

Разработка приложения для анализа скорости интернет-соединения с визуализацией данных и проверкой потоковой загрузки

Малик Александр Игоревич

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данной работе описывается разработка приложения для анализа скорости интернет-соединения с использованием библиотеки Python speedtest-cli и графического интерфейса Tkinter. Приложение предоставляет возможность измерять скорость загрузки и выгрузки, задержку соединения, проверять доступность популярных веб-сайтов, оценивать потоковую загрузку видео для различных разрешений, а также сохранять результаты для последующей визуализации. Реализованы механизмы автоматического выбора доступных серверов и расчета времени загрузки видео. Программное решение предоставляет удобные инструменты для анализа качества интернет-соединения, что особенно актуально в условиях глобальной цифровизации.

Ключевые слова: интернет-соединение, скорость загрузки, Tkinter, Python, анализ данных, потоковая загрузка, визуализация, speedtest-cli.

Development of an application for analyzing Internet connection speed with data visualization and streaming download verification

Malik Aleksandr Igorevich

Sholom-Aleichem Priamursky State University

Student

Abstract

This paper describes the development of an application for analyzing the speed of an Internet connection using the Python library speedtest-cli and the graphical interface Tkinter. The application provides the ability to measure download and upload speeds, connection latency, check the availability of popular websites, evaluate video streaming for various resolutions, and save the results for later visualization. The mechanisms of automatic selection of available servers and calculation of video download time are implemented. The software solution provides convenient tools for analyzing the quality of Internet connections, which is especially important in the context of global digitalization.

Keywords: internet connection, download speed, Tkinter, Python, data analysis, streaming, visualization, speedtest-cli.

1 Введение

1.1 Актуальность

В эпоху активной цифровизации качество интернет-соединения играет важную роль в работе, обучении и досуге пользователей. Возможности оценки скорости подключения и стабильности сети необходимы для контроля качества предоставляемых услуг интернет-провайдерами. Разработка удобного приложения для мониторинга скорости соединения, визуализации данных и проверки возможности потоковой загрузки видео является важной задачей, способствующей улучшению пользовательского опыта и повышению качества интернет-услуг.

1.2 Обзор исследований

А.Э. Фёдорова, Е.Ю. Герлинг, К.А. Ахрамеева и В.И. Андрианов в своём исследовании разработали веб-интерфейс для мониторинга беспроводных сетей семейства IEEE 802.11, уделив внимание его оптимизации для маломощных устройств и обеспечению информационной безопасности [1]. В другой работе Е.Ю. Герлинг, М.М. Ковцур и А.Э. Фёдорова акцентировали внимание на преимуществах веб-приложений, таких как доступ через браузер, централизованное хранение данных и простота резервного копирования, что упрощает настройку сетевых устройств [2]. А.В. Брюхачев, А.В. Светлов, И.В. Ханин и С.С. Шигуров предложили архитектуру информационно-измерительной системы для мониторинга скорости интернет-соединения в сетях WCDMA, проанализировав активные и пассивные методы измерения, что позволяет эффективно оценивать качество соединения [3]. А.А. Ермаков и А.Д. Сатыбаев провели исследование развития широкополосного интернета в Кыргызстане, выявив различия между городскими и сельскими районами и обозначив ключевые направления для улучшения интернет-доступа [4]. В статье Д. А. Алигуловича, З. И. Махмудовой и Р. Ф. Сайфуллаева рассматриваются плюсы и минусы использования Python и Tkinter для разработки приложений с графическим пользовательским интерфейсом (GUI). Авторы подробно анализируют потенциальные области применения Tkinter, акцентируя внимание на его возможностях для создания кроссплатформенных решений, что делает этот инструмент особенно привлекательным для использования [5].

1.3 Цель исследования

Разработать приложение для измерения скорости интернет-соединения, проверки возможности потоковой загрузки видео и визуализации данных с использованием Python и библиотеки speedtest-cli.

2 Материалы и методы

В процессе разработки использовались современные инструменты программирования: библиотека speedtest-cli для взаимодействия с серверами скорости, Tkinter для создания графического интерфейса и matplotlib для визуализации данных. Программа измеряет ключевые параметры интернет-

соединения: скорость загрузки, скорость выгрузки и задержку (пинг). Проверка потоковой загрузки видео основана на расчетах минимальных требований для различных разрешений. Реализован автоматический выбор доступного сервера для предотвращения сбоев при тестировании.

3 Результаты и обсуждения

1. Импорт необходимых библиотек (рис.1).

```
import speedtest
import tkinter as tk
from tkinter import ttk
import threading
import matplotlib.pyplot as plt
from datetime import datetime
```

Рисунок 1. Импорт библиотек

Программа начинается с подключения модулей:

- **speedtest**: для работы с API Speedtest и измерения скорости интернета.
- **tkinter**: для создания графического интерфейса.
- **matplotlib.pyplot**: для построения графиков.
- **datetime**: для работы с датами и временем.
- **threading**: для выполнения задач в параллельных потоках (чтобы интерфейс не зависал).

2. Функция определения доступности сервера (рис.2).

```
def get_available_server(): 1 usage
    try:
        st = speedtest.Speedtest()
        servers = st.get_servers()
        for server_list in servers.values():
            for server in server_list:
                try:
                    st.get_best_server([server])
                    return server
                except Exception:
                    continue
        raise Exception("Нет доступных серверов.")
    except Exception as e:
        result_var.set(f"Ошибка при получении серверов: {e}")
    return None
```

Рисунок 2. Код функции определения доступности сервера

Эта функция определяет доступный сервер для тестирования:

- Создается объект Speedtest.
- Перебираются доступные серверы из метода get_servers.
- Каждый сервер проверяется на доступность. Если сервер доступен, он возвращается для последующего использования.

3. Функция измерения скорости (рис.3).

```
def test_speed(): 1 usage
    try:
        # Создание объекта Speedtest
        st = speedtest.Speedtest()

        # Получение доступного сервера
        available_server = get_available_server()
        if not available_server:
            return

        st.get_best_server([available_server])

        # Измерение скорости загрузки (download)
        download_speed = st.download() / 1_000_000 # Перевод в Мбит/с

        # Измерение скорости выгрузки (upload)
        upload_speed = st.upload() / 1_000_000 # Перевод в Мбит/с

        # Измерение пинга (задержки)
        ping = st.results.ping

        # Обновление результатов в интерфейсе
        result_var.set(f"Скорость загрузки: {download_speed:.2f} Мбит/с\n"
                       f"Скорость выгрузки: {upload_speed:.2f} Мбит/с\n"
                       f"Пинг: {ping:.2f} мс")

        # Сохранение результатов для графика
        save_results(download_speed, upload_speed, ping)

    except Exception as e:
        result_var.set(f"Ошибка: {e}")
```

Рисунок 3. Код функции измерения скорости

Эта функция проводит измерение скорости:

- Инициализация объекта Speedtest.

- Получение доступного сервера через `get_available_server`.
- Измерение скорости загрузки, выгрузки и пинга.
- Отображение результатов в интерфейсе через `result_var`.
- Сохранение результатов в файл с помощью функции `save_results`.

4. Функция запуска измерения скорости в отдельном потоке (рис.4).

```
def start_test(): 1 usage
    result_var.set("Тестирование... Пожалуйста, подождите.")
    threading.Thread(target=test_speed).start()
```

Рисунок 4. Код функции

Запускает функцию `test_speed` в отдельном потоке, чтобы основной интерфейс оставался отзывчивым.

5. Функция получения информации о провайдере (рис.5).

```
def show_provider_info(): 1 usage
    try:
        st = speedtest.Speedtest()
        best_server = st.get_best_server()
        isp = st.results.client["isp"]
        ip = st.results.client["ip"]
        location = f"{st.results.client['lat']}, {st.results.client['lon']}"

        provider_var.set(f"Ваш провайдер: {isp}\nIP-адрес: {ip}\nМестоположение: {location}")
    except Exception as e:
        provider_var.set(f"Ошибка получения данных провайдера: {e}")
```

Рисунок 5. Код функции

Получает информацию о провайдере, IP-адресе и географическом местоположении пользователя:

- Используется метод `get_best_server` из библиотеки `speedtest`.
- Данные провайдера (`isp`), IP-адрес и координаты (широта и долгота) выводятся в интерфейс.

6. Функция сохранения результатов (рис.6).

```
def save_results(download, upload, ping): 1 usage
    try:
        now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        with open("speedtest_results.txt", "a") as file:
            file.write(f"Дата: {now}\n")
            file.write(f"Скорость загрузки: {download:.2f} Мбит/с\n")
            file.write(f"Скорость выгрузки: {upload:.2f} Мбит/с\n")
            file.write(f"Пинг: {ping:.2f} мс\n\n")
    except Exception as e:
        result_var.set(f"Ошибка при сохранении результатов: {e}")
```

Рисунок 6. Код функции

Сохраняет результаты тестов в файл speedtest_results.txt:

- В файл записывается текущая дата и время теста, скорость загрузки, выгрузки и пинг.
- Каждый тест записывается в отдельной секции.

7. Функция отображения графиков результатов измерения скорости (рис.7).

```
def plot_results(): 1 usage
    try:
        times, downloads, uploads = [], [], []

        # Чтение данных из файла
        with open("speedtest_results.txt", "r") as file:
            lines = file.readlines()

        for i in range(0, len(lines), 5):
            times.append(lines[i].split(": ")[1].strip())
            downloads.append(float(lines[i + 1].split(": ")[1].split()[0]))
            uploads.append(float(lines[i + 2].split(": ")[1].split()[0]))

        # Построение графика
        plt.figure(figsize=(10, 5))
        plt.plot(*args: times, downloads, label="Скорость загрузки (Мбит/с)", marker="o")
        plt.plot(*args: times, uploads, label="Скорость выгрузки (Мбит/с)", marker="o")
        plt.xticks(rotation=45, fontsize=8)
        plt.xlabel("Время теста")
        plt.ylabel("Скорость (Мбит/с)")
        plt.title("История скорости интернет-соединения")
        plt.legend()
        plt.tight_layout()
        plt.show()
    except Exception as e:
        result_var.set(f"Ошибка при построении графика: {e}")
```

Рисунок 7. Код функции

Читает файл с результатами и строит график:

- Данные времени, скорости загрузки и выгрузки извлекаются из файла.
 - Построение линейного графика с помощью matplotlib, где:
 - Ось X — время теста.
 - Ось Y — скорость соединения.
- На графике отображаются метки для загрузки и выгрузки.

8. Функция проверки доступности сайтов (рис.8).

```
def ping_websites(): 1 usage
    try:
        import os, platform

        websites = ["google.com", "youtube.com", "github.com"]
        ping_results = []

        for site in websites:
            if platform.system().lower() == "windows":
                response = os.system(f"ping -n 1 {site}")
            else:
                response = os.system(f"ping -c 1 {site}")

            if response == 0:
                ping_results.append(f"{site}: Доступен")
            else:
                ping_results.append(f"{site}: Недоступен")

        result_var.set("\n".join(ping_results))
    except Exception as e:
        result_var.set(f"Ошибка проверки сайтов: {e}")
```

Рисунок 8. Код функции

Проверяет доступность популярных сайтов:

- Использует системную команду ping (в зависимости от ОС).
- Выводит список сайтов с пометками "Доступен" или "Недоступен".

9. Функция потокового воспроизведения видео (рис.9).

```
def check_video_streaming(): 1 usage
    try:
        st = speedtest.Speedtest()
        download_speed = st.download() / 1_000_000 # Перевод в Мбит/с

        # Минимальные требования к скорости для различных разрешений
        video_requirements = {
            "480p": 2.5,
            "720p": 5.0,
            "1080p": 8.0,
            "4K": 25.0
        }

        streaming_results = []

        for quality, required_speed in video_requirements.items():
            if download_speed >= required_speed:
                load_time = (60 * required_speed) / download_speed
                streaming_results.append(f"Видео {quality}: Поддерживается (1 мин загружается за {load_time:.2f} сек)")
            else:
                streaming_results.append(f"Видео {quality}: Не поддерживается")

        result_var.set("\n".join(streaming_results))
    except Exception as e:
        result_var.set(f"Ошибка проверки видео: {e}")
```

Рисунок 9. Код функции

Проверяет возможность потокового воспроизведения видео:

- Сравнивает скорость загрузки с минимальными требованиями для воспроизведения видео в разных разрешениях (480p, 720p, 1080p, 4K).
- Рассчитывает время загрузки одной минуты видео для каждого разрешения.

10. Создание графического интерфейса

Интерфейс построен с помощью tkinter.

Окно приложения (root) (рис.10):

```
# Создание основного окна
root = tk.Tk()
root.title("Тест скорости интернет-соединения")
root.geometry("600x600")
root.configure(bg="#2e2e2e")

# Стили
style = ttk.Style()
style.theme_use("clam")
style.configure(style="TLabel", background="#2e2e2e", foreground="white", font=("Arial", 12))
style.configure(style="TButton", background="#444", foreground="white", font=("Arial", 12), padding=6)
style.map(style="TButton", background=[("active", "#555")])
```

Рисунок 10. Код

- Устанавливается размер, заголовок и фон.

- Применяется стиль для кнопок и меток с темной темой.

Кнопки:

- Для запуска теста скорости (рис.11).

```
# Кнопка для запуска теста
start_button = ttk.Button(root, text="Запустить тест скорости", command=start_test)
start_button.pack(pady=10)
```

Рисунок 11. Код

- Для отображения информации о провайдере (рис.12).

```
provider_button = ttk.Button(root, text="Показать данные провайдера", command=show_provider_info)
provider_button.pack(pady=10)
```

Рисунок 12. Код

- Для построения графика (рис.13).

```
plot_button = ttk.Button(root, text="Построить график скорости", command=plot_results)
plot_button.pack(pady=10)
```

Рисунок 13. Код

- Для проверки доступности сайтов (рис.14).

```
ping_button = ttk.Button(root, text="Проверить доступность сайтов", command=ping_websites)
ping_button.pack(pady=10)
```

Рисунок 14. Код

- Для проверки возможности воспроизведения видео (рис.15).

```
video_button = ttk.Button(root, text="Проверить возможность загрузки видео", command=check_video_streaming)
video_button.pack(pady=10)
```

Рисунок 15. Код

Текстовые метки:

- Для вывода результатов теста скорости (рис.16).

```
result_var = tk.StringVar()
result_label = ttk.Label(root, textvariable=result_var, wraplength=500)
result_label.pack(pady=10)
```

Рисунок 16. Код

- Для отображения информации о провайдере (рис.17).

```
provider_var = tk.StringVar()
provider_label = ttk.Label(root, textvariable=provider_var, wraplength=500)
provider_label.pack(pady=10)
```

Рисунок 17. Код

11. Многопоточность

Все операции, требующие времени (например, тест скорости), выполняются в отдельных потоках:

- Используется библиотека `threading`.
- Это предотвращает зависание интерфейса во время выполнения длительных задач.

Программа последовательно:

- Определяет доступный сервер.
- Тестирует соединение.
- Сохраняет результаты и отображает их.
- Проверяет доступность популярных ресурсов.
- Анализирует возможность потокового воспроизведения видео.
- Построение графиков для визуализации данных.

Выводы

Разработанное приложение обеспечивает удобный инструмент для мониторинга и анализа качества интернет-соединения. Автоматический выбор серверов, сохранение истории измерений и построение графиков позволяют получить детализированную картину производительности сети. Возможность проверки потоковой загрузки видео делает приложение полезным как для пользователей, так и для провайдеров услуг, предоставляя объективную оценку качества интернет-соединения.

Библиографический список

1. Федорова А.Э., Герлинг Е.Ю., Ахрамаева К.А., Андрианов В.И. Разработка веб-интерфейса для системы мониторинга беспроводных сетей семейства IEEE 802.11 // Региональная информатика и информационная безопасность. 2021. С. 381-385.
2. Герлинг Е.Ю., Ковцур М.М., Федорова А.Э. Разработка веб-интерфейса для системы мониторинга беспроводных сетей семейства IEEE 802.11 // Актуальные проблемы инфотелекоммуникаций в науке и образовании. 2021. С. 224-228.
3. Брюхачев А.В., Светлов А.В., Ханин И.В., Шигуров С.С. Информационно-измерительная система для мониторинга скорости интернет-соединения в сетях WCDMA // Известия высших учебных заведений. Поволжский регион. Технические науки. 2012. №. 2(22). С. 33-43.

4. Ермаков А.А., Сатыбаев А.Д. Обзор и анализ широкополосного интернет-соединения в кыргызстане // Проблемы автоматизации и управления. 2014. №. 1(26). С. 88-96.
5. Aligulovich D.A., Mahmudova Z.I., Sayfullayev R.F. Tibbiyot yo'nalishida zamonaviy pedagogik texnologiyalarini qo'llab tkinter paketida gui dasturlarini tuzish // Евразийский журнал математической теории и компьютерных наук. 2022. №2(12). С. 31-35.