

Лабораторная работа

Знакомство с интерфейсом сокетов

Цель работы:

Приобрести навыки написания программ с сетевым взаимодействием.
Изучить структуру интерфейса сокетов

Задачи работы:

- 1) Изучить концепцию сокетов и их типы соединения
- 2) Описать функции предоставляемые интерфейсом сокетов
- 3) Разработать демонстративную программу на языке Python

Теория

Концепция сокетов.

Интерфейс сокетов - это API для сетей TCP/IP (см. рис. 1). Другими словами, он описывает набор программных функций или процедур, позволяющий разрабатывать приложения для использования в сетях TCP/IP.

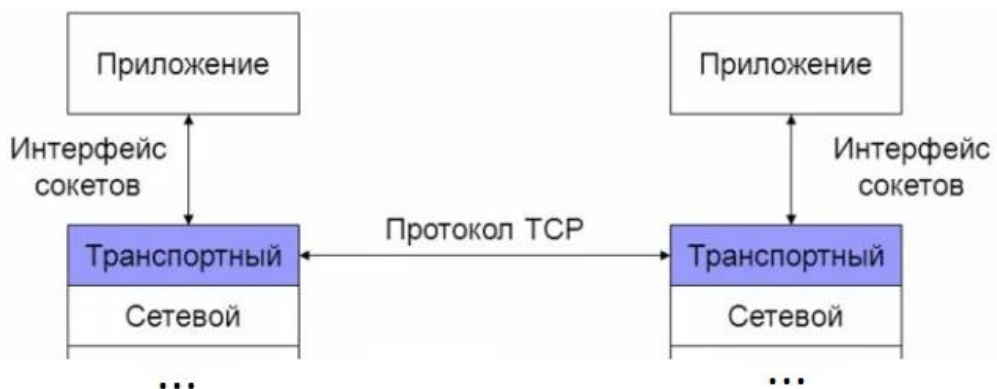


Рисунок 1 - Положение интерфейса сокетов в модели OSI и TCP/IP

Абстракция сокетов

Сокет можно рассматривать, как конечный пункт передачи данных по сети. Сетевое соединение — это процесс передачи данных по сети между двумя компьютерами или процессами. Сокет — конечный пункт передачи данных. Другими словами, когда программы используют сокет, для них он является абстракцией, представляющий одно из окончаний сетевого соединения. Для установления соединения в абстрактной модели сокетов необходимо, чтобы каждая из сетевых программ имела свой собственный сокет. Связь между двумя сокетами может быть ориентирована на соединение, а может быть и нет. Сетевая модель интерфейса сокетов использует цикл открыть-считать-записать-закрыть.

Создание сокета на сервере

Чтобы создать сокет, программа сервера вызывает функцию `socket`. Она, в свою очередь, возвращает дескриптор сокета, подобный дескриптору файла. Другими словами, дескриптор сокета указывает на таблицу, содержащую описание свойств и структуры сокета.

Далее, вызывается метод `bind`, который используется для присоединения сокета к определенному IP адресу и порту.

Затем – метод `listen`, который сообщает о том что сокет готов принимать сообщения по сети. При его вызове создается очередь для сообщений, которую необходимо указать. Например, если указана очередь в размере – 3, а сервер получил большее количество запросов на соединение, а предыдущие запросы еще не отработаны, то все новые запросы будут отбрасываться.

Следующий этап – вызов `accept`, для перехода в режим пассивного ожидания (ждет запросов от клиентов на установку соединения).

Для того чтобы прочитывать принятые сообщения, используется метод receive.

На рисунке 2 представлены описанные методы, используемые сервером для сетевых соединений.

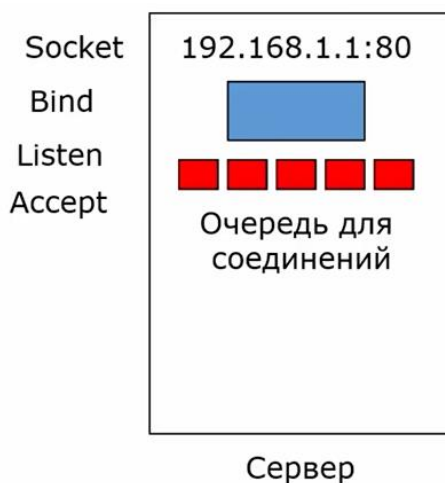


Рисунок 2 - Методы, используемые сервером

Создание сокета для клиента

Для настройки соединения клиента, используется следующий порядок вызова функций

1. **Socket.** Для клиента не имеет значения какой порт выбрать, номер порта назначается операционной системой автоматически. Поэтому метод bind на клиенте зачастую не вызывается
2. **Connect.** Указывается IP адрес и порт, с которыми нужно установить соединение.
3. **Send.** Используется для передачи данных по сети.
4. **Close.** Требуется для разрыва соединения с сервером.

На рисунке ниже представлена схема вызовов методов хостов для установки соединения, передачи и считывания данных, а также их конечного разъединения

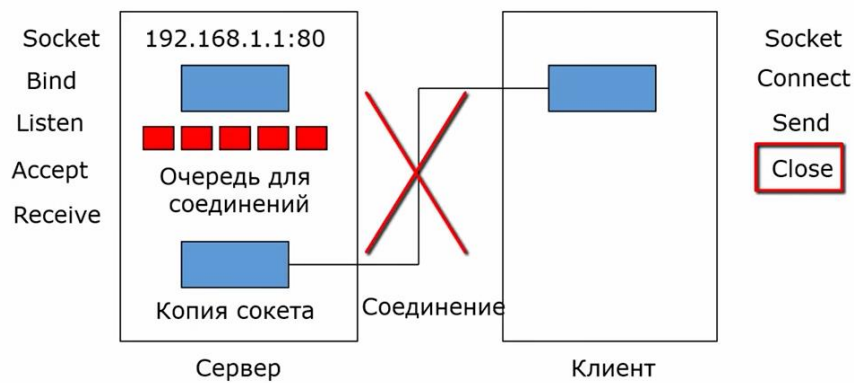


Рисунок 3 - Настройка хостов и их взаимодействие

Лабораторная работа

Получить практические навыки, разработав программу на языке Python для обмена текстовыми сообщениями между клиентом и сервером

Задание 1

Перед началом написания программы необходимо создать два файла формата “.ру”, в один из которых будет записан скрипт сервера, а в другом – клиента.

Написать код программы для сервера и изучить его описание представленное ниже

```

1  import socket
2
3  sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4
5  sock.bind(('', 9090))
6  sock.listen(1)
7
8  sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
9
10 print ('Server succesfully!')
11
12 conn, addr = sock.accept()
13 print("--Connected ", addr)
14 msg = b"Welcome to server!"
15 conn.sendall(msg)
16
17
18 while True:
19     msg = input()
20     conn.sendall(msg.encode())
21
22     data = conn.recv(1024)
23     if data:
24         print("CLIENT: ", data.decode())
25

```

Рисунок 4 - Код программы сервера

Первым делом, необходимо подключить библиотеку для работы с сокетами `import socket`. Затем, создание самого сокета в виде переменной `sock`, в настройках сокета указывается тип IP адреса `AF_INET`, что означает IPv4 и указание протокола транспортного уровня (`SOCK_STREAM` – означает протокол TCP, а `SOCK_DGRAM` - UDP). Далее, сокет привязывается к IP адресу и порту методом `bind()`, в данном случае указан пустой IP, что означает – прослушивать со всех доступных адресов (название или ip адрес домена, “localhost”, “127.0.0.1” и т.д.), а порт указан на 9090. Остается использовать `listen()` для перевода сервера в режим прослушки, в нем указывается количество клиентов, которое может подключиться, в данном случае – 1 клиент.

Метод `setsockopt` необходим для случая перезапуска скрипта сервера, так как используемый порт может быть еще занят, данный метод предотвращает это.

После успешного запуска, сервер выводит указанное сообщение. После, методом `accept` – извлекаем информацию о клиенте и отправляем ему сообщение функцией `sendall()` об успешном подключении.

Окончательный этап – использование цикла, который предлагает владельцу сервера ввести сообщение для клиента через `input()`, а затем отправить его в кодировке `sendall()`. Только после отправки сообщения сервер может получить сообщение от клиента методом `recv()`, в котором указан максимальный объем данных, а затем вывести его на экран `print()`.

Задание 2

Написать код программы для клиента и изучить его описание представленное ниже

Для клиента используется почти тот же набор функций что и для сервера, отличие лишь составляет то, что клиенту не требуется привязывать сокет к адресу и порту, так как функцией connect() он подключается, указав адрес и порт сервера. Порт клиента назначается операционной системой автоматически. На рисунке ниже представлен код для клиента, с упрощенным пояснением.

```
1 import socket
2
3 sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4 ipAdr = input("enter ip adress:")
5
6 sock.connect((ipAdr, 9090))
7
8 print ('Connecting to server...')
9 data = sock.recv(1024)
10 print(data.decode('UTF-8'))
11
12 while True:
13     msg = input()
14     if(msg=='exit'):
15         conn.close()
16         quit()
17
18     sock.sendall(msg.encode())
19     if data:
20         data = sock.recv(1024)
21         print("SERVER:", data.decode())
22
```

Поддержка сокетов

Указать тип IPv4 и TCP

Ввести IP сервера

Подключиться к серверу

Получить сообщение от сервера о подключении

Декодировать полученное сообщение

Ввод сообщения для сервера

Вывести полученное сообщение от сервера

Рисунок 5 - Код программы для клиента

Задание 3

Запустить и протестировать программу сетевого обмена.

Примечание: В случае отсутствия Python на устройстве, выполнить установку по ссылке: <https://www.python.org/downloads/windows/>

Необходимо первым делом убедиться, что брандмауэр на компьютерах выключен, или необходимые порты открыты для приема входящих и исходящих соединений (см. рис. 6).

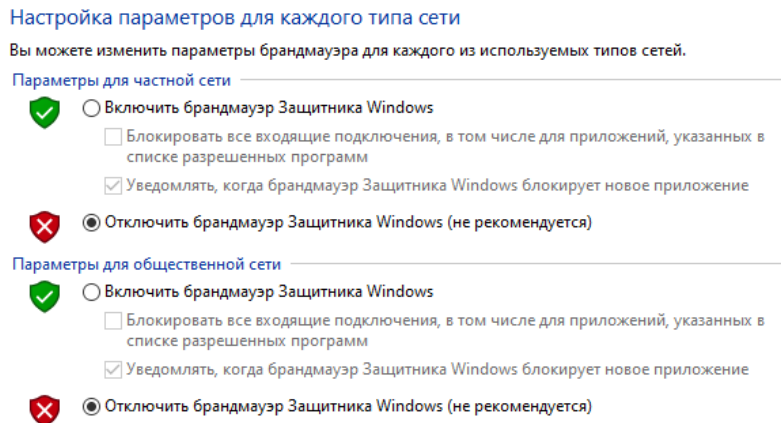


Рисунок 6 - Отключение брандмауэра

Тестирование будет проводиться в локальной сети, поэтому необходимо убедиться, что оба устройства: клиент и сервер подключены к одной сети (см. рис. 7).

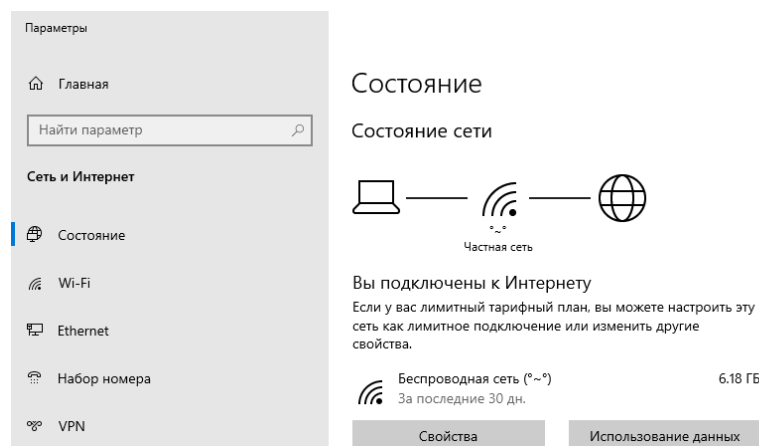


Рисунок 7 - Проверка состояния сети

Следующий шаг – запуск серверной программы.

Примечание: Запуск программы выполняется через командную строку “C:\filename.py”. Либо скомпилировать файл формата “.py” в исполняемый “.exe” руководствуясь одной из инструкций:

- <https://habr.com/ru/companies/vdsina/articles/557316/>
- <https://vc.ru/newtechaudit/122327-kompiliruem-kod-python-v-fayl-exe>

Важно учесть, что скомпилированный файл на 64-разрядной системе не выполнится на 32-битной, тогда как компиляция на 32-битной может запускаться на 64-разрядной.

При запуске можно заметить – сервер уведомляет о корректном включении. (см. рис. 8)



Рисунок 8 - Запуск серверной программы

Далее, необходимо запустить программу с клиентского устройства и ввести IP адрес сервера для подключения к нему. Сервер сразу же отправляет приветственное сообщение, означающее успешное подключение (см. рис. 9)

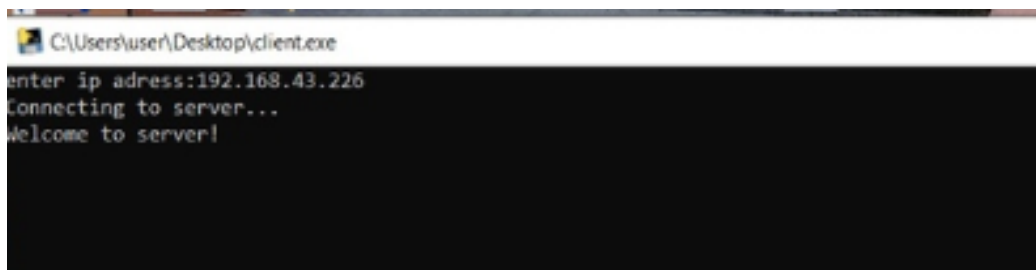
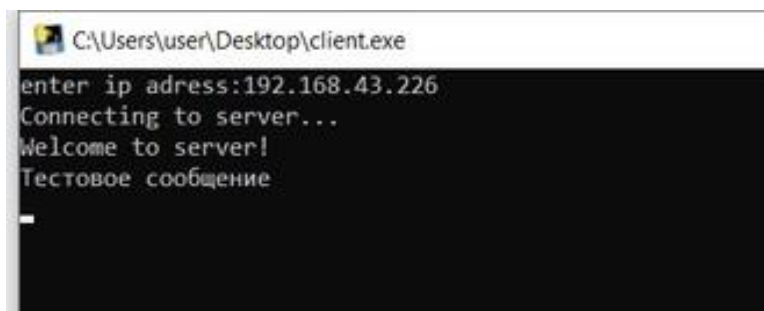


Рисунок 9 - Подключение клиента к серверу

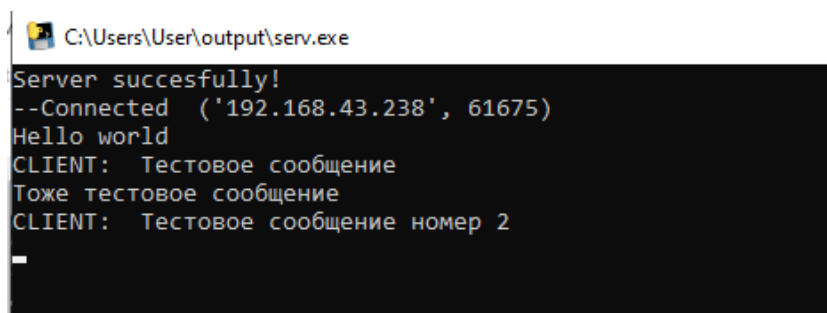
После успешного соединения можно отправить текстовое сообщение “Тестовое сообщение” с устройства клиента серверу (см. рис. 10)



```
C:\Users\user\Desktop\client.exe
enter ip adress:192.168.43.226
Connecting to server...
Welcome to server!
Тестовое сообщение
_
```

Рисунок 10 - Отправка данных серверу

Последний шаг – проверка доставки сообщения серверу, а также ответ клиенту (см. рис. 11).



```
C:\Users\User\output\serv.exe
Server succesfully!
--Connected ('192.168.43.238', 61675)
Hello world
CLIENT: Тестовое сообщение
Тоже тестовое сообщение
CLIENT: Тестовое сообщение номер 2
_
```

Рисунок 11 - Обмен сообщениями

В результате действий должно осуществиться корректное соединение между двумя устройствами, а также появиться возможность вести обмен текстовыми сообщениями через консоль.

Примечание: Недостатки программы заключаются в том, что необходимо избежать блокирующего ввода данных, так как невозможно получить ответное сообщение, без отправки своего. Также отправка пустого сообщения вызывает некорректное поведение со стороны клиента и сервера, для этого необходимо изучить особенности функции отправки. Далее, нужно учитывать максимальный размер буфера $data = recv()$, в случае его малого размера, как вариант – использовать цикл для фрагментированного сбора данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1) Сети передачи данных: учеб.-метод. пособие / Г.Ф. Масич. – Пермь: Изд-во Перм. нац. исслед. политехн. ун-та, 2014. – 192 с.
- 2) Информационные технологии – Обзорная лекция по курсу «Сети ЭВМ» [Электронный ресурс] / Электрон. текстовые дан. 2023. – Режим доступа: https://mf.grsu.by/Kafedry/kaf001/academic_process/048/40 свободный. (Дата обращения: 01.04.2023 г.)
- 3) Свободная энциклопедия – Сокеты Беркли [Электронный ресурс] / Электрон. текстовые дан. 2023. – Режим доступа: https://ru.wikipedia.org/wiki/Сокеты_Беркли, свободный. (Дата обращения: 01.04.2023 г.)

Код программы сервера

```
import socket

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

sock.bind(('', 9090))
sock.listen(1)

sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

print ('Server succesfully!')

conn, addr = sock.accept()
print("--Connected ", addr)
msg = b"Welcome to server!"
conn.sendall(msg)

while True:
    msg = input()
    conn.sendall(msg.encode())

    data = conn.recv(1024)
    if data:
        print("CLIENT: ", data.decode())
```

Код программы клиента

```
import socket

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
ipAdr = input("enter ip adress:")

sock.connect((ipAdr, 9090))

print ('Connecting to server...')
data = sock.recv(1024)
print(data.decode('UTF-8'))

while True:
    msg = input()
    if(msg=='exit'):
        conn.close()
        quit()

sock.sendall(msg.encode())
if data:
    data = sock.recv(102400)
    print("SERVER:", data.decode())
```

Подготовил студент группы КС-19-1Б
Дворянских Демид Анатольевич
17.04.2023