POLITECHNIKA POZNAŃSKA

WYDZIAŁ ELEKTRYCZNY

Instytut Automatyki, Robotyki i Inżynierii Informatycznej

Tomasz Adamczyk

Jacek Eichler

Sprawozdanie z zajęć laboratoryjnych

PROGRAMOWANIE SIECIOWE – PROTOKOŁY BINARNE

30 listopada 2018

1. TREŚĆ ZADANIA.

Temat: Komunikacja pomiędzy klientami poprzez serwer (2:1), w oparciu o autorski protokół binarny.

Protokół:

- połączeniowy,
- wszystkie dane przesyłane w postaci binarnej,
- pole operacji o długości 4 bitów,
- pole odpowiedzi o długości 3 bitów,
- pole długości danych o rozmiarze 64 bitów,
- pole danych o zmiennym rozmiarze,
- dodatkowe pola zdefiniowane przez programistę, następujące po polu danych.

Funkcje oprogramowania:

- klienta: o nawiązanie połączenia z serwerem, o uzyskanie identyfikatora sesji, o wysłanie zaproszenia do drugiego klienta, o przyjęcie/odrzucenie zaproszenie, o przesłanie wiadomości tekstowej (binarna postać znaków ASCII), o zamknięcie sesji komunikacyjnej,
  o zakończenie połączenia.

 serwera: o wygenerowanie identyfikatora sesji, o informowanie klienta, czy drugi węzeł jest osiągalny:

  - w przypadku braku osiągalności, należy zwrócić błąd.

  o pośredniczenie w transmisji.

Inne:

 identyfikator sesji powinien być przesyłany w trakcie komunikacji.

2. OPIS PROTOKOŁU (FORMAT KOMUNIKATU, ZBIÓR KOMEND I ODPOWIEDZI).

Budowa pakietu

Operacja	Odpowiedź	Długość Danych	Dane	ID Sesji
4 bity	3 bity	64 bity	zmienny rozmiar	9 bitów

Komendy:

- `!accept` – kod 1 (0001),
- `!available` – kod 2 (0010),
- `!disconnect` – kod 3 (0011),
- `!exit` – kod 4 (0100), `!invite` – kod 5 (0101), `!reject` – kod 6 (0110).

Kody operacji:

- kod 0 (0000) – wysłanie zwykłej wiadomości,
- kod 1 (0001) – akceptacja zaproszenia wysłanego przez drugiego klienta,
- kod 2 (0010) – sprawdzenie dostępności drugiego klienta,
- kod 3 (0011) – rozłączenie się z drugim klientem,
- kod 4 (0100) – odłączenie się od serwera,
- kod 5 (0101) – zaproszenie drugiego klienta do czatu,
- kod 6 (0110) – odrzucenie zaproszenia wysłanego przez drugiego klienta.

Kody odpowiedzi:

- kod 0 (000) – wysłanie pakietu inicjalizującego id sesji (pakiet testowy),
- kod 1 (001) – wysłanie wiadomości,
- kod 2 (010) – serwer odpowiada, że klient zaakceptował zaproszenie do czatu,
- kod 3 (011) – serwer odpowiada, że drugi klient jest dostępny,
- kod 4 (100) – serwer odpowiada, że drugi klient rozłączył się z czatu,
- kod 5 (101) – serwer odpowiada, że drugi klient wyszedł z czatu,
- kod 6 (110) – serwer odpowiada, że zostałeś zaproszony do czatu,
- kod 7 (111) – serwer odpowiada, że klient odrzucił twoje zaproszenie do czatu.

3. APLIKACJA UŻYTKOWNIKA ORAZ APLIKACJA SERWERA (ZALEŻNIE OD WARIANTU) W WERSJI ŹRÓDŁOWEJ Z KOMENTARZAMI (W SZCZEGÓLNOŚCI DOTYCZĄCYMI FRAGMENTÓW PROGRAMÓW ZWIĄZANYCH Z TRANSMISJĄ).

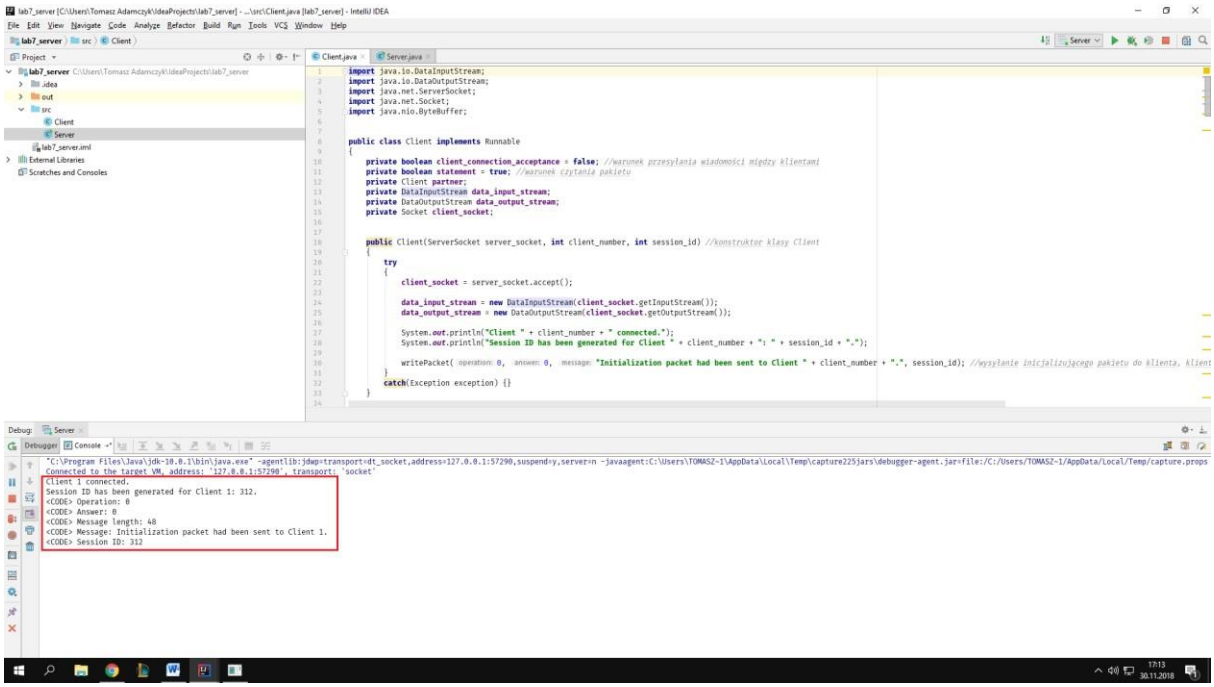
Aplikacja klienta:

- https://github.com/WangHoHan/chat-with-binary-protocol/blob/master/Chat%20With%20Binary%20Protocol/Client/src/Client.java

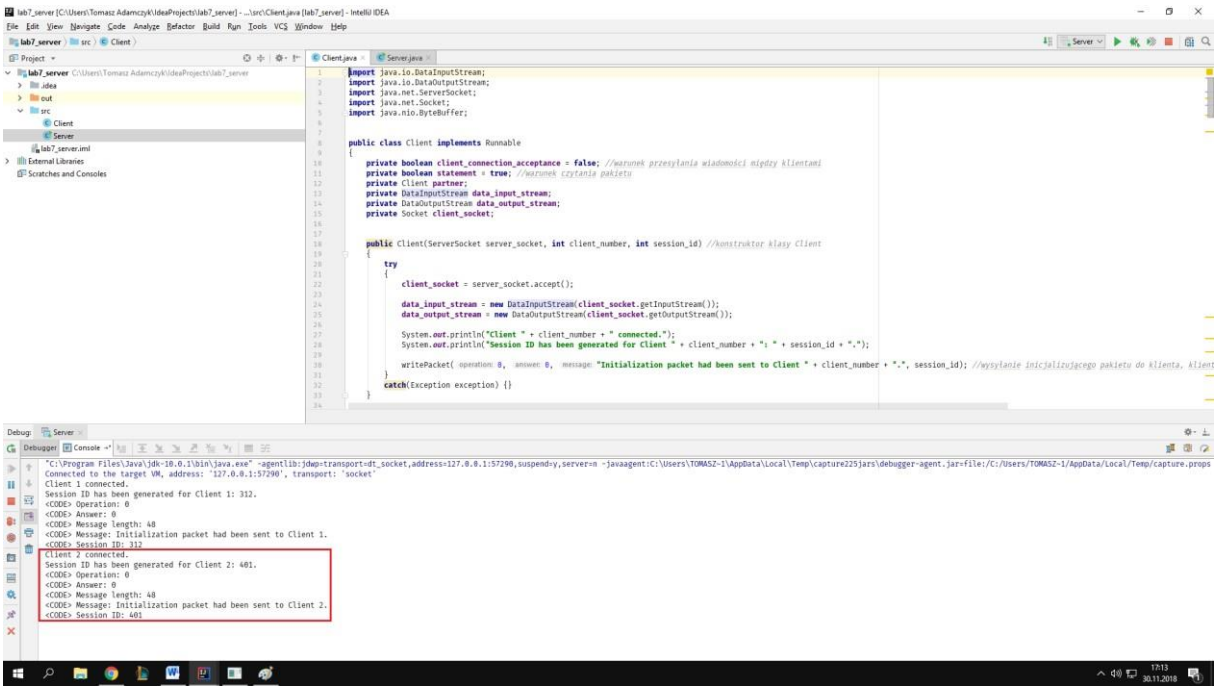
Aplikacja serwera (zawiera dwa pliki typu java):

- https://github.com/WangHoHan/chat-with-binary-protocol/blob/master/Chat%20With%20Binary%20Protocol/Server/src/Client.java
- https://github.com/WangHoHan/chat-with-binary-protocol/blob/master/Chat%20With%20Binary%20Protocol/Server/src/Server.java

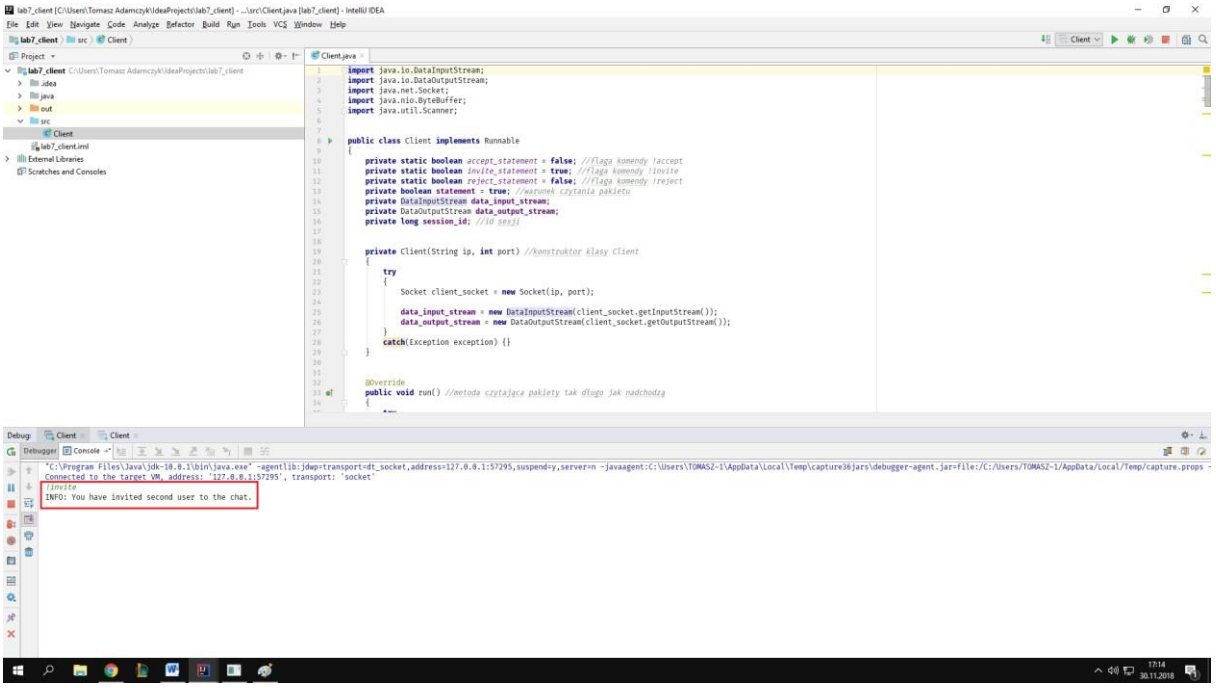
4. PRZEBIEG PRZYKŁADOWEJ SESJI KOMUNIKACYJNEJ – OPIS SŁOWNY ORAZ OBRAZ SESJI ZAREJESTROWANY PRZEZ PROGRAM *Wireshark*, WRAZ ZE STOSOWNYMI OBJAŚNIENIAMI.



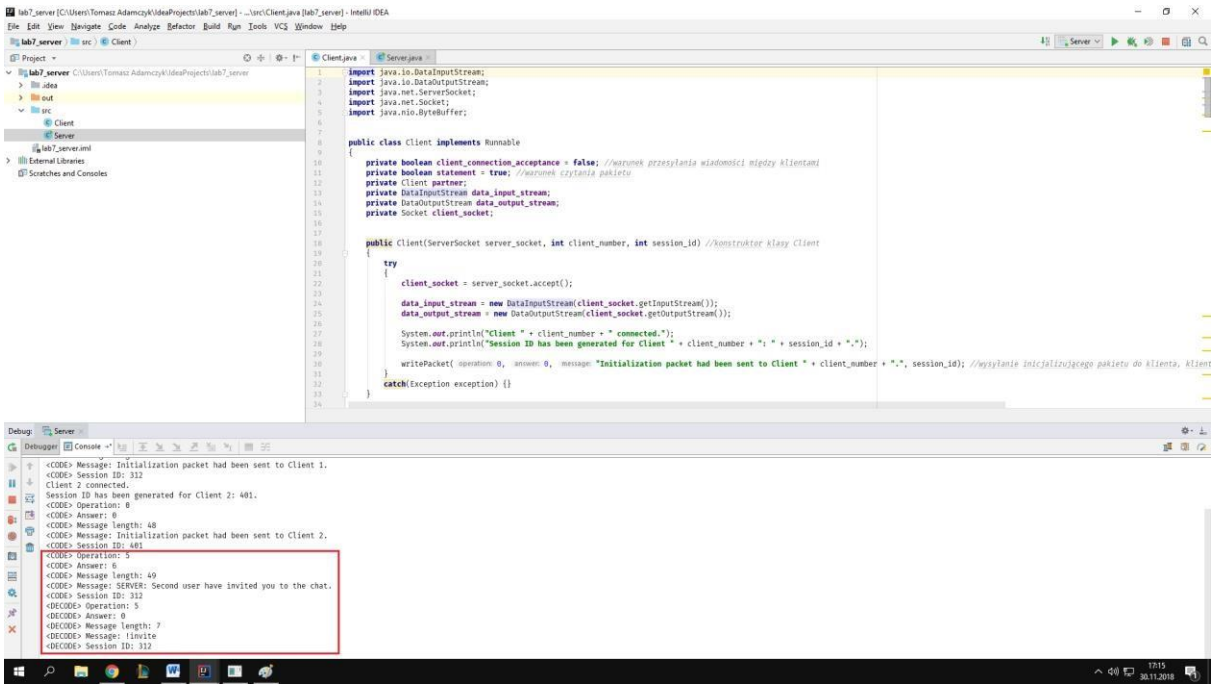
Rysunek 1: Połączenie się pierwszego klienta z serwerem. Serwer wysyła do klienta „pusty” pakiet zawierający jedynie id sesji. Klient odczytuje id sesji i przypisuje sobie jego wartośd.



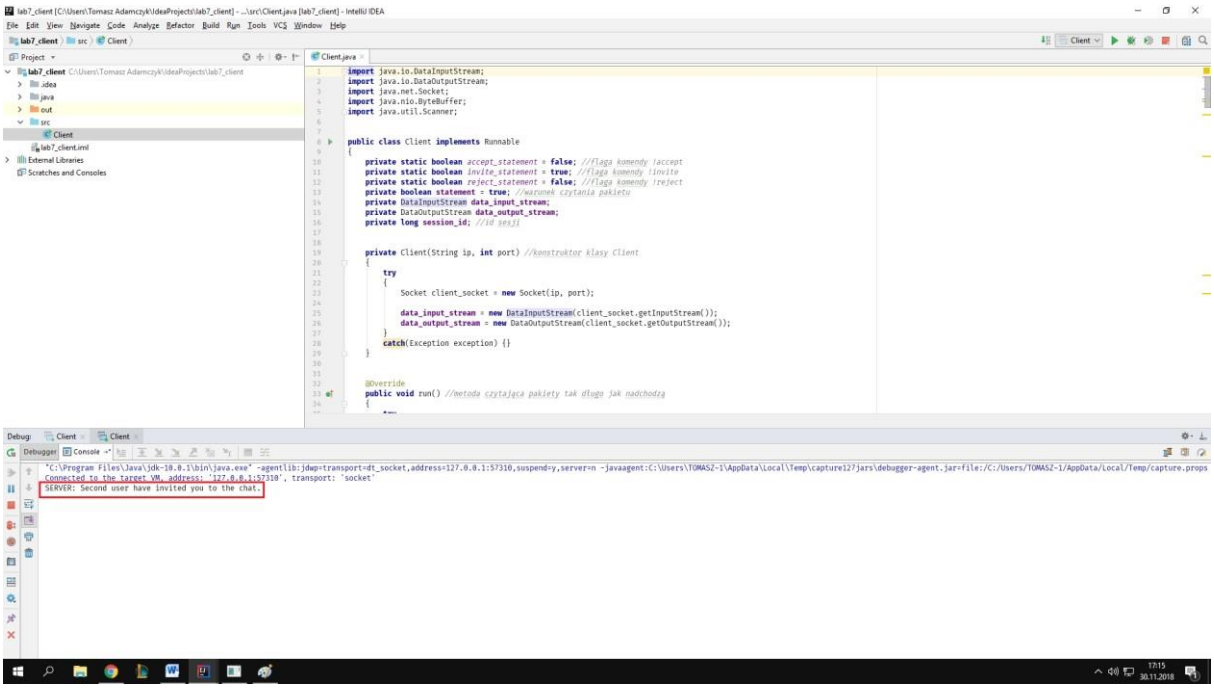
Rysunek 2: Połączenie się drugiego klienta z serwerem. Serwer wysyła do klienta „pusty” pakiet zawierający jedynie id sesji. Klient odczytuje id sesji i przypisuje sobie jego wartośd.



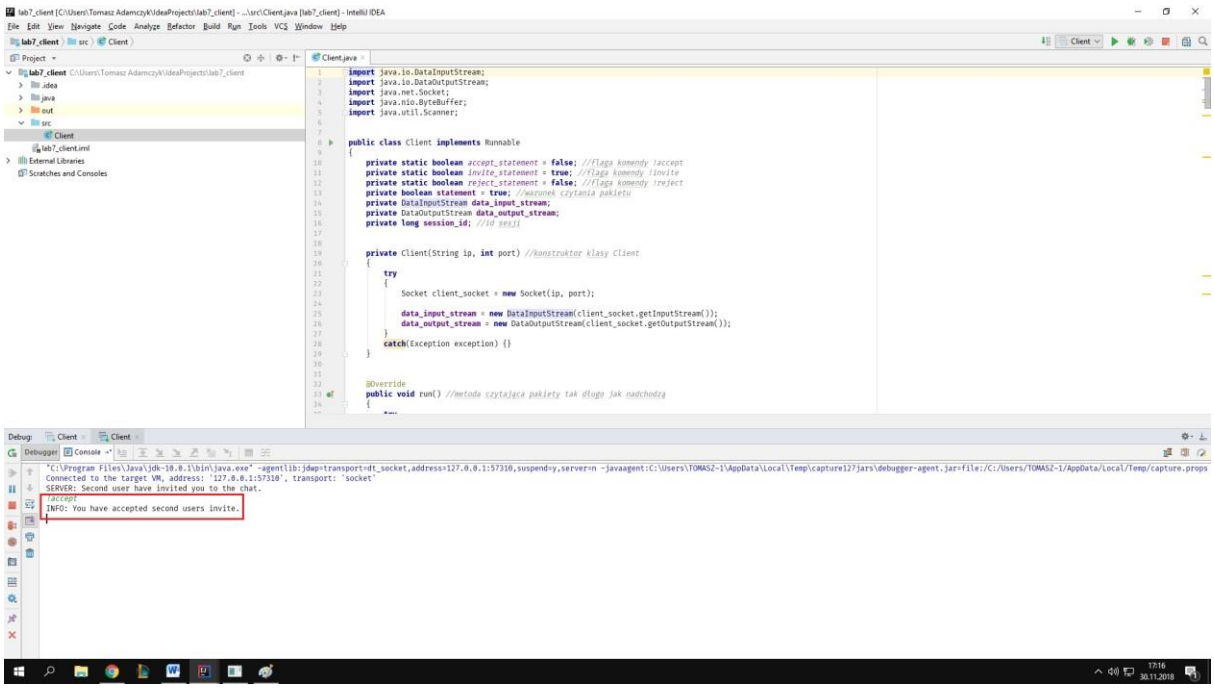
Rysunek 3: Wysłanie zaproszenia przez pierwszego użytkownika. Kod operacji dla komendy !invite to 5 (0101). Użytkownikowi wyświetla się komunikat potwierdzający wysłanie zaproszenia.



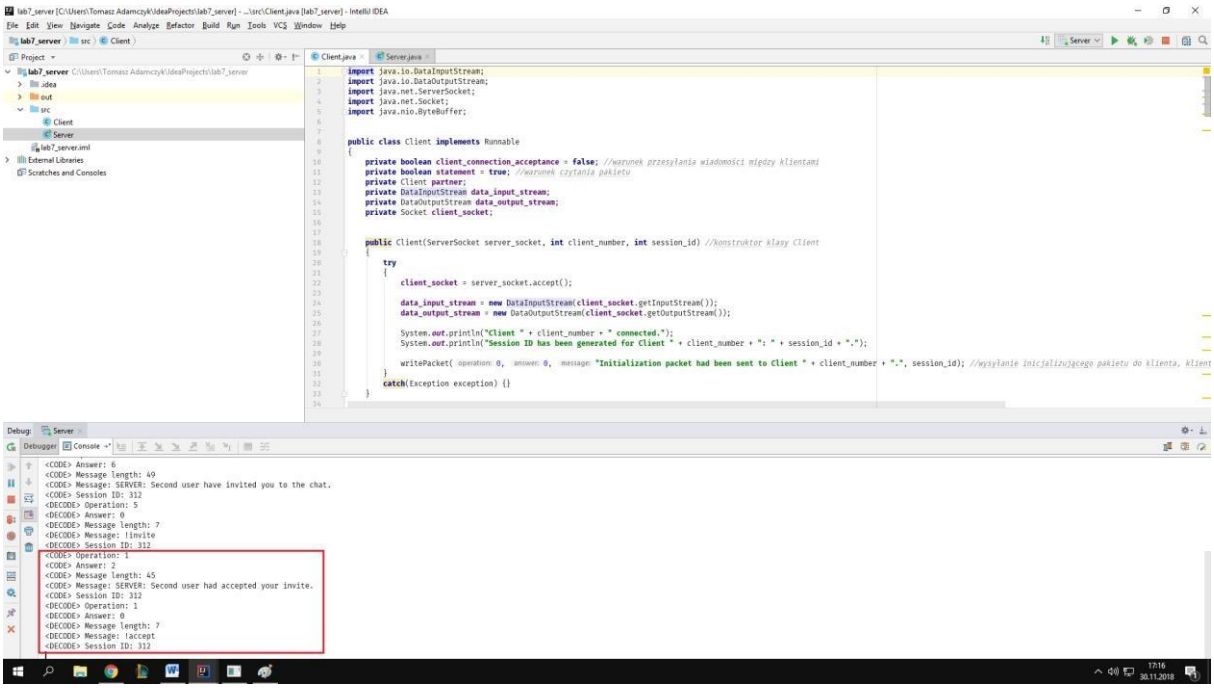
Rysunek 4: Odebranie zaproszenia przez serwer. Serwer przetwarza otrzymaną informację i wysyła pakiet informujący drugiego użytkownika o zaproszeniu do czatu (kod odpowiedzi to 6 (110)).



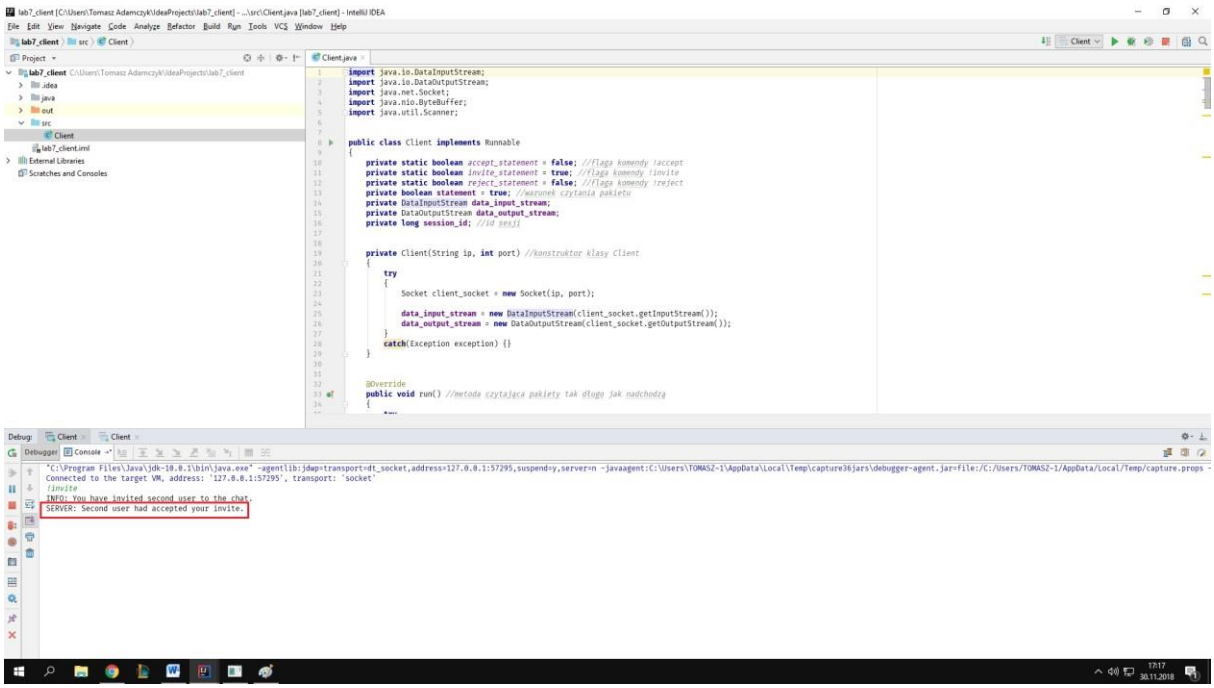
Rysunek 5: Odebranie zaproszenia przez drugiego użytkownika. Użytkownik otrzymuje informację od serwera o chęci nawiązania połączenia przez drugą osobę dzięki przesłaniu odpowiedniej wartości w polu operacji oraz odpowiedzi.



Rysunek 6: Zaakceptowanie zaproszenia przez drugiego użytkownika. Kod operacji dla komendy !accept to 1 (0001). Użytkownikowi wyświetla się stosowny komunikat.

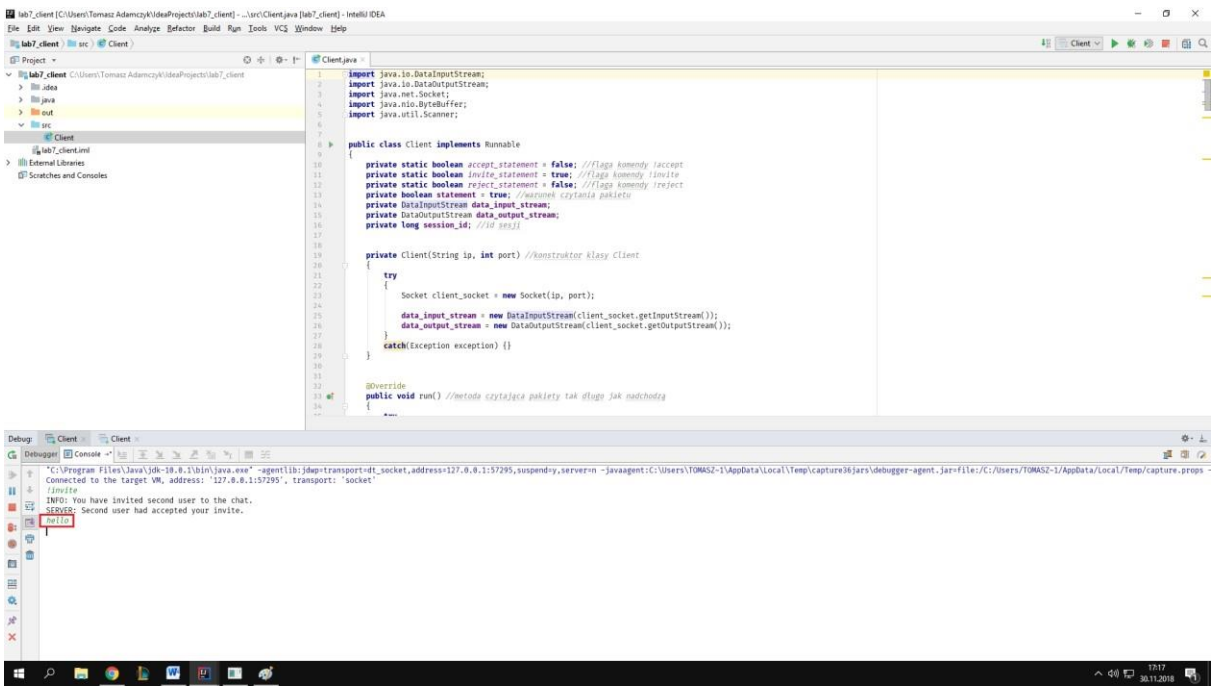


Rysunek 7: Otrzymanie informacji o zaakceptowaniu zaproszenia przez drugiego użytkownika. Serwer otrzymuje odpowiedź i wysyła pakiet informujący pierwszego użytkownika o sukcesie połączenia (kod odpowiedzi 2 (010)).

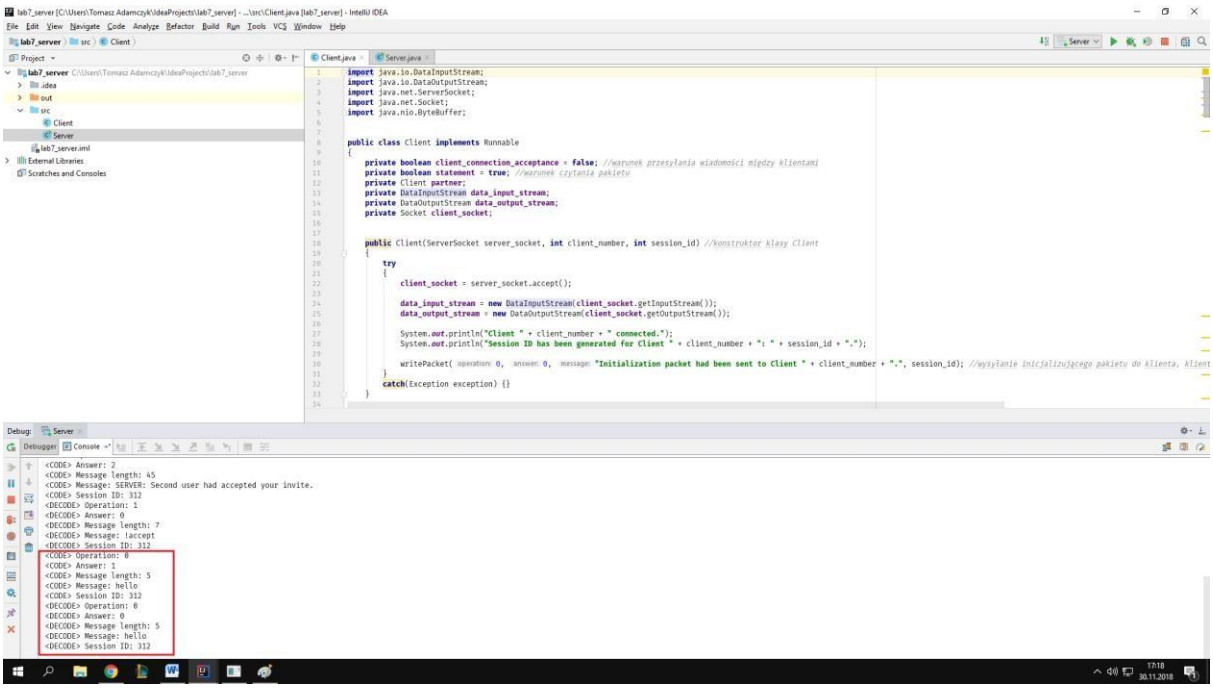


Rysunek 8: Użytkownik wysyłający zaproszenie dostaje informację od serwera o tym, że druga osoba zaakceptowała jego zaproszenie. Użytkownikowi wyświetla się stosowny komunikat dzięki odebraniu pakietu o ustalonej wartości pól operacji oraz odpowiedzi.

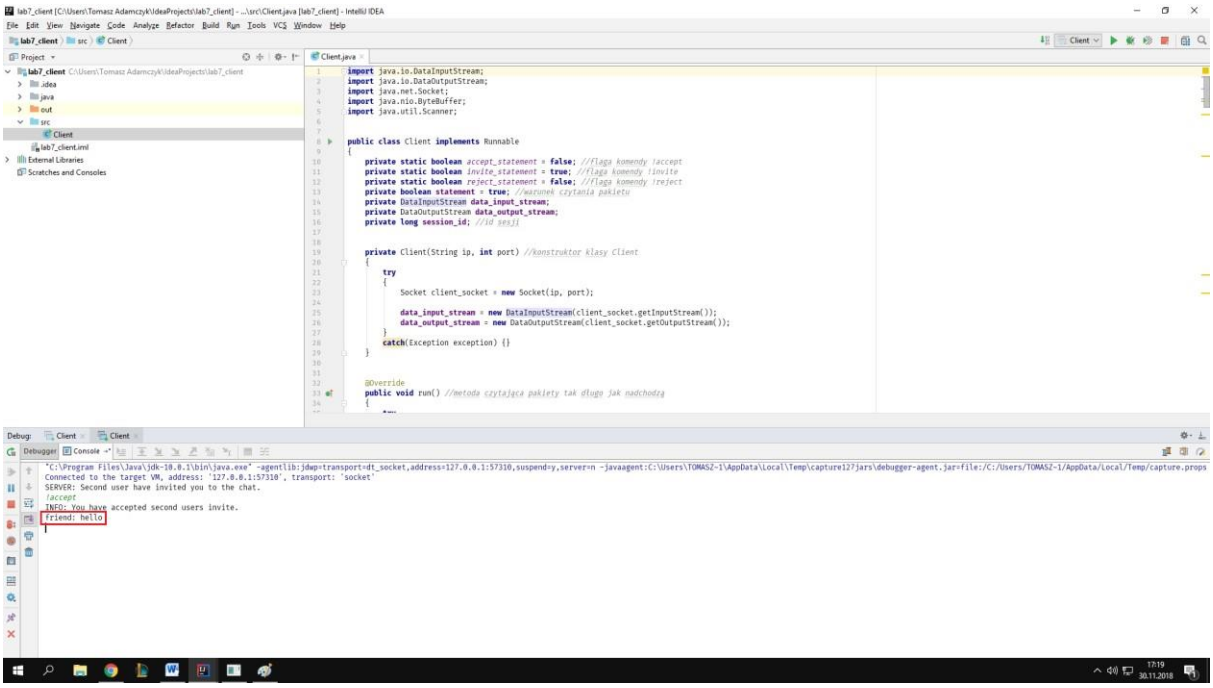
‘



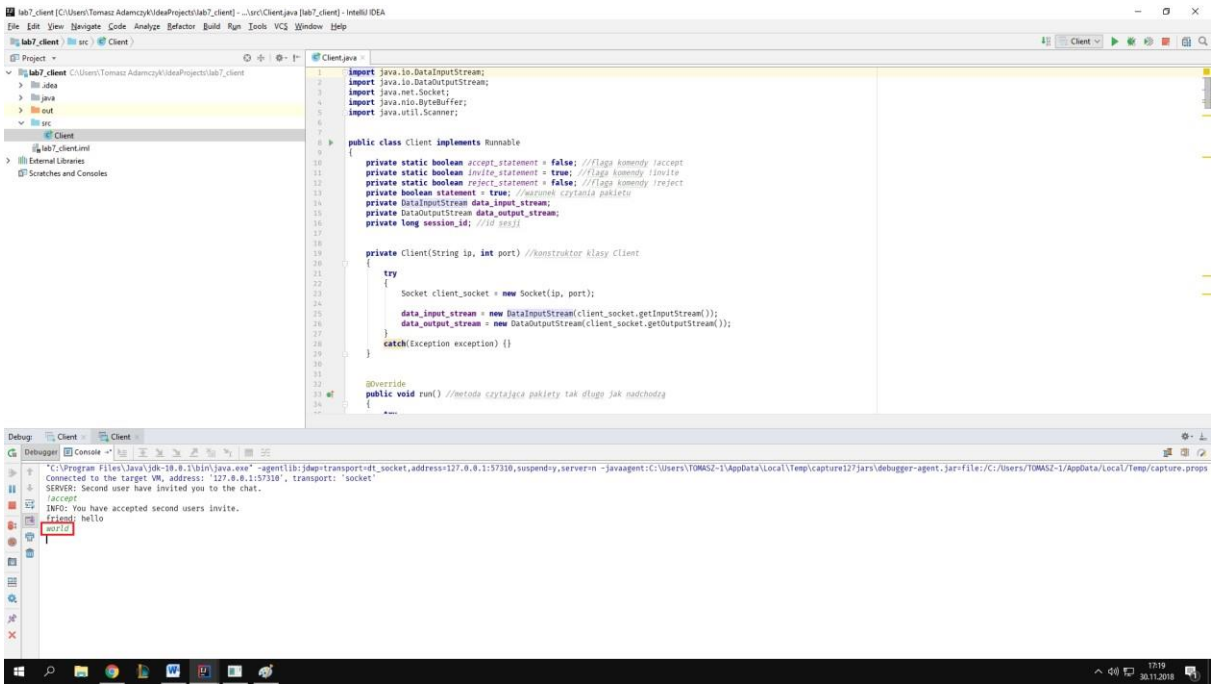
Rysunek 9: Wysłanie wiadomości przez pierwszego klienta. Kod operacji do wysłania wiadomości to 0 (0000).



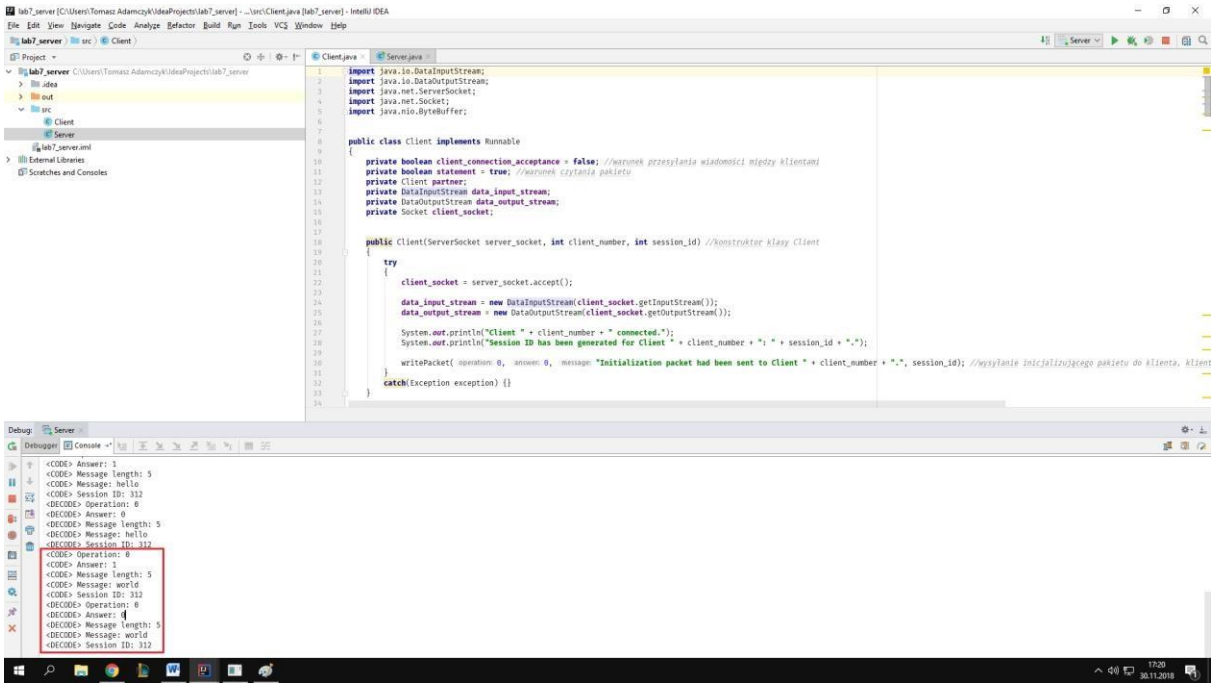
Rysunek 10: Serwer otrzymuje wiadomośd wysłaną przez pierwszego klienta i wysyła ją do drugiej osoby. Kod odpowiedzi w tym przypadku to 1 (001).



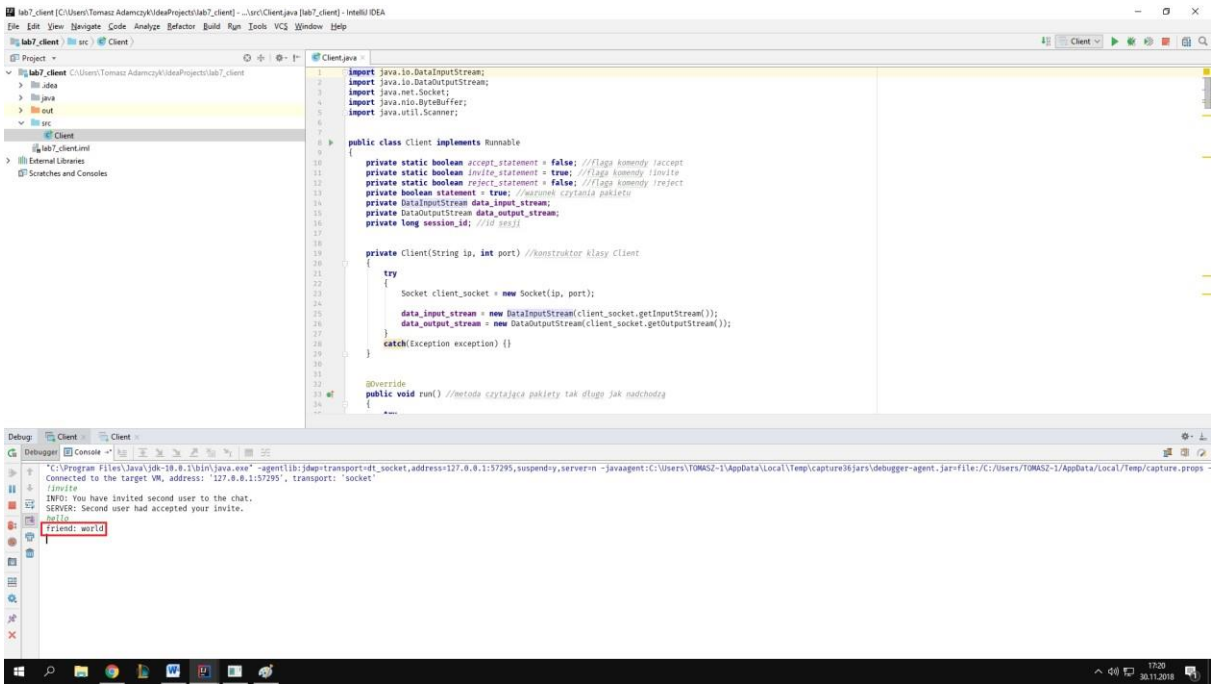
Rysunek 11: Drugi klient otrzymuje wiadomośd wysłaną przez drugą osobę. Wyświetla mu się przed wiadomością „friend:", aby nie pomylid osoby wysyłającej wiadomośd. „friend" nie jest częścią wiadomości.



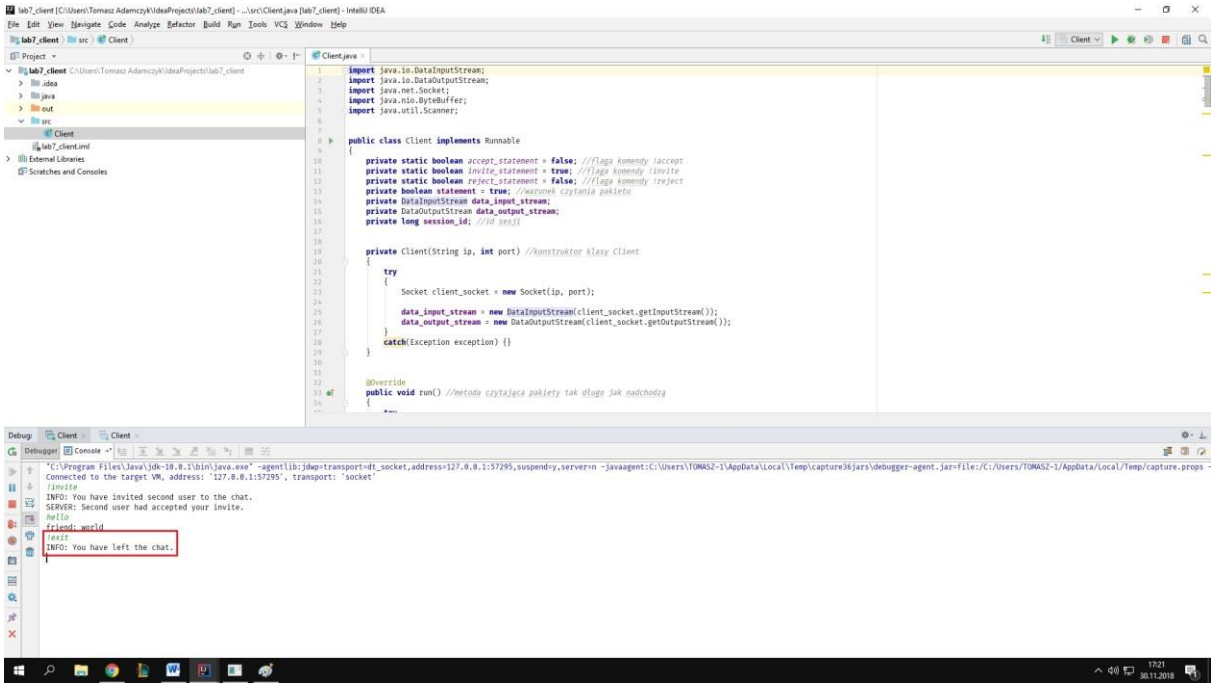
Rysunek 12: Wysłanie wiadomości przez klienta numer 2. Wszystko dzieje się analogicznie jak przy przesyłaniu wiadomości przez klienta numer 1 (wyżej).



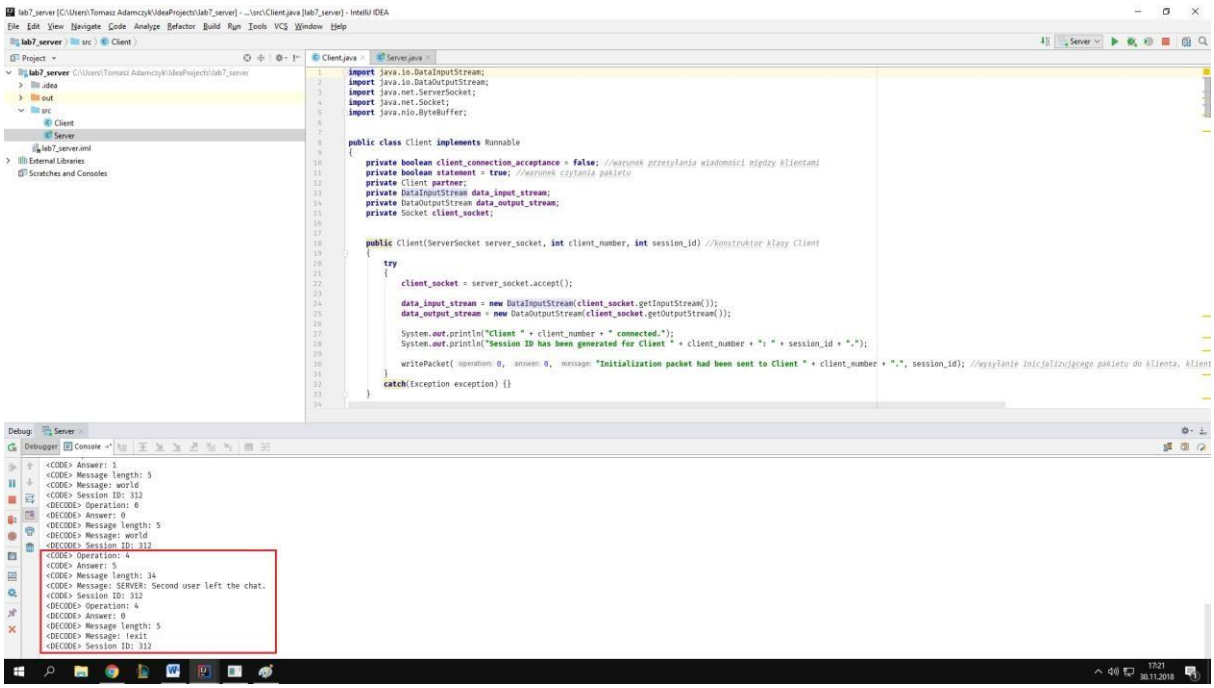
Rysunek 13: Odebranie wiadomości wysyłanej przez klienta numer 2. Serwer przesyła wiadomośd do klienta numer 1.



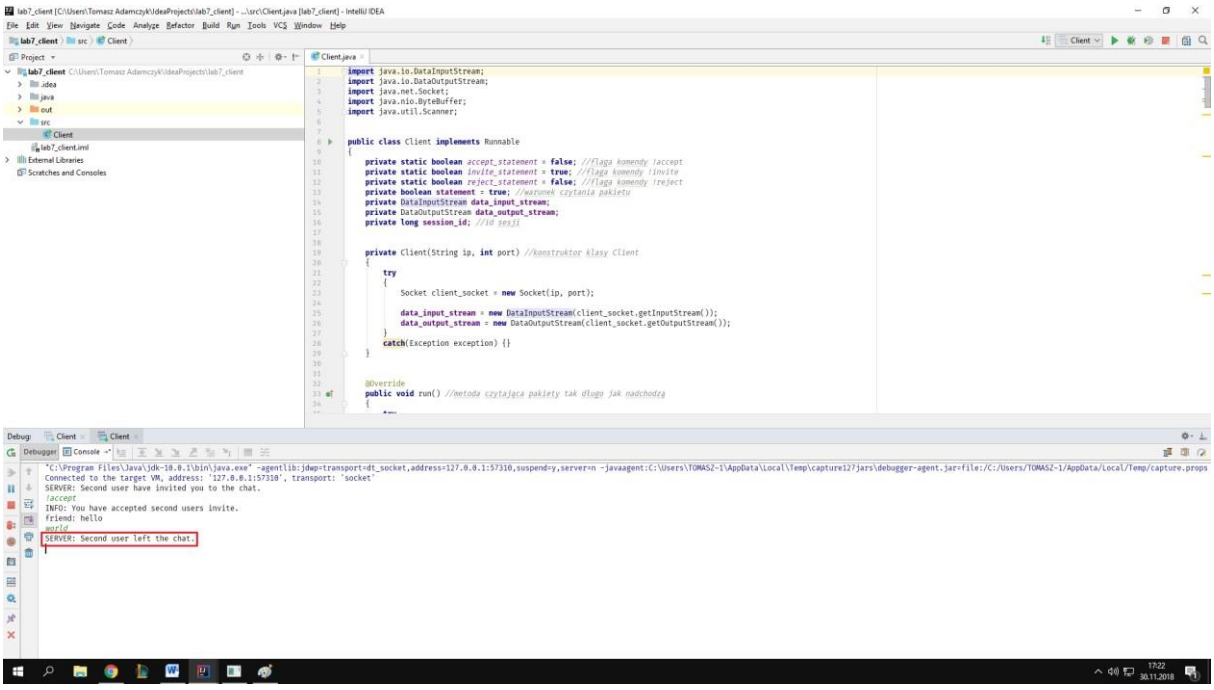
Rysunek 14: Klient numer 1 otrzymuje wiadomośd wysłaną przez klienta numer 2.



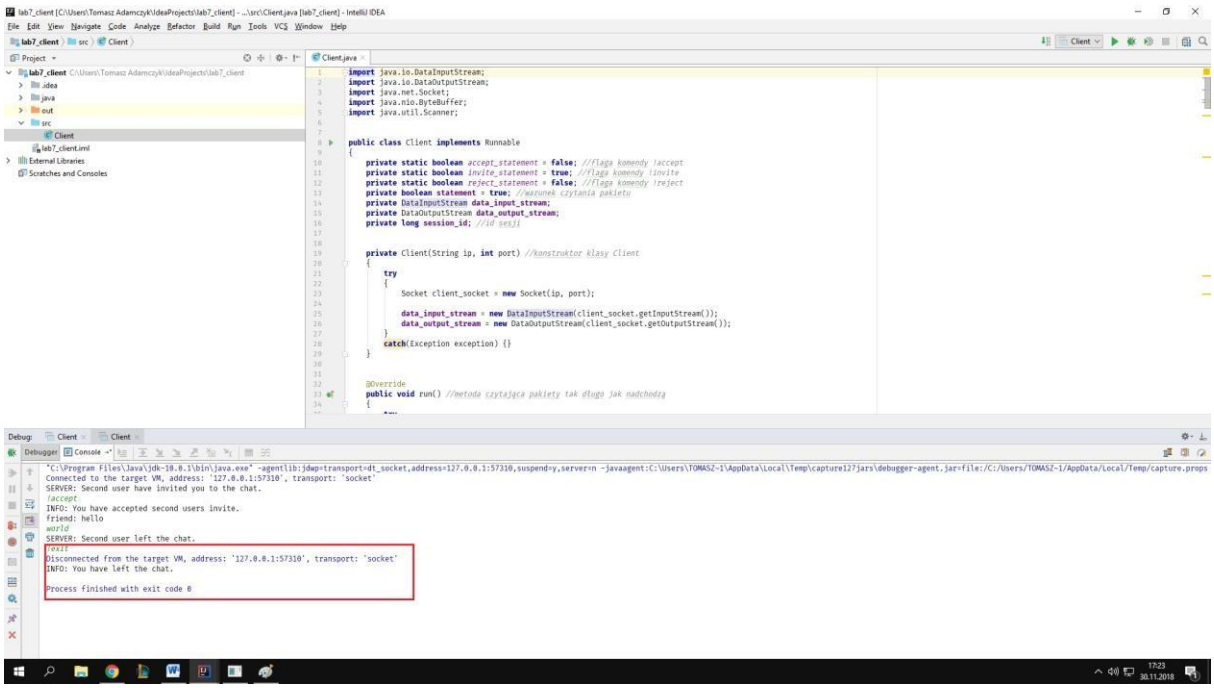
Rysunek 15: Klient numer 1 opuszcza czat korzystając z komendy !exit (kod operacji: 4 (0100)). Przesyła swój ostatni pakiet serwerowi przed rozłączeniem połączenia. Użytkownikowi wyświetla się stosowny komunikat.



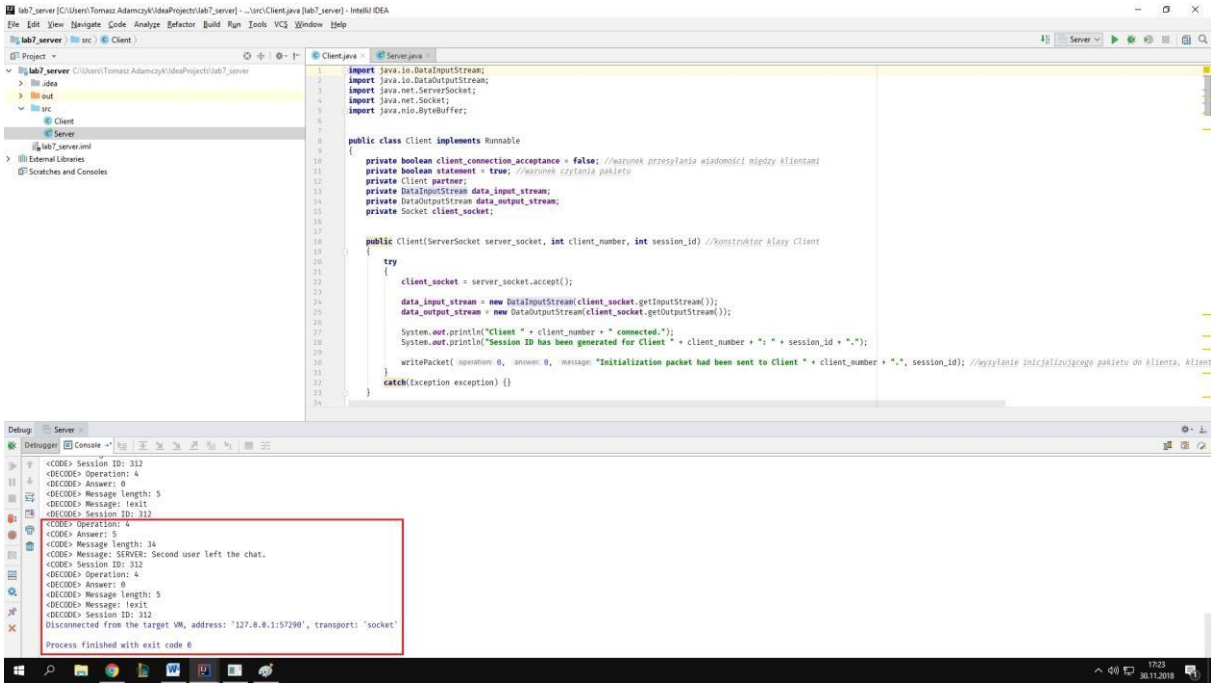
Rysunek 16: Serwer otrzymuje informację o rozłączeniu się pierwszego klienta. Wysyła pakiet do drugiego klienta informujący o tym, że drugi użytkownik się rozłączył (kod odpowiedzi: 5 (101).



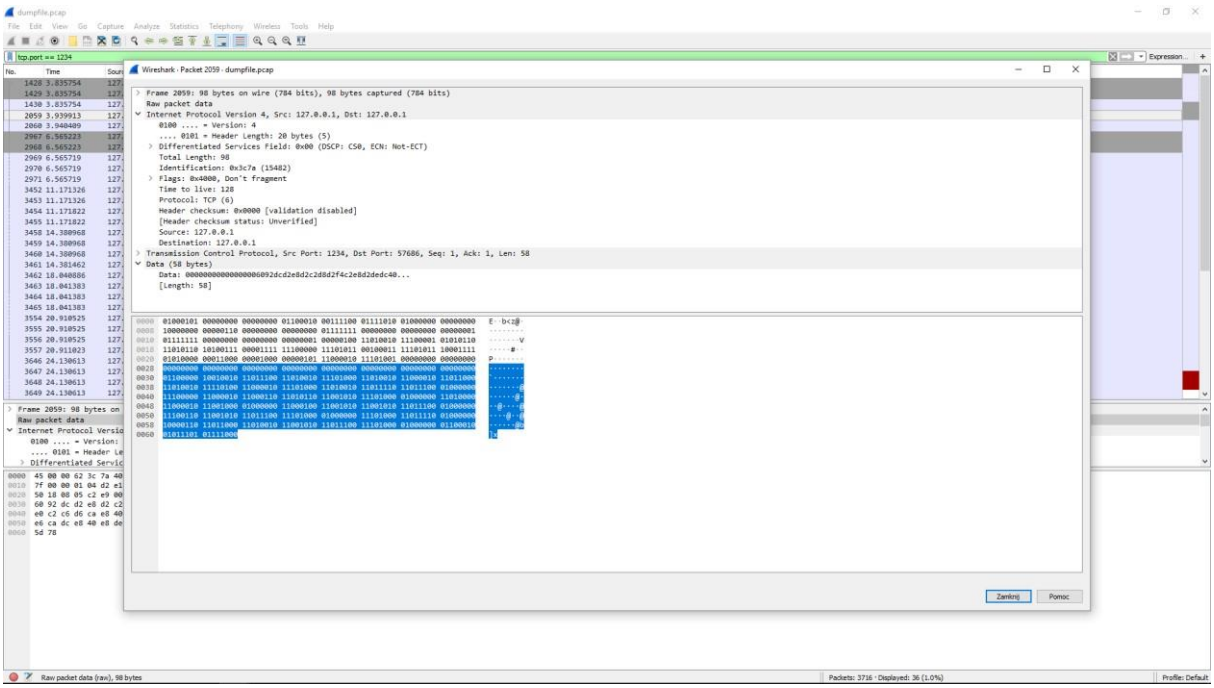
Rysunek 17: Klient numer 2 dostaje wiadomośd od serwera o opuszczeniu czatu przez klienta numer 1. Użytkownikowi wyświetla się odpowiednie powiadomienie dzięki odczytaniu pola operacji oraz odpowiedzi w wysłanym przez serwer pakiecie.



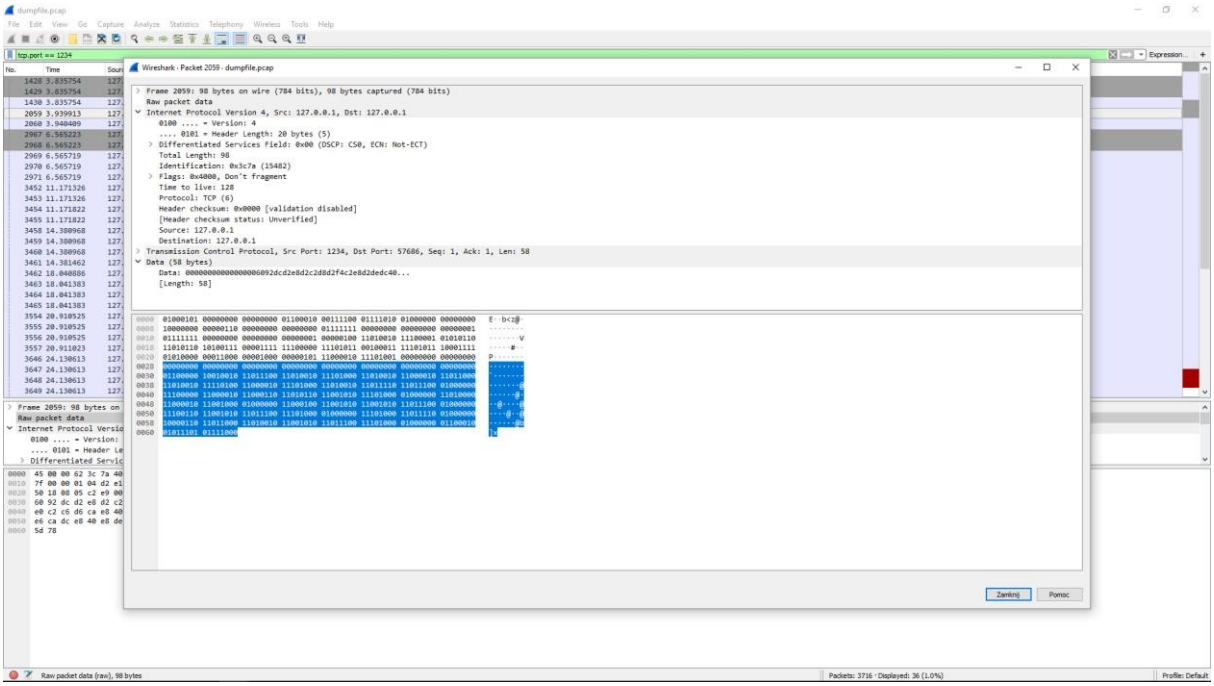
Rysunek 18: Klient numer 2 rozłącza się z czatu komendą !exit analogicznie do klienta numer 1.



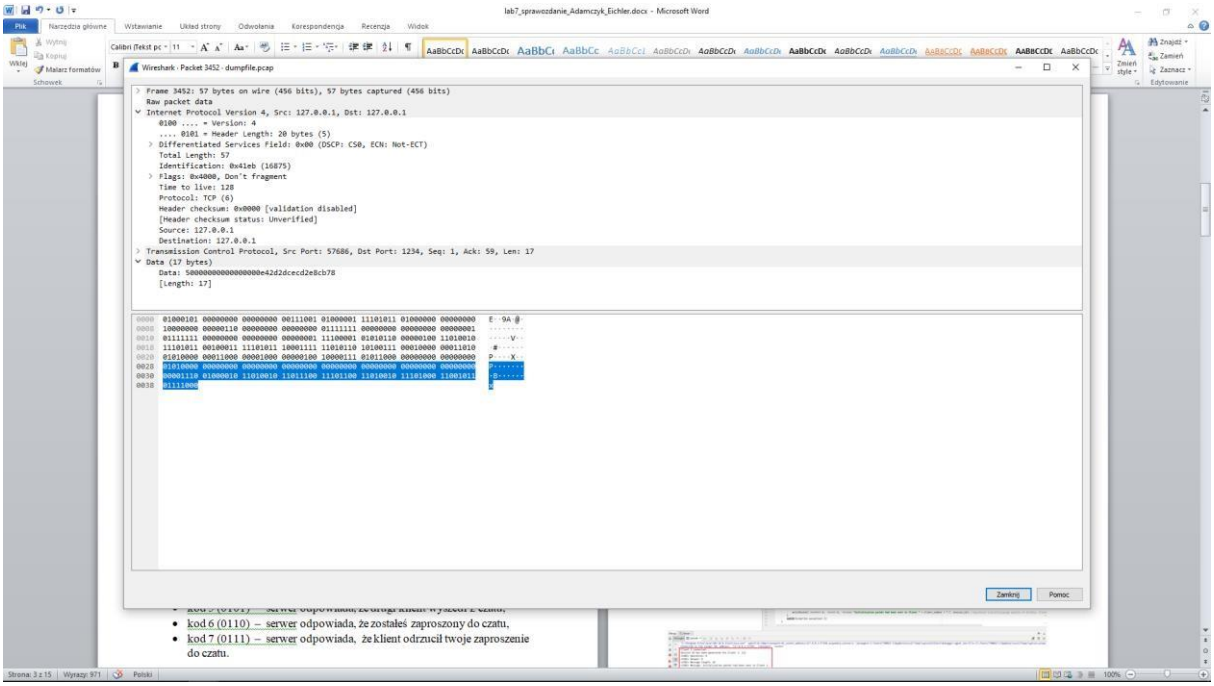
Rysunek 19: Serwer otrzymuje swój ostatni pakiet informujący go o zakooczeniu czatu przez klienta numer 2 i sam się wyłącza.



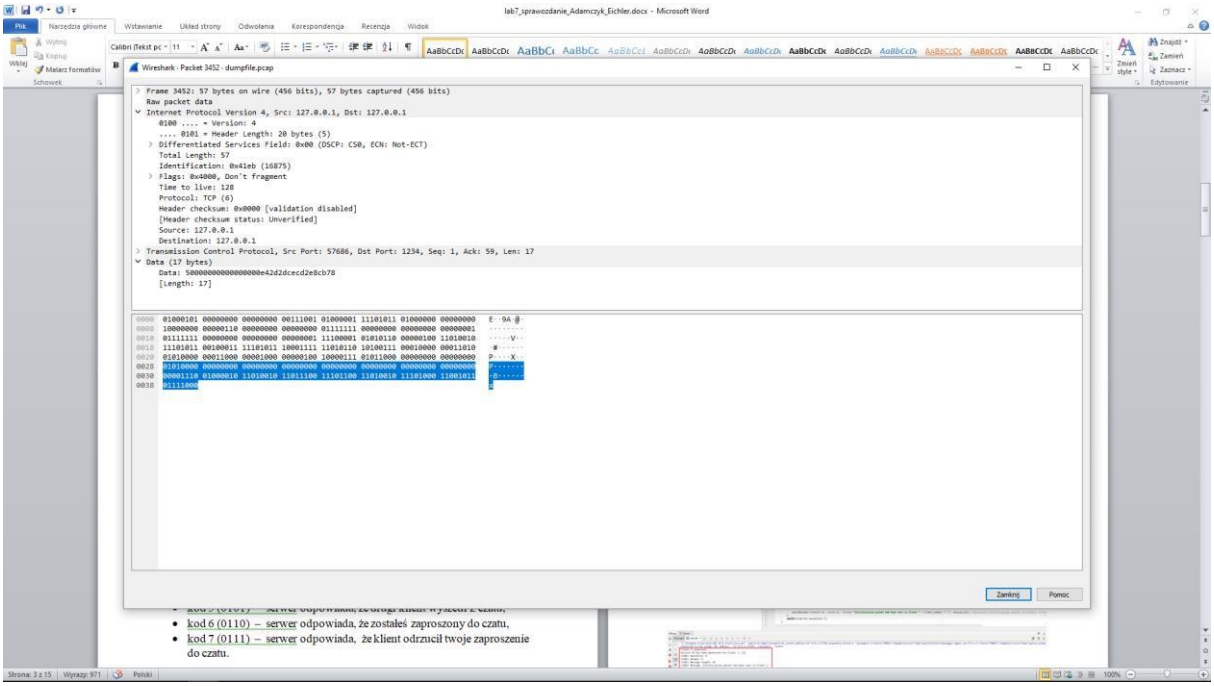
Rysunek 20: Pakiet inicjalizujący dla klienta 1. Wysyła on id sesji: 312 (101111000) – zapisane na 9 bitach oraz wiadomośd: „Initialization packet had been sent to Client 1”. Pole operacji oraz odpowiedzi ustawione są na 0. Id sesji może różnid się od tych na wyższych rysunkach (serwer wylosował inne), gdyż konieczne było ponowne podsłuchanie połączenia. Reszta informacji zawartych w dumpfile.pcap się zgadza z powyższymi zdjęciami.



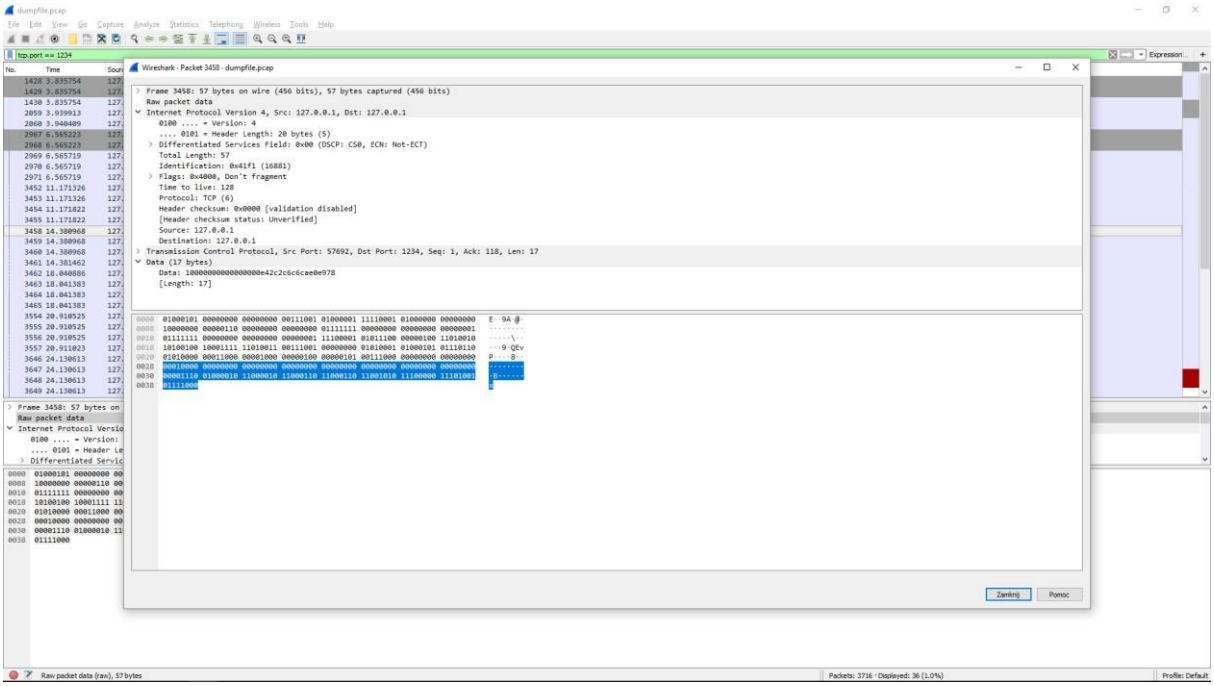
Rysunek 21: Pakiet inicjalizujący dla klienta 2. Wysyła on id sesji: 492 (111101100) – zapisane na 9 bitach oraz wiadomośd: „Initialization packet had been sent to Client 2”. Pole operacji oraz odpowiedzi ustawione są na 0. Id sesji może różnid się od tych na wyższych rysunkach (serwer wylosował inne), gdyż konieczne było ponowne podsłuchanie połączenia. Reszta informacji zawartych w dumpfile.pcap się zgadza z powyższymi zdjęciami.



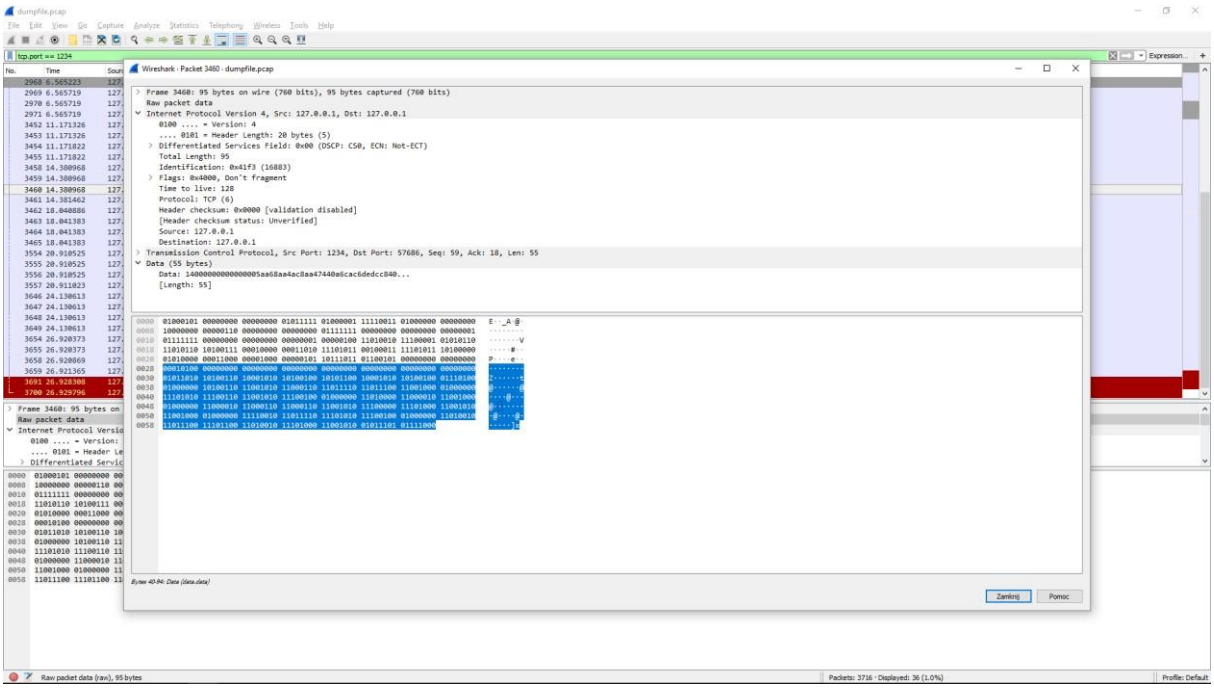
Rysunek 22: Zaproszenie drugiego klienta do czatu. Pole operacji: 5 (0101), pole odpowiedzi 0 (000), wiadomośd: !invite.



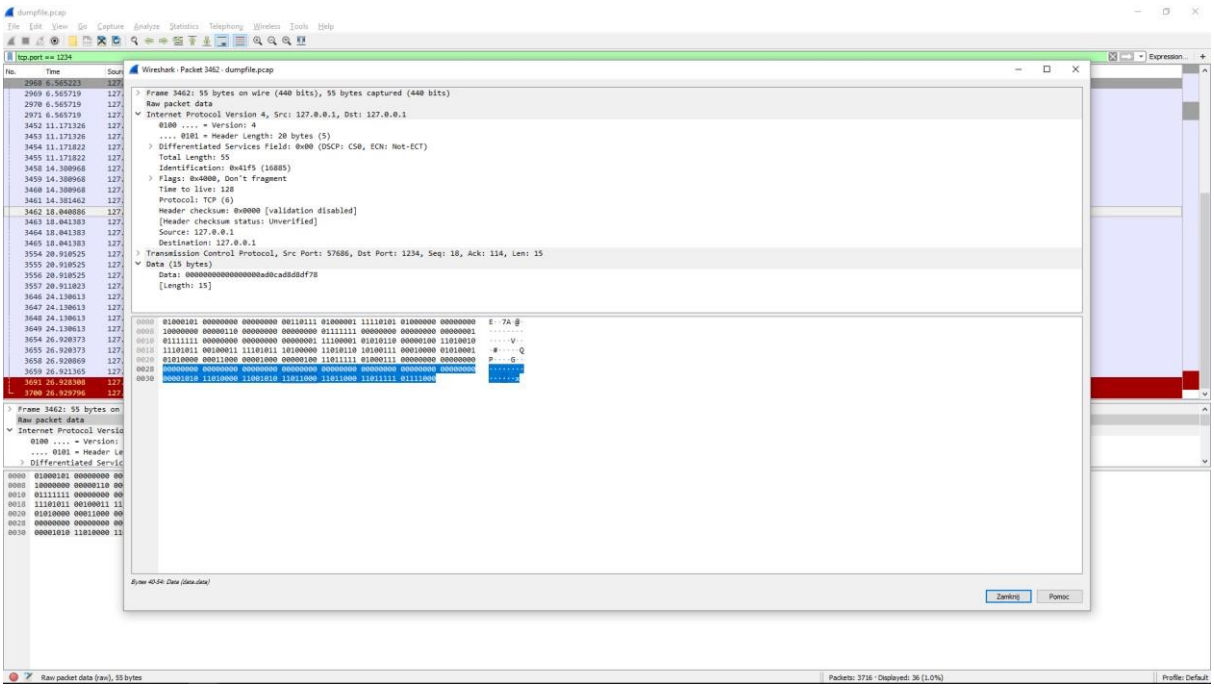
Rysunek 23: Serwer dostaje wiadomośd od klienta numer 1 (!invite) i wysyła do klienta numer 2 wiadomośd: „SERVER: Second user have invited you to the chat.”, pole operacji nie zmienia się w stosunku do wiadomości od klienta 1. Pole odpowiedzi ustawia się na 6 (110).



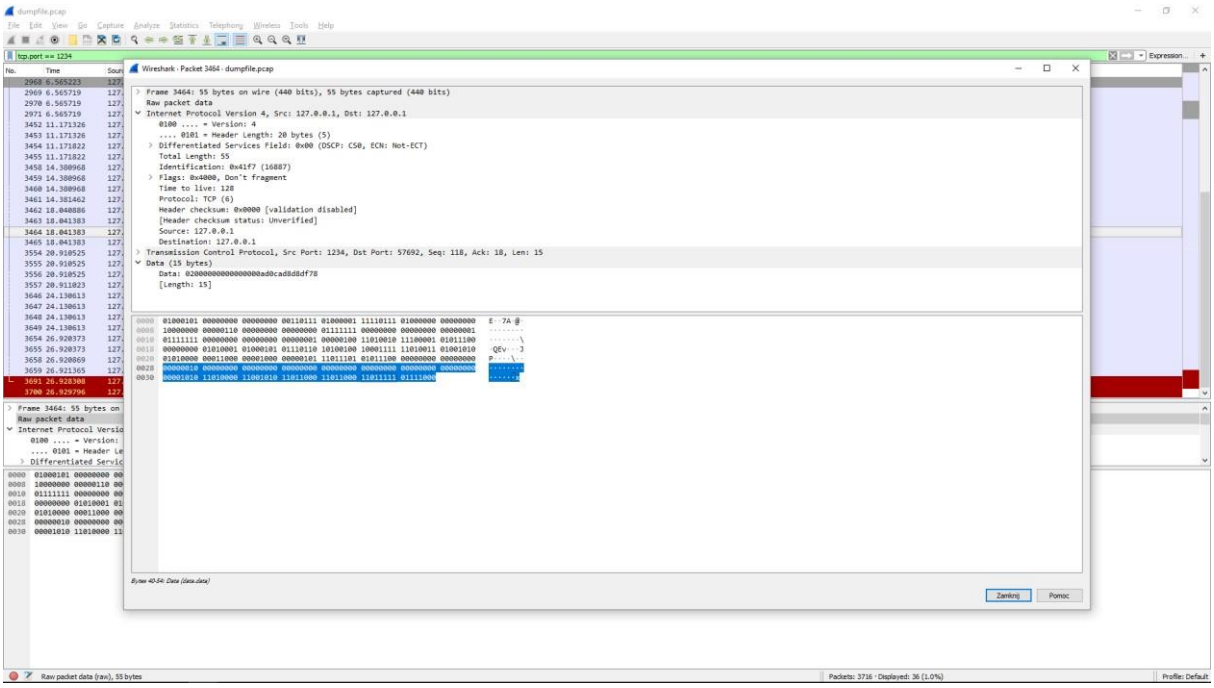
Rysunek 24: Klient akceptuje zaproszenie od klienta 1 i wysyła do serwera wiadomośd: „!accept”. Pole operacji ustawia na 1 (0001), pole odpowiedzi 0 (000).



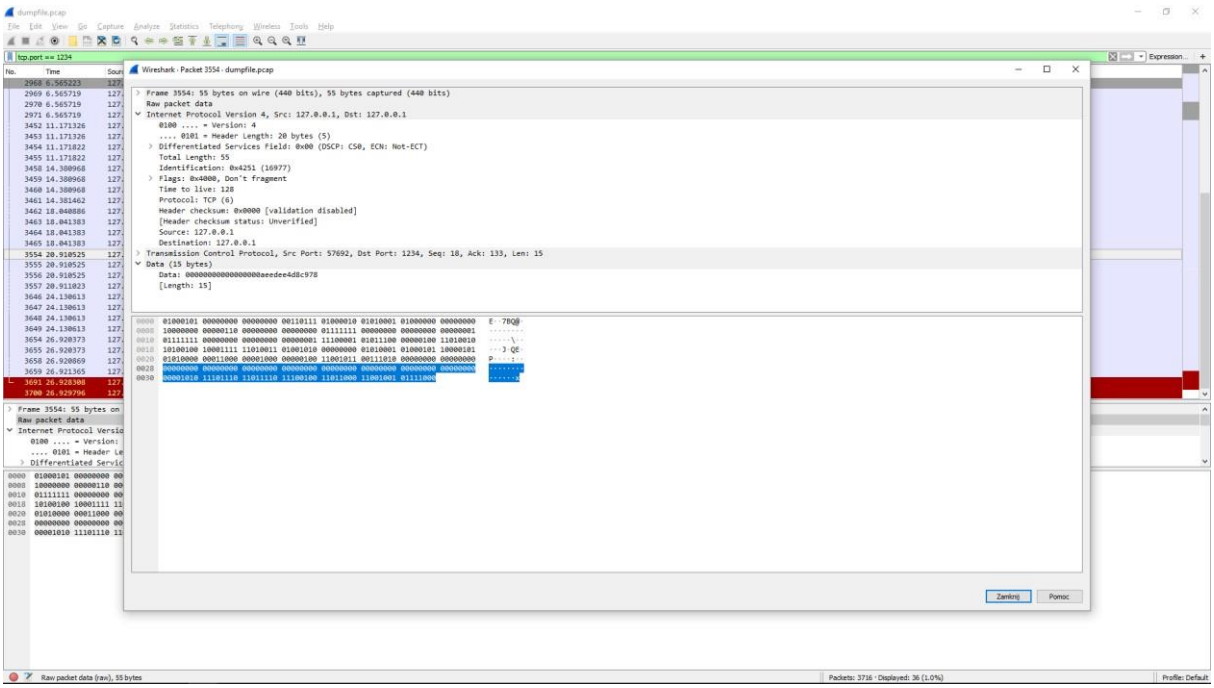
Rysunek 25: Serwer otrzymuje informację o akceptacji od klienta numer 2. Serwer wysyła pakiet z wiadomością: „SERVER: Second user had accepted your invite.”. Pole operacji się nie zmienia w stosunku do poprzedniego pakietu otrzymanego od klienta numer 2, pole odpowiedzi wynosi 2 (010).



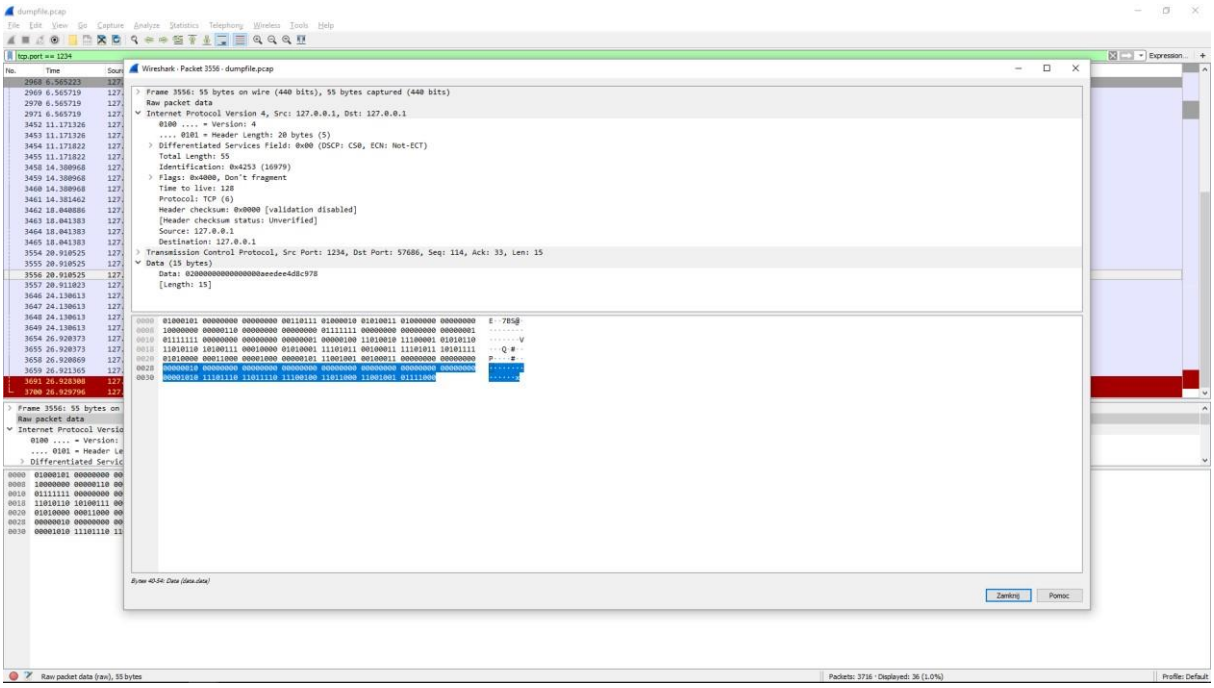
Rysunek 26: Klient wysyła wiadomośd: „hello” do drugiego klienta. Pole operacji wynosi 0 (0000) oraz pole odpowiedzi wynosi 0 (000).



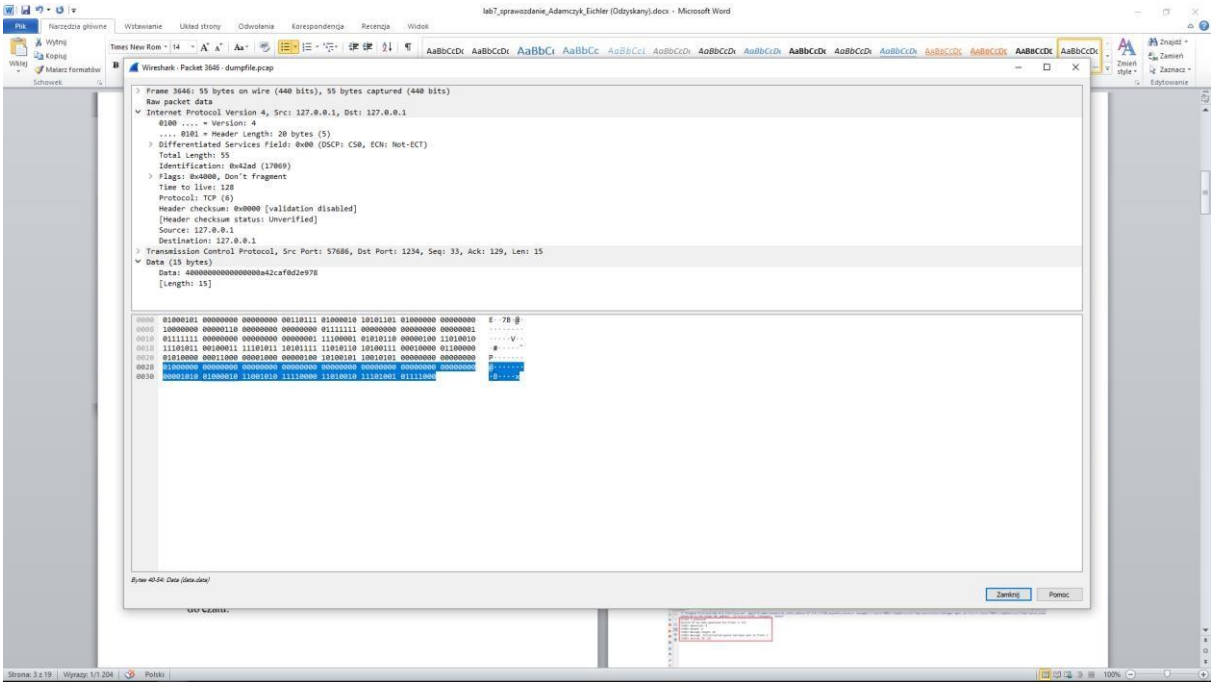
Rysunek 27: Serwer otrzymuje wiadomośd hello od klienta 1 i wysyła ją do klienta 2. Pole odpowiedzi ustala na 1 (001). Pole operacji się nie zmienia i wynosi 0 (0000).



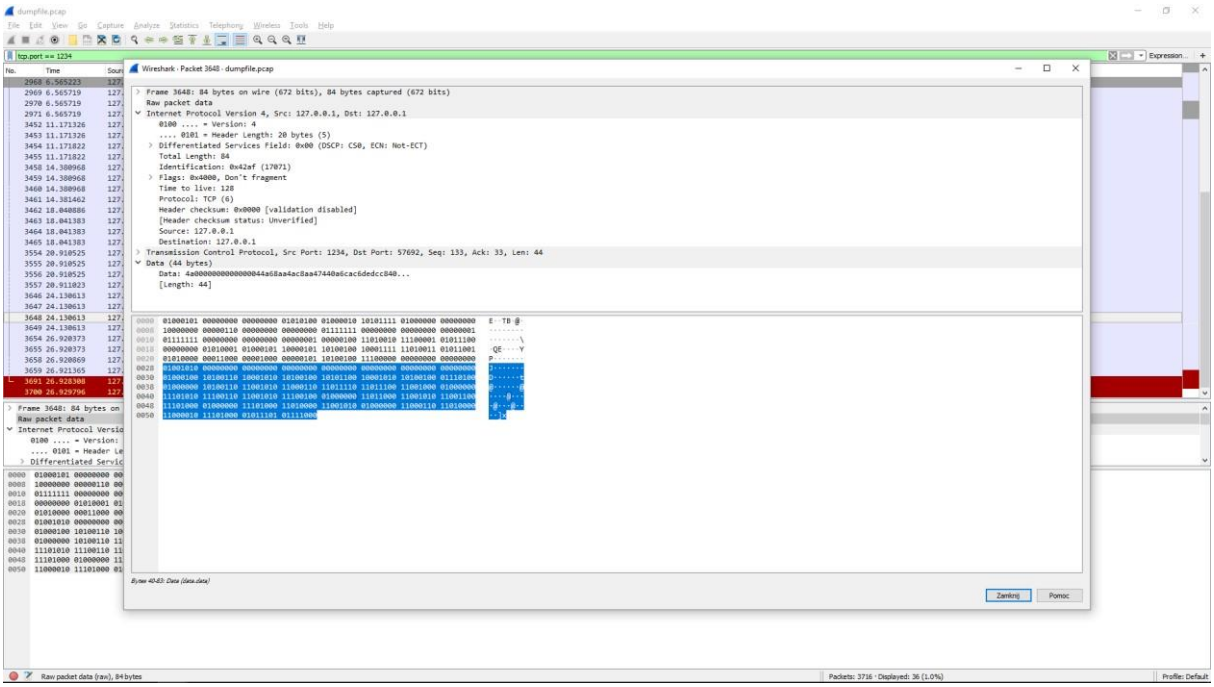
Rysunek 28: Klient numer 2 wysyła wiadomośd: „world” do klienta numer 1. Pole operacji ustawia na 0 (0000), pole odpowiedzi również wynosi 0 (000).



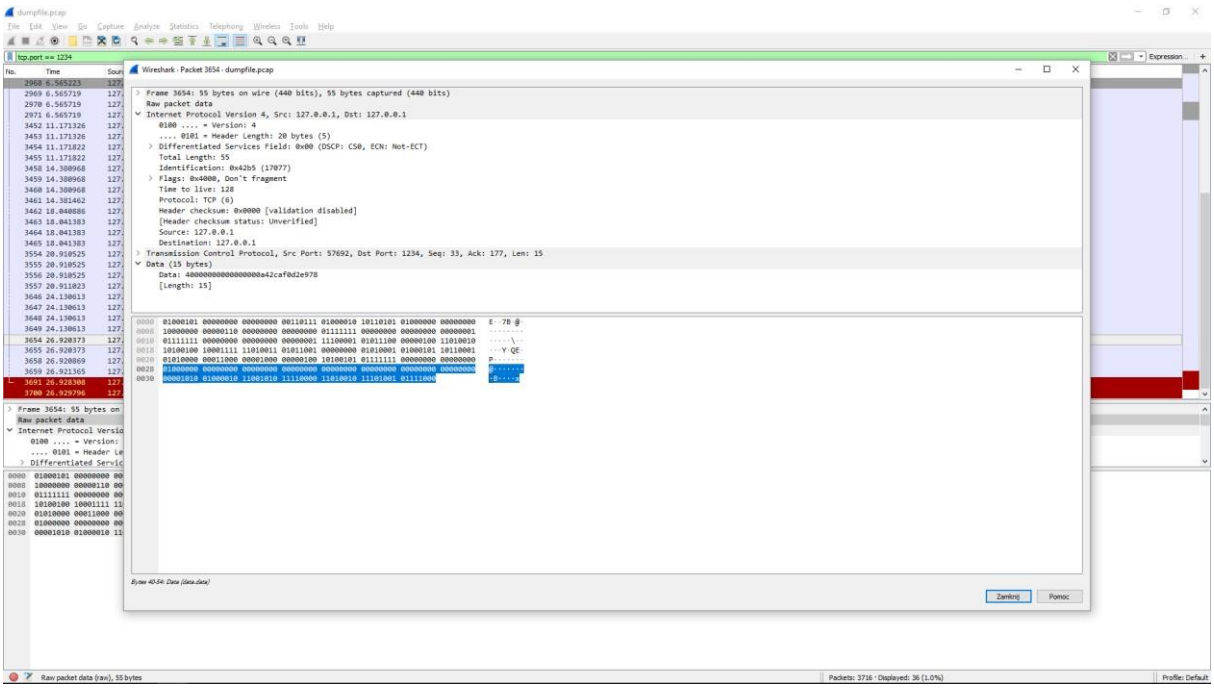
Rysunek 29: Serwer otrzymuje od klienta 2 wiadomośd „world” i przesyła ją do klienta 1 ustawiając pole odpowiedzi na 1 (001). Pole operacji się nie zmienia i wynosi 0 (0000).



Rysunek 30: Klient przesyła pakiet z wiadomością „!invite” tym samym rozłączając się z serwerem. Kod operacji dla tej komendy wynosi 4 (0100). Pole odpowiedzi wynosi 0 (000).



Rysunek 31: Serwer otrzymuje pakiet od klienta numer 1 o rozłączeniu i wysyła klientowi numer 2 pakiet z wiadomością: „SERVER: Second user left the chat.”. Kod odpowiedzi serwera wynosi 5 (101). Kod operacji nie zmienia się w stosunku do pakietu wysłanego do serwera od klienta 1 o rozłączeniu.



Rysunek 32: Drugi klient wysyła do serwera wiadomośd „!exit” tym samym koocząc połączenie. Kod dla operacji !exit wynosi 4 (0100). Kod odpowiedzi wynosi 0 (000).

5. ODPOWIEDZI NA PYTANIA POSTAWIONE W SEKCJI *„ZADANIA SZCZEGÓŁOWE”*.

Zadania szczegółowe

1. Przygotuj implementacje protokołu komunikacyjnego, aplikacji klienckiej oraz aplikacji serwerowej w dowolnym języku wysokiego poziomu.

Protokół, aplikacja kliencka oraz aplikacja serwerowa została zaimplementowana w języku Java. Link do wersji źródłowej wraz z komentarzami znajduje się na czwartej stronie (zadanie 3).

2. Przetestuj połączenie pomiędzy programami, rejestrując całość transmisji. Przeanalizuj przechwycone dane. Czy przesłane dane są w pełni binarne?

Całość transmisji została przechwycona. Zostanie ona przesłana w pliku o nazwie dumpfile.pcap (ip: 127.0.0.1, port: 1234) wraz ze sprawozdaniem. Przesyłane dane są w pełni binarne.

3. Określ teoretyczną oraz rzeczywistą wielkość komunikatów. Czy rozmiar jest zależny od przesyłanych danych? Czy istnieje możliwość łatwej rozbudowy protokołu?

Wielkość pakietu zależy od długości danych. W naszym przypadku wielkość pakietu wynosi: `10 + message.length()` bajtów. Zgodnie z tabelką przedstawiającą długość pakietu (strona 2) można zauważyć, że gdybyśmy wysyłali pusty tekst wiadomość zajmowałaby 10 bajtów (jest to minimum jaki trzeba przesłać). Z każdym wysłanym znakiem długość pakietu zwiększa się o 1 bajt. Rozbudowa protokołu wiązałaby się z zakodowaniem kolejnej ilości danych na poszczególnych miejscach w pakiecie. W naszym protokole nie korzystamy z biblioteki Bitset więc zakodowanie danego pola w danym miejscu mogłoby wiązać się ze sporymi komplikacjami. Konieczne byłoby także zaktualizowanie funkcji kodującej oraz dekodującej.