

Создание системы мониторинга сетевого трафика на языке программирования Python

Эрдман Александр Алексеевич

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данном исследовании рассматривается процесс создания графического приложения для мониторинга сетевого трафика на компьютере. Программа реализована на языке программирования Python с применением двух модулей Psutil и Matplotlib. Результатом исследования является практическое применение приложения и описание его работы.

Ключевые слова: python, matplotlib, psutil

Creating a network traffic monitoring system in the Python programming language

Erdman Alexander Alekseevich

Sholom-Aleichem Priamursky State University

Student

Abstract

This study examines the process of creating a graphical application for monitoring network traffic on a computer. The program is implemented in the Python programming language using two modules Psutil and Matplotlib. The result of the research is the practical application of the application and a description of its work.

Keywords: python, matplotlib, psutil

1 Введение

1.1 Актуальность

Мониторинг сетевого трафика является критически важной задачей в современном цифровом мире, где интернет и сетевые технологии играют ключевую роль в повседневной жизни и бизнесе. С ростом объема данных, передаваемых через сети, и увеличением числа устройств, подключенных к интернету, становится необходимым эффективное управление сетевыми ресурсами. Мониторинг сетевого трафика позволяет выявлять узкие места, оптимизировать использование пропускной способности и обеспечивать безопасность сети, что особенно актуально в условиях увеличения кибератак и необходимости защиты конфиденциальной информации.

Кроме того, мониторинг сетевого трафика играет важную роль в диагностике и устранении неполадок в сети. Анализ данных о сетевом трафике позволяет выявлять аномалии и потенциальные проблемы, такие как

перегрузка сети, неэффективное использование ресурсов или подозрительная активность. Это особенно важно для организаций, где стабильность и надежность сети являются критическими факторами для успешной работы. Внедрение систем мониторинга сетевого трафика способствует повышению эффективности сетевой инфраструктуры и улучшению качества обслуживания конечных пользователей, что в конечном итоге способствует достижению стратегических целей организации.

1.2 Обзор исследований

М.В. Васильев и И.И. Митричев в совместном исследовании разработали программное приложение для мониторинга программных процессов и обработки квантово-химических расчётов, произведённых с помощью пакета Orca [1]. Д.С. Бондар и Е.А. Ружицкая привели пример реализации web-сервиса мониторинга коммутаторов по протоколу SNMP с использованием языка Python [2]. Т.В. Маркосян в своей статье рассмотрел широкий спектр инструментов и библиотек для разработки систем мониторинга и анализа данных на языке программирования Python [3]. Д.А. Дрянкова описала процесс визуализации данных из таблицы Pandas с использованием библиотеки Matplotlib [4]. В.Д. Семенова и Е.Д. Паршина в совместном труде изучили возможности решения математических задач при помощи языка программирования Python и библиотеки Matplotlib на примере решения дифференциального уравнения в частных производных [5].

1.3 Цель исследования

Целью исследования является создание системы для мониторинга сетевого трафика на компьютере с помощью языка программирования Python.

2 Материалы и методы

Для решения поставленной задачи используется язык программирования Python, библиотеки Psutil, Matplotlib и Tkinter. В качестве среды разработки используется PyCharm.

3 Результаты и обсуждения

Для работы программы необходимо установить три библиотеки: psutil, matplotlib, tkinter. Первая библиотека отвечает за получения информации о системе, включая данные о сетевом трафике. Второй модуль используется для создания статических, анимированных и интерактивных визуализаций в Python. Последний набор инструментов необходим для создания графических интерфейсов в Python.

Следующим шагом является программирования события получения данных о сетевом трафике (рис. 1).

```
def get_network_traffic():
    net_io = psutil.net_io_counters()
    sent = net_io.bytes_sent / (1024 * 1024)
    recv = net_io.bytes_recv / (1024 * 1024)
    return sent, recv
```

Рисунок 1. Получение данных о сетевом трафике

Функция “psutil.net_io_counters()” возвращает объект, содержащий информацию о сетевом трафике, включая количество отправленных и полученных байт. Эти значения переводятся в мегабайты для удобства отображения.

Далее реализуется функция “update_data”, которая обновляет данные и графики каждую секунду (рис. 2).

```
def update_data(i):
    sent, recv = get_network_traffic()
    sent_data.append(sent)
    recv_data.append(recv)
    times.append(time.time())
```

Рисунок 2. Обновление данных и графиков

Создание графиков является ключевым элементом программы для мониторинга сетевого трафика, так как именно графики позволяют пользователям визуально оценивать состояние сети в реальном времени. В программе создаются четыре основных графика, каждый из которых служит для отображения различных аспектов сетевого трафика (рис. 3 и 4).

```
# Обновление графика загрузки
ax_recv.clear()
ax_recv.plot(*args, times, recv_data, label='Полученные данные (МБ)')
ax_recv.legend()
ax_recv.set_xlabel('Время (сек)')
ax_recv.set_ylabel('МБ')
ax_recv.set_title('Полученные данные')

# Обновление графика выгрузки
ax_sent.clear()
ax_sent.plot(*args, times, sent_data, label='Отправленные данные (МБ)')
ax_sent.legend()
ax_sent.set_xlabel('Время (сек)')
ax_sent.set_ylabel('МБ')
ax_sent.set_title('Отправленные данные')

# Обновление графика кумулятивного трафика
cumulative_recv = [sum(recv_data[:i+1]) for i in range(len(recv_data))]
cumulative_sent = [sum(sent_data[:i+1]) for i in range(len(sent_data))]
ax_cumulative.clear()
ax_cumulative.plot(*args, times, cumulative_recv, label='Кумулятивные полученные данные (МБ)', color='blue')
ax_cumulative.plot(*args, times, cumulative_sent, label='Кумулятивные отправленные данные (МБ)', color='orange')
ax_cumulative.legend()
ax_cumulative.set_xlabel('Время (сек)')
ax_cumulative.set_ylabel('МБ')
ax_cumulative.set_title('Кумулятивный сетевой трафик')

# Обновление круговой диаграммы
total = sum(recv_data) + sum(sent_data)
recv_percent = (sum(recv_data) / total) * 100 if total > 0 else 0
sent_percent = (sum(sent_data) / total) * 100 if total > 0 else 0
ax_pie.clear()
ax_pie.pie([recv_percent, sent_percent], labels=['Получено', 'Отправлено'], autopct='%1.1f%%', startangle=140)
ax_pie.set_title('Процент загрузки и отправки данных')
```

Рисунок 3. Обновления графиков мониторинга

```
# Создание графиков
fig_recv, ax_recv = plt.subplots()
fig_sent, ax_sent = plt.subplots()
fig_cumulative, ax_cumulative = plt.subplots()
fig_pie, ax_pie = plt.subplots()

# Создание графического интерфейса
root = tk.Tk()
root.title("Монитор сетевого трафика")

text_var = tk.StringVar()
text_label = ttk.Label(root, textvariable=text_var)
text_label.pack()

# Встраивание графика загрузки в Tkinter
canvas_recv = FigureCanvasTkAgg(fig_recv, master=root)
canvas_recv.draw()
canvas_recv.get_tk_widget().pack(side=tk.LEFT, fill=tk.BOTH, expand=1)

# Встраивание графика выгрузки в Tkinter
canvas_sent = FigureCanvasTkAgg(fig_sent, master=root)
canvas_sent.draw()
canvas_sent.get_tk_widget().pack(side=tk.RIGHT, fill=tk.BOTH, expand=1)

# Встраивание графика кумулятивного трафика в Tkinter
canvas_cumulative = FigureCanvasTkAgg(fig_cumulative, master=root)
canvas_cumulative.draw()
canvas_cumulative.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

# Встраивание круговой диаграммы в Tkinter
canvas_pie = FigureCanvasTkAgg(fig_pie, master=root)
canvas_pie.draw()
canvas_pie.get_tk_widget().pack(side=tk.BOTTOM, fill=tk.BOTH, expand=1)

# Создание анимации для обновления графиков
ani_recv = animation.FuncAnimation(fig_recv, update_data, interval=1000, cache_frame_data=False)
ani_sent = animation.FuncAnimation(fig_sent, update_data, interval=1000, cache_frame_data=False)
ani_cumulative = animation.FuncAnimation(fig_cumulative, update_data, interval=1000, cache_frame_data=False)
ani_pie = animation.FuncAnimation(fig_pie, update_data, interval=1000, cache_frame_data=False)
```

Рисунок 4. Графическое создание графиков и анимации

Первый график, который отображает количество полученных данных, создается с помощью функции “subplots” из библиотеки “matplotlib”. Эта функция возвращает объект “Figure” и объект “Axes”, которые используются для построения графика. График полученных данных отображает количество данных, полученных за последние 60 секунд, в мегабайтах. Для этого используется метод “plot”, который строит линейный график на основе данных, собранных функцией “get_network_traffic”. График обновляется каждую секунду, что позволяет пользователям видеть динамику изменения трафика в реальном времени.

Второй график, который отображает количество отправленных данных, создается аналогично первому. Он также использует метод “plot” для построения линейного графика, но в этом случае отображаются данные об отправленных мегабайтах за последние 60 секунд. Этот график позволяет пользователям видеть, сколько данных было отправлено через сеть, и как это количество изменяется со временем.

Третий график, который отображает кумулятивный трафик, показывает накопленное количество полученных и отправленных данных. Для этого используется метод “plot”, который строит два линейных графика: один для

кумулятивных полученных данных и другой для кумулятивных отправленных данных. Кумулятивные данные рассчитываются как сумма всех значений, полученных до текущего момента времени. Этот график позволяет пользователям видеть общий объем данных, прошедших через сеть с момента запуска программы.

Четвертый график представляет собой круговую диаграмму, которая показывает процентное соотношение загрузки и отправки данных. Для этого используется метод “pie”, который строит круговую диаграмму на основе процентных значений. Процентные значения рассчитываются как доля полученных и отправленных данных от общего объема трафика. Этот график позволяет пользователям легко оценить, какая часть трафика приходится на полученные данные, а какая — на отправленные.

Все графики обновляются каждую секунду с помощью функции “FuncAnimation” из библиотеки “matplotlib.animation”. Эта функция вызывает метод “update_data”, который обновляет данные и перерисовывает графики. Таким образом, пользователи могут видеть актуальную информацию о сетевом трафике в реальном времени. Графики встраиваются в графический интерфейс, созданный с помощью библиотеки “tkinter”. Для этого используется класс “FigureCanvasTkAgg”, который позволяет отображать графики “matplotlib” в окне “Tkinter”. Каждый график встраивается в отдельный виджет “Canvas”, который затем добавляется в окно “Tkinter” с помощью метода “pack”.

Реализация программы завершена. Необходимо провести тестирования и убедиться в корректной работе программы (рис. 5).

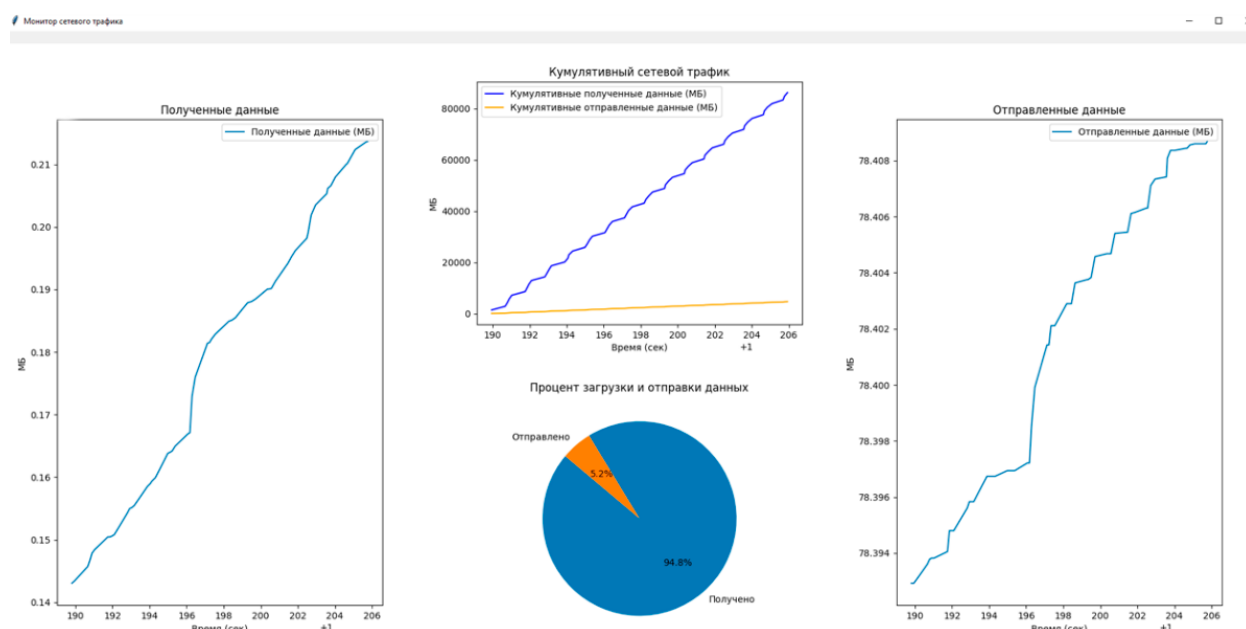


Рисунок 5. Результат работы программы

Вывод

В результате исследования была реализована программа на языке программирования Python для мониторинга сетевого трафика с графическим

представлением в виде диаграмм. В основу программы были заложены методы модулей psutil, matplotlib, tkinter. Программа может быть полезна как для обычных пользователей, так и для администраторов сетей, обеспечивая наглядное представление данных и возможность анализа сетевого трафика.

Библиографический список

1. Васильев М.В., Митричев И.И. Программное приложение для мониторинга квантово-химических расчётов в Linux-системах на языке Python 3 // Успехи в химии и химической технологии. 2020. Т. 34. № 6 (229). С. 103-105.
2. Бондар Д.С., Ружицкая Е.А. Разработка web-сервиса мониторинга коммутаторов по протоколу SNMP с использованием языка Python // В сборнике: Новые математические методы и компьютерные технологии в проектировании, производстве и научных исследованиях. Материалы XIX Республиканской научной конференции студентов и аспирантов в 2 частях. Гомель, 2016. С. 96-97.
3. Маркосян Т.В. Использование Python для создания системы мониторинга и анализа данных в реальном времени // Научный Лидер. 2024. № 23 (173). С. 14-20.
4. Дрянкова Д.А. Визуализация данных с помощью библиотек Pandas и Matplotlib для языка программирования Python // Дневник науки. 2023. № 6 (78).
5. Семенова В.Д., Паршина Е.Д. Использование Python - библиотеки Matplotlib для построения графиков математических функций // В сборнике: Математические модели техники, технологий и экономики. Материалы Всероссийской студенческой научно-практической конференции. Санкт-Петербург, 2024. С. 375-378.