AKADEMIA GÓRNICZO-HUTNICZA

Dokumentacja do projektu

Prosta lista zadań

z przedmiotu

Programowanie Sieciowe

Elektronika i Telekomunikacja 3 rok

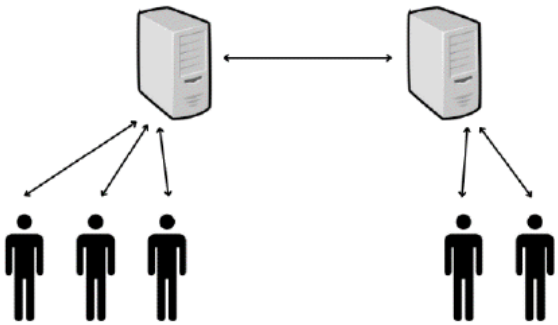
Jakub Kostecki
Szymon Krzyworzeka
Marcel Włosek

prowadzący: mgr. inż. Karol Salwik

22.01.2023

1. Opis projektu

Tematem zrealizowanego projektu było stworzenie komunikatora tekstowego typu klient-serwer, wykorzystując dwa skomunikowane ze sobą serwery. Klienci maja możliwość podłączenia się do jednego z serwerów i tym samym dołączenia się do usługi „chatu”. Wiadomość wysłana przez klienta do serwera do którego jest podłączony jest rozsyłana do wszystkich klientów podłączanych do tego samego serwera jak również do wszystkich klientów połączonych do drugiego serwera.



Rysunek 1- diagram architektury komunikatora

2. Realizacja projektu

Projekt został wykonany przy użyciu języka Python3. W oparciu o funkcje sieciowe stworzono dwa programy. Program serwera działający, jednocześnie do komunikacji z klientami i drugim serwerem, oraz program klienta pozwalający na polaczenie się z serwerem. Wykorzystano protokół komunikacyjny TCP – Transmission Control Protocol, oraz adresacje Ipv4 unicast. Wykorzystanie biblioteki threading umożliwiło realizowanie każdego połączenia w obrębie osobnego wątku. Działanie programów wymagało umożliwienia wielu procesom pracowania jednocześnie

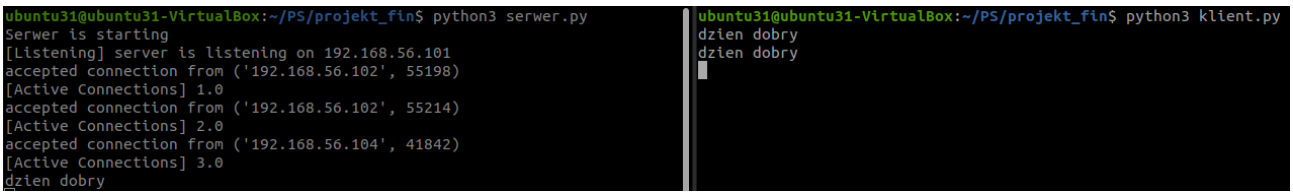
Kodowanie wiadomości będzie wykonywane w programie w sposób następujący: Proces wysyła dwie zakodowane wiadomości z czego pierwsza to strumień bajtów odpowiadający długości wysyłanej wiadomości właściwej uzupełniony przestrzenią bitową do długości 64. Druga wiadomość to już właściwa zakodowana wiadomość pobrana z klawiatury klienta.

3. Prezentacja działania programu

Prezentacja została wykonana na przykładzie polaczenie jednego klienta do pierwszego serwera, dwóch klientów do drugiego serwera, oraz nawiązaniu polaczenia pomiędzy serwerami.

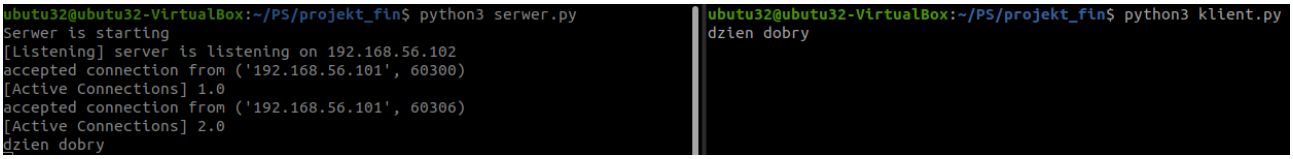
Pierwsza maszyna wirtualna

Konsole serwera o adresie .101 i klienta podłączony do serwera .102:



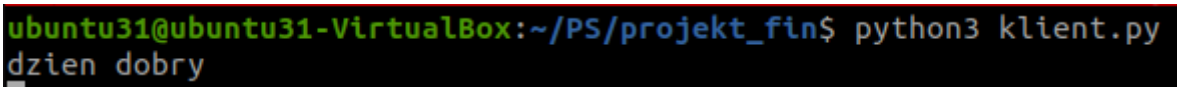
Druga maszyna wirtualna

Konsole serwera o adresie .102 i klienta podłączony do serwera .101:



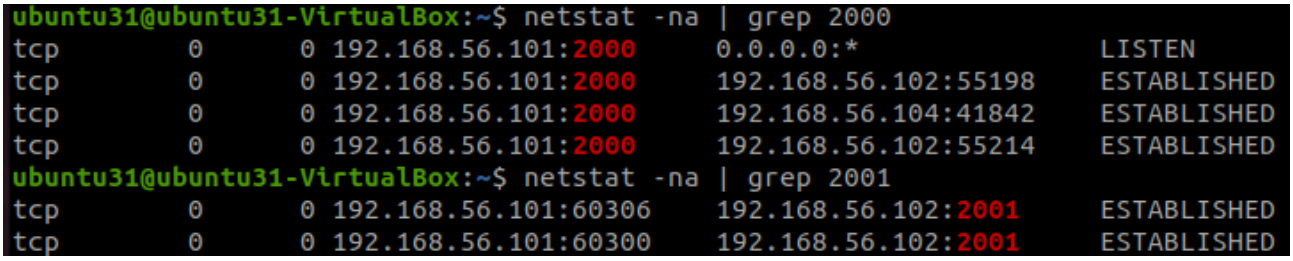
Trzecia maszyna wirtualna

Konsola klienta podłączonego do serwera o adresie .101

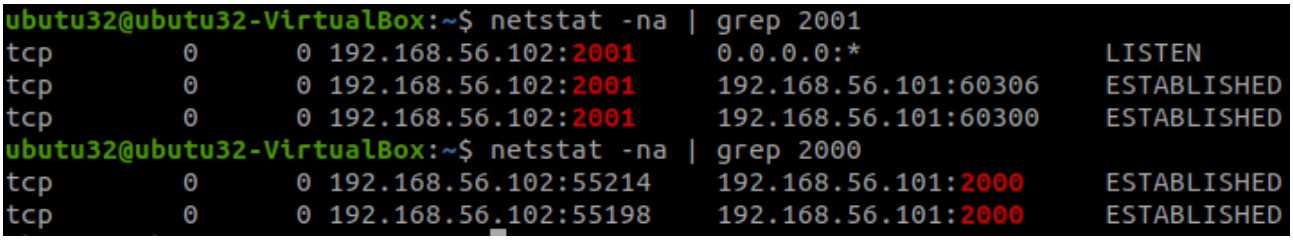


Po wysłaniu wiadomości z klienta podłączonego do serwera .102, została ona przekazana do wszystkich klientów wliczając w to nadawcę.

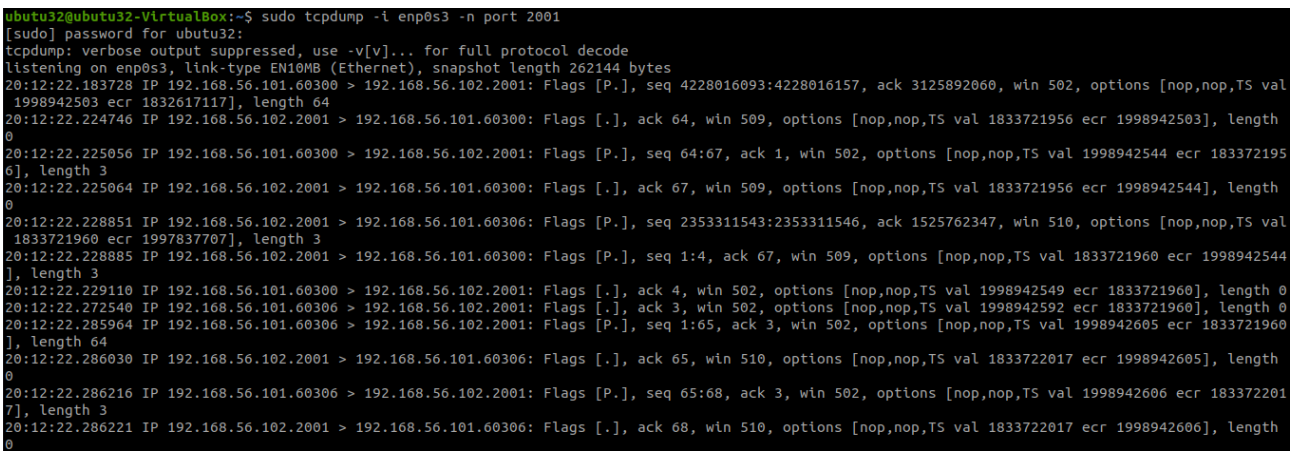
Stany portów na pierwszej maszynie wirtualnej:



Stany portów na drugiej maszynie wirtualnej:



Przepływ pakietów na drugiej maszynie podczas wysłania komunikatu:



4. Wykorzystane bibliotek i funkcje

- Biblioteka socket
    1. Funkcja socket() – tworzenie gniazd w programie serwera i klienta
    2. Funkcja gethostbyname() – pobieranie adresu IP
    3. Funkcja gethostname() – pobieranie nazwy urządzenia
    4. Funkcja bind() – powiazanie z gniazdem
    5. Funkcja connect() – nawiązywanie połącznia
    6. Funkcja listen() – nasłuchiwanie na serwerze
    7. Funkcja accept() – akceptacja polaczeń przez serwer
    8. Funkcja send() – wysyłanie informacji
    9. Funkcja recv() – odbieranie informacji
- Biblioteka threading

    Funkcja decode() – umożliwiająca zamienienie stringa bitowego na stringa w kodzie Unicode

    Funkcja encode() – umożliwiająca zakodowanie zmiennej string jako strumienia bitowego

    Funkcja Thread() – umożliwia wielowątkowość

5. Napotkane problemy i ich rozwiązania

Problemem okazała się implementacja zakładanego w fazie konspektu protokołu SCTP. Mimo usilnej próby postawienia gniazda IPPROTO_SCTP lub SOCK_SEQPACK nie udało się znaleźć sposobu, aby kompilator nie wyrzucał błędu z nieobsługiwaną rodziną adresową. Zdecydowano się zatem zrealizować połączenie między serwerami za pomocą gniazd strumieniowych.

Podczas pisania funkcjonalności komunikacji serwera z serwerem, ponieważ serwery obsługują się nawzajem tak samo jak obsługują klientów. Napotkaliśmy na problem w którym serwer odbierający wiadomość od drugiego serwera wysyła ją do wszystkich połączonych ze sobą klientów. Ponieważ w liście klientów znajduje się też serwer od którego otrzymano wiadomość, wiadomość jest wysyłana z powrotem. Sposobem na rozwiązanie takiego problemu okazało się być wprowadzenie zmiennej, która trzyma treść poprzednio otrzymanej wiadomości. Nie jest to rozwiązanie idealne z dwóch powodów:
Nie jest możliwe wysłanie wiadomości o takiej samej treści dwa razy z rzędu oraz serwer wysyła wiadomość o jeden raz za dużo np. w przypadku, gdy klient połączony z serwerem o adresie x.x.x.12 wyśle wiadomość i serwer x.x.x.12 wyśle tę wiadomość do serwera x.x.x.34 to serwer x.x.x.34 wysyła tę wiadomość z powrotem, mimo że serwer x.x.x.12 ją zignoruje.

6. Dalszy rozwój i ulepszenia

Implementacja mechanizmów zabezpieczających takich jak, bezpieczne rozłączanie klientów w momencie kiedy serwer nagle przestanie działać.
Na ten moment klient może bezpiecznie rozłączyć się z serwerem jedynie poprzez wysłanie wiadomości o treści DISCONNECT, nie jest to jednak wygodne

Wdrożenie interfejsu użytkownika po stronie klienta. Możliwość wybrania pseudonimu który będzie wyświetlany podczas wysyłania wiadomości na komunikatorze.