+
".rrd", valor)
113            rrdtool.dump(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".
rrd", self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".xml")

```

```

114     ret = rrdtool.graphv(self.agente.hostname+"_"+self.agente.id_agente+"_"+
115         archivo+".png",
116         "--start",str(ultimo-100),
117         "--end",str(ultimo+100),
118         "--vertical-label=Porcentaje/s",
119         "--title="+header,
120         "--lower-limit","0",
121         "--upper-limit","100",
122         "DEF:val1="+self.agente.hostname+"_"+self.agente.
123         id_agente+"_"+archivo+".rrd:size:AVERAGE",
124         "DEF:val2="+self.agente.hostname+"_"+self.agente.
125         id_agente+"_"+archivo+".rrd:used:AVERAGE",
126         "DEF:val3="+self.agente.hostname+"_"+self.agente.
127         id_agente+"_"+archivo+".rrd:all:AVERAGE",
128         "CDEF:totalHDD=val1,val3,*",
129         "CDEF:usoHDDporcentaje=val2,val3,*,100,*,totalHDD,/ ",
130         "CDEF:usoHDD=val2,val3,*",
131         "CDEF:umbral"+str(umbral-20)+"_1=usoHDDporcentaje,"+str(umbral
132         -20)+",GT,usoHDDporcentaje,"+str(umbral-10)+",LT,EQ,usoHDDporcentaje,0,IF",
133         "CDEF:umbral"+str(umbral-10)+"_1=usoHDDporcentaje,"+str(umbral
134         -10)+",GT,usoHDDporcentaje,"+str(umbral)+",LT,EQ,usoHDDporcentaje,0,IF",
135         "CDEF:umbral"+str(umbral)+"_1=usoHDDporcentaje,"+str(umbral)+",
136         GT,usoHDDporcentaje,"+str(umbral+(100-umbral))+",LT,EQ,usoHDDporcentaje,0,IF",
137         "AREA:usoHDDporcentaje#FFBB0077:HDD en uso",
138         "VDEF:m=usoHDDporcentaje,LSLSLOPE",
139         "VDEF:b=usoHDDporcentaje,LSLINT",
140         "CDEF:avg=usoHDDporcentaje,POP,m,COUNT,*,b,+",
141         "CDEF:umbral"+str(umbral-20)+"=avg,"+str(umbral-20)+",",
142         "+str(umbral-10)+",LIMIT",
143         "CDEF:umbral"+str(umbral-10)+"=avg,"+str(umbral-10)+",",
144         "+str(umbral)+",LIMIT",
145         "CDEF:umbral"+str(umbral)+"=avg,"+str(umbral)+",",
146         "+str(umbral+(100-umbral))+",LIMIT",
147         "VDEF:minUmbral"+str(umbral)+"=umbral"+str(umbral)+",FIRST",
148         "VDEF:last=usoHDDporcentaje,LAST",
149         "PRINT:last:%6.2lf %S",
150         "GPRINT:minUmbral"+str(umbral)+": Se alcanzará el "+str(umbral)
151         +"%@ %c :strftime",
152         "LINE1:avg#FF9F00",
153         "LINE3:"+str(umbral-20),
154         "AREA:10#FF000022::STACK",
155         "AREA:10#FF000044::STACK",
156         "AREA:"+str(100-umbral)+"#FF000066::STACK",
157         "AREA:umbral"+str(umbral-20)+"#0077FF77:Uso mayor que el "+str(
158         umbral-20)+" y menor que el "+str(umbral-10)+" por ciento",
159         "AREA:umbral"+str(umbral-10)+"#00880077:Uso mayor que el "+str(
160         umbral-10)+" y menor que el "+str(umbral)+" por ciento",
161         "AREA:umbral"+str(umbral)+"#FF000088:Uso mayor que el "+str(
162         umbral)+" y menor que el "+str(umbral+(100-umbral))+", por ciento",
163         "AREA:umbral"+str(umbral-20)+"_1#0077FF77",
164         "AREA:umbral"+str(umbral-10)+"_1#00880077",
165         "AREA:umbral"+str(umbral)+"_1#FF000088")
166     valor=float(ret["print[0]"])
167     with open("umbral_HDD.txt","a") as f:
168         f.write(str(valor)+"\n")
169         f.flush()
170
171 def graficarRAM(self, grupo, oid1, oid2, oid3, archivo, header, numero):
172     umbral=int(self.umbralRAM)
173     ultimo=rrdtool.last(self.agente.hostname+"_"+self.agente.id_agente+"_"+

```



```

archivo+".rrd")
primero=rrdtool.first(self.agente.hostname+"_"+self.agente.id_agente+"_"+
archivo+".rrd")
consulta=consultav5SNMP(self.agente.comunidad,self.agente.ip,int(self.agente
.version),numero,grupo,oid1,oid2,oid3,int(self.agente.puerto))
if consulta[0]!="" or consulta[1]!="" or consulta[2]!="":
    valor = "N:" + str(consulta[0]) + ':' + str(consulta[1]) + ':' + str(
consulta[2])
    rrdtool.update(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+
".rrd", valor)
    rrdtool.dump(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".
rrd",self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".xml")
    ret = rrdtool.graphv(self.agente.hostname+"_"+self.agente.id_agente+"_"+
archivo+".png",
        "--start",str(ultimo-100),
        "--end","+100",
        "--vertical-label=Porcentaje/s",
        "--title="+header,
        "--lower-limit","0",
        "--upper-limit","100",
        "DEF:val1="+self.agente.hostname+"_"+self.agente.
id_agente+"_"+archivo+".rrd:size:AVERAGE",
        "DEF:val2="+self.agente.hostname+"_"+self.agente.
id_agente+"_"+archivo+".rrd:used:AVERAGE",
        "DEF:val3="+self.agente.hostname+"_"+self.agente.
id_agente+"_"+archivo+".rrd:all:AVERAGE",
        "CDEF:totalRAM=val1,val3,*",
        "CDEF:usoRAMporcentaje=val2,val3,*,100,*,totalRAM,/ ",
        "CDEF:usoRAM=val2,val3,*",
        "CDEF:umbral"+str(umbral-20)+"_1=usoRAMporcentaje,"+str(umbral
-20)+",GT,usoRAMporcentaje,"+str(umbral-10)+",LT,EQ,usoRAMporcentaje,0,IF",
        "CDEF:umbral"+str(umbral-10)+"_1=usoRAMporcentaje,"+str(umbral
-10)+",GT,usoRAMporcentaje,"+str(umbral)+",LT,EQ,usoRAMporcentaje,0,IF",
        "CDEF:umbral"+str(umbral)+"_1=usoRAMporcentaje,"+str(umbral)+",
GT,usoRAMporcentaje,"+str(umbral+(100-umbral))+",LT,EQ,usoRAMporcentaje,0,IF
",
        "AREA:usoRAMporcentaje#FFBB0077:En uso",
        "VDEF:m=usoRAMporcentaje,LSLSLOPE",
        "VDEF:b=usoRAMporcentaje,LSLINT",
        "CDEF:avg=usoRAMporcentaje,POP,m,COUNT,*,b,+",
        "CDEF:umbral"+str(umbral-20)+"=avg,"+str(umbral-20)+"_"+str(
umbral-10)+"_1,LIMIT",
        "CDEF:umbral"+str(umbral-10)+"=avg,"+str(umbral-10)+"_"+str(
umbral)+"_1,LIMIT",
        "CDEF:umbral"+str(umbral)+"=avg,"+str(umbral)+"_"+str(umbral
+(100-umbral))+",LIMIT",
        "VDEF:minUmbral"+str(umbral)+"=umbral"+str(umbral)+"_1,FIRST",
        "VDEF:last=usoRAMporcentaje,LAST",
        "PRINT:last:%6.2lf %S",
        "GPRINT:minUmbral"+str(umbral)+"_1: Se alcanzará el "+str(umbral)
+"%@ %c :strftime",
        "LINE1:avg#FF9F00",
        "LINE3:"+str(umbral-20),
        "AREA:10#FF000022::STACK",
        "AREA:10#FF000044::STACK",
        "AREA:"+str(100-umbral)+"#FF000066::STACK",
        "AREA:umbral"+str(umbral-20)+"#0077FF77:Uso mayor que el "+str(
umbral-20)+" y menor que el "+str(umbral-10)+" por ciento",
        "AREA:umbral"+str(umbral-10)+"#00880077:Uso mayor que el "+str(
umbral-10)+" y menor que el "+str(umbral)+" por ciento",
        "AREA:umbral"+str(umbral)+"#FF000088:Uso mayor que el "+str(

```

```

201     umbral)+" y menor que el "+str(umbral+(100-umbral))+" por ciento",
202         "AREA:umbral"+str(umbral-20)+"_1#0077FF77",
203         "AREA:umbral"+str(umbral-10)+"_1#00880077",
204         "AREA:umbral"+str(umbral)+"_1#FF000088")
205     valor=float(ret["print[0]"])
206     with open("umbral_RAM.txt","a") as f:
207         f.write(str(valor)+"\n")
208         f.flush()
209 detector=DetectorUmbrales("4",10,0)
210 detector.obtenerUmbrales()

```

Para enviar los correos electrónicos de notificación cuando se sobrepasa un umbral se uso el código siguiente.

notificador.py:

```

1 import smtplib
2 from email.mime.image import MIMEImage
3 from email.mime.text import MIMEText
4 from email.mime.multipart import MIMEMultipart
5 import datetime
6
7 class Notificador():
8     def __init__(self, agente):
9         self.remitente = "admiredes3.4CMI@gmail.com"
10        self.destinatario = "admiredes3.4CMI@gmail.com"
11        self.servidor = 'smtp.gmail.com: 587'
12        self.contra = '#Equipo5'
13        self.agente=agente
14    def enviarCorreo(self, tipo, imagen=""):
15        asunto, texto=self.obtenerContenido(tipo)
16        hora=self.obtenerHora()
17        texto="El agente: "+self.agente+" "+texto+hora
18        #Definimos contenido del correo
19        mensaje = MIMEMultipart()
20        mensaje['Subject'] = asunto
21        mensaje['From'] = self.remitente
22        mensaje['To'] = self.destinatario
23        mensaje.attach(MIMEText(texto, "plain"))
24        #Verificamos si hay que enviar una imagen
25        if imagen!="":
26            with open(imagen, 'rb') as f:
27                img = MIMEImage(f.read())
28                f.close()
29                mensaje.attach(img)
30        #Enviamos el correo
31        servidor_correo = smtplib.SMTP(self.servidor)
32        servidor_correo.starttls()
33        servidor_correo.login(self.remitente, self.contra)
34        servidor_correo.sendmail(self.remitente, self.destinatario, mensaje.as_string())
35        servidor_correo.quit()
36    def obtenerContenido(self, tipo):
37        with open("mensajes.txt") as f:
38            for i, linea in enumerate(f):
39                if i==tipo:
40                    return linea.split(";")
41            else:
42                return ["", ""]
43    def obtenerHora(self):
44        return str(datetime.datetime.now())

```


Al igual que se envía una notificación, también se escribe en un archivo de log, el cual es controlado por el siguiente script.

logger.py:

```

1 import datetime
2 class Logger():
3     def __init__(self, agente):
4         self.agente=agente
5     def escribirLog(self, tipo):
6         contenido=self.obtenerContenido(tipo)
7         hora=self.obtenerHora()
8         with open('log.txt', 'a') as f:
9             f.write(self.agente+contenido.replace("\n", "")+hora+"\n")
10            f.flush()
11            print("Problema detectado, revisar archivo de log o correo electrónico")
12    def obtenerContenido(self, tipo):
13        with open("mensajes.txt") as f:
14            for i, linea in enumerate(f):
15                if i==tipo:
16                    return linea.split(";")[1]
17            else:
18                return ""
19    def obtenerHora(self):
20        return str(datetime.datetime.now())

```

4.2. Examen 2

En este examen, se hizo uso de un script hecho en python el cual permite graficar un archivo rrd dado, el script usado es el siguiente.

Graficador.py:

```

1 from pysnmp.hlapi import *
2 import rrdtool
3 import time
4 from convertidor import convertir_a_numero
5 from notificador import Notificador
6 from logger import Logger
7
8 archivo_rrd="predict.rrd"
9 ultimo=rrdtool.last(archivo_rrd)
10 inicio=ultimo-86400
11 ayerInicio=(inicio-40000)
12 ayerFinal=ultimo-40000
13 ret = rrdtool.graphv( "predict.png",
14                      "—start", str(inicio),
15                      "—end", str(ultimo),
16                      "—vertical-label=Bytes/s",
17                      '—slope-mode',
18                      "DEF:obs="+archivo_rrd+":inoctets:AVERAGE",
19                      "DEF:pred="+archivo_rrd+":inoctets:HWPREDICT",
20                      "DEF:dev="+archivo_rrd+":inoctets:DEVPREDICT",
21                      "DEF:fail="+archivo_rrd+":inoctets:FAILURES",
22                      "DEF:yvalue="+archivo_rrd+":inoctets:AVERAGE:start=" + str(
ayrInicio) + ":end=" + str(ayerFinal),
23                      'SHIFT:yvalue:40000',
24                      #"RRA:DEVSEASONAL:1d:0.1:2",
25                      #"RRA:DEVPREDICT:5d:5",
26                      #"RRA:FAILURES:1d:7:9:5"

```

```

27         "CDEF: scaledobs=obs,8,*" ,
28         "CDEF: upper=pred,dev,2,*,+" ,
29         "CDEF: lower=pred,dev,2,*,-" ,
30         "CDEF: scaledupper=upper,8,*" ,
31         "CDEF: scaledlower=lower,8,*" ,
32         "CDEF: scaledh=yvalue,8,*" ,
33         "CDEF: scaledpred=pred,8,*" ,
34         "VDEF: lastfail=fail, LAST" ,
35     "PRINT: lastfail:%6.2lf %S" ,
36     "PRINT: lastfail: %e :strftime" ,
37         # "VDEF: lastobs=obs, LAST" ,
38         # "VDEF: lastmax=upper, LAST" ,
39         # "VDEF: lastmin=lower, LAST" ,
40     # "PRINT: lastmax:%6.2lf %S" ,
41     # "PRINT: lastmin:%6.2lf %S" ,
42     # "PRINT: lastobs:%6.2lf %S" ,
43     "AREA: scaledh#C9C9C9: Ayer" ,
44     "TICK: fail#FDD017:1.0: FFallas" ,
45     "LINE1: scaledobs#00FF00: In traffic" ,
46     "LINE1: scaledpred#FF00FF: Prediccion\\n" ,
47     #"LINE1: outoctets#0000FF: Out traffic" ,
48     "LINE1: scaledupper#ff0000: Upper Bound Average bits in\\n" ,
49     "LINE1: scaledlower#0000FF: Lower Bound Average bits in")

```

Además se hizo uso del siguiente script que permite actualizar un archivo rrd con información nueva.

P2Ejercicio5.py:

```

1 from pysnmp.hlapi import *
2 import rrdtool
3 import time
4 from convertidor import convertir_a_numero
5 from notificador import Notificador
6 from logger import Logger
7 #Funcion para hacer consultas a un solo objeto con referencias al nombre
8 def consultav2SNMP(comunidad, host, version, interfaz, grupo, objeto, puerto):
9     resultado=""
10    errorIndication, errorStatus, errorIndex, varBinds = next(getCmd(SnmpEngine(),
11        CommunityData(comunidad, mpModel=version), UdpTransportTarget((host, puerto)),
12        ContextData(),
13        ObjectType(ObjectIdentity(grupo, objeto, interfaz).addAsn1MibSource('file:///usr/share/snmp', 'http://mibs.snmplabs.com/asn1/@mib@'))))
14    if errorIndication:
15        print(errorIndication)
16    elif errorStatus:
17        print('%s at %s' % (errorStatus.prettyPrint(),
18            errorIndex and varBinds[int(errorIndex) - 1][0] or '?'))
19    else:
20        for varBind in varBinds:
21            VarB=(' = '.join([x.prettyPrint() for x in varBind]))
22            resultado=VarB.partition(' = ')[2]
23    return resultado
24
25 def consultaSNMP(comunidad, host, version, interfaz, grupo, objeto1, objeto2, puerto):
26     resultado=[]
27     errorIndication, errorStatus, errorIndex, varBinds = next(getCmd(SnmpEngine(),
28         CommunityData(comunidad, mpModel=version), UdpTransportTarget((host, puerto)),
29         ContextData(),
30         ObjectType(ObjectIdentity(grupo, objeto1, interfaz).addAsn1MibSource('file:///

```

```

usr/share/snmp', 'http://mibs.snmplabs.com/asn1/@mib@')),
28     ObjectType( ObjectIdentity(grupo, objeto2, interfaz).addAsn1MibSource('file:///
usr/share/snmp', 'http://mibs.snmplabs.com/asn1/@mib@'))))

29
30 if errorIndication:
31     print(errorIndication)
32     resultado.append("")
33     resultado.append("")
34 elif errorStatus:
35     print('%s at %s' % (errorStatus.prettyPrint(),
36                         errorIndex and varBinds[int(errorIndex) - 1][0] or '?')
37 ))
38 else:
39     for varBind in varBinds:
40         VarB=(' = '.join([x.prettyPrint() for x in varBind]))
41         resultado.append(VarB.partition(' = ')[2])
42 return resultado
43
44 #Funcion para crear el archivo .rrd con un valor
45 def crearRRDUno(nombre, inicio, step, tiempo, steps1, steps2, row1, row2):
46     ret=rrdtool.create(nombre, '—start', inicio, '—step', step,
47                        "DS: valor1:COUNTER:" + tiempo + ":U:U",
48                        "RRA:AVERAGE:0.5:" + steps1 + ":" + row1,
49                        "RRA:AVERAGE:0.5:" + steps2 + ":" + row2)
50     if ret:
51         print(rrdtool.error())
52
53 #Funcion para crear el archivo .rrd con dos valores
54 def crearRRDDos(nombre, inicio, step, tiempo, steps1, steps2, row1, row2):
55     ret=rrdtool.create(nombre, '—start', inicio, '—step', step,
56                        "DS: valor1:COUNTER:" + tiempo + ":U:U",
57                        "DS: valor2:COUNTER:" + tiempo + ":U:U",
58                        "RRA:AVERAGE:0.5:" + steps1 + ":" + row1,
59                        "RRA:AVERAGE:0.5:" + steps2 + ":" + row2)
60     if ret:
61         print(rrdtool.error())
62
63 def crearRRDHW(nombre, inicio, step, tiempo, steps1, row1):
64     ret=rrdtool.create(nombre, '—start', inicio, '—step', step,
65                        "DS: inoctets:COUNTER:" + tiempo + ":U:U",
66                        "RRA:AVERAGE:0.5:" + steps1 + ":" + row1,
67                        "RRA:HWpredict:300:0.1:0.0035:10:3",
68                        "RRA:SEASONAL:10:0.1:2",
69                        "RRA:DEVSEASONAL:10:0.1:2",
70                        "RRA:DEVPredict:300:4",
71                        "RRA:FAILURES:300:3:5:4")
72     if ret:
73         print(rrdtool.error())
74
75 #crearRRDHW("prueba.rrd", "N", "1", "600", "1", "600")
76 final=False
77 fecha_inicio=fecha_final=""
78 noti=Notificador("Compu Profa")
79 log=Logger("Compu Profa")
80 archivo_rrd="predict.rrd"
81 while 1:
82     # consulta=consultav2SNMP("public", "localhost", 1, 1, 'IF-MIB', 'ifInOctets', 161)
83     consulta=consultav2SNMP("variation/virtualtable", "10.100.71.200", 1, 1, 'IF-MIB',
84                             'ifInOctets', 1024)
85     valor = "N:" + str(consulta)
86     # print(valor)

```

```

85 ultimo=rrdtool.last(archivo_rdd)
86 inicio=ultimo-3600
87 ayerInicio=(inicio-86400)
88 ayerFinal=ultimo-86400
89 primero=rrdtool.first(archivo_rdd)
90 rrdtool.update(archivo_rdd, valor)
91 rrdtool.dump(archivo_rdd, 'prueba.xml')
92 ret = rrdtool.graphv( "prueba.png",
93     "--start",str(inicio),
94     "--end",str(ultimo),
95     "--vertical-label=Bytes/s",
96     "--slope-mode",
97     "DEF:obs="+archivo_rdd+":inoctets:AVERAGE",
98     "DEF:pred="+archivo_rdd+":inoctets:HWPREDICT",
99     "DEF:dev="+archivo_rdd+":inoctets:DEVPREDICT",
100     "DEF:fail="+archivo_rdd+":inoctets:FAILURES",
101     "DEF:yvalue="+archivo_rdd+":inoctets:AVERAGE:start="+str(
    ayerInicio)+":end="+str(ayerFinal),
102     "SHIFT:yvalue:86400",
103     "#RRA:DEVSEASONAL:1d:0.1:2",
104     "#RRA:DEVPREDICT:5d:5",
105     "#RRA:FAILURES:1d:7:9:5",
106     "CDEF:scaledobs=obs,8,*",
107     "CDEF:upper=pred,dev,2,*,+",
108     "CDEF:lower=pred,dev,2,*,-",
109     "CDEF:scaledupper=upper,8,*",
110     "CDEF:scaledlower=lower,8,*",
111     "CDEF:scaledh=yvalue,8,*",
112     "CDEF:scaledpred=pred,8,*",
113     "VDEF:lastfail=fail,LAST",
114     "PRINT:lastfail:%6.2lf %S",
115     "PRINT:lastfail: %c :strftime",
116     # "VDEF:lastobs=obs,LAST",
117     # "VDEF:lastmax=upper,LAST",
118     # "VDEF:lastmin=lower,LAST",
119     # "PRINT:lastmax:%6.2lf %S",
120     # "PRINT:lastmin:%6.2lf %S",
121     # "PRINT:lastobs:%6.2lf %S",
122     "AREA:scaledh#C9C9C9:Ayer",
123     "TICK:fail#FDD017:1.0:Fallas",
124     "LINE1:scaledobs#00FF00:In traffic",
125     "LINE1:scaledpred#FF00FF:Prediccion\\n",
126     # "LINE1:outoctets#0000FF:Out traffic",
127     "LINE1:scaledupper#ff0000:Upper Bound Average bits in\\n",
128     "LINE1:scaledlower#0000FF:Lower Bound Average bits in")
129 # print("Máximo: "+ret["print[0]"])
130 # print("Mínimo: "+ret["print[1]"])
131 # print("Medido: "+ret["print[2]"])
132 # print("Fecha: "+ret["print[3]"])
133 # print(float(ret["print[0]"]))
134 if "nan" not in ret["print[0]"]:
135     ultima_falla=float(ret["print[0]"])
136     if ultima_falla==0:
137         # print("No soy una falla")
138         if final:
139             final=not final
140             # print("Fecha inicio: "+fecha_inicio)
141             # print("Fecha final: "+fecha_final)
142             noti.enviarCorreo(4, fecha_final, "prueba.png")
143             log.escribirLog(4, fecha_final)
144     else:

```

```

145     # print("Soy una falla")
146     fecha_final=ret["print[1]"]
147     if not final:
148         # print("Falla detectada")
149         fecha_inicio=ret["print[1]"]
150         final=not final
151         noti.enviarCorreo(3, fecha_inicio, "prueba.png")
152         log.escribirLog(3, fecha_inicio)
153
154 if ret:
155     print (rrdtool.error())
156     time.sleep(300)

```

Y por último, se realizaron algunas modificaciones a los programas de notificación y logger que se mostraron en la sección anterior, el código se puede ver a continuación. `notificador.py`:

```

1 import smtplib
2 from email.mime.image import MIMEImage
3 from email.mime.text import MIMEText
4 from email.mime.multipart import MIMEMultipart
5 import datetime
6
7 class Notificador():
8     def __init__(self, agente):
9         self.remitente = "admiredes3.4CM1@gmail.com"
10        self.destinatario = "tanibet.escom@gmail.com"
11        self.servidor = 'smtp.gmail.com: 587'
12        self.contra = '#Equipo5'
13        self.agente=agente
14    def enviarCorreo(self, tipo, fecha, imagen=""):
15        asunto, texto=self.obtenerContenido(tipo)
16        # hora=self.obtenerHora()
17        texto="El agente: "+self.agente+" "+texto+fecha
18        #Definimos contenido del correo
19        mensaje = MIMEMultipart()
20        mensaje['Subject'] = asunto
21        mensaje['From'] = self.remitente
22        mensaje['To'] = self.destinatario
23        mensaje.attach(MIMEText(texto, "plain"))
24        #Verificamos si hay que enviar una imagen
25        if imagen!="":
26            with open(imagen, 'rb') as f:
27                img = MIMEImage(f.read())
28                f.close()
29                mensaje.attach(img)
30        #Enviamos el correo
31        servidor_correo = smtplib.SMTP(self.servidor)
32        servidor_correo.starttls()
33        servidor_correo.login(self.remitente, self.contra)
34        servidor_correo.sendmail(self.remitente, self.destinatario, mensaje.as_string())
35        servidor_correo.quit()
36    def obtenerContenido(self, tipo):
37        with open("mensajes.txt") as f:
38            for i, linea in enumerate(f):
39                if i==tipo:
40                    return linea.split(";")
41            else:
42                return ["", ""]
43    def obtenerHora(self):

```

```
44 return str(datetime.datetime.now())
```

logger.py:

```
1 import datetime
2 class Logger():
3     def __init__(self, agente):
4         self.agente=agente
5     def escribirLog(self, tipo, fecha):
6         contenido=self.obtenerContenido(tipo)
7         # hora=self.obtenerHora()
8         with open('log.txt', 'a') as f:
9             f.write(self.agente+contenido.replace("\n", "")+fecha+"\n")
10            f.flush()
11            print("Problema detectado, revisar archivo de log o correo electrónico")
12    def obtenerContenido(self, tipo):
13        with open("mensajes.txt") as f:
14            for i, linea in enumerate(f):
15                if i==tipo:
16                    return linea.split(";")[1]
17            else:
18                return ""
19    def obtenerHora(self):
20        return str(datetime.datetime.now())
```

Capítulo 5

Conclusiones

Hernández Pineda Miguel Angel:

En el desarrollo de esta practica se trabajaron con dos algoritmos que nos permiten obtener la información cuando se presenta una falla, esto se puede traducir como el momento en que el comportamiento del objeto que se está monitorizando presenta un comportamiento anómalo, sin embargo la detección de dicho comportamiento varía de acuerdo con la lectura de los datos, pues si los datos recibidos tienen un comportamiento lineal se utilizan algunos algoritmos basados en predicción de datos mientras que los datos que tienen un comportamiento no lineal se utilizan algoritmos de muestreo y detección de errores. En esta ocasión los algoritmos utilizados fueron el de Mínimos cuadrados para el comportamiento lineal y Holt-Winters para el comportamiento no lineal. En cuestiones de implementación el algoritmo de Holt-Winters es más complicado debido a todos los parametros necesarios que deben definirse sobre la marcha para que el muestreo de la información sea satisfactorio y no se identifiquen errores donde no los hay por lo que, basados en su complejidad en la implementación, podemos decir que es bastante robusto y efectivo con los parámetros correctos.

Monroy Martos Elioth:

Considero que el desarrollo de esta práctica se fortalecieron distintas habilidades y hubo que hacer uso de varios conocimientos, sin embargo lo más importante desde mi perspectiva, es la importancia que hay detrás de un sistema de detección de fallas. Temas como mínimos cuadrados, métodos de linea base, método de Holt-Winters, e incluso una estrategia para la propuesta de umbrales para la detección de fallas, componen en conjunto una de las partes con mayor importancia funcional en nuestra herramienta. Además de que se desarrollaron varios scripts que resultan ser útiles para todo tipo de situaciones como el detector de umbrales programado, el logger, el notificador o el graficador. Los cuales pueden ser scripts que nos faciliten la vida en un futuro.

Zuñiga Hernández Carlos:

La práctica tuvo como principal objetivo la predicción del rendimiento y de fallas de los recursos disponibles en una computadora, así como la detección de fallas y

el monitoreo del rendimiento en tiempo real. La predicción se implementó por medio de los algoritmos Mínimos cuadrados y Holt-Winters, cada uno con diferentes características; aunque Holt-winters demostró ser más preciso y eficiente, debido a que hacía uso de datos históricos para hacer la predicción. Es claro que es necesario monitorizar la actividad en los dispositivos que conforman una red, para que cuando se presente una falla, se pueda actuar a tiempo para resolverla o reducir el impacto y que todo el proceso que conlleva esta detección es complejo y debe hacerse de manera detallada.

Referencias Bibliográficas

- [1] SNMP library for Python. Marzo 20, 2019, de - Sitio web: <http://snmplabs.com/pysnmp/index.html>
- [2] PySNMP architecture. Marzo 20, 2019, de - Sitio web: <http://snmplabs.com/pysnmp/docs/pysnmp-architecture.html>
- [3] Library reference. Marzo 20, 2019, de - Sitio web: <http://snmplabs.com/pysnmp/docs/api-reference.html>
- [4] K. McCloghrie. (March 1991). Management Information Base for Network Management of TCP/IP-based internets: MIB-II. Marzo 20, 2019, de IETF Sitio web: <https://www.ietf.org/rfc/rfc1213.txt>
- [5] Maddox, Segán. *FCAPS ISO Model*. Management. Disponible en: <https://sites.google.com/site/smaddoxcloud/management/fcaps>
- [6] Cisco Systems. *White Paper de las mejores prácticas del proceso de línea de base*. Cisco Systems.
- [7] Pérez, Tanibet. *Método de Mínimos Cuadrados*.
- [8] Maguiña, Omar. *El Método de Pronóstico Holt-Winters*. Disponible en: <https://administration21.files.wordpress.com/2017/01/pronc3b3sticos-holt-winters-omr-nov2016.pdf>