Trabalho 1 da disciplina de Gerência de Redes - Gerente de contabilidade

Enrico D. Pizzol[1], **Cássio Entrudo**[1], **Giovani D. Silva**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)

{enrico.pizzol,giovani.silva,cmentrudo}@inf.ufrgs.br

Resumo. *O trabalho proposto teve como objetivo desenvolver uma ferramenta gráfica capaz de realizar três tarefas de gerenciamento de contabilização. A biblioteca SNMPv3 foi utilizada para a elaboração da tarefa. A biblioteca PySimpleGui foi utilizada para a elaboração da interface gráfica devido à sua facilidade de uso.*

1. Objetos presentes na MiBs

Na área da contabilidade, as MIBs podem prover dados relevantes acerca da utilização de recursos de rede e do fluxo de tráfego. Alguns dos elementos disponíveis nas MIBs de gestão contábil são:

Interfaces: possibilitam a supervisão do tráfego em cada interface de rede, incluindo a quantidade de dados transmitidos e recebidos, a taxa de transferência, e outras informações pertinentes.

TCP e UDP: permitem a supervisão de conexões TCP e UDP, incluindo dados como o número de pacotes enviados e recebidos, as portas de origem e destino, e outras informações.

Roteamento: possibilita a supervisão da tabela de roteamento, incluindo informações acerca das rotas existentes, da métrica das rotas, e do próximo salto.

Conexões: possibilita a supervisão das conexões ativas na rede, incluindo informações como o endereço IP e as portas de origem e destino.

QoS (Qualidade de Serviço): possibilita a supervisão e gestão da qualidade de serviço na rede, incluindo informações acerca da largura de banda disponível e da priorização do tráfego.

Esses são apenas alguns exemplos de elementos disponíveis nas MIBs de gestão contábil do SNMP. As informações coletadas por esses elementos são utilizadas para a supervisão e gestão dos recursos de rede, permitindo a identificação de problemas e a otimização do desempenho da rede.

2. Objetos escolhidos

Uma MIBS (Management Information Base for SNMP - Base de Informações de Gerenciamento para o Simple Network Management Protocol) de gerenciamento de contabilidade pode incluir vários objetos que são usados para monitorar e gerenciar informações contábeis em uma rede. Alguns dos objetos comuns incluem:

Contadores de tráfego de rede: Esses objetos coletam informações sobre o número de pacotes enviados e recebidos em uma rede. Utilização da largura de banda: Esses objetos monitoram a quantidade de largura de banda utilizada em uma rede. Taxas de erro: Esses objetos coletam informações sobre a taxa de erro em uma rede. Contadores de tempo de atividade: Esses objetos monitoram o tempo de atividade de dispositivos na rede, como switches e roteadores. Contadores de erros de hardware: Esses objetos coletam informações sobre erros de hardware em dispositivos na rede, como falhas de placas ou unidades de disco. Utilização do processador: Esses objetos monitoram o uso de CPU em servidores e outros dispositivos na rede.

3. Implementação do código

3.1. Informações iiciais sobre o código

O código foi desenvolvido em Python, utilizando-se das bibliotecas PySimpleGUI(para interface gráfica) e EASYSnmp, para o SNMNP. Junto ao código está um arquivo requirements.txt, que pode ser utilizado para instalar as bibliotecas necessárias na maquina, com os comandos a seguir:

```
pip install --upgrade pip
pip install -r requirements.txt
```

A aplicação desenvolidada é extramamente simples, contendo apenas um **launcher** e uma classe **App** para quem delegamos a criação da interface. A classe **App** também é responsável por escutar os eventos e atualizar os valores na tela.

3.2. Uptime

Fornece a quantidade de tempo que o dispositivo está em execução desde o último reinício. Essas informações podem ser usadas para calcular métricas de disponibilidade e confiabilidade do dispositivo.

```
        uptime = session.get('sysUpTime.0')
        uptime_ticks = uptime.value
        uptime_secs = int(uptime_ticks) // 100
        self.window['-UPTIME-'].update(uptime_secs)
```

3.3. CPU-USAGE

The subsection titles must be in boldface, 12pt, flush left.

```
        cpu_usage = int(session.get('1.3.6.1.4.1.2021.11.11.0').value)
        cpu_usage_percent = 100 - cpu_usage
        #cpu_usage_percent = random.random()
        self.window['-PROCESSOR-'].update(cpu_usage_percent)
```

3.4. Número de interfaces

.

```
        oid = '1.3.6.1.2.1.2.1.0'
        value = int(session.get(oid).value)
        self.window['-INTERFACES-'].update(value)
```

3.5. Uso de Memória

Fornece a porcentagem atual de uso da CPU e memória do dispositivo. monitorar o uso da CPU e memória é importante para um gerente de contabilidade devido a alocação de recursos, pois ao monitorar o uso da CPU e da memória, um gerente de contabilidade pode identificar quais aplicativos ou serviços estão consumindo mais recursos e alocar esses recursos de acordo. Por exemplo, se um determinado aplicativo estiver consumindo muita CPU, um gerente de contabilidade pode priorizar os recursos da CPU para garantir que outros aplicativos críticos continuem funcionando corretamente. Sobre o uso da memória, essas informações podem ser usadas para detectar possíveis vazamentos de memória ou problemas de contenção de recursos, garantindo que os recursos do sistema estejam sendo utilizados de forma eficiente.

```
    def memory(self, session):
        total_memory_oid = 'memTotalReal.0'
        used_memory_oid = 'memAvailReal.0'

        total_memory = int(session.get(total_memory_oid).value)
        used_memory = int(session.get(used_memory_oid).value)
        free_memory = total_memory - used_memory
        used_memory_percent = (used_memory / total_memory) * 100

        print(f'Total memory: {total_memory} bytes')
        print(f'Used memory: {used_memory} bytes ({used_memory_percent}%
        print(f'Free memory: {free_memory} bytes')

        return
        total_memory,used_memory,used_memory_percent,free_memory
    total_memory,used_memory,used_memory_percent,free_memory = self.mem
    self.window['-MEMORYTOTAL-'].update(total_memory)
    self.window['-MEMORYUSE-'].update(used_memory)
    self.window['-MEMORYFREE-'].update(free_memory)
    self.window['-MEMORYFREEPERCENT-'].update(used_memory_percent)
```

3.6. Disk

Fornece a porcentagem atual de uso do disco do dispositivo,monitorar o uso de disco é importante para um gerente de contabilidade devido a alocação de ao planejamento de capacidade que é necessário. O monitoramento do uso do disco pode ajudar a identificar quando um disco está se aproximando da capacidade, permitindo que um gerente de contabilidade planeje recursos de armazenamento adicionais antes que eles sejam necessários. Isso pode ajudar a evitar perda de dados ou travamentos do sistema devido a espaço em disco insuficiente.

```
    def disk_space(self,session):
        try:
            #disk_space_oid = 'hrStorageUsed.31'
            #disk_space_oid = '1.3.6.1.4.1.2021.9.1.7.1'
```

```
            disk_space_oid = 'UCD-SNMP-MIB::dskAvail.1'
            disk_space_used = int(session.get(disk_space_oid).value)

            if disk_space_used > 0:
                #disk_space_total_oid = 'hrStorageSize.31'
                disk_space_total_oid = 'UCD-SNMP-MIB::dskAvail.1'
                disk_space_total = int(session.get(disk_space_total_oid
                disk_space_free = disk_space_total - disk_space_used
                disk_space_used_percent = (disk_space_used / disk_space
                print(f'Total disk space: {disk_space_total} bytes')
                print(f'Used disk space: {disk_space_used} bytes ({disk
                print(f'Free disk space: {disk_space_free} bytes')
                return disk_space_total,disk_space_used,disk_space_used
            else:
                sg.popup("N o foi poss vel obter o espa o em disco")
        except ValueError:
            return ["Nao foi possivel ler o disco", "Nao foi possivel l
    disk_space_total,disk_space_used,disk_space_used_percent,disk_space
    self.window['-DISKTOTAL-'].update(disk_space_total)
    self.window['-DISKUSE-'].update(disk_space_used)
    self.window['-DISKFREE-'].update(disk_space_free)
    self.window['-DISKFREEPERCENT-'].update(disk_space_used_percent)
```

3.7. Erros de entrada e saida

Fornece o número de erros de entrada e saída presentes naquela conexão.

```
        ifInErrors = session.get('IF-MIB::ifInErrors.2').value
        self.window['-INERRORS-'].update(ifInErrors)

        ifOutErrors = session.get('IF-MIB::ifOutErrors.2').value
        self.window['-OUTERRORS-'].update(ifOutErrors)
```

3.8. Temperatura

Fornece a temperatura atual do dispositivo, que pode ser usada para monitorar equipamentos sensíveis à temperatura e detectar possíveis problemas de superaquecimento. Ao monitorar métricas ambientais, como temperatura, um gerente de contabilidade pode detectar possíveis falhas de hardware antes que ocorram, permitindo manutenção proativa para evitar tempo de inatividade e perda de dados.

```
            def temperature(self,session):
        #temperature_oid = 'tempSensorValue.1'
        #temperature_oid = '1.3.6.1.4.1.318.1.1.10.2.3.2.1.4'
        temperature_oid = '1.3.6.1.4.1.9148.3.3.1.3.1.1'

        try:
            temperature_value = int(session.get(temperature_oid).value)
```

```
            if temperature_value > 0:
                temperature_celsius = temperature_value / 10
                return temperature_celsius
            else:
                sg.popup("N o foi poss vel obter a temperatura")

        except ValueError:
            return "Conecte um Sensor de Temperatura"

    temperature = self.temperature(session)
    self.window['-TEMPERATURE-'].update(temperature)
```

3.9. Pacotes enviados e recebidos

The subsection titles must be in boldface, 12pt, flush left.

```
        packets_sent,packets_received = self.packets(session)
        self.window['-PACKETSSENT-'].update(packets_sent)
        self.window['-PACKETSRECEIVED-'].update(packets_received)
```

3.10. Contador de conexões TCP

Fornece o número de conexões TCP atuais que o dispositivo está manipulando. Essas informações podem ser usadas para identificar possíveis congestionamentos de rede e determinar se recursos adicionais são necessários para suportar a rede.Por exemplo, um gerente de contabilidade pode notar um aumento no número de conexões TCP em um firewall. Eles podem usar essa informação para determinar se o firewall está lidando com a carga de tráfego atual e se é necessário adicionar mais recursos ao firewall para garantir a estabilidade da rede.

```
        active_tcp_oid = 'tcpCurrEstab.0'
        active_tcp_connections = int(session.get(active_tcp_oid).value)
        self.window['-TCP-'].update(active_tcp_connections)
```

3.11. Informações de trafego

The subsection titles must be in boldface, 12pt, flush left.

```
    def trafego(self,session):
        input_traffic_oid = 'ifInOctets.1'
        output_traffic_oid = 'ifOutOctets.1'

        input_traffic = int(session.get(input_traffic_oid).value)
        output_traffic = int(session.get(output_traffic_oid).value)
        total_traffic = input_traffic + output_traffic

        return input_traffic,output_traffic,total_traffic
    input_traffic, output_traffic,total_traffic = self.trafego(session)
```

```
    self.window['-INTRAFFIC-'].update(input_traffic)
    self.window['-OUTTRAFFIC-'].update(output_traffic)
    self.window['-TOTALTRAFFIC-'].update(total_traffic)
```

3.12. Taxa de Transferência

Serve para a otimização de desempenho, ao monitorar a taxa de transferência da rede e o uso da largura de banda pode ajudar a identificar problemas de desempenho da rede, permitindo que um gerente de contabilidade otimize as configurações de rede e garanta que a rede esteja funcionando em seu pico. A largura de banda fornece a capacidade máxima de largura de banda do dispositivo. Essas informações podem ser usadas para identificar se a rede está sendo utilizada em seu pleno potencial e determinar se é necessário mais largura de banda.

```
     def transfer_rate(self,session,intervalo):
        import time
        initial_time = time.time()

        interface_oid = 'IF-MIB::ifInOctets.1'
        traffic_start = session.get(interface_oid).value

        time.sleep(intervalo)


        traffic_end = session.get(interface_oid).value


        traffic = float(traffic_end) - float(traffic_start)


        transfer_rate = traffic / (time.time() - initial_time)
        return transfer_rate
    transferrate = self.transfer_rate(session,intervalo = 2)
    self.window['-TRAFFIC-'].update(transferrate)
```

3.13. Código completo

```
 import PySimpleGUI as sg
from easysnmp import Session
import random
from time import time

sg.theme('DarkTeal')


class App:

    def __init__(self):
```

```
        self.window = self.create_screen()

    def create_screen(self):
        window = sg.Window('An lise de M tricas', self.get_layout(),
                           button_color='purple', margins=(10, 10))
        #window = sg.Window("My Window", self.get_layout(), background_
        return window

    def get_layout(self):
        return [

            [sg.Text('Tempo de atividade (s) = ', font=('Helvetica', 15)
             sg.Text(size=(40,1), key='-UPTIME-')],
            [sg.Text('Utiliza  o do processador em % = ', font=('Helv
             sg.Text(size=(40,1), key='-PROCESSOR-')],

            #Numero de interfaces
            [sg.Text('N mero de interfaces = ', font=('Helvetica', 15)
                sg.Text(size=(40,1), key='-INTERFACES-')],


            #Memoria
            [sg.Text('Total de mem ria (Mb)= ', font=('Helvetica', 15)
             sg.Text(size=(40,1), key='-MEMORYTOTAL-')],
            [sg.Text('Uso de mem ria(Mb) = ', font=('Helvetica', 15),
             sg.Text(size=(40,1), key='-MEMORYUSE-')],
            [sg.Text('Memoria livre(Mb) = ', font=('Helvetica', 15), te
             sg.Text(size=(40,1), key='-MEMORYFREE-')],
            [sg.Text('Memoria livre em % =
', font=('Helvetica', 15), text_color='white'),
             sg.Text(size=(40,1), key='-MEMORYFREEPERCENT-')],

            #Disco
            [sg.Text('Espa o em disco total (Mb ) = ', font=('Helvetic
             sg.Text(size=(40,1), key='-DISKTOTAL-')],
            [sg.Text('Espa o em disco em uso(Mb) = ', font=('Helvetica
             sg.Text(size=(40,1), key='-DISKUSE-')],
            [sg.Text('Espa o em disco livre (Mb)=
', font=('Helvetica', 15), text_color='white'),
             sg.Text(size=(40,1), key='-DISKFREE-')],
            [sg.Text('Espa o em disco livre em %= ', font=('Helvetica'
             sg.Text(size=(40,1), key='-DISKFREEPERCENT-')],

            #Erros de entrada e saida
            [sg.Text('Contadores de erros de entrada = ', font=('Helvet
```

```
            sg.Text(size=(40,1), key='-INERRORS-')],
           [sg.Text('Contadores de erros de sa da = ', font=('Helveti
            sg.Text(size=(40,1), key='-OUTERRORS-')],

           #Temperatura
           [sg.Text('Temperatura (Celsius) = ', font=('Helvetica', 15)
            sg.Text(size=(40,1), key='-TEMPERATURE-')],

           #Pacotes enviados e recebidos
           [sg.Text('Pacotes enviados = ', font=('Helvetica', 15), tex
            sg.Text(size=(40,1), key='-PACKETSSENT-')],
           [sg.Text('Pacotes recebidos = ', font=('Helvetica', 15), te
            sg.Text(size=(40,1), key='-PACKETSRECEIVED-')],



           #TCP
           [sg.Text('Contador de conex es TCP = ', font=('Helvetica',
            sg.Text(size=(40,1), key='-TCP-')],

           #Informacoes de trafego entrada,saida e total
           [sg.Text('Tr fego de entrada (bytes) = ', font=('Helvetica
            sg.Text(size=(40,1), key='-INTRAFFIC-')],
           [sg.Text('Tr fego de sa da (bytes) = ', font=('Helvetica'
            sg.Text(size=(40,1), key='-OUTTRAFFIC-')],
           [sg.Text('Tr fego total (bytes) = ', font=('Helvetica', 15
            sg.Text(size=(40,1), key='-TOTALTRAFFIC-')],



           [sg.Text('Utiliza  o da largura de banda (Mb/s) = ', font
            sg.Text(size=(40,1), key='-BANDWIDTH-')],
           [sg.Text('Taxa de transfer ncia de rede (bytes/s) = ', fon
            sg.Text(size=(40,1), key='-TRAFFIC-')],
           ]

    def run(self):
        endereco_ip = 0
        ip = sg.popup_get_text('Digite o endere o IP do agente')
        while True:
            event, values = self.window.read(timeout=0)
            endereco_ip = ip
            print(f'Endere o IP digitado: {endereco_ip}')

            self.update(endereco_ip)
            import time
```

```
            time.sleep(1)
            if event in (sg.WIN_CLOSED, 'Exit'):
                break



                #while event not in (sg.WIN_CLOSED, 'Exit'):
                #    self.update(endereco_ip)

            #self.update(endereco_ip)

    def update(self,endereco_ip):

        session = Session(hostname=endereco_ip, community='public', ver

        uptime = session.get('sysUpTime.0')
        uptime_ticks = uptime.value
        uptime_secs = int(uptime_ticks) // 100

        #uptime_secs = random.random()
        self.window['-UPTIME-'].update(uptime_secs)

        #cpu_usage = int(session.get('ssCpuRawIdle.0').value)
        cpu_usage = int(session.get('1.3.6.1.4.1.2021.11.11.0').value)
        cpu_usage_percent = 100 - cpu_usage
        #cpu_usage_percent = random.random()
        self.window['-PROCESSOR-'].update(cpu_usage_percent)

        #numero de interfaces
        oid = '1.3.6.1.2.1.2.1.0'
        value = int(session.get(oid).value)
        #value = random.random()
        self.window['-INTERFACES-'].update(value)


        #total_memory,used_memory,used_memory_percent,free_memory = sel

        #####MEMORIA
        total_memory,used_memory,used_memory_percent,free_memory = self
        #total_memory,used_memory,used_memory_percent,free_memory = [ra
        self.window['-MEMORYTOTAL-'].update(total_memory)
        self.window['-MEMORYUSE-'].update(used_memory)
        self.window['-MEMORYFREE-'].update(free_memory)
        self.window['-MEMORYFREEPERCENT-'].update(used_memory_percent)
```

```
disk_space_total,disk_space_used,disk_space_used_percent,disk_s
#disk_space_total, disk_space_used, disk_space_used_percent, disk_

self.window['-DISKTOTAL-'].update(disk_space_total)
self.window['-DISKUSE-'].update(disk_space_used)
self.window['-DISKFREE-'].update(disk_space_free)
self.window['-DISKFREEPERCENT-'].update(disk_space_used_percent

####Erros de entrada e saida
ifInErrors = session.get('IF-MIB::ifInErrors.2').value
#ifInErrors = random.random()
self.window['-INERRORS-'].update(ifInErrors)
ifOutErrors = session.get('IF-MIB::ifOutErrors.2').value
#ifOutErrors = random.random()
self.window['-OUTERRORS-'].update(ifOutErrors)

####Temperatura
temperature = self.temperature(session)
#temperature = random.random()
self.window['-TEMPERATURE-'].update(temperature)

# Pacotes enviados e recebidos
packets_sent,packets_received = self.packets(session)
#packets_sent, packets_received = [random.random(),random.random
self.window['-PACKETSSENT-'].update(packets_sent)
self.window['-PACKETSRECEIVED-'].update(packets_received)

# Contador de conex es TCP
active_tcp_oid = 'tcpCurrEstab.0'
active_tcp_connections = int(session.get(active_tcp_oid).value)
#active_tcp_connections = random.random()
self.window['-TCP-'].update(active_tcp_connections)

# Informa  es de trafego
input_traffic, output_traffic,total_traffic = self.trafego(sess
#input_traffic, output_traffic, total_traffic = [random.random()
self.window['-INTRAFFIC-'].update(input_traffic)
self.window['-OUTTRAFFIC-'].update(output_traffic)
self.window['-TOTALTRAFFIC-'].update(total_traffic)



utilizacao_bps= self.utilizacao_largura_banda(session,intervalo
#utilizacao_bps = random.random()
self.window['-BANDWIDTH-'].update(utilizacao_bps)
```

```
        trafego = self.transfer_rate(session,intervalo = 2)
        #trafego = random.random()
        self.window['-TRAFFIC-'].update(trafego)

        self.verifica_erros(variavel = 'INERRORS',valor = ifInErrors ,l
        self.verifica_erros(variavel = 'OUTERRORS',valor = ifOutErrors
        self.verifica_erros(variavel = 'Utiliza  o da bandwidth',valo
        self.verifica_erros(variavel = 'Uso de memoria',valor = used_me



    def close(self):
        self.window.close()

    def utilizacao_largura_banda(self,session,intervalo = 5):
        interface_index = 1
        in_octets = int(session.get(f'IF-MIB::ifInOctets.{interface_ind
        out_octets = int(session.get(f'IF-MIB::ifOutOctets.{interface_i
        if_speed = int(session.get(f'IF-MIB::ifSpeed.{interface_index}'

        # Aguardar por um intervalo de tempo
        import time
        time.sleep(intervalo)

        # Obter os novos valores de entrada e sa da do contador de lar
        in_octets_new = int(session.get(f'IF-MIB::ifInOctets.{interface
        out_octets_new = int(session.get(f'IF-MIB::ifOutOctets.{interfa

        # Calcular a diferen a entre os valores de entrada e sa da em
        delta_in_octets = in_octets_new - in_octets
        delta_out_octets = out_octets_new - out_octets

        # Calcular a utiliza  o da largura de banda em bits por segun
        intervalo_de_tempo = 5  # segundos
        taxa_de_transferencia_bps = (delta_in_octets + delta_out_octets

        utilizacao_da_largura_de_banda = taxa_de_transferencia_bps / if

        return utilizacao_da_largura_de_banda

    def memory(self, session):
        total_memory_oid = 'memTotalReal.0'
        used_memory_oid = 'memAvailReal.0'

        total_memory = int(session.get(total_memory_oid).value)
```

```
        used_memory = int(session.get(used_memory_oid).value)
        free_memory = total_memory - used_memory
        used_memory_percent = (used_memory / total_memory) * 100

        print(f'Total memory: {total_memory} bytes')
        print(f'Used memory: {used_memory} bytes ({used_memory_percent}
        print(f'Free memory: {free_memory} bytes')

        return total_memory,used_memory,used_memory_percent,free_memory

    def disk_space(self,session):
        try:
            #disk_space_oid = 'hrStorageUsed.31'
            #disk_space_oid = '1.3.6.1.4.1.2021.9.1.7.1'
            disk_space_oid = 'UCD-SNMP-MIB::dskAvail.1'
            disk_space_used = int(session.get(disk_space_oid).value)

            if disk_space_used > 0:
                #disk_space_total_oid = 'hrStorageSize.31'
                disk_space_total_oid = 'UCD-SNMP-MIB::dskAvail.1'
                disk_space_total = int(session.get(disk_space_total_oid
                disk_space_free = disk_space_total - disk_space_used
                disk_space_used_percent = (disk_space_used / disk_space
                print(f'Total disk space: {disk_space_total} bytes')
                print(f'Used disk space: {disk_space_used} bytes ({disk
                print(f'Free disk space: {disk_space_free} bytes')
                return disk_space_total,disk_space_used,disk_space_used
            else:
                sg.popup("N o foi poss vel obter o espa o em disco")
        except ValueError:
            return ["Nao foi possivel ler o disco", "Nao foi possivel l


    def transfer_rate(self,session,intervalo):
        import time
        initial_time = time.time()

        interface_oid = 'IF-MIB::ifInOctets.1'
        traffic_start = session.get(interface_oid).value

        time.sleep(intervalo)


        traffic_end = session.get(interface_oid).value
```

```
        traffic = float(traffic_end) - float(traffic_start)


        transfer_rate = traffic / (time.time() - initial_time)
        return transfer_rate


    def temperature(self,session):
        #temperature_oid = 'tempSensorValue.1'
        #temperature_oid = '1.3.6.1.4.1.318.1.1.10.2.3.2.1.4'
        temperature_oid = '1.3.6.1.4.1.9148.3.3.1.3.1.1'

        try:
            temperature_value = int(session.get(temperature_oid).value)

            if temperature_value > 0:
                temperature_celsius = temperature_value / 10
                return temperature_celsius
            else:
                sg.popup("N o foi poss vel obter a temperatura")

        except ValueError:
            return "Conecte um Sensor de Temperatura"


    def packets(self,session):
        packets_sent_oid = 'ifOutUcastPkts.1'
        packets_sent = int(session.get(packets_sent_oid).value)

        packets_received_oid = 'ifInUcastPkts.1'
        packets_received = int(session.get(packets_received_oid).value)

        return packets_sent,packets_received


    def trafego(self,session):
        input_traffic_oid = 'ifInOctets.1'
        output_traffic_oid = 'ifOutOctets.1'

        input_traffic = int(session.get(input_traffic_oid).value)
        output_traffic = int(session.get(output_traffic_oid).value)
        total_traffic = input_traffic + output_traffic

        return input_traffic,output_traffic,total_traffic

    def verifica_erros(self,variavel,valor,limite):
```

```
        if float(valor) > float(limite):
            message = "A vari vel " + variavel + " est  acima do limi
            button = sg.popup(message, button_type=sg.POPUP_BUTTONS_YES

            if button == "Yes":
                self.window.close()
                exit(8)
            else:
                pass
```

3.14. Execucao do programa

O resultado da execução do programa foi o seguinte:

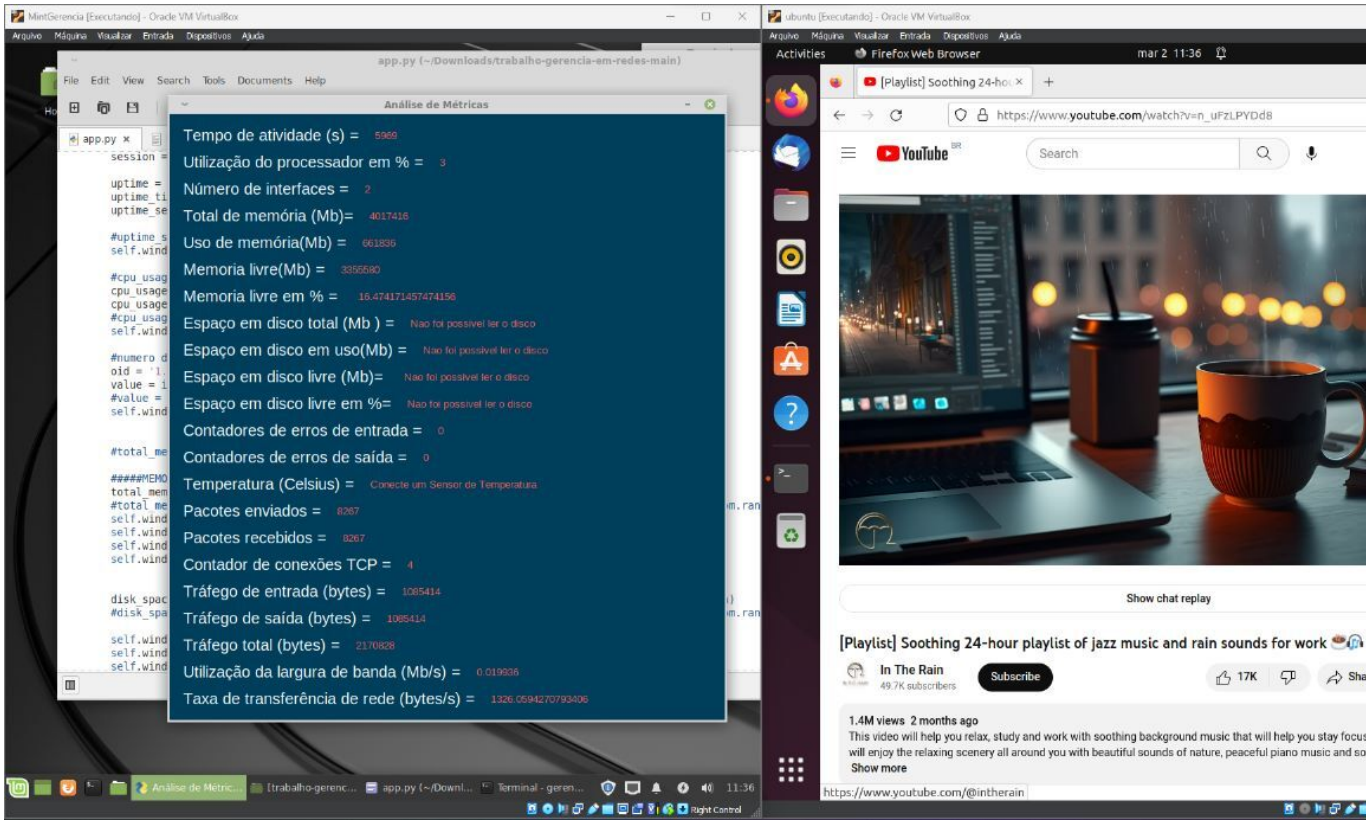

Figura 1. Resultado da execução do código com uma aba do youtube aberta.

4. Metodologia de Análise

A metodologia de analise, além do visual, baseia-se na função de verificação de erros, ela recebe o valor da variável e um parâmetro de limite, caso passe do limite, avisa o gerente e pergunta se quer fechar a aplicação.

```
        def verifica_erros(self,variavel,valor,limite):

        if valor > limite:
            message = "A variavel " + variavel + " esta acima do limite
            button = sg.popup(message, button_type=sg.POPUP_BUTTONS_YES

            if button == "Yes":
                self.window.close()
                exit(8)
            else:
                pass
    self.verifica_erros(variavel = 'INERRORS',valor = ifInErrors ,limit
        self.verifica_erros(variavel = 'OUTERRORS',valor = ifOutErrors
    self.verifica_erros(variavel = 'Utilizacao da bandwidth',valor = ut
    self.verifica_erros(variavel = 'Uso de memoria',valor = used_memory
    self.verifica_erros(variavel = 'Uso de disco',valor = disk_space_us
    self.verifica_erros(variavel = 'Temperatura',valor = temperature ,l
    self.verifica_erros(variavel = 'Uso de cpus',valor = cpu_usage_perc
```

5. Referências

http://www.linux-admins.net/2012/02/linux-snmp-oids-for-cpumemory-and-disk.html https://iphostmonitor.com/mib/tags/temperature-status.html https://easysnmp.readthedocs.io/en/latest/ https://www.pysimplegui.org/