Politechnika Gdańska

Architektura systemów komputerowych

Laboratorium

Sprawozdanie z ćwiczenia 4

Autorzy:
Kewin Trochowski
Michał Zarzycki

Kwiecień 23, 2024

1 Wprowadzenie

W ramach projektu wykonano symulację transmisji szeregowej, implementując prosty system komunikacji pomiędzy aplikacją serwerową a kliencką. System został zrealizowany w języku Python z wykorzystaniem biblioteki PyQt6 dla interfejsu graficznego oraz modułu `socket` do obsługi połączeń sieciowych. Projekt skupia się na prezentacji sposobu kodowania i dekodowania wiadomości, ustanawianiu połączenia oraz obsłudze interfejsu użytkownika.

2 Struktura projektu

Projekt składa się z dwóch głównych komponentów:

1. **Aplikacja serwerowa (`ServerApp`)** – nasłuchuje na określonym adresie IP i porcie, odbiera połączenia od klientów, dekoduje otrzymane wiadomości i wyświetla je w interfejsie graficznym.
2. **Aplikacja kliencka (`ClientApp`)** – łączy się z serwerem, wysyła zakodowane wiadomości po ich odpowiednim przefiltrowaniu (np. cenzura przekleństw), a także pozwala na interakcję z użytkownikiem poprzez graficzny interfejs użytkownika.

3 Kodowanie i dekodowanie wiadomości

Kluczowym aspektem symulacji jest metoda kodowania i dekodowania wiadomości. Kodowanie odbywa się po stronie klienta, gdzie każdy znak wiadomości jest przekształcany do postaci binarnej, a następnie obracany i opatrywany dodatkowymi bitami kontrolnymi:

```
def encode(self, text):
    encoded = ""
    for char in text:
        ascii_value = ord(char)
        bit_string = '0' + format(ascii_value, '08b')[::-1] + '11'
        encoded += bit_string
    return encoded
```

Dekodowanie odbywa się po stronie serwera. Odbiera on ciąg bitów, interpretuje go zgodnie z ustalonym schematem, a następnie odtwarza oryginalną wiadomość:

```
def decode(self, bit_string):
    chars = []
    for i in range(0, len(bit_string), 11):
```

```
        byte = bit_string[i+1:i+9]
        ascii_value = int(byte[::-1], 2)
        chars.append(chr(ascii_value))
    return ''.join(chars)
```

4 Ustanawianie połączenia

Aplikacja serwerowa i kliencka korzystają z modułu `socket`, aby nawiązać połączenie przez TCP/IP. Serwer nasłuchuje na ustalonym adresie i porcie, a klient próbuje się połączyć, używając tych samych danych:

- **Serwer:**

  $\texttt{self.server}_{s}ocket.bind((self.ip_{a}ddress, self.port))$

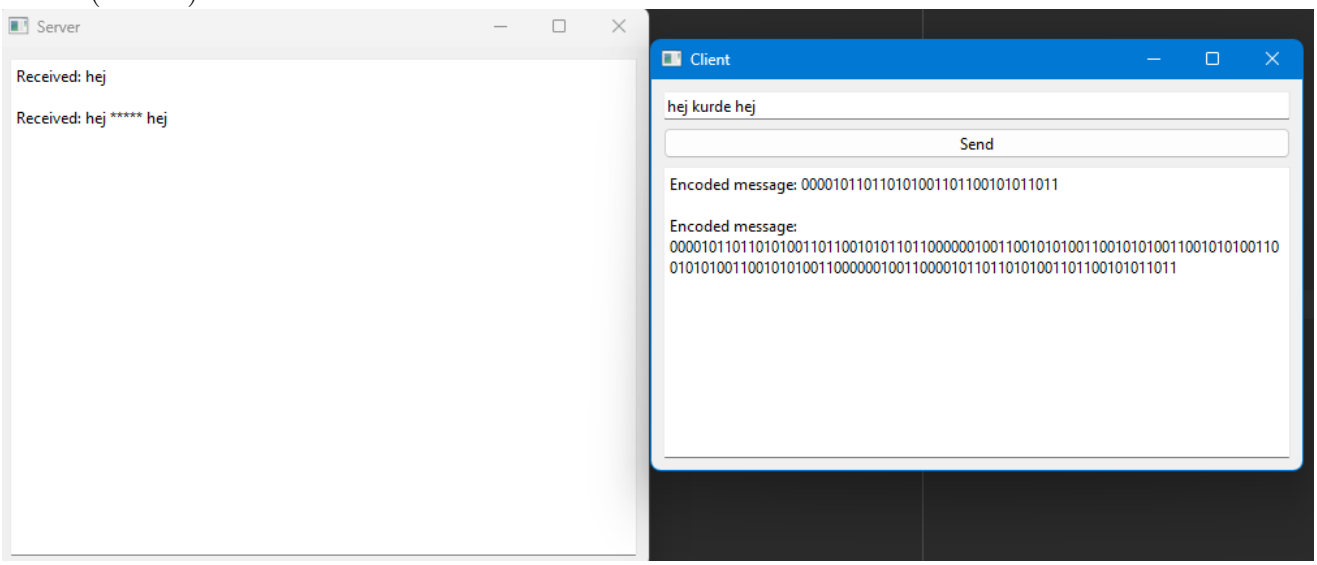
  $\texttt{self.server}_{s}ocket.listen(1)$

- **Klient:**

$\texttt{client}_{s}ocket.connect((self.ip_{a}ddress, self.port))$

5 Interfejs użytkownika

Zarówno klient, jak i serwer posiadają prosty interfejs graficzny stworzony z wykorzystaniem PyQt6. Umożliwia on użytkownikom wprowadzanie adresu IP i portu poprzez okno dialogowe, a także interakcję z aplikacją, taką jak wysyłanie wiadomości (klient) oraz ich odbiór i wyświetlanie (serwer).



6 Podsumowanie

Projekt demonstruje efektywne wykorzystanie socketów w Pythonie do symulacji transmisji szeregowej, a także integrację z PyQt6 dla interaktywnego interfejsu użytkownika. Użytkownik może w czasie rzeczywistym obserwować proces kodowania, wysyłania, odbierania i dekodowania wiadomości, co stanowi praktyczną demonstrację komunikacji w sieci TCP/IP.