**Gniazda TCP i UDP**

**Laboratorium 1**

**Wiktor Zmiedak**

**Obraz zawierający tekst, zrzut ekranu, oprogramowanie, numer

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.Zadanie 1**

W przeprowadzonej analizie za pomocą Wireshark zarejestrowano przebieg komunikacji TCP pomiędzy klientem a serwerem. Zaobserwowano standardowy three-way handshake, czyli trójfazowe ustanowienie połączenia TCP:

* SYN – pakiet inicjalizujący połączenie wysłany przez klienta,
* SYN-ACK – odpowiedź serwera potwierdzająca przyjęcie połączenia,
* ACK – finalne potwierdzenie od klienta, które kończy proces nawiązywania sesji.

Po ustanowieniu połączenia widoczna jest wymiana danych (PSH, ACK), w której przesyłane są konkretne pakiety. Następnie dochodzi do zamykania połączenia:

* FIN, ACK – pakiet zamknięcia sesji przez jedną ze stron,
* ACK – potwierdzenie zamknięcia przez drugą stronę.

W trakcie trwania połącznie możemy zauważyć zmianę wartości pól Seq oraz Ack.

**Zadanie 2**

**Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.**

W przeprowadzonej analizie komunikacji między klientem a serwerem UDP za pomocą Wireshark zaobserwowano poprawną wymianę pakietów.

* Klient wysłał pakiet UDP z portu 43634 na port 8080 zawierający 5 bajtów danych,
* Serwer odpowiedział pakietem UDP z portu 8080 na port 43634 klienta, co potwierdza mechanizm echo, gdzie serwer odsyła otrzymaną wiadomość.

W analizie pakietów nie występuje nawiązanie i zamykanie połączenia, jak w TCP, ponieważ UDP działa bezstanowo, a więc każda wiadomość jest wysyłana i odbierana niezależnie.

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.**

Dodatkowo przeprowadzono analizę przy pomocy netstat. Jak widać na załączonym screenie przed uruchomieniem serwera w tabelce widoczne były jedynie standardowe nasłuchania usługi UDP. Po uruchomieniu serwera na porcie 8080 pokazał się nowy wpis 0.0.0.0:8080 wskazujący na to że serwer UDP rozpoczął swoje nasłuchiwanie.

**Zadanie 3**

Co robi serwer:

* Nasłuchuje na podanym porcie przy użyciu gniazda UDP,
* Odbiera dane od klienta za pomocą recvfrom().
* Sprawdza, czy odebrana wiadomość jest liczbą całkowitą oraz czy jest palindromem,
* Odsyła do klienta odpowiedź "TAK, to palindrom" lub "NIE, to nie jest palindrom",
* Wysłanie pustego datagramu przez klienta kończy działanie serwera.

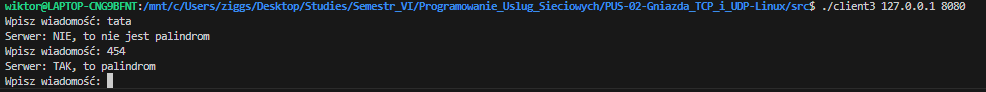
Co robi klient:

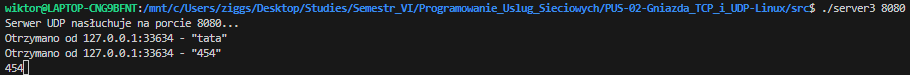
* Tworzy gniazdo UDP i łączy się z serwerem za pomocą connect(),
* Pobiera dane od użytkownika i wysyła je do serwera przy pomocy send(),
* Oczekuje na odpowiedź serwera i wypisuje ją na ekranie,
* Wysłanie pustego pakietu kończy działanie programu.

Co robi libpalindrome:

* Funkcja is\_palindrome() sprawdza, czy podany ciąg znaków jest palindromem,
* Ignoruje białe znaki oraz wiodące zera,
* Jeśli ciąg zawiera litery – zwraca błąd.

Wywołanie:

****

****

**Zadanie 4**

Tworzenie gniazda TCP:

* socket() – tworzy gniazdo serwera,
* bind() – wiąże gniazdo z adresem i portem,
* listen() – umożliwia serwerowi nasłuchiwanie połączeń.

Pętla główna serwera:

* select() monitoruje serwer i klientów, aby sprawdzić, kto chce wysłać dane,
* Jeśli nowy klient się łączy to accept() dodaje go do listy klientów,
* Jeśli klient wysyła wiadomość to serwer odbiera ją i przesyła do klientów za pomocą send(),
* Jeśli klient rozłącza się to serwer usuwa go z listy aktywnych klientów.

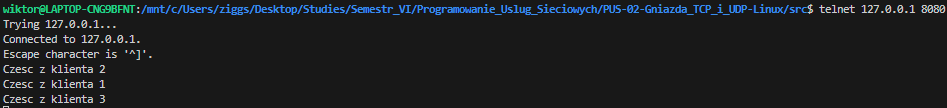
Obsługa klientów:

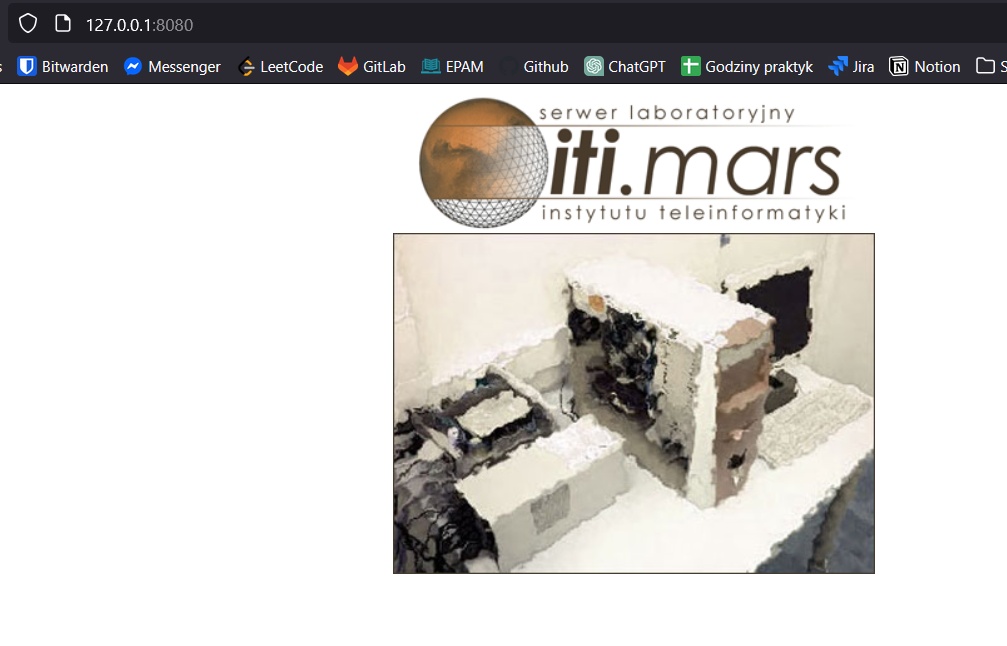
* FD\_SET() – dodaje gniazda klientów do listy monitorowanych deskryptorów,
* FD\_ISSET() – sprawdza, czy klient wysłał dane,
* read() – odbiera wiadomość od klienta,
* send() – przesyła wiadomość do pozostałych klientów,
* Jeśli klient się rozłączy to serwer zamyka jego gniazdo i usuwa go z listy.

Wywołanie:

**Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.**

****

**Zadanie 5**