

**Przemysłowe Sieci Informacyjne**

**Laboratorium 4 - Ethernet**

09.06.2020

Oleksandr Burkotenko (176226)

**Opis ćwiczenia:** wykorzystując zdobytą wiedzę oraz opierając się na materiałach dostępnych w sieci, zaprezentowałem działający mechanizm komunikacji wykorzystujący standard Ethernet TCP oraz UDP.

**Przebieg ćwiczenia:** najpierw zostały pobrane środowisko Visual Studio Code oraz Node.js . Musiałem dalej stworzyć działający kod w JS, który pozwala na komunikację typu serwer-klient. Pliki utworzone zostały w jednym folderze oraz zastosowano było terminal wbudowany w VSC. Z powodu tego, że mam w dostępie wyłącznie 1 komputer badania przeprowadzone, więc mogłoby to spowodować jakieś błędy w trakcie ćwiczenia. Dlatego konfiguracja została przeprowadzona na jednym komputerze.

Także należało uważać aby powoływać najpierw Serwer, a dopiero po nim Klient. Wykonałem to w ten sposób, że zrobiłem 2 oddzielne okna i 2 terminala, gdzie na pierwszym był napisany kod Serwera, a na drugim kod Klienta.



**Rys. 1:** sposób wywołania w terminalu (np dla TCP).

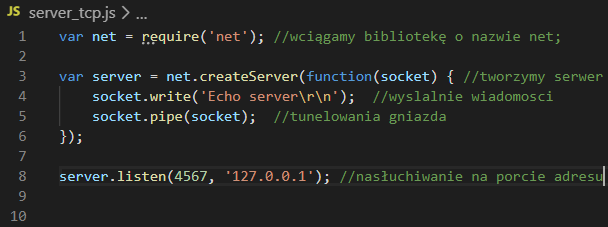
**Protokół TCP:**

Najpopularniejszym protokołem warstwy transportowej jest TCP. Podczas transmisji danych, TCP zestawia połączenie pomiędzy komunikującymi się stronami przez zainicjowanie tzw. sesji. TCP jest protokołem niezawodnym , w którym odbiorca potwierdza otrzymanie każdej wiadomości. Wszystkie wiadomości dostarczane są w takiej samej kolejności, w jakiej zostały wysłane.

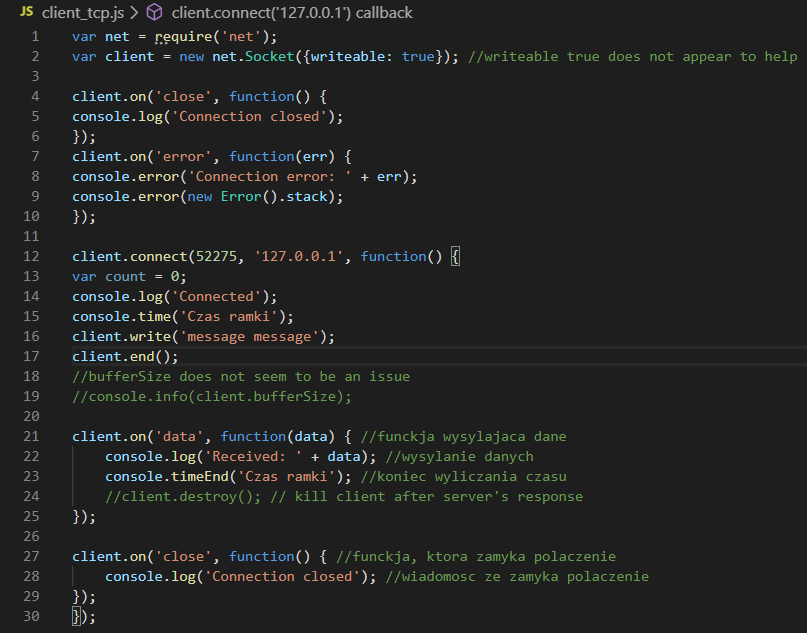
Wszystkie cechy wymienione powyżej są zapewniane przy warstwę TCP. Oznacza to, że TCP może współdziałać z innymi, bardziej zawodnymi protokołami niższych warstw i nie powinno to afektować komunikacji z perspektywy warstwy aplikacji.

W czasie wysyłanie danych, TCP zapewnia, że wszystkie wiadomości zostały dostarczone do miejsca przeznaczenia. Odbiorca testuje każdy otrzymany pakiet pod kątem błędów transmisji (poprzez wyliczanie sumy kontrolnej danych). Jeśli wiadomość jest poprawna, odbiorca wysyła potwierdzenie do nadawcy. Jeśli nadawca nie otrzyma potwierdzenia w przeciągu określonego (konfigurowalnego) czasu, to ponownie wysyła zagubiony pakiet.

Po kilku nieudanych próbach, TCP zakłada, że odbiorca jest nieosiągalny i informuje warstwę aplikacji, że transmisja zakończyła się niepowodzeniem.



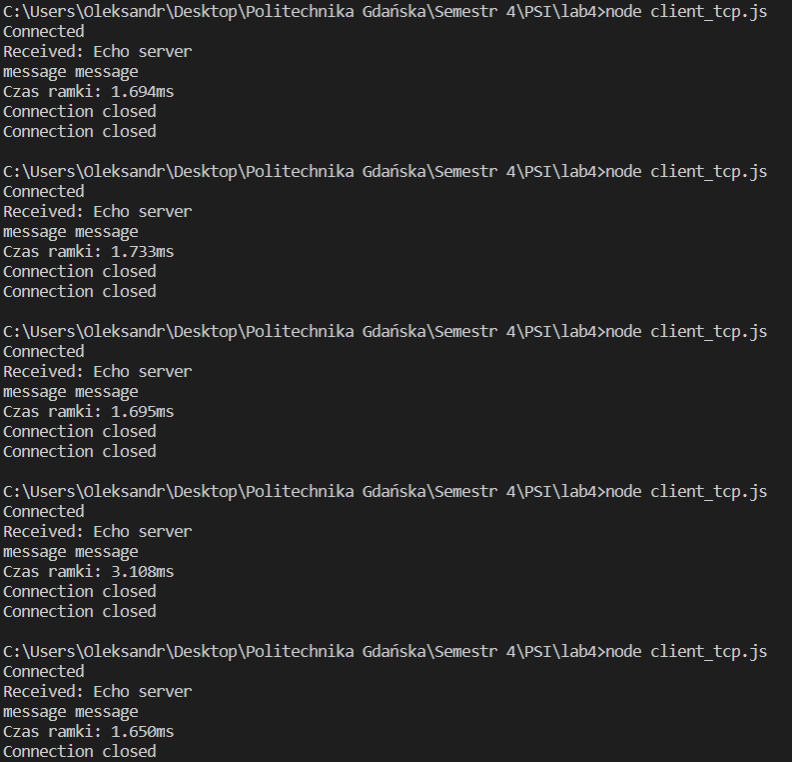
**Rys. 2:** kod dla Serwera oparty na Node.js (TCP).



**Rys. 3:** kod Klienta oparty na Node.js (TCP).



**Rys. 4:** wywołanie serwera w terminale.

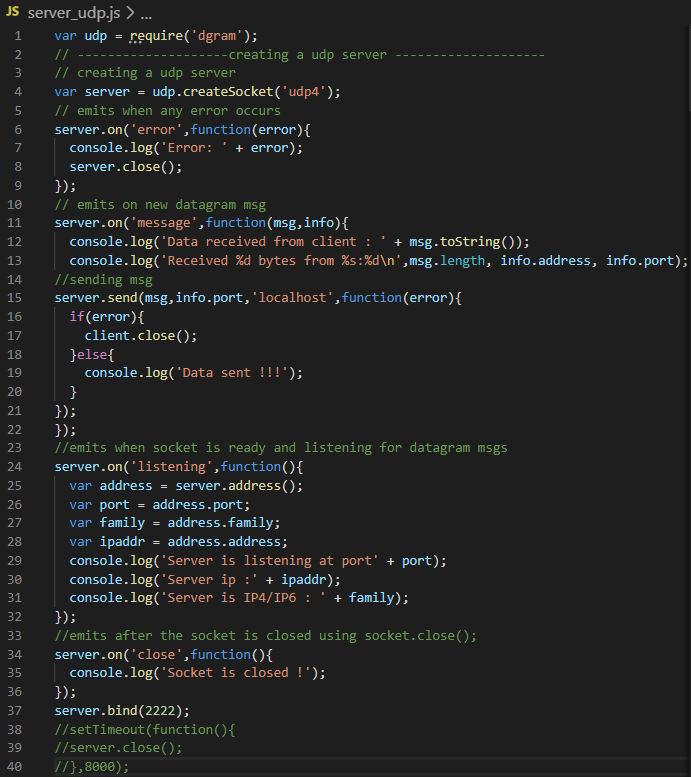


**Rys. 5:** przesyłanie “message message” z czasem działania.

**Protokół UDP:**

Drugim popularnym protokołem używanym w warstwie transportowej jest UDP. Jest to prostszy protokół, w którym komunikacja odbywa się bez nawiązywania żadnego stałego połączenia pomiędzy aplikacjami. Wszystkie pakiety wysyłane są niezależnie od siebie. Dzięki swojej prostocie UDP jest szybsze niż TCP. Z drugiej jednak strony, nie zapewnia takiej niezawodności działania jak TCP. Przede wszystkim UDP nie gwarantuje, że wiadomości rzeczywiście dotarły do odbiorcy. UDP nie dostarcza pakietów w takiej samej kolejności, w jakiej zostały one wysłane. Ciężar uporządkowania otrzymywanych wiadomości i sprawdzenia czy nie nastąpiły błędy transmisji spoczywa na otrzymującej je aplikacji.

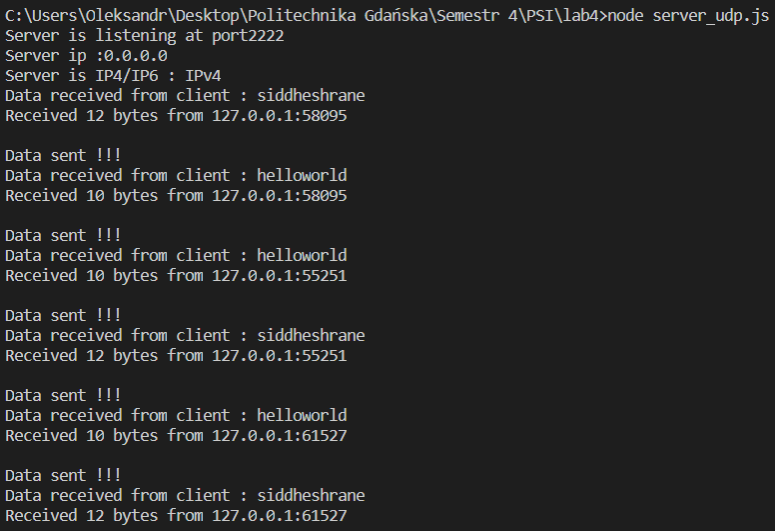
UDP jest preferowane jeśli przesyłane pakiety danych są nieistotne lub komunikacja musi odbywać się z wyjątkowo dużą prędkością. Przykładowo UDP jest używane do przesyłania zapytań DNS (z powodu bardzo dużej liczby zapytań kierowanych do relatywnie niewielu serwerów DNS). UDP jest używane również podczas transmisji audio i video, gdzie utrata pewnej liczby pakietów nie jest bardzo uciążliwa dla odbiorcy.



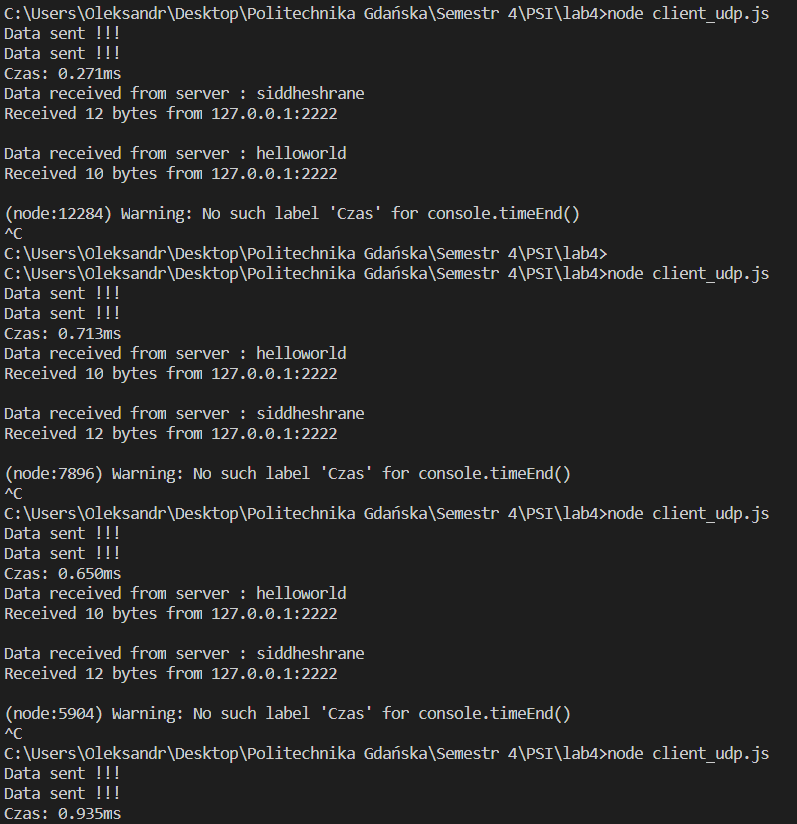
**Rys. 6:** kod dla Serwera oparty na Node.js (UDP).



**Rys. 7:** kod Klienta oparty na Node.js (UDP).



**Rys. 8:** zachowanie Serwera w protokole UDP.



**Rys. 9:** zachowanie klienta w protokole UDP.

**Wnioski:**

1. Szybszą metodą do przesyłania danych okazała się metoda poprzez protokół UDP. Najmniejsze opóźnienie pakietu w tym protokole wyniosło **0.271ms**, a w protokole TCP - **1.650ms**.
2. Głównie w trakcie sprawozdania napotkałem tylko błędy związane z tym że wykorzystywałem localhost i robiłem wszystko poprzez jeden komputer - bardziej wrażliwa na to była metoda TCP. Udałoby się te błędy minąć używając np. inny PC.
3. Zdecydowanie mogę powiedzieć że najtrudniejszą częścią tego ćwiczenia jest napisanie kodu oraz możliwe błędy przy konfiguracji połączeń.