

Содержание

1 Постановка задачи

Цель работы

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

Задачи

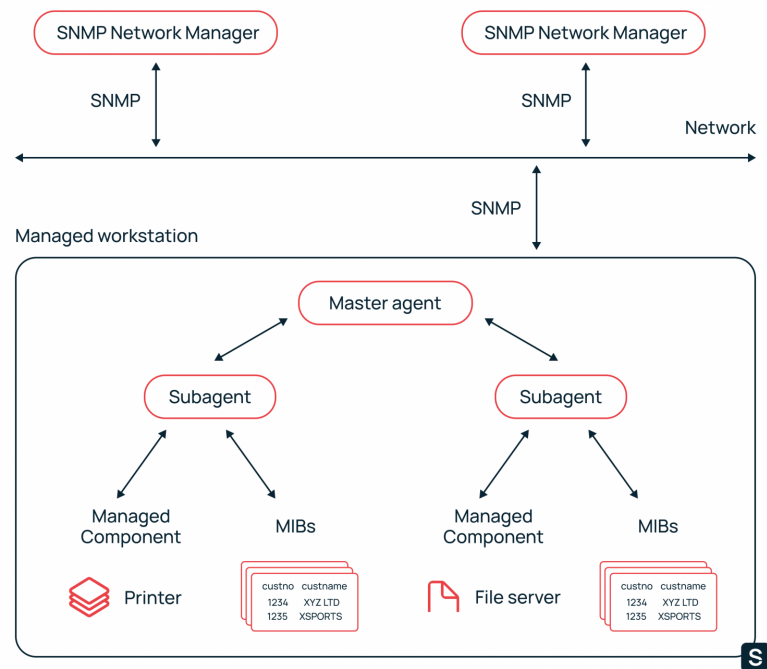
1. Исследовать протокол SNMP.
2. Настроить конфигурацию службы SNMP на целевых устройствах.
3. Определить стек разработки.
4. Разработка приложения для визуализации метрик с пользовательских устройств, полученных с помощью протокола SNMP.

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 суб-агенты, 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.

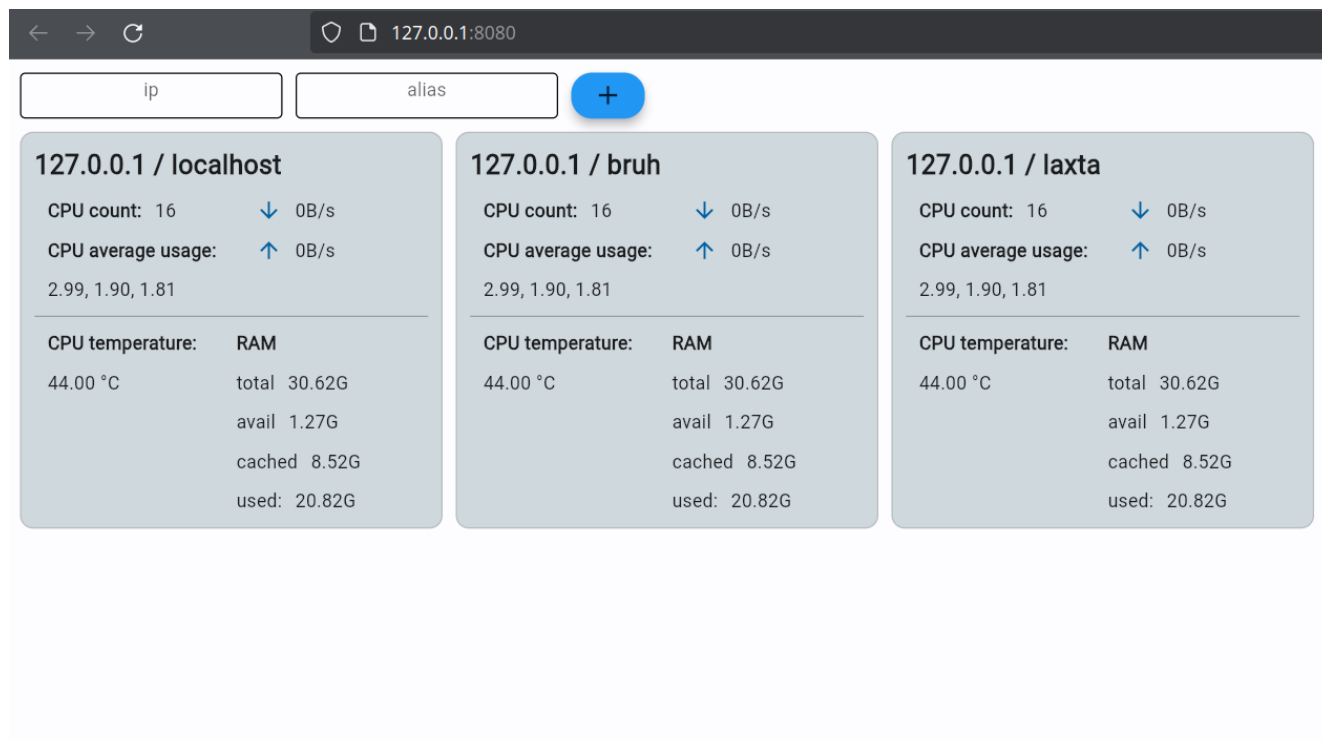


Рис. 1: Внешний вид приложения

```
1 import re
2 import flet as ft
3 from item import MachineInfo
4 from dialog_window import ErrorAlert
5 from utils import check_valid_ip
6
7 class App(ft.UserControl):
8     def __init__(self, path, page):
9         super().__init__()
10         self.path_to_config = path
11         self.machines = self.read_config(self.path_to_config)
12         self.page = page
13
14     def build(self):
15         self.add_item_button = ft.FloatingActionButton(
```

```

16         icon=ft.icons.ADD, height=35, on_click=self.add_item, bgcolor=ft.
colors.BLUE
17     )
18     self.ip_input = ft.TextField(hint_text="ip", width=200, height=35,
text_align=ft.TextAlign.CENTER, text_size=14)
19     self.alias_input = ft.TextField(hint_text="alias", width=200, height=35,
text_align=ft.TextAlign.CENTER, text_size=14)
20
21     self.items = ft.Row(
22         wrap=True,
23         spacing=10,
24         run_spacing=10,
25         controls=self.machines,
26         width=1600,
27     )
28
29     self.new_input = ft.Row(
30         width=600,
31         spacing=10,
32         controls=[self.ip_input, self.alias_input, self.add_item_button],
33     )
34
35     return ft.Column(controls=[self.new_input, self.items])
36
37 def add_item(self, e):
38     if not check_valid_ip(self.ip_input.value):
39         self.open_error_dialog(e, "Invalid IP", "Please enter valid IP adress"
)
40
41         return
42
43     new_machine = MachineInfo(self.ip_input.value, self.alias_input.value)
44     if not new_machine.client.check_connection():
45         self.open_error_dialog(e, "Error connection", "Please check your
machine connection")
46
47         return
48
49     self.machines.append(new_machine)
50     with open(self.path_to_config, 'a') as file:
51         file.write(f"\n{new_machine.ip} {new_machine.alias}\n")
52     self.update()
53
54 def open_error_dialog(self, e, err_text, main_text):
55     self.err_dialog = ft.AlertDialog(
56         title=ft.Text(err_text),
57         content=ft.Text(main_text),
58         actions_alignment=ft.MainAxisAlignment.END
59     )
60     self.page.dialog = self.err_dialog

```

```

59     self.err_dialog.open = True
60     self.page.update()
61
62     def read_config(self, file_path):
63         machines = []
64         with open(file_path, 'r') as file:
65             for line in file:
66                 values = line.strip().split()
67                 if len(values) == 1:
68                     ip = values[0]
69                     alias = ""
70                 elif len(values) == 2:
71                     ip = values[0]
72                     alias = values[1]
73                 else:
74                     continue
75                 if not check_valid_ip(ip):
76                     continue
77                 machine = MachineInfo(ip, alias)
78                 machines.append(machine)
79         return machines
80
81     def main(page: ft.Page):
82
83         page.add(
84             App("config.txt", page)
85         )
86
87     if __name__ == "__main__":
88         ft.app(target=main, view=ft.WEB_BROWSER, port=8080)

```

```

1  import threading
2  import time
3  import flet as ft
4  from snmp.snmp import SNMPClient
5  from hurry.filesize import size, si
6  from utils import kilobyte_to_gigabyte
7
8
9  class MachineInfo(ft.UserControl):
10     def __init__(self, ip, alias):
11         super().__init__()
12         self.ip = ip
13         self.client = SNMPClient(ip)
14         self.alias = alias
15         self.counter = 0
16
17     def did_mount(self):
18         self.running = True

```



```

19     self.th = threading.Thread(target=self.update_values, args=(), daemon=True
20 )
21     self.th.start()
22
23     def will_unmount(self):
24         self.running = False
25
26     def update_values(self):
27         while self.running:
28             self.cpu.value = f"{self.client.get_cpu_usage_1min()}, {self.client.
29 get_cpu_usage_5min()}, {self.client.get_cpu_usage_10min()}"
30             self.temp.value = "{:.2f} °C".format(int(self.client.
31 get_cpu_temperature()) / 1000)
32
33             self.counter += 1
34             if self.counter == 5:
35                 network_in = int(self.client.get_internet_traffic_in())
36                 network_out = int(self.client.get_internet_traffic_out())
37
38                 self.network_in.value = "{}/s".format(size( (network_in - self.
39 network_in_prev) / 5 ))
40                 self.network_out.value = "{}/s".format(size( (network_out - self.
41 network_out_prev) / 5 ))
42
43                 self.network_in_prev = network_in
44                 self.network_out_prev = network_out
45                 self.counter = 0
46
47                 total_ram = int(self.client.get_total_ram())
48                 avail_ram = int(self.client.get_avail_ram())
49                 cached_ram = int(self.client.get_cached_ram())
50                 used_ram = total_ram - avail_ram - cached_ram
51
52                 self.total_ram.value = "{:.2f}G".format(kilobyte_to_gigabyte(total_ram
53 ))
54                 self.avail_ram.value = "{:.2f}G".format(kilobyte_to_gigabyte(avail_ram
55 ))
56                 self.used_ram.value = "{:.2f}G".format(kilobyte_to_gigabyte(used_ram))
57                 self.cached_ram.value = "{:.2f}G".format(kilobyte_to_gigabyte(
58 cached_ram))
59
60             self.update()
61             time.sleep(1)
62
63     def build(self):
64         self.cpu = ft.Text()
65         self.temp = ft.Text()
66         self.network_in = ft.Text()

```

```

59     self.network_out = ft.Text()
60     self.total_ram = ft.Text()
61     self.avail_ram = ft.Text()
62     self.cached_ram = ft.Text()
63     self.shared_ram = ft.Text()
64     self.used_ram = ft.Text()
65     self.cpu_count = ft.Text()
66
67     self.network_in_prev = int(self.client.get_internet_traffic_in())
68     self.network_out_prev = int(self.client.get_internet_traffic_out())
69     self.cpu_count.value = self.client.get_cpu_count()
70
71     cpu_container = ft.Container(
72         content=ft.Column(
73             controls=[
74                 ft.Row(
75                     controls=[
76                         ft.Text("CPU count:", weight=ft.FontWeight.BOLD),
77                         self.cpu_count
78                     ]
79                 ),
80                 ft.Text("CPU average usage:", weight=ft.FontWeight.BOLD),
81                 self.cpu
82             ]),
83         padding=10,
84         margin = 0,
85         height = 100,
86         alignment=ft.alignment.center,
87     )
88
89     network_container = ft.Container(
90         content=ft.Column(
91             controls=[
92                 ft.Row(
93                     controls=[
94                         ft.Icon(name=ft.icons.ARROW_DOWNWARD, size=20),
95                         self.network_in
96                     ]
97                 ),
98                 ft.Row(
99                     controls=[
100                         ft.Icon(name=ft.icons.ARROW_UPWARD, size=20),
101                         self.network_out
102                     ]
103                 )
104             ]),
105         padding=10,
106         margin = 0,

```

```

107         height = 100,
108         alignment=ft.alignment.center,
109     )
110
111     ram_container = ft.Container(
112         content=ft.Column(
113             controls=[
114                 ft.Column(
115                     controls=[
116                         ft.Text("RAM", weight=ft.FontWeight.BOLD),
117                         ft.Row(
118                             controls = [
119                                 ft.Text("total"),
120                                 self.total_ram,
121                             ],
122                         ),
123                         ft.Row(
124                             controls = [
125                                 ft.Text("avail"),
126                                 self.avail_ram,
127                             ],
128                         ),
129                         ft.Row(
130                             controls = [
131                                 ft.Text("cached"),
132                                 self.cached_ram
133                             ],
134                         ),
135                         ft.Row(
136                             controls = [
137                                 ft.Text("used:"),
138                                 self.used_ram
139                             ],
140                         )
141                     ]
142                 )
143             ]
144         ),
145         padding=10,
146         margin = 0,
147         height = 150,
148         alignment=ft.alignment.center,
149     )
150
151     temp_container = ft.Container(
152         content = ft.Column(
153             controls=[
154                 ft.Text("CPU temperature:", weight=ft.FontWeight.BOLD),

```

```

155         self.temp
156     ]),
157     padding=10,
158     margin = 0,
159     height = 150,
160     alignment=ft.alignment.center,
161 )
162
163 main_content = ft.Column(
164     width=300,
165     controls=[
166         ft.Text(value="{} / {}".format(self.ip, self.alias), size=20,
weight=ft.FontWeight.BOLD),
167         ft.Row(
168             controls=[
169                 cpu_container,
170                 network_container
171             ],
172         ),
173         ft.Container(
174             bgcolor=ft.colors.BLACK26,
175             border_radius=ft.border_radius.all(30),
176             height=1,
177             alignment=ft.alignment.center,
178             width=300
179         ),
180         ft.Row(
181             controls=[
182                 temp_container,
183                 ram_container
184             ],
185         )
186     ],
187     spacing=0,
188 )
189
190 return ft.Container(
191     content=main_content,
192     border=ft.border.all(1, ft.colors.BLACK12),
193     bgcolor=ft.colors.BLUE_GREY_100,
194     padding=10,
195     border_radius=10
196 )

```

```

1 import re
2
3 def check_valid_ip(ip_string):
4     aa = re.match(
5         r"~\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}$",

```

```
6         ip_string
7     )
8     if aa is None:
9         return False
10    return True
11
12 def kilobyte_to_gigabyte(kilobyte):
13     return kilobyte / 1024 / 1024
```

5 Выводы

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