Raport 6 – Sieci komputerowe 1

Główne cele tego laboratorium to:

1. zapoznanie się z detalami dotyczącymi przesyłania danych przez sieć – na podstawie języka Python,
2. zauważenie różnicy między działaniem TCP i UDP.

W ramach zadania 1 stworzymy dwa programy – aplikację kliencką oraz serwer zbierający nadsyłane przez klientów rekordy. Komunikację między programami oprzemy na protokole TCP. W zadaniu 2 naszym celem będzie ponowne stworzenie takich programów jak w zadaniu 1, tylko tym razem korzystających z UDP zamiast TCP. W zadaniu 3 i 4 skupimy się na porównaniu działania TCP i UDP

Zadanie 1

Wyobraźmy sobie, że naszym celem jest stworzenie aplikacji zbierającej numery wychodzących połączeń z telefonu (oczywiście za zgodą właściciela). Potrzebujemy do tego dwóch programów:

1. klienta – programu uruchamiającego się na komórce i przesyłającego podsłuchane numery telefonów
2. serwera – programu, który będzie nasłuchiwał na wiadomości przysyłane przez klientów.

Na potrzeby naszego laboratorium skupimy się na samym aspekcie przesyłania danych.

Serwer

Zadaniem serwera jest:

• nasłuchiwanie na wybranym porcie

• przyjmowanie połączeń pochodzących od klienta

• odbieranie wiadomości przychodzących od klienta

• wypisywanie otrzymanych wiadomości na linię poleceń.

Aby stworzyć serwer, do którego będą mogli się podłączać klienci, musimy przede wszystkim nasłuchiwać na wiadomości przychodzące na nasz lokalny adres oraz jakiś konkretny port. Aby to zrobić zaczniemy od stworzeniu socketu (gniazda).

Kod serwera:

```
import socket

s=socket.socket()
```

```
host = "localhost"
port = 7777
s.bind((host,port))
s.listen()
(client_socket, client_address) = s.accept()
print("Connection from ", str(client_address))

while True:
   data = client_socket.recv(1024).decode("utf-8")
   if not data:
      break
   print(data)

client_socket.close()
s.close()
```

Klient

Zadaniem programu klienckiego jest:

• połączenie się z serwerem

• odczytanie tekstu wpisanego przez użytkownika

• wysłanie odczytanego tekstu do serwera.

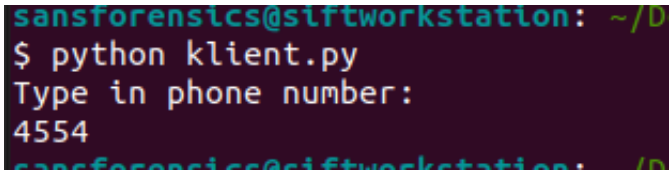
Kod klienta:

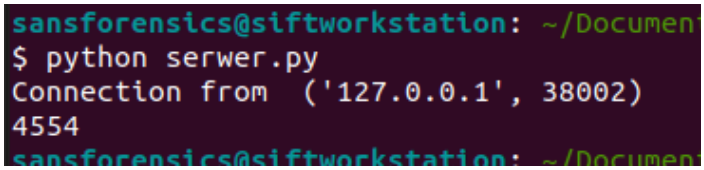
```
import socket

s=socket.socket()
 #server address
dest_host = "localhost"
dest_port = 7777
s.connect((dest_host, dest_port))

message=input("Type in phone number: \n")
s.send(message.encode("utf-8"))

s.close()
```

1.3 Zadanie 1a

Wykorzystując opisany powyżej kod stwórz program serwera oraz klienta realizujące zadania opisane w powyższych podpunktach. Testem na sprawdzenie poprawności działania programu jest przesłanie wiadomości z klienta do serwera.

Przeprowadzone testy:





Opisz (w punktach) proces tworzenia i wykorzystywania socketa z perspektywy serwera i z perspektywy klienta. Wyszczególnij które funkcje używane w naszym programie są odpowiedzialne za dane punkty

Perspektywa serwera	Perspektywa klienta
1. Utworzenie socketa (socket.socket()).	1. Utworzenie socketa (socket.socket())
2. Przypisanie socketowi adresu i portu (socket.bind((host, port))).	2. Nawiązanie połączenia (socket.connect().
3. Nasłuchiwanie na wcześniej podanym adresie i porcie (socket.listen()).	3. Wysłanie danych do socketa (socket.send()).
4.Nawiązanie (akceptacja) połączenia i utworzenie socketa klienta (socket.accept())	4.Odebranie danych z socketa (socket.recv()).
5.Odebranie danych z socketa klienta (socket.recv() – tutaj na obiekcie client socket).	5. Zamknięcie socketa (socket.close())
6.(Jedna z możliwości) Wysłanie danych do socketa klienta (socket.send() – tutaj na obiekcie client socket).	
7. Zamknięcie socketa klienta (socket.close() – na obiekcie client socket).	
8. Zamknięcie socketa serwera (socket.close()).	

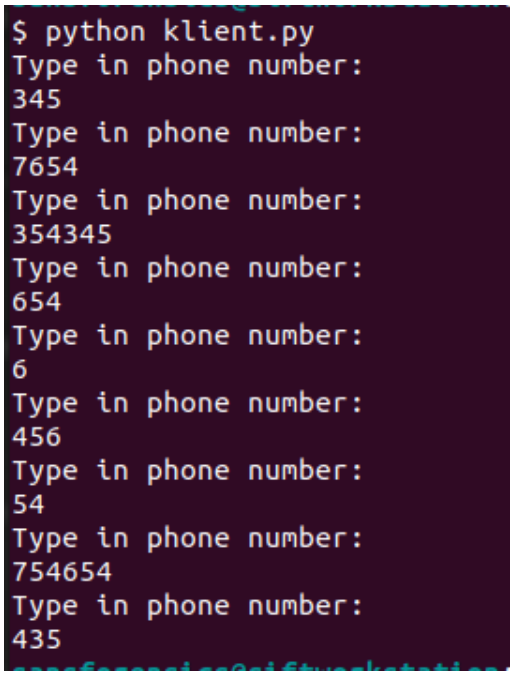
Zadanie 1b

Zmodyfikuj:

• klienta – aby etap wczytywania i wysyłania wiadomości był powtarzany 10-krotnie (użytkownik może podać 10 razy wiadomość)

• serwer – aby zbierał do listy wiadomości nadsyłane przez klienta.

Wpisane numery:



Output z serwera:

Zadanie 2 - UDP

Na początku przyjrzyjmy się dokładniej tworzeniu socketa. Możemy wtedy podać 3 opcjonalne argumenty:

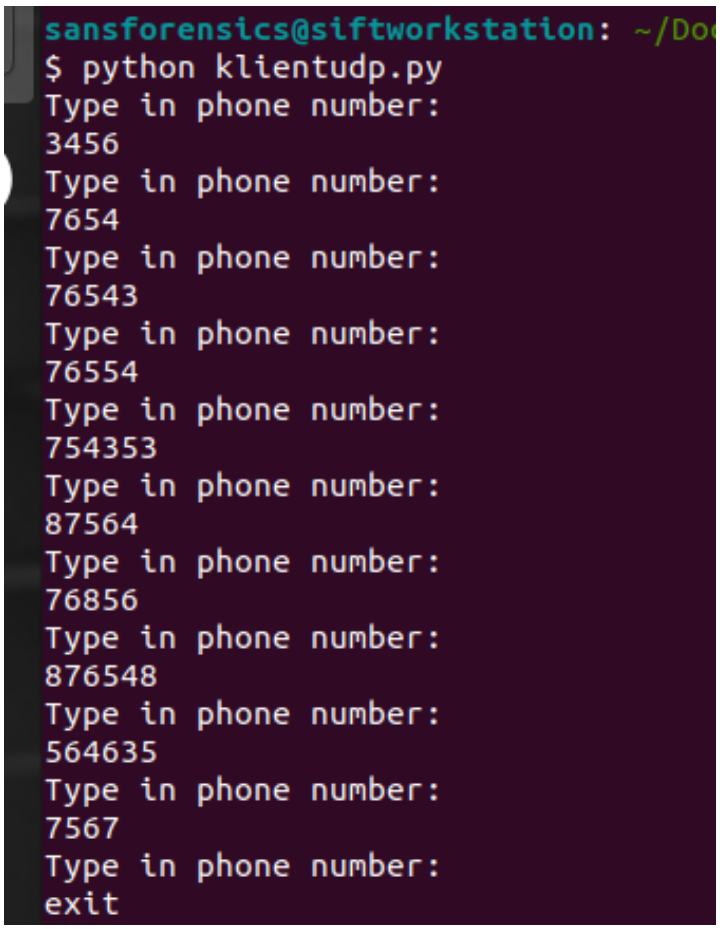
• socket_family – rodzina protokołów, które będą używane w połączeniu z socketem. Tutaj wystarcza nam domyślna wartość (AF_INET) oznaczająca używanie IPv4.

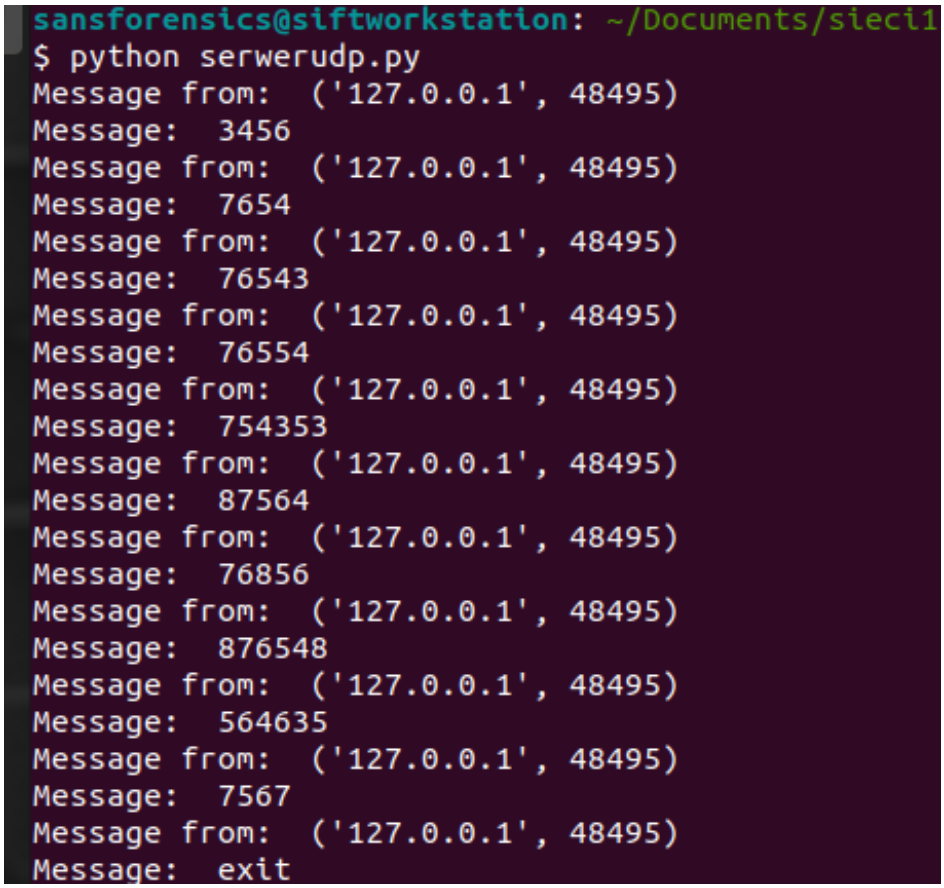
• type – typ komunikacji za pomocą socketa. Dla IPv4 jest to zasadniczo wybór pomiędzy TCP a UDP. Domyślnie wartością jest socket.SOCK_STREAM (TCP). Aby używać UDP możemy podać wartość: socket.SOCK_DGRAM.

• protocol – tym argumentem nie musimy się tutaj zajmować, pozostawiamy zawartość domyślną.

W poprzednim zadaniu stworzyliśmy serwer i klienta korzystając z protokołu TCP. Tym razem chcielibyśmy użyć UDP. Podobnie jak w zadaniu 1 zaczynamy od stworzenia socketa (zwróć uwagę, że podajemy wartość parametru type)

Przeprowadzone testy:





Perspektywa serwera	Perspektywa klienta
1. Utworzenie socketa (socket.socket()).	1.Utworzenie socketa (socket.socket()).
2. Przypisanie socketowi adresu i portu (socket.bind((host, port))).	2. Wysłanie danych do socketa (socket.sendto()).
3. Odebranie danych z socketa (socket.recvfrom()).	3. Odebranie danych z socketa (socket.recvfrom()).
4.Wysłanie danych do socketa (socket.sendto()).	4. Zamknięcie socketa (socket.close()).
5.Zamknięcie socketa (socket.close()).	

Zadanie 3 - porównanie

Porównaj krótko przygotowany przez Ciebie opis procesu działania socketów z zadania 1a do tego z zadania 2a pod kątem następujących pytań

• Jakie są różnice pomiędzy oboma procesami?

• Z czego wynikają te różnice?

Podstawową, a zarazem główną różnicą między wykorzystaniem socketów do komunikacji za pomocą TCP, a UDP jest potrzeba utrzymywania połączenia w

przypadku protokołu TCP. Serwer prowadzi nasłuchiwanie na sockecie(na wcześniej podanym adresie i porcie(socket.listen()).), akceptuje połączenie od klienta (socket.accept()), a po przeprowadzonej komunikacji zostaje zamknięty socket klienta. W przypadku UDP, po utworzeniu socketa dane są natychmiastowo odbierane, nie tworzy się socketu dla klienta, nie zamyka go, a jedynie wykorzystuje otrzymany podczas odbierania danych adres oraz port docelowy i to tylko wtedy, kiedy serwer musi odesłać komunikat zwrotny. Różnice te wynikają ze standardu komunikacji wykorzystywanego przez protokoły:

TCP jest połączeniowy (Gdy maszyna A wysyła dane do maszyny B, maszyna B jest powiadamiana o nadejściu danych, a następnie przesyła potwierdzenie odbioru) podczas gdy UDP jest bezpołączeniowy (gdy maszyna A wysyła pakiety do komputera B, strumień jest niekierunkowy. Oznacza to, że transmisja danych odbywa się bez ostrzeżenia odbiorcy (maszyna B), a odbiorca otrzymuje dane bez konieczności potwierdzania (maszyna A)). Z tego wynika, że UDP nie wymaga nawiązywania, utrzymywania, ani zakańczania połączeń, a jedynie odbiera otrzymane na dany socket dane.

Zadanie 4 – Porównanie na podstawie wykładu

Celem tego zadania jest powtórzenie sobie podstawowych różnic między TCP i UDP. To zadanie można oprzeć w całości na informacjach podanych podczas wykładu.

Porównaj TCP i UDP pod kątem:

- tworzenia połączenia,
- niezawodności,
- potwierdzania dostarczenia wiadomości,
- prędkości działania,
- przypadków użycia.

	TCP	UTP
Tworzenia połączenia	Połączeniowy	Bezpołączeniowy
Niezawodność	Niezawodny	Zawodny
Potwierdzanie dostarczenia wiadomości	Potwierdzenie dostarczenia danych	Brak potwierdzenia dostarczenia danych
Prędkość działania	Wolniejszy, ważniejsza niezawodność	Szybszy, ważniejsza prędkość
Przypadki użycia	Komunikacja dwukierunkowa, duża ilość danych, wymaganie poprawności danych (np. strony internetowe, autoryzacja użytkownika)	Komunikacja jednokierunkowa, mniejsze ilości danych, szybkość ważniejsza od poprawności (np. streaming multimediów, wideo, telekonferencje)