Michał Matak
nr indeksu: 304071

Paweł Müller
nr indeksu: 304080

Jakub Robaczewski
nr indeksu: 304119

Grzegorz Rusinek
nr indeksu: 304083

Programowanie Sieciowe

Laboratorium nr 2

Zestaw programów klient – serwer wysyłające datagramy TCP

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH

10.12.2021

Część pierwsza - zadanie 2.1

Cel ćwiczenia

Celem ćwiczenia było stworzenie programów w języku C/C++ oraz Python, służących do komunikowania się między sobą przy wykorzystaniu gniazd sieciowych oraz protokołu TCP, zarówno przy użyciu adresów IPv4, jak i IPv6. Należy zaznaczyć, że programy te z założenia mają potrafić komunikować się ze sobą niezależnie od języka, w jakim zostały napisane – a więc serwer napisany w Pythonie potrafi komunikować się z klientem napisanym w C i vice versa.

Implementacja

W każdym z języków stworzone zostały dwa programy – jeden z nich jest implementacją klienta, a drugi serwera. Klient wysyła wiadomości do serwera – najpierw posługując się adresem IPv4, następnie IPv6. Serwer przez cały czas nasłuchuje na połączenia na dwóch gniazdach – jedno z nich obsługuje połączenia IPv4, drugie – IPv6. Oba gniazda nasłuchują na tym samym porcie. Serwer napisany w języku C++ działa na jednym wątku, a wyborem obsługiwanego gniazda zajmuje się funkcja `select()`. Serwer w języku Python został natomiast stworzony jako program dwuwątkowy, gdzie jeden z wątków nasłuchuje na połączenia z adresów IPv4, natomiast drugi – z adresów IPv6.

Zarówno klient, jak i serwer informują na bieżąco o przesyłanych i odbieranych wiadomościach poprzez stosowne komunikaty wysyłane na standardowe wyjście.

Wnioski

Mechanizm gniazd umożliwia relatywnie prostą realizację komunikacji sieciowej między dwoma programami. Dzięki temu, że programy korzystają z zapewnionej implementacji gniazd BSD, programista nie tylko może skorzystać z gotowych rozwiązań implementujących gniazda, ale nie musi się też martwić o niskopoziomowe różnice w formie zapisu poszczególnych zmiennych, ponieważ zajmują się tym za niego poszczególne warstwy modelu ISO/OSI.

Chcąc zrealizować program obsługujący kilka gniazd jednocześnie, nie musimy realizować obsługi każdego z gniazd w osobnym procesie, ani nawet w osobnym wątku – używane języki zapewniają konstrukcje pozwalające obsługiwać wiele gniazd w jednym wątku. Jedną z takich konstrukcji jest funkcja `select()`, która pozwala nasłuchiwać na wielu gniazdach równocześnie, nie tworząc przy tym ogromnej ilości wątków oczekujących na odbiór komunikatów sieciowych. Może to być przydatne szczególnie w sytuacjach, w których komunikaty przychodzą rzadko. Nawet jeżeli zdarzyłoby się, że w trakcie obsługi komunikatu na jednym z gniazd, drugie gniazdo zgłosiłoby komunikat do obsłużenia, to zostanie ono umieszczone w kolejce gniazd oczekujących na obsługę. Zakładając, że komunikaty są odbierane relatywnie rzadko, jest to optymalne rozwiązanie problemu.

Python -> C++

```
robak@RobakPc:/mnt/d/Dokumenty/Studia/Sem 5/PSI/Z2/python$ python3 client.py
IPv4: Trying to connect to: ('localhost', 8888)
IPv4: Sending b'11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111122222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222'
IPv4: Trying to connect to: ('::1', 8888)
IPv6: Sending b'11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111122222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222'
robak@RobakPc:/mnt/d/Dokumenty/Studia/Sem 5/PSI/Z2/python$
```

```
IPv4: Opened socket connection
IPv6: Opened socket connection
IPv4: Connection by 127.0.0.1:40212
IPv4: Received message: 1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111 from 127.0.0.1:4021
IPv4: Received message: 1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111 from 127.0.0.1:4021
IPv4: Received message: 1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111 from 127.0.0.1:4021
IPv4: Received message: 1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111 from 127.0.0.1:4021
IPv4: Received message: 1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111 from 127.0.0.1:4021
IPv4: Received message: 2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222 from 127.0.0.1:4021
IPv4: Received message: 2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222 from 127.0.0.1:4021
IPv4: Received message: 2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222 from 127.0.0.1:4021
IPv4: Received message: 2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222 from 127.0.0.1:4021
IPv4: Received message: 2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222 from 127.0.0.1:4021
IPv4: Connection with 127.0.0.1:40212 ended
IPv6: Connection by ::1:44136
IPv6: Received message: 1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111::1 from ::1:44136
IPv6: Received message: 1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111::1 from ::1:44136
IPv6: Received message: 1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111::1 from ::1:44136
IPv6: Received message: 1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111::1 from ::1:44136
IPv6: Received message: 1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111::1 from ::1:44136
IPv6: Received message: 2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222::1 from ::1:44136
IPv6: Received message: 2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222::1 from ::1:44136
IPv6: Received message: 2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222::1 from ::1:44136
IPv6: Received message: 2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222::1 from ::1:44136
IPv6: Received message: 2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222::1 from ::1:44136
IPv6: Received message: 2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222::1 from ::1:44136
IPv6: Connection with ::1:44136 ended
```

C++ -> Python

```
IPv4: Sending message:
 11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111122222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222
IPv6: Sending message:
 33333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333344444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444
```

```
IPv4: Opened socket connection: <socket.socket fd=3, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 8888)>
IPv6: Opened socket connection: <socket.socket fd=5, family=AddressFamily.AF_INET6, type=SocketKind.SOCK_STREAM, proto=0, laddr=('::1', 8888, 0, 0)>
IPv4: Connected by ('127.0.0.1', 40216)
IPv4: Received message: 1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111[100] from 127.0.0.1:40216
IPv4: Received message: 1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111[100] from 127.0.0.1:40216
IPv4: Received message: 1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111[100] from 127.0.0.1:40216
IPv4: Received message: 1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111[100] from 127.0.0.1:40216
IPv4: Received message: 1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111[100] from 127.0.0.1:40216
IPv4: Received message: 2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222[100] from 127.0.0.1:40216
IPv4: Received message: 2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222[100] from 127.0.0.1:40216
IPv4: Received message: 2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222[100] from 127.0.0.1:40216
IPv4: Received message: 2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222[100] from 127.0.0.1:40216
IPv4: Received message: 2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222[100] from 127.0.0.1:40216
IPv4: Connection with ('127.0.0.1', 40216) ended
IPv6: Connected by ('::1', 44140, 0, 0)
IPv6: Received message: 3333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333[100] from ::1:44140
IPv6: Received message: 3333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333[100] from ::1:44140
IPv6: Received message: 3333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333[100] from ::1:44140
IPv6: Received message: 3333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333[100] from ::1:44140
IPv6: Received message: 3333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333[100] from ::1:44140
IPv6: Received message: 4444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444[100] from ::1:44140
IPv6: Received message: 4444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444[100] from ::1:44140
IPv6: Received message: 4444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444[100] from ::1:44140
IPv6: Received message: 4444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444[100] from ::1:44140
IPv6: Received message: 4444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444[100] from ::1:44140
IPv6: Connection with ('::1', 44140, 0, 0) ended
```

Część druga - zadanie 2.2

Cel ćwiczenia

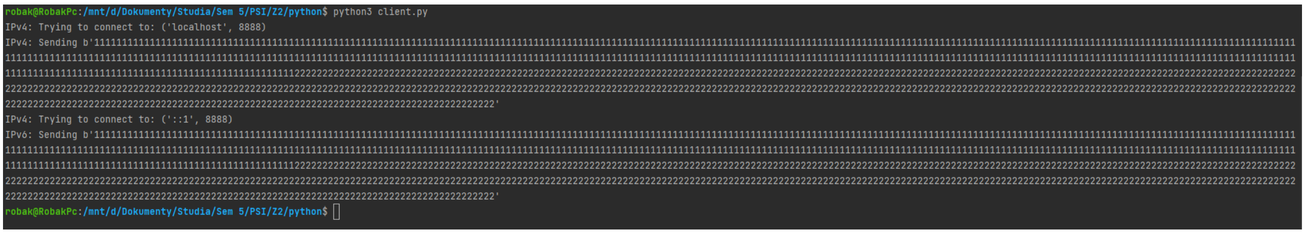
Zadanie polegało na modyfikacji stworzonego programu w celu zapewnienia prawidłowego podziału plików w przypadku wysłania przez klienta danych większych niż bufor odbiorczy serwera.

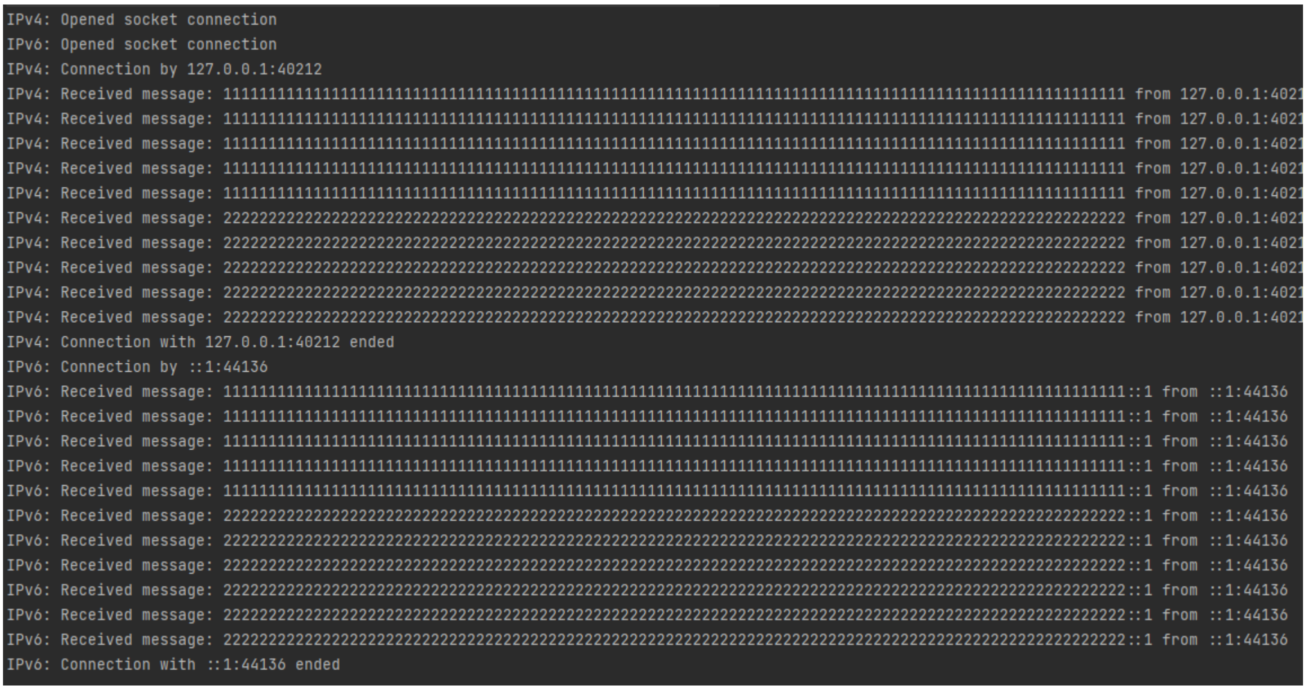
Implementacja

Nasza implementacja klienta w języku Pythonie wykorzystuje funkcję `sendall()`, która automatycznie dzieli dane na odpowiednie segmenty. Dodatkowo należało zmodyfikować funkcję odbioru danych na serwerze, by obsługiwała pobieranie danych w pętli.

Przykład działania

Klient wysyła wiadomość o długości 1000 bajtów, przy czym serwer posiada jedynie 100 bajtowy bufor. Dlatego dane są przesyłane w formie 10 segmentów po 100 bajtów.





Część trzecia - zadanie 2.3

Cel ćwiczenia

Zadanie polegało na rozszerzeniu implementacji programu w Pythonie, by obsługiwało timeouty, dla funkcji `connect()` i `accept()` oraz zrealizowaniu podobnych funkcjonalności w C++ za pomocą funkcji `select()`.

Implementacja w Pythonie

Kody serwera i klienta zostały rozszerzone o funkcje `settimeout()`, ustawioną na przykładowy czas 10 sekund. Po tym czasie serwer przestanie oczekiwać na połączenia z klientem, a klient przestanie czekać na zestawienie połączenia z niedostępnym serwerem. Działanie funkcji można zaobserwować uruchamiając serwer bez klienta (po 10 sekundach bez połączenia serwer rozłączy się) oraz przez próbę połączenia klienta z nieroutowanym adresem 10.0.0.0 (klient po 10 zaprzestanie prób połączenia).

Timeout serwera.

```
C:\Users\kubar\AppData\Local\Programs\Python\Python39\python.exe "D:/Dokumenty/Studia/Sem 5/PSI/Z2/python/server.py"
IPv4: Opened socket connection: <socket.socket fd=300, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 8888)>
IPv6: Opened socket connection: <socket.socket fd=320, family=AddressFamily.AF_INET6, type=SocketKind.SOCK_STREAM, proto=0, laddr=('::1', 8888, 0, 0)>
IPv4: Connection ended: Timeout
IPv6: Connection ended: Timeout

Process finished with exit code 0
```

Timeout klienta.

```
C:\Users\kubar\AppData\Local\Programs\Python\Python39\python.exe "D:/Dokumenty/Studia/Sem 5/PSI/Z2/python/client.py"
IPv4: Trying to connect to: ('10.0.0.0', 8888)
IPv4: Connection ended: Timeout

Process finished with exit code 0
```

Implementacja w C++

Dotychczasowe wymogi dotyczące obsługi adresów IPv4 i IPv6 sprawiły, że dodanie timeoutu do serwera wymagało minimalnych zmian, ze względu na to, że funkcja `select()` była już obecna i wystarczyło przechwycić, kiedy będzie równa 0. W kliencie zmiany dotyczyły przestawienia gniazda w tryb nie-blokujący i zrealizowania oczekiwania przez 10 sekund za pomocą funkcji `select()`. W tym celu stworzono pomocniczy dodatkowy zestaw deskryptorów, który zawiera jedynie deskryptor klienta.

Timeout serwera.

```
Ubuntu-20.04: '/mnt/d/Dokumenty/Studia/Sem 5/PSI/Z2/c/cmake-build-debug/server'
IPv4: Opened socket connection
IPv6: Opened socket connection
No incoming connections. Server timeout.

Process finished with exit code 0
```

Timeout klienta.

```
Ubuntu-20.04: '/mnt/d/Dokumenty/Studia/Sem 5/PSI/Z2/c/cmake-build-debug/client'
IPv4: Trying to connect with: 10.0.0.0:8888
IPv4: Connection timeout

Process finished with exit code 0
```