Содержание

1 Постановка задачи

Цель работы

Разработка приложения для визуализации метрик с пользовательских устройств, полученных с помощью протокола SNMP

Задачи

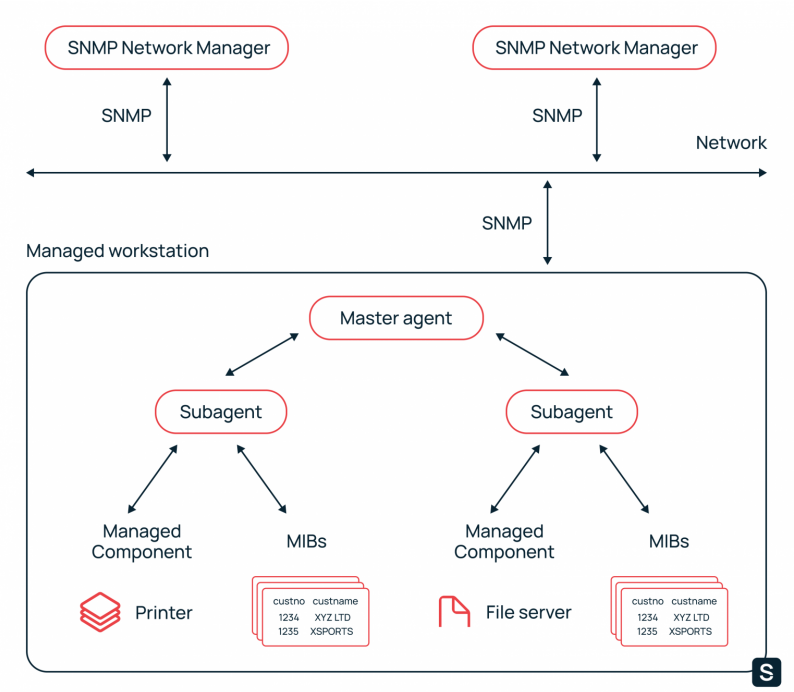
1. Исследовать протокол SNMP.
2. Настроить конфигурацию службы SNMP на целевых устройствах.
3. Определить стэк разработки.
4. Разработка приложения для визуализации метрик с пользовательских устройств, полученных с помощью протокола SNMP.

2 Обзор литературы

SNMP (Simple Network Management Protocol) [**Mauro2005**] представляет собой протокол на прикладном уровне, разработанный Советом по архитектуре Интернета (IETF) для обмена данными между сетевыми устройствами в сетях любого размера. Он позволяет системным администраторам мониторить, контролировать производительность сети и изменять конфигурацию устройств без необходимости ввода команд вручную.

Архитектура SNMP включает в себя несколько компонентов:

1. Сетевая станция управления (NMS): Она представляет собой программное обеспечение, которое мониторит и управляет устройствами в сети. NMS взаимодействует с агентами устройств через протокол SNMP.
2. Агенты: Это программное обеспечение, установленное на управляемых устройствах. Оно собирает информацию о состоянии устройства и передает её NMS.
3. Мастер-агенты: Программы, которые связывают сетевые менеджеры и субагенты, а также анализируют запросы от NMS и управляют передачей данных между ними.
4. Субагенты: Это программное обеспечение, поставляемое вендором вместе с управляемыми устройствами. Они собирают информацию об устройстве и передают её мастер-агенту.
5. Управляемые компоненты: Это сетевые устройства или программное обеспечение с установленными агентами. Они могут быть различными, от маршрутизаторов и коммутаторов до IP-видеокамер и антивирусных программ.



Для организации обмена данными между NMS и агентами используются специальные единицы данных, называемые Protocol Data Unit (PDU)[**Alani2014**]. Существует семь типов PDU, таких как:

1. **GET** – запрос данных
2. **SET** – изменение данных
3. **RESPONSE** – ответ на запрос
4. **TRAP** – уведомление об событии
5. **GETBULK** – запрос агенту на извлечение с устройства массива данных. Это улучшенный вариант запроса GETNEXT.
6. **INFORM** – сообщение, аналогичное TRAP, но с подтверждением получения. Агент будет отправлять уведомление, пока менеджер не подтвердит, что оно дошло.

Изначально SNMP разрабатывался для управления интернетом, однако его гибкая архитектура позволяет мониторить и управлять всеми сетевыми устройствами с помощью единого интерфейса. Наиболее распространенной версией является SNMPv2c.

3 Выбор технологий

Для взаимодействия с SNMP используется приложение, разработанное в рамках другого курсового проекта. Оно предоставляет интерфейс для получения следующих метрик:

1. Входящий и исходящий интернет трафик.
2. Температура процессора.
3. Нагрузка процессора.
4. Количество RAM на устройстве.

Для визуализации метрик с пользовательских устройств, полученных с помощью протокола SNMP, используется Flet [**fletIntroductionFlet**]. Flet - это мощный фреймворк для визуализации данных, разработанный на языке программирования Python. Он позволяет быстро и просто создавать веб/десктоп приложения.

4 Ход Работы

Разработка приложения

Приложение представляет собой веб приложение, которое получает данные о пользовательских устройствах через файл config. Файл config может обновляться пользователем через интерфейс приложения, для добавления новых устройств для мониторинга.

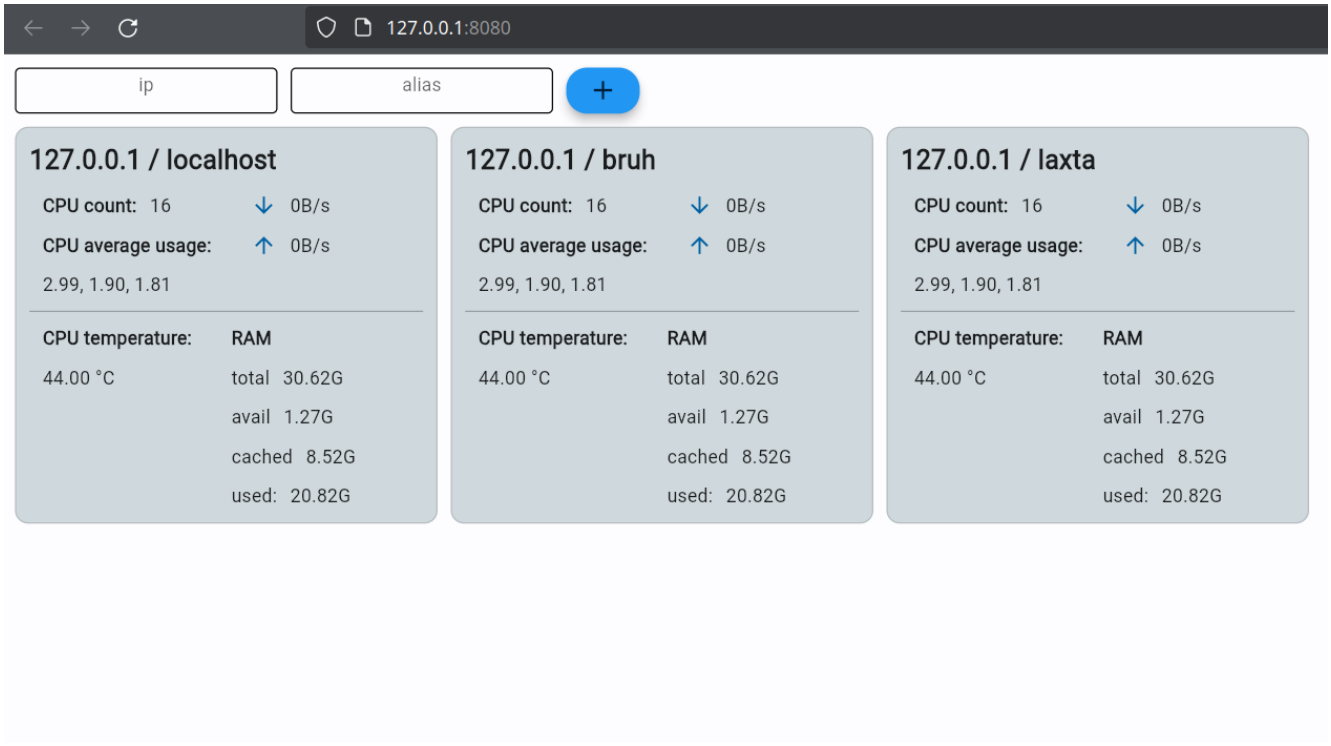


Рис. 1: Внешний вид приложения

```
import re
import flet as ft
from item import MachineInfo
from dialog_window import ErrorAlert
from utils import check_valid_ip

class App(ft.UserControl):
    def __init__(self, path, page):
        super().__init__()
        self.path_to_config = path
        self.machines = self.read_config(self.path_to_config)
        self.page = page

    def build(self):
        self.add_item_button = ft.FloatingActionButton(
```

```
            icon=ft.icons.ADD, height=35, on_click=self.add_item, bgcolor=ft.colors.BLUE
        )
        self.ip_input = ft.TextField(hint_text="ip", width=200, height=35, text_align=ft.TextAlign.CENTER, text_size=14)
        self.alias_input = ft.TextField(hint_text="alias", width=200, height=35, text_align=ft.TextAlign.CENTER, text_size=14)

        self.items = ft.Row(
            wrap=True,
            spacing=10,
            run_spacing=10,
            controls=self.machines,
            width=1600,
        )

        self.new_input = ft.Row(
            width=600,
            spacing=10,
            controls=[self.ip_input, self.alias_input, self.add_item_button],
        )

        return ft.Column(controls=[self.new_input, self.items])

    def add_item(self, e):
        if not check_valid_ip(self.ip_input.value):
            self.open_error_dialog(e, "Invalid IP", "Please enter valid IP adress")
            return

        new_machine = MachineInfo(self.ip_input.value, self.alias_input.value)
        if not new_machine.client.check_connection():
            self.open_error_dialog(e, "Error connection", "Please check your machine connection")
            return

        self.machines.append(new_machine)
        with open(self.path_to_config, 'a') as file:
            file.write(f"\n{new_machine.ip} {new_machine.alias}\n")
        self.update()

    def open_error_dialog(self, e, err_text, main_text):
        self.err_dialog = ft.AlertDialog(
            title=ft.Text(err_text),
            content=ft.Text(main_text),
            actions_alignment=ft.MainAxisAlignment.END
        )
        self.page.dialog = self.err_dialog
```

```
        self.err_dialog.open = True
        self.page.update()

    def read_config(self, file_path):
        machines = []
        with open(file_path, 'r') as file:
            for line in file:
                values = line.strip().split()
                if len(values) == 1:
                    ip = values[0]
                    alias = ""
                elif len(values) == 2:
                    ip = values[0]
                    alias = values[1]
                else:
                    continue
                if not check_valid_ip(ip):
                    continue
                machine = MachineInfo(ip, alias)
                machines.append(machine)
        return machines

def main(page: ft.Page):

    page.add(
        App("config.txt", page)
    )

if __name__ == "__main__":
    ft.app(target=main, view=ft.WEB_BROWSER, port=8080)
```

```
import threading
import time
import flet as ft
from snmp.snmp import SNMPClient
from hurry.filesize import size, si
from utils import kilobyte_to_gigabyte


class MachineInfo(ft.UserControl):
    def __init__(self, ip, alias):
        super().__init__()
        self.ip = ip
        self.client = SNMPClient(ip)
        self.alias = alias
        self.counter = 0

    def did_mount(self):
        self.running = True
```

```
        self.th = threading.Thread(target=self.update_values, args=(), daemon=True)
        self.th.start()

    def will_unmount(self):
        self.running = False

    def update_values(self):
        while self.running:
            self.cpu.value = f"{self.client.get_cpu_usage_1min()}, {self.client.get_cpu_usage_5min()}, {self.client.get_cpu_usage_10min()}"
            self.temp.value = "{:.2f} °C".format(int(self.client.get_cpu_temperature()) / 1000)

            self.counter += 1
            if self.counter == 5:
                network_in = int(self.client.get_internet_traffic_in())
                network_out = int(self.client.get_internet_traffic_out())

                self.network_in.value = "{}/s".format(size( (network_in - self.network_in_prev) / 5 ))
                self.network_out.value = "{}/s".format(size( (network_out - self.network_out_prev) / 5 ))

                self.network_in_prev = network_in
                self.network_out_prev = network_out
                self.counter = 0

            total_ram = int(self.client.get_total_ram())
            avail_ram = int(self.client.get_avail_ram())
            cached_ram = int(self.client.get_cached_ram())
            used_ram = total_ram - avail_ram - cached_ram

            self.total_ram.value = "{:.2f}G".format(kilobyte_to_gigabyte(total_ram))
            self.avail_ram.value = "{:.2f}G".format(kilobyte_to_gigabyte(avail_ram))
            self.used_ram.value = "{:.2f}G".format(kilobyte_to_gigabyte(used_ram))
            self.cached_ram.value = "{:.2f}G".format(kilobyte_to_gigabyte(cached_ram))

            self.update()
            time.sleep(1)

    def build(self):
        self.cpu = ft.Text()
        self.temp = ft.Text()
        self.network_in = ft.Text()
```

```
        self.network_out = ft.Text()
        self.total_ram = ft.Text()
        self.avail_ram = ft.Text()
        self.cached_ram = ft.Text()
        self.shared_ram = ft.Text()
        self.used_ram = ft.Text()
        self.cpu_count = ft.Text()

        self.network_in_prev = int(self.client.get_internet_traffic_in())
        self.network_out_prev = int(self.client.get_internet_traffic_out())
        self.cpu_count.value = self.client.get_cpu_count()

        cpu_container = ft.Container(
            content=ft.Column(
                controls=[
                    ft.Row(
                        controls=[
                            ft.Text("CPU count:", weight=ft.FontWeight.BOLD),
                            self.cpu_count
                        ]
                    ),
                    ft.Text("CPU average usage:", weight=ft.FontWeight.BOLD),
                    self.cpu
                ]),
            padding=10,
            margin = 0,
            height = 100,
            alignment=ft.alignment.center,
        )

        network_container = ft.Container(
            content=ft.Column(
                controls=[
                    ft.Row(
                        controls=[
                            ft.Icon(name=ft.icons.ARROW_DOWNWARD, size=20),
                            self.network_in
                        ]
                    ),
                    ft.Row(
                        controls=[
                            ft.Icon(name=ft.icons.ARROW_UPWARD, size=20),
                            self.network_out
                        ]
                    )
                ]),
            padding=10,
            margin = 0,
```

```
            height = 100,
            alignment=ft.alignment.center,
        )

        ram_container = ft.Container(
            content=ft.Column(
                controls=[
                    ft.Column(
                        controls=[
                        ft.Text("RAM", weight=ft.FontWeight.BOLD),
                        ft.Row(
                            controls =[
                                ft.Text("total"),
                                self.total_ram,
                            ],
                        ),
                        ft.Row(
                            controls =[
                            ft.Text("avail"),
                            self.avail_ram,
                            ],
                        ),
                        ft.Row(
                            controls =[
                            ft.Text("cached"),
                            self.cached_ram
                            ],
                        ),
                        ft.Row(
                            controls =[
                            ft.Text("used:"),
                            self.used_ram
                            ],
                        )
                        ]
                    )
                ]
                ),
            padding=10,
            margin = 0,
            height = 150,
            alignment=ft.alignment.center,
        )

        temp_container = ft.Container(
            content = ft.Column(
                controls=[
                    ft.Text("CPU temperature:", weight=ft.FontWeight.BOLD),
```

```
                    self.temp
                ]),
            padding=10,
            margin = 0,
            height = 150,
            alignment=ft.alignment.center,
        )

        main_content = ft.Column(
            width=300,
            controls=[
                ft.Text(value="{} / {}".format(self.ip, self.alias), size=20,
    weight=ft.FontWeight.BOLD),
                ft.Row(
                    controls=[
                        cpu_container,
                        network_container
                    ],
                ),
                ft.Container(
                    bgcolor=ft.colors.BLACK26,
                    border_radius=ft.border_radius.all(30),
                    height=1,
                    alignment=ft.alignment.center,
                    width=300
                ),
                ft.Row(
                    controls=[
                        temp_container,
                        ram_container
                    ],
                )
            ],
            spacing=0,
        )

        return ft.Container(
            content=main_content,
            border=ft.border.all(1, ft.colors.BLACK12),
            bgcolor=ft.colors.BLUE_GREY_100,
            padding=10,
            border_radius=10
        )
```

```
import re

def check_valid_ip(ip_string):
    aa = re.match(
            r"^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}$",
```

```
            ip_string
    )
    if aa is None:
        return False
    return True

def kilobyte_to_gigabyte(kilobyte):
    return kilobyte / 1024 / 1024
```

5 Выводы

В рамках курсового проекта было реализовано веб приложение для визуализации метрик, полученных с помощью SNMP с пользовательских устройств.