

**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.**

8.

**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:
  - 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,
  - zakończenie połączenia.
- serwera:
  - wygenerowanie identyfikatora sesji, o informowanie klienta, czy drugi węzeł jest osiągalny:
    - w przypadku braku osiągalności, należy zwrócić błąd.
  - 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).**

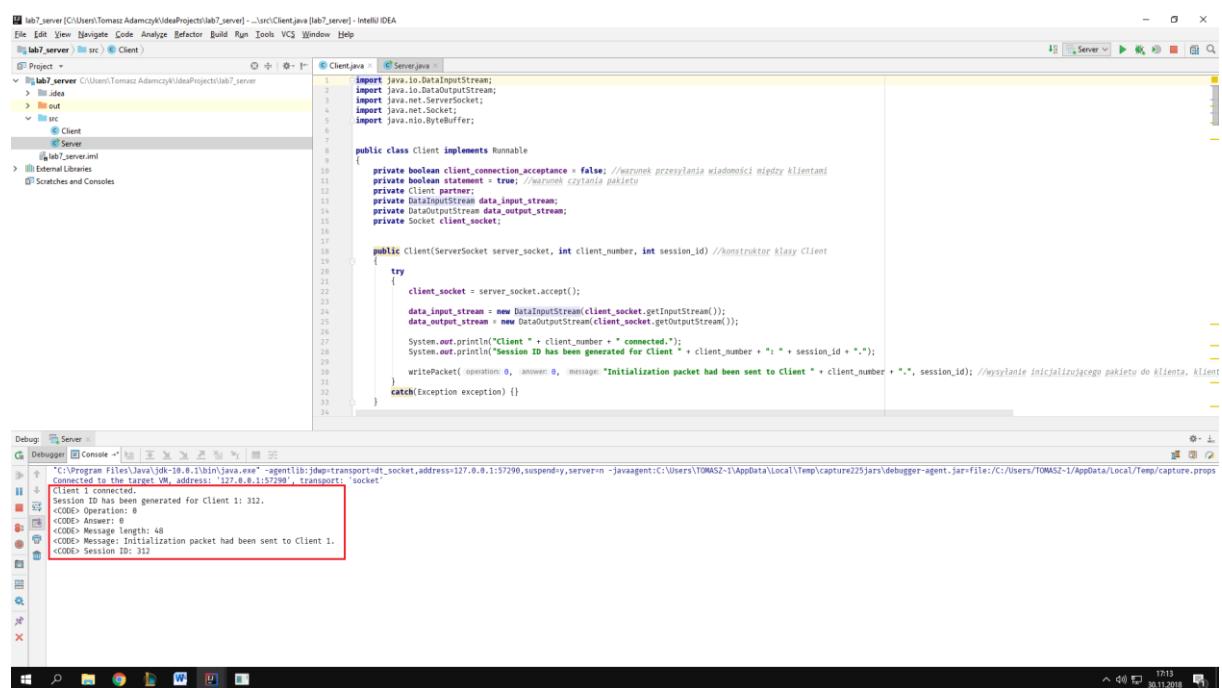
## Aplikacja klienta:

- [https://github.com/WangHoHan/chatapplicationusingbinaryprotocol/blob/master/lab7\\_client/src/Client.java](https://github.com/WangHoHan/chatapplicationusingbinaryprotocol/blob/master/lab7_client/src/Client.java)

#### Aplikacja serwera (zawiera dwa pliki typu java):

- [https://github.com/WangHoHan/chatapplicationusingbinaryprotocol/blob/master/lab7\\_server/src/Client.java](https://github.com/WangHoHan/chatapplicationusingbinaryprotocol/blob/master/lab7_server/src/Client.java)
  - [https://github.com/WangHoHan/chatapplicationusingbinaryprotocol/blob/master/lab7\\_server/src/Server.java](https://github.com/WangHoHan/chatapplicationusingbinaryprotocol/blob/master/lab7_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 wartość.

```

1 import java.io.DataInputStream;
2 import java.io.DataOutputStream;
3 import java.net.ServerSocket;
4 import java.net.Socket;
5 import java.nio.ByteBuffer;
6
7
8 public class Client implements Runnable
9 {
10     private boolean client_connection_acceptance = false; //warunek przesyłania wiadomości między klientem
11     private boolean statement = true; //warunek czytania pakietu
12     private Client partner;
13     private DataInputStream data_input_stream;
14     private DataOutputStream data_output_stream;
15     private Socket client_socket;
16
17     public Client(ServerSocket server_socket, int client_number, int session_id) //konstruktor klasy Client
18     {
19         try
20         {
21             client_socket = server_socket.accept();
22
23             data_input_stream = new DataInputStream(client_socket.getInputStream());
24             data_output_stream = new DataOutputStream(client_socket.getOutputStream());
25
26             System.out.println("Client " + client_number + " connected.");
27             System.out.println("Session ID has been generated for Client " + client_number + ":" + session_id + ".");
28
29             writePacket( operation: 0, answer: 0, message:"Initialization packet had been sent to Client " + client_number + ".*", session_id); //wysyłanie inicjalizującego pakietu do klienta, klient
30
31         } catch(Exception exception) {}
32     }
33
34 }

```

Debug: Server

```

Connected to the target VM, address: '127.0.0.1:57290', transport: 'socket'
Session ID has been generated for Client 1: 312.
<CODE> Operation: 0
<CODE> Answer: 0
<CODE> Length: 48
<CODE> Message: Initialization packet had been sent to Client 1.
<CODE> Session ID: 312
Client 1 connected.
Session ID has been generated for Client 2: 401.
<CODE> Operation: 0
<CODE> Answer: 0
<CODE> Length: 48
<CODE> Message: Initialization packet had been sent to Client 2.
<CODE> Session ID: 401

```

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 wartość.

```

1 import java.io.DataInputStream;
2 import java.io.DataOutputStream;
3 import java.net.Socket;
4 import java.nio.ByteBuffer;
5 import java.util.Scanner;
6
7
8 public class Client implements Runnable
9 {
10     private static boolean accept_statement = false; //flaga komendy /accept
11     private static boolean invite_statement = true; //flaga komendy /invite
12     private static boolean reject_statement = false; //flaga komendy /reject
13     private boolean statement = true; //warunek czytania pakietu
14     private DataInputStream data_input_stream;
15     private DataOutputStream data_output_stream;
16     private long session_id; //id sesji
17
18     public Client(String ip, int port) //konstruktor klasy Client
19     {
20         try
21         {
22             Socket client_socket = new Socket(ip, port);
23
24             data_input_stream = new DataInputStream(client_socket.getInputStream());
25             data_output_stream = new DataOutputStream(client_socket.getOutputStream());
26         } catch(Exception exception) {}
27     }
28
29     @Override
30     public void run() //metoda czyszcząca pakietę tak dugo jak nadchodzi
31     {
32         ...
33     }
34 }

```

Debug: Client

```

Connected to the target VM, address: '127.0.0.1:57295', transport: 'socket'
!invite
INFO: You have invited second user to the chat.

```

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

```

lab7_server [C:\Users\Tomasz Adamczyk\IdeaProjects\lab7_server] - ...src\Client.java [lab7_server] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
lab7_server [src] Client
Project
lab7_server [C:\Users\Tomasz Adamczyk\IdeaProjects\lab7_server]
  > idea
  > out
  < src
    < Client
      < Server
        lab7_server.iml
External Libraries
Scratches and Consoles

Client.java
1 import java.io.DataInputStream;
2 import java.io.DataOutputStream;
3 import java.net.ServerSocket;
4 import java.net.Socket;
5 import java.nio.ByteBuffer;
6
7
8 public class Client implements Runnable
9 {
10     private boolean client_connection_acceptance = false; //warunek przesyłania wiadomości między klientami
11     private boolean statement_true; //warunek czytania pakietu
12     private Client partner;
13     private DataInputStream data_input_stream;
14     private DataOutputStream data_output_stream;
15     private Socket client_socket;
16
17
18     public Client(ServerSocket server_socket, int client_number, int session_id) //konstruktor klasy Client
19     {
20         try
21         {
22             client_socket = server_socket.accept();
23
24             data_input_stream = new DataInputStream(client_socket.getInputStream());
25             data_output_stream = new DataOutputStream(client_socket.getOutputStream());
26
27             System.out.println("Client " + client_number + " connected.");
28             System.out.println("Session ID has been generated for Client " + client_number + ":" + session_id + ".");
29
30             writePacket(operation_0, answer_0, message);
31
32         } catch(Exception exception) {}
33     }
34 }

Debug: Server
Debugger Console
<CODE> Message: Initialization packet had been sent to Client 1.
<CODE> Session ID: 312
<CODE> Client ID has been generated for Client 2: 481.
<CODE> Operation: 0
<CODE> Answer: 0
<CODE> Message length: 48
<CODE> Message: Initialization packet had been sent to Client 2.
<CODE> Session ID: 481
<CODE> Operation: 5
<CODE> Answer: 0
<CODE> Message length: 49
<CODE> Message: SERVER: Second user have invited you to the chat.
<CODE> Operation: 5
<CODE> Answer: 0
<CODE> Message length: 7
<CODE> Message: SERVER: I invite
<CODE> Session ID: 312

```

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)).

```

lab7_client [C:\Users\Tomasz Adamczyk\IdeaProjects\lab7_client] - ...src\Client.java [lab7_client] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
lab7_client [src] Client
Project
lab7_client [C:\Users\Tomasz Adamczyk\IdeaProjects\lab7_client]
  > idea
  > java
  > out
  < src
    < Client
      < lab7_client.iml
External Libraries
Scratches and Consoles

Client.java
1 import java.io.DataInputStream;
2 import java.io.DataOutputStream;
3 import java.net.Socket;
4 import java.nio.ByteBuffer;
5 import java.util.Scanner;
6
7
8 public class Client implements Runnable
9 {
10     private static boolean accept_statement = false; //flaga komendy accept
11     private static boolean invite_statement = true; //flaga komendy invite
12     private static boolean reject_statement = false; //flaga komendy reject
13     private boolean statement_true; //warunek czytania pakietu
14     private DataInputStream data_input_stream;
15     private DataOutputStream data_output_stream;
16     private long session_id; //id sesji
17
18     private Client(String ip, int port) //konstruktor klasy Client
19     {
20         try
21         {
22             Socket client_socket = new Socket(ip, port);
23
24             data_input_stream = new DataInputStream(client_socket.getInputStream());
25             data_output_stream = new DataOutputStream(client_socket.getOutputStream());
26         } catch(Exception exception) {}
27     }
28
29     @Override
30     public void run() //metoda czytająca pakietę tak dugo jak nadchodzi
31     {
32
33     }
34 }

Debug: Client
Debugger Console
C:\Program Files\Java\jdk-10.0.1\bin\java.exe -agentlib:jdwp=transport=dt_socket,address=127.0.0.1:57310,suspend=y,server=n -javaagent:C:\Users\TOMASZ-1\AppData\Local\Temp\capture127\jars\debugger-agent.jar=file:C:/Users/TOMASZ-1/AppData/Local/Temp/capture.props
Connected to the target VM, address: "127.0.0.1:57310", transport: "socket"
SERVER: Second user have invited you to the chat.

```

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.

```

lab7_client [C:\Users\Tomasz Adamczyk\IdeaProjects\lab7_client] - ...src\Client.java [lab7_client] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
lab7_client src Client
Project lab7_client C:\Users\Tomasz Adamczyk\IdeaProjects\lab7_client
> idea
> java
> out
> src
Client
lab7_client
External Libraries
Scratches and Consoles

Client.java
1 import java.io.DataInputStream;
2 import java.io.DataOutputStream;
3 import java.net.Socket;
4 import java.nio.ByteBuffer;
5 import java.util.Scanner;
6
7
8 public class Client implements Runnable
9 {
10     private static boolean accept_statement = false; // flaga komendy !accept
11     private static boolean invite_statement = true; // flaga komendy !invite
12     private static boolean reject_statement = false; // flaga komendy !reject
13     private boolean statement = true; // warunek czytania pakietu
14     private DataInputStream data_input_stream;
15     private DataOutputStream data_output_stream;
16     private long session_id; // id sesji
17
18
19     private Client(String ip, int port) // konstruktor klasy Client
20     {
21         try
22         {
23             Socket client_socket = new Socket(ip, port);
24
25             data_input_stream = new DataInputStream(client_socket.getInputStream());
26             data_output_stream = new DataOutputStream(client_socket.getOutputStream());
27         }
28         catch(Exception exception) {}
29     }
30
31
32     @Override
33     public void run() // metoda czyszcząca pakietę tak dugo jak nadchodzi
34     {
35
36
37
38
39
39     }
}

```

Java console output:

```

Connected to the target VM, address: "127.0.0.1:57310", transport: socket
SERVER: Second user have invited you to the chat.
ACCEPT
INFO: You have accepted second users invite.

```

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

```

lab7_server [C:\Users\Tomasz Adamczyk\IdeaProjects\lab7_server] - ...src\Client.java [lab7_server] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
lab7_server src Client
Project lab7_server C:\Users\Tomasz Adamczyk\IdeaProjects\lab7_server
> idea
> java
> out
> src
Client
lab7_server
External Libraries
Scratches and Consoles

Client.java
1 import java.io.DataInputStream;
2 import java.io.DataOutputStream;
3 import java.net.ServerSocket;
4 import java.net.Socket;
5 import java.nio.ByteBuffer;
6
7
8 public class Client implements Runnable
9 {
10     private boolean client_connection_acceptance = false; // warunek przesyłania wiadomości między klientami
11     private boolean statement = true; // warunek czytania pakietu
12     private Client partner;
13     private DataInputStream data_input_stream;
14     private DataOutputStream data_output_stream;
15     private Socket client_socket;
16
17
18     public Client(ServerSocket server_socket, int client_number, int session_id) // konstruktor klasy Client
19     {
20         try
21         {
22             client_socket = server_socket.accept();
23
24             data_input_stream = new DataInputStream(client_socket.getInputStream());
25             data_output_stream = new DataOutputStream(client_socket.getOutputStream());
26
27             System.out.println("Client " + client_number + " connected.");
28             System.out.println("Session ID has been generated for Client " + client_number + ":" + session_id + ".");
29
30             writePacket(operation_0, answer_0, message("Initialization packet had been sent to Client " + client_number + ":", session_id)); // wysyłanie inicjalizującego pakietu do klienta, klient
31
32         }
33         catch(Exception exception) {}
34     }
}

```

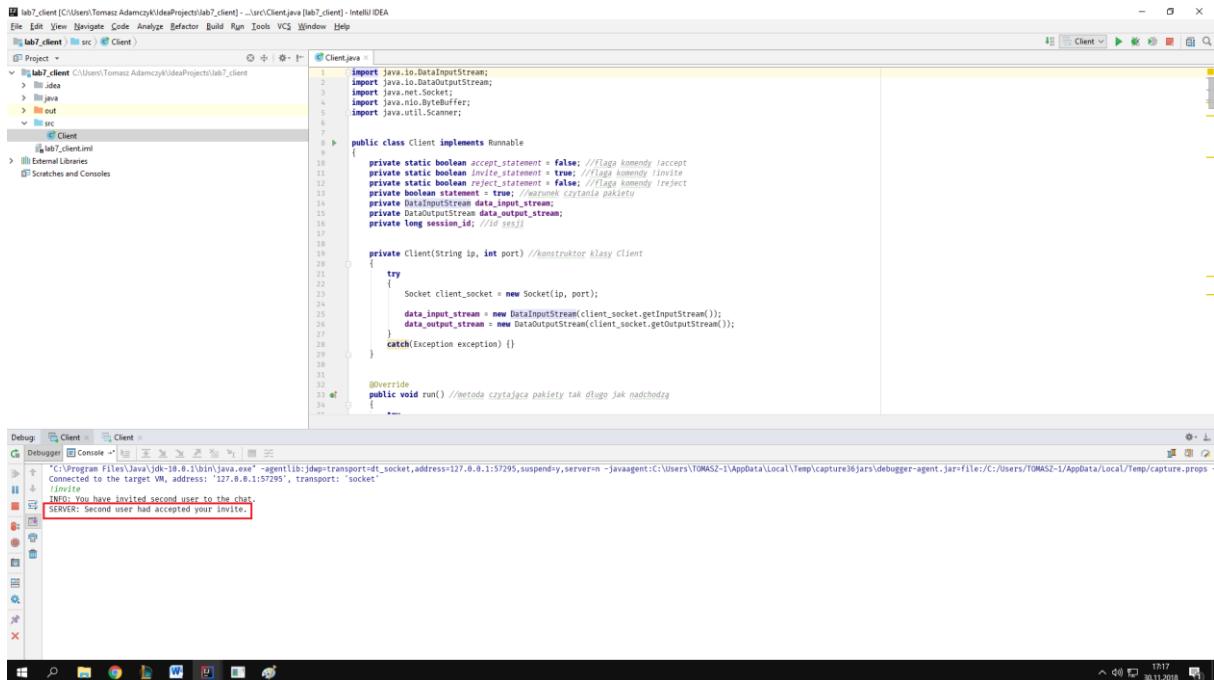
Java console output:

```

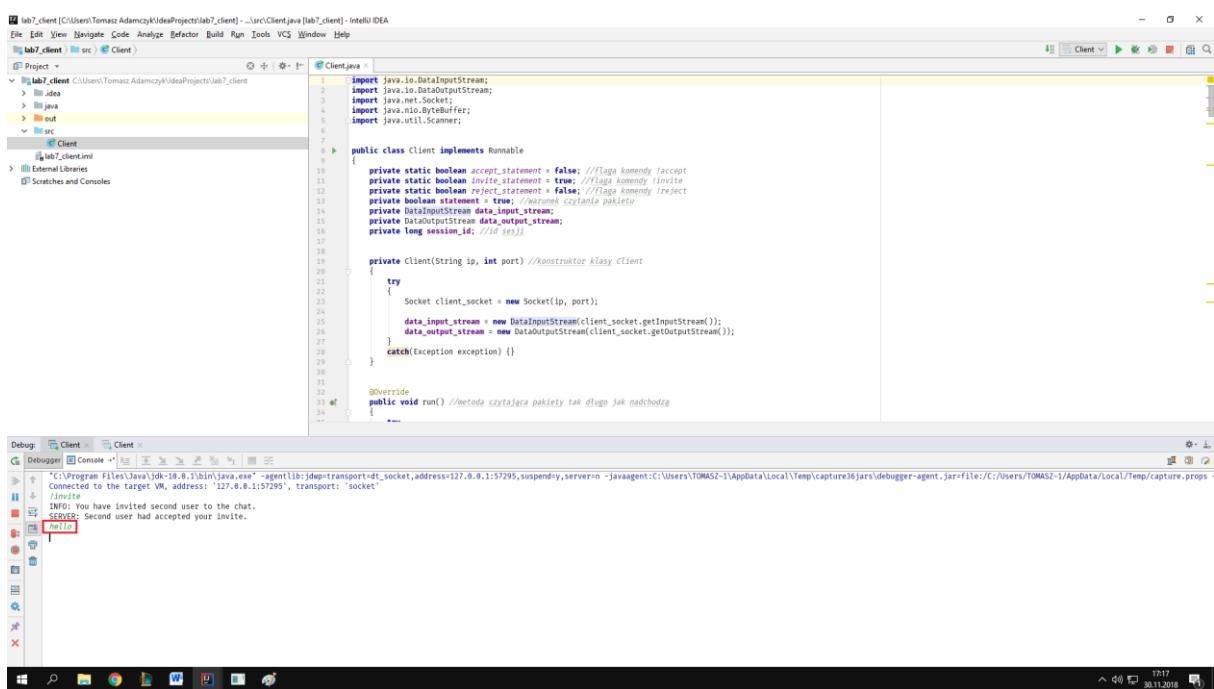
<CODE> Answer: 6
<CODE> Message length: 49
<CODE> Message: SERVER: Second user have invited you to the chat.
<CODE> Session ID: 312
<CODE> Operation: 5
<CODE> Answer: 8
<CODE> Message: linvite
<CODE> Session ID: 312
<CODE> Operation: 1
<CODE> Message length: 45
<CODE> Message: SERVER: Second user had accepted your invite.
<CODE> Session ID: 312
<CODE> Operation: 1
<CODE> Answer: 7
<CODE> Message length: 7
<CODE> Message: !accept
<CODE> Session ID: 312

```

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 wysyłania wiadomości to 0 (0000).

The screenshot shows the IntelliJ IDEA interface with the project 'lab7\_server' open. The 'Client.java' file is selected in the editor. The code implements a client socket to handle connections from clients. The 'Server.java' tab is also visible. Below the editor is the 'Console' tab, which displays the server's log output. The log shows a sequence of messages between two clients, with one client sending an invite message and the other responding with an accept message.

```

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.nio.ByteBuffer;
import java.util.Scanner;

public class Client implements Runnable
{
    private boolean client_connection_acceptance = false; //warunek przesyłania wiadomości między klientami
    private boolean statement_true; //warunek czytania pakietu
    private Client partner;
    private ServerSocket server_socket;
    private DataInputStream data_input_stream;
    private DataOutputStream data_output_stream;
    private Socket client_socket;
}

public Client(ServerSocket server_socket, int client_number, int session_id) //konstruktor klasy Client
{
    try
    {
        client_socket = server_socket.accept();
        data_input_stream = new DataInputStream(client_socket.getInputStream());
        data_output_stream = new DataOutputStream(client_socket.getOutputStream());
        System.out.println("Client " + client_number + " connected.");
        System.out.println("Session ID has been generated for Client " + client_number + ":" + session_id + ".");
        writePacket(operation_0, answer_0, message("Initialization packet had been sent to Client " + client_number + ".", session_id)); //wysyłanie inicjalizującego pakietu do klienta, klient
    }
    catch(Exception exception) {}
}

```

```

<CODE> Answer: 2
<CODE> Message length: 45
<CODE> Message: Second user had accepted your invite.
<CODE> Session ID: 312
<CODE> Operation: 1
<CODE> Answer: 4
<CODE> Message length: 7
<CODE> Message: accept
<CODE> Session ID: 312
<CODE> Operation: 0
<CODE> Message length: 5
<CODE> Message: hello
<CODE> Session ID: 312
<CODE> Operation: 0
<CODE> Answer: 0
<CODE> Message length: 5
<CODE> Message: hello
<CODE> Session ID: 312

```

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

The screenshot shows the IntelliJ IDEA interface with the project 'lab7\_client' open. The 'Client.java' file is selected in the editor. The code implements a client socket to connect to a server and handle messages. The 'Console' tab shows the client's log output, including a connection to the server and a message indicating it accepted an invitation from another user.

```

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.Socket;
import java.nio.ByteBuffer;
import java.util.Scanner;

public class Client implements Runnable
{
    private static boolean accept_statement = false; //flaga komendy accept
    private static boolean invite_statement = true; //flaga komendy invite
    private static boolean reject_statement = false; //flaga komendy reject
    private boolean statement_true; //warunek czytania pakietu
    private DataInputStream data_input_stream;
    private DataOutputStream data_output_stream;
    private long session_id; //id sesji

    private Client(String ip, int port) //konstruktor klasy Client
    {
        try
        {
            Socket client_socket = new Socket(ip, port);
            data_input_stream = new DataInputStream(client_socket.getInputStream());
            data_output_stream = new DataOutputStream(client_socket.getOutputStream());
        }
        catch(Exception exception) {}
    }

    @Override
    public void run() //metoda czytająca pakietę tak dugo jak nadchodzi
    {
        ...
    }
}

```

```

C:\Program Files\Java\jdk-10.0.1\bin\java.exe -agentlib:jdwp=transport=dt_socket,address=127.0.0.1:57310,suspend=y,server=n -javaagent:C:\Users\TOMASZ-1\AppData\Local\Temp\capture127\jars\debugger-agent.jar=file:C:/Users/TOMASZ-1/AppData/Local/Temp/capture.props
Connected to the target VM, address: "127.0.0.1:57310", transport: "socket"
SERVER: Second user have invited you to the chat.
INFO: You have accepted second users invite.
friend: hello

```

Rysunek 11: Drugi klient otrzymuje wiadomość wyslaną przez drugą osobę. Wyświetla mu się przed wiadomością „friend:”, aby nie pomylić osoby wysyłającej wiadomość. „friend” nie jest częścią wiadomości.

```

lab7.client [C:\Users\Tomasz Adamczyk\IdeaProjects\lab7_client] - ...src\Client.java [lab7_client] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
lab7.client src Client
Project lab7_client C:\Users\Tomasz Adamczyk\IdeaProjects\lab7_client
> idea
> java
> out
> src
Client
lab7_client
External Libraries
Scratches and Consoles

Client.java
1 import java.io.DataInputStream;
2 import java.io.DataOutputStream;
3 import java.net.Socket;
4 import java.nio.ByteBuffer;
5 import java.util.Scanner;
6
7
8 public class Client implements Runnable
9 {
10     private static boolean accept_statement = false; // flaga komendy accept
11     private static boolean invite_statement = true; // flaga komendy invite
12     private static boolean reject_statement = false; // flaga komendy reject
13     private boolean statement = true; // warunek czytania pakietu
14     private DataInputStream data_input_stream;
15     private DataOutputStream data_output_stream;
16     private long session_id; // id sesji
17
18
19     private Client(String ip, int port) // konstruktor klasy Client
20     {
21         try
22         {
23             Socket client_socket = new Socket(ip, port);
24
25             data_input_stream = new DataInputStream(client_socket.getInputStream());
26             data_output_stream = new DataOutputStream(client_socket.getOutputStream());
27         }
28         catch(Exception exception) {}
29     }
30
31     @Override
32     public void run() // metoda czyszcząca pakietę tak dugo jak nadchodzi
33     {
34
35     }
}

```

Java console output:

```

C:\Program Files\Java\jdk-10.0.1\bin\java.exe" -agentlib:jdwp=transport=dt_socket,address=127.0.0.1:57310,suspend=y,server=n -javaagent:C:\Users\TOMASZ-1\AppData\Local\Temp\capture127\jars\debugger-agent.jar=file:C:/Users/TOMASZ-1/AppData/Local/Temp/capture.props
Connected to the target VM, address: '127.0.0.1:57310', transport: 'socket'
INFO: You have accepted second users invite.
friend: hello
world

```

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

```

lab7.server [C:\Users\Tomasz Adamczyk\IdeaProjects\lab7_server] - ...src\Client.java [lab7_server] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
lab7.server src Client
Project lab7_server C:\Users\Tomasz Adamczyk\IdeaProjects\lab7_server
> idea
> java
> out
> src
Client
lab7_server
External Libraries
Scratches and Consoles

Client.java
1 import java.io.DataInputStream;
2 import java.io.DataOutputStream;
3 import java.net.ServerSocket;
4 import java.net.Socket;
5 import java.nio.ByteBuffer;
6
7
8 public class Client implements Runnable
9 {
10     private boolean client_connection_acceptance = false; // warunek przesyłania wiadomości między klientami
11     private boolean statement = true; // warunek czytania pakietu
12     private Client partner;
13     private DataInputStream data_input_stream;
14     private DataOutputStream data_output_stream;
15     private Socket client_socket;
16
17
18     public Client(ServerSocket server_socket, int client_number, int session_id) // konstruktor klasy Client
19     {
20         try
21         {
22             client_socket = server_socket.accept();
23
24             data_input_stream = new DataInputStream(client_socket.getInputStream());
25             data_output_stream = new DataOutputStream(client_socket.getOutputStream());
26
27             System.out.println("Client " + client_number + " connected.");
28             System.out.println("Session ID has been generated for Client " + client_number + ":" + session_id + ".");
29
30             writePacket(operation_0, answer_0, message("Initialization packet had been sent to Client " + client_number + ".", session_id)); // wysyłanie inicjalizującego pakietu do klienta, klient
31
32         }
33         catch(Exception exception) {}
34     }
}

```

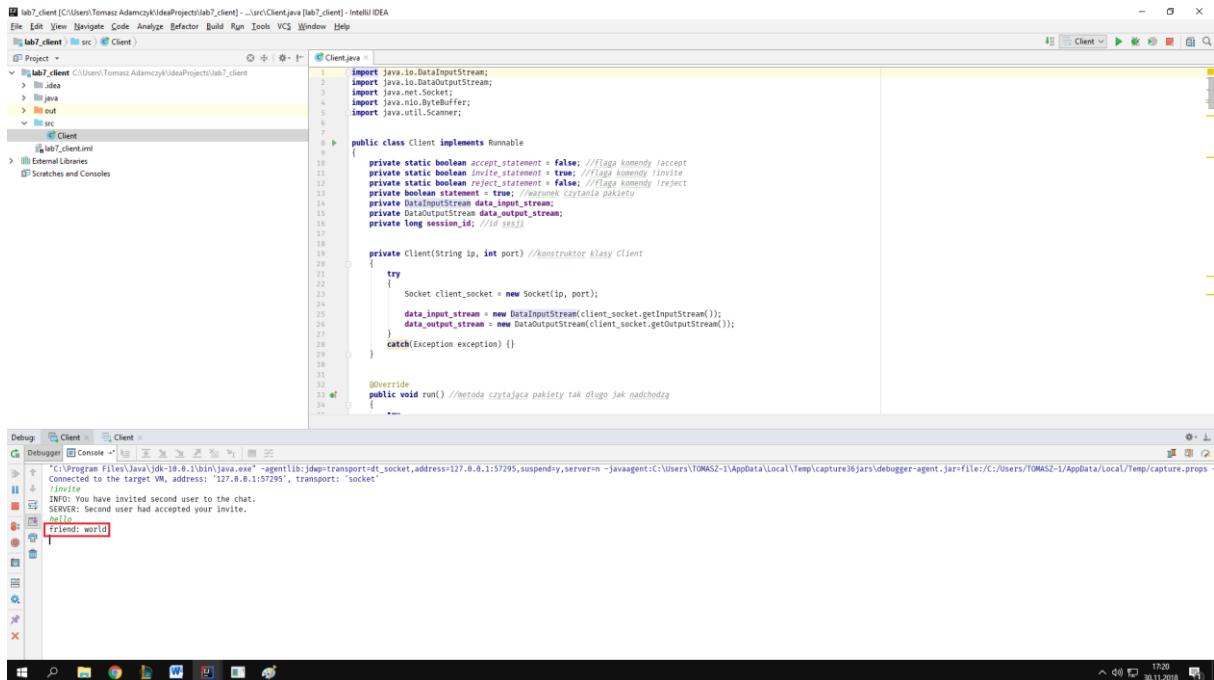
Java console output:

```

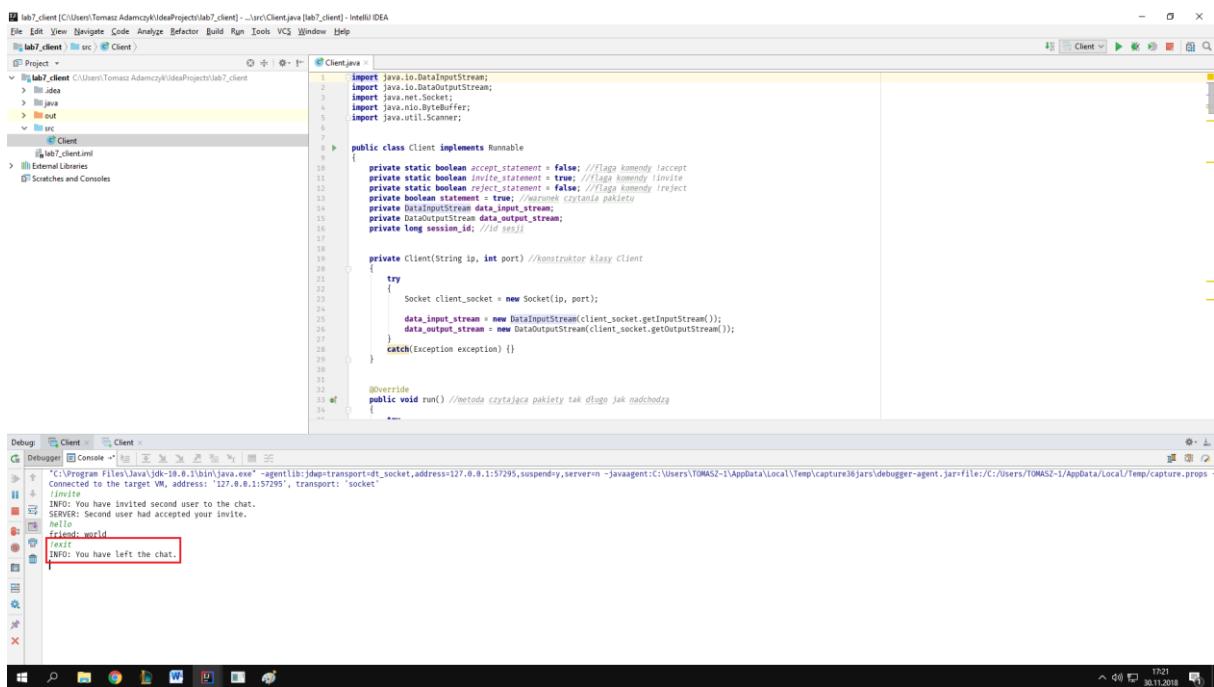
<CODE> Answer: 1
<CODE> Message length: 5
<CODE> Message: hello
<CODE> Session ID: 332
<CODE> Operation: 0
<CODE> Answer: 4
<CODE> Message length: 5
<CODE> Message: world
<CODE> Session ID: 332
<CODE> Operation: 0
<CODE> Answer: 4
<CODE> Message length: 5
<CODE> Message: world
<CODE> Session ID: 332

```

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



Rysunek 14: Klient numer 1 otrzymuje wiadomość 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.

```

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.nio.ByteBuffer;
import java.util.Scanner;

public class Client implements Runnable
{
    private boolean client_connection_acceptance = false; //warunek przesyłania wiadomości między klientami
    private boolean statement_true; //warunek czekania na pakiet
    private Client partner;
    private DataInputStream data_input_stream;
    private DataOutputStream data_output_stream;
    private Socket client_socket;
}

public Client(ServerSocket server_socket, int client_number, int session_id) //konstruktor klasy Client
{
    try
    {
        client_socket = server_socket.accept();
        data_input_stream = new DataInputStream(client_socket.getInputStream());
        data_output_stream = new DataOutputStream(client_socket.getOutputStream());
        System.out.println("Client " + client_number + " connected.");
        System.out.println("Session ID has been generated for Client " + client_number + ":" + session_id);
        writePacket(operation_0, answer_0, message("Initialization packet had been sent to Client " + client_number + ".", session_id));
    }
    catch(Exception exception) {}
}

```

Debug: Server

```

<CODE> Answer: 1
<CODE> Message length: 5
<CODE> Message: 
<CODE> Session ID: 332
<CODE> Operation: 0
<CODE> Answer: 0
<CODE> Message length: 5
<CODE> Message: world
<CODE> Session ID: 332
<CODE> Operation: 4
<CODE> Message length: 34
<CODE> Message: SERVER: Second user left the chat.
<CODE> Operation: 4
<CODE> Answer: 0
<CODE> Message length: 5
<CODE> Message: 
<CODE> Session ID: 332

```

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)).

```

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.Socket;
import java.nio.ByteBuffer;
import java.util.Scanner;

public class Client implements Runnable
{
    private static boolean accept_statement = false; //flaga komendy accept
    private static boolean invite_statement = true; //flaga komendy invite
    private static boolean reject_statement = false; //flaga komendy reject
    private static String friend_name; //nazwa przyjaciela pakietu
    private DataInputStream data_input_stream;
    private DataOutputStream data_output_stream;
    private long session_id; //id sesji

    private Client(String ip, int port) //konstruktor klasy Client
    {
        try
        {
            Socket client_socket = new Socket(ip, port);
            data_input_stream = new DataInputStream(client_socket.getInputStream());
            data_output_stream = new DataOutputStream(client_socket.getOutputStream());
        }
        catch(Exception exception) {}
    }

    @Override
    public void run() //metoda czyszcząca pakiety tak dugo jak nadchodzi
    {
        ...
    }
}

Connected to the target VM, address: '127.0.0.1:57310', transport: 'socket'.
/accept
INFO: You have accepted second users invite.
friend: hello
/SERVER: Second user left the chat.

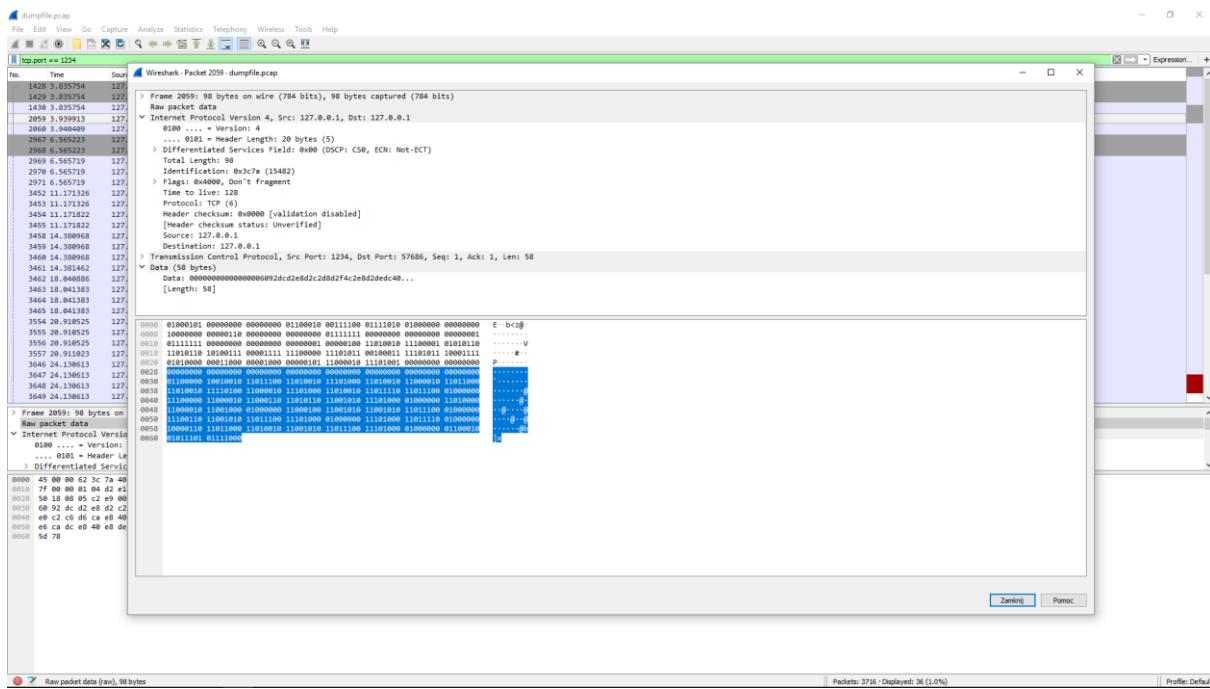
```

Rysunek 17: Klient numer 2 dostaje wiadomość 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.

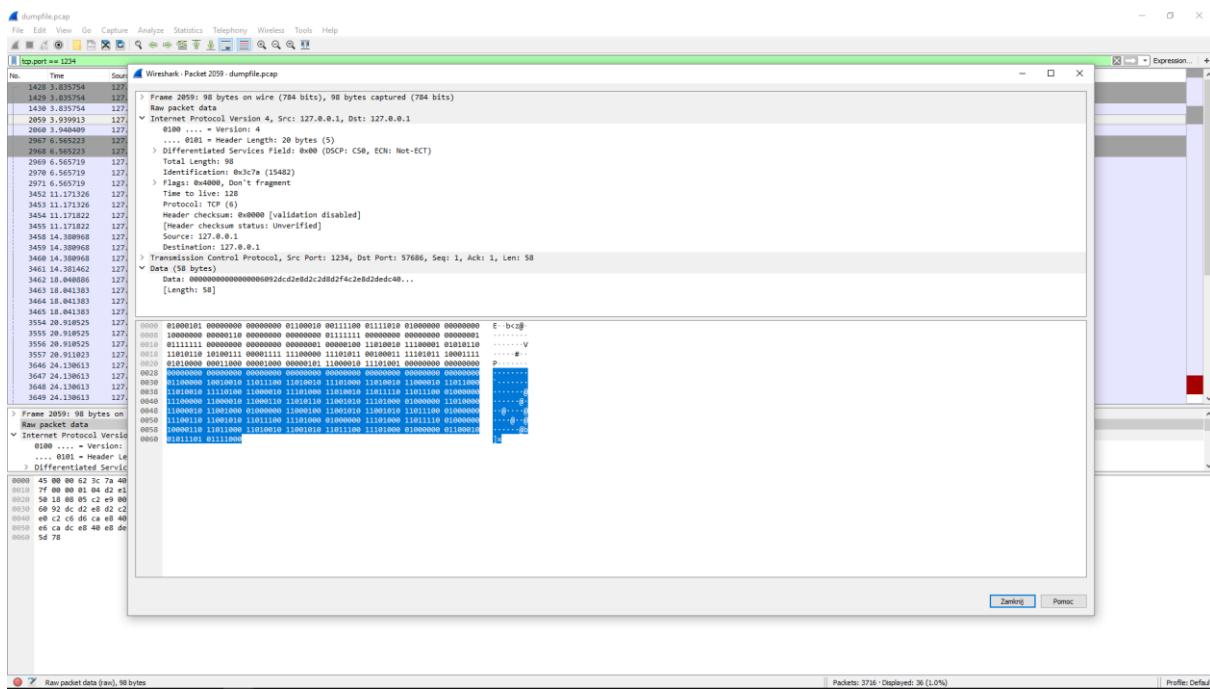
```

lab7_client [C:\Users\Tomasz Adamczyk\IdeaProjects\lab7_client] - ...src\Client.java [lab7_client] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
lab7_client src Client
Project lab7_client C:\Users\Tomasz Adamczyk\IdeaProjects\lab7_client
> idea
> java
> out
> src
Client
lab7_client
External Libraries
Scratches and Consoles

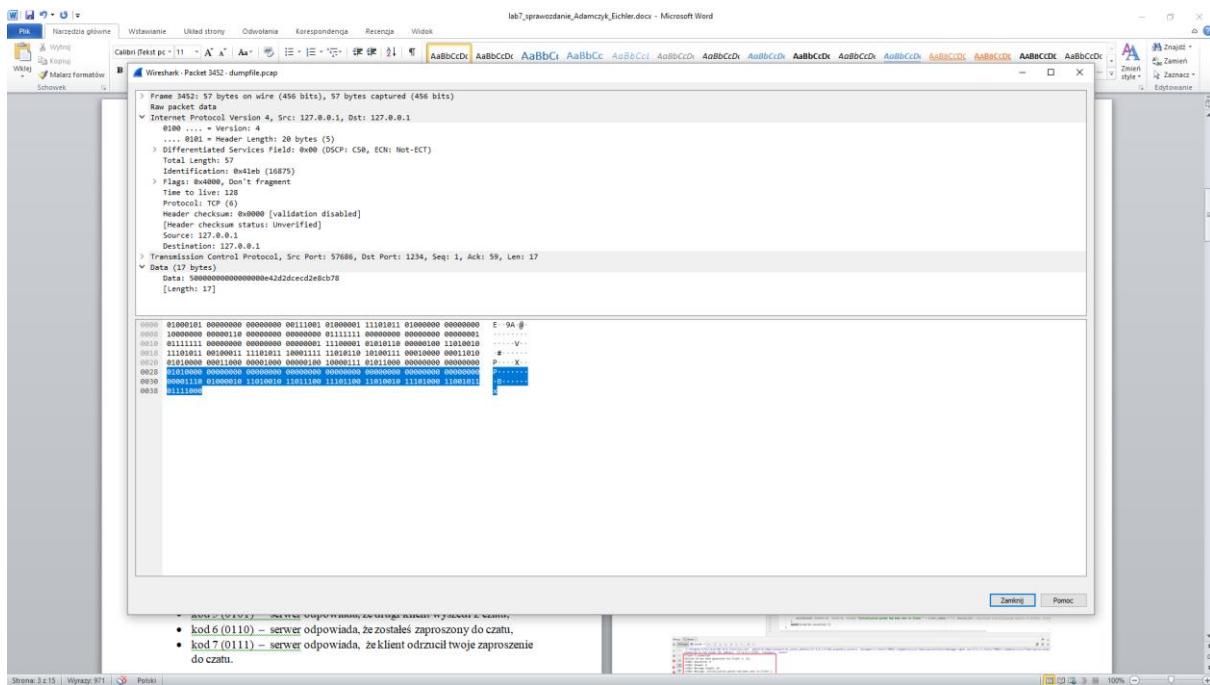
Client.java
1 import java.io.DataInputStream;
2 import java.io.DataOutputStream;
3 import java.net.Socket;
4 import java.nio.ByteBuffer;
5 import java.util.Scanner;
6
7
8 public class Client implements Runnable
9 {
10     private static boolean accept_statement = false; // flaga komendy accept
11     private static boolean invite_statement = true; // flaga komendy invite
12     private static boolean reject_statement = false; // flaga komendy reject
13     private boolean statement = true; // warunek czytania pakietu
14     private DataInputStream data_input_stream;
15     private DataOutputStream data_output_stream;
16     private long session_id; // id sesji
17
18
19     private Client(String ip, int port) // konstruktor klasy Client
20     {
21         try
22         {
23             Socket client_socket = new Socket(ip, port);
24
25             data_input_stream = new DataInputStream(client_socket.getInputStream());
26             data_output_stream = new DataOutputStream(client_socket.getOutputStream());
27         }
28         catch(Exception exception) {}
29     }
30
31
32     @Override
33     public void run() // metoda czyszcząca pakietę tak dugo jak nadchodzi
34     {
35
36
37
38
39
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
59
60
61
62
63
64
65
66
67
68
69
69
70
71
72
73
74
75
76
77
78
79
79
80
81
82
83
84
85
86
87
88
89
89
90
91
92
93
94
95
96
97
98
99
99
100
101
102
103
104
105
106
107
108
109
109
110
111
112
113
114
115
116
117
118
119
119
120
121
122
123
124
125
126
127
128
129
129
130
131
132
133
134
135
136
137
138
139
139
140
141
142
143
144
145
146
147
148
149
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
168
169
169
170
171
172
173
174
175
176
177
178
179
179
180
181
182
183
184
185
186
187
188
189
189
190
191
192
193
194
195
196
197
198
199
199
200
201
202
203
204
205
206
207
208
209
209
210
211
212
213
214
215
216
217
218
219
219
220
221
222
223
224
225
226
227
228
229
229
230
231
232
233
234
235
236
237
238
239
239
240
241
242
243
244
245
246
247
248
249
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
267
268
269
269
270
271
272
273
274
275
275
276
277
278
279
279
280
281
282
283
284
285
285
286
287
288
289
289
290
291
292
293
294
295
295
296
297
298
299
299
300
301
302
303
304
305
305
306
307
308
309
309
310
311
312
313
314
315
315
316
317
318
319
319
320
321
322
323
324
325
325
326
327
328
329
329
330
331
332
333
334
335
335
336
337
338
339
339
340
341
342
343
344
345
345
346
347
348
349
349
350
351
352
353
354
355
355
356
357
358
359
359
360
361
362
363
364
364
365
366
367
368
368
369
370
371
372
373
373
374
375
376
376
377
378
379
379
380
381
382
383
383
384
385
386
386
387
388
389
389
390
391
392
393
393
394
395
395
396
397
398
398
399
399
400
401
402
402
403
404
404
405
406
406
407
408
408
409
409
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
```



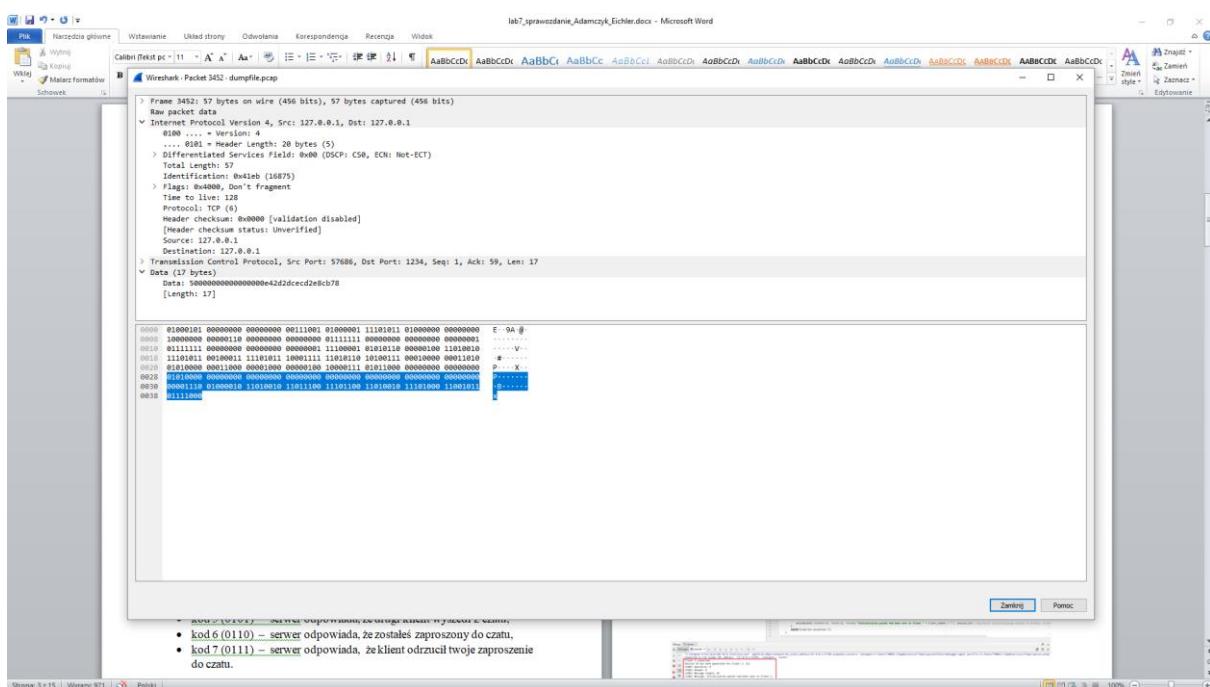
Rysunek 20: Pakiet inicjalizujący dla klienta 1. Wysyła on id sesji: 312 (101111000) – zapisane na 9 bitach oraz wiadomość: „Initialization packet had been sent to Client 1”. Pole operacji oraz odpowiedzi ustawione są na 0. Id sesji może różnić się od tych na wyższych rysunkach (serwer wylosował inne),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



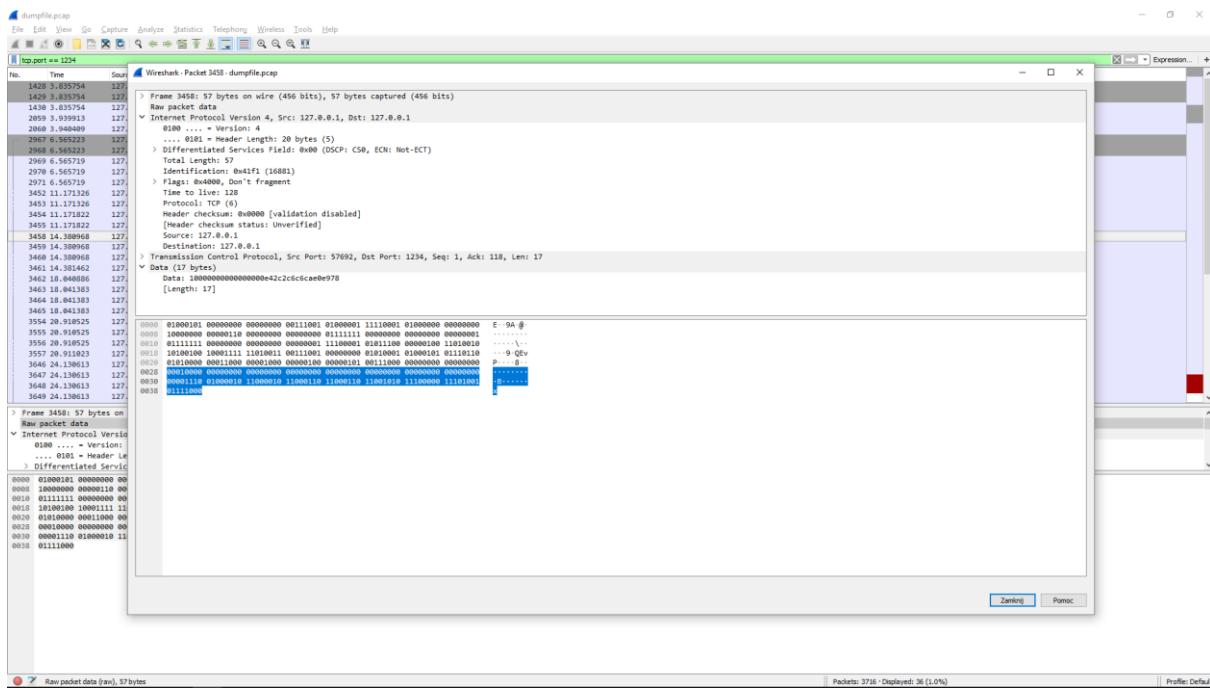
Rysunek 21: Pakiet inicjalizujący dla klienta 2. Wysyła on id sesji: 492 (111101100) – zapisane na 9 bitach oraz wiadomość: „Initialization packet had been sent to Client 2”. Pole operacji oraz odpowiedzi ustawione są na 0. Id sesji może różnić się od tych na wyższych rysunkach (serwer wylosował inne),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



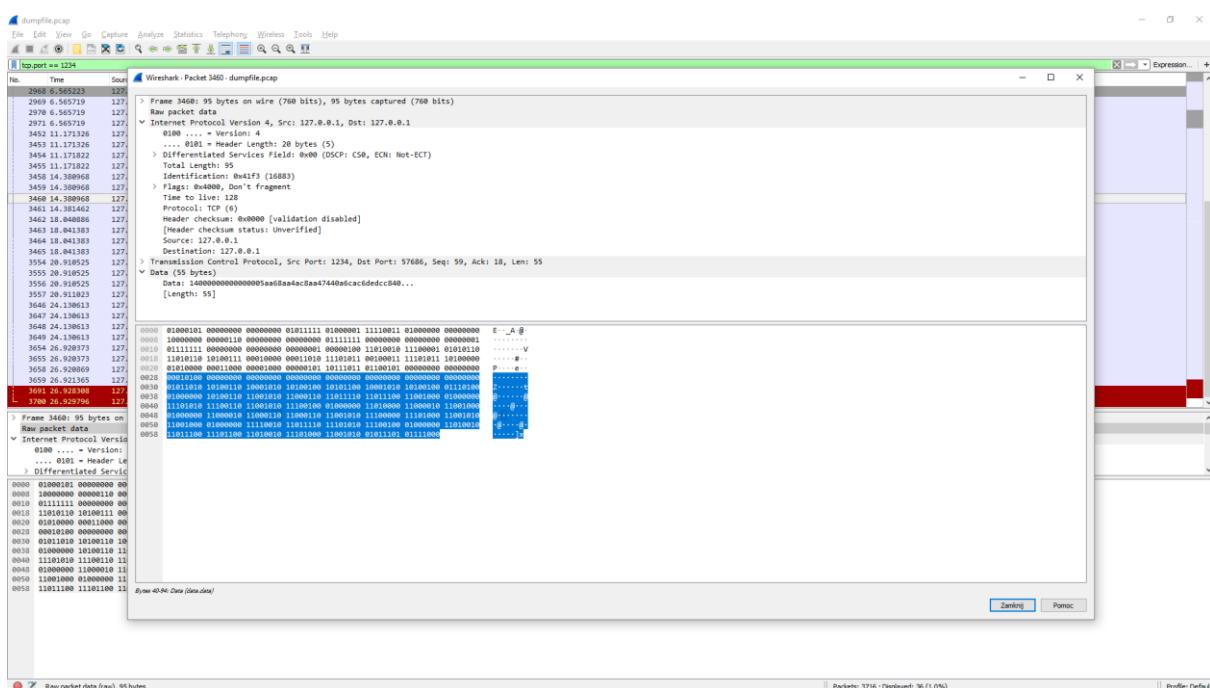
Rysunek 22: Zaproszenie drugiego klienta do czatu. Pole operacji: 5 (0101), pole odpowiedzi 0 (000), wiadomość: !invite.



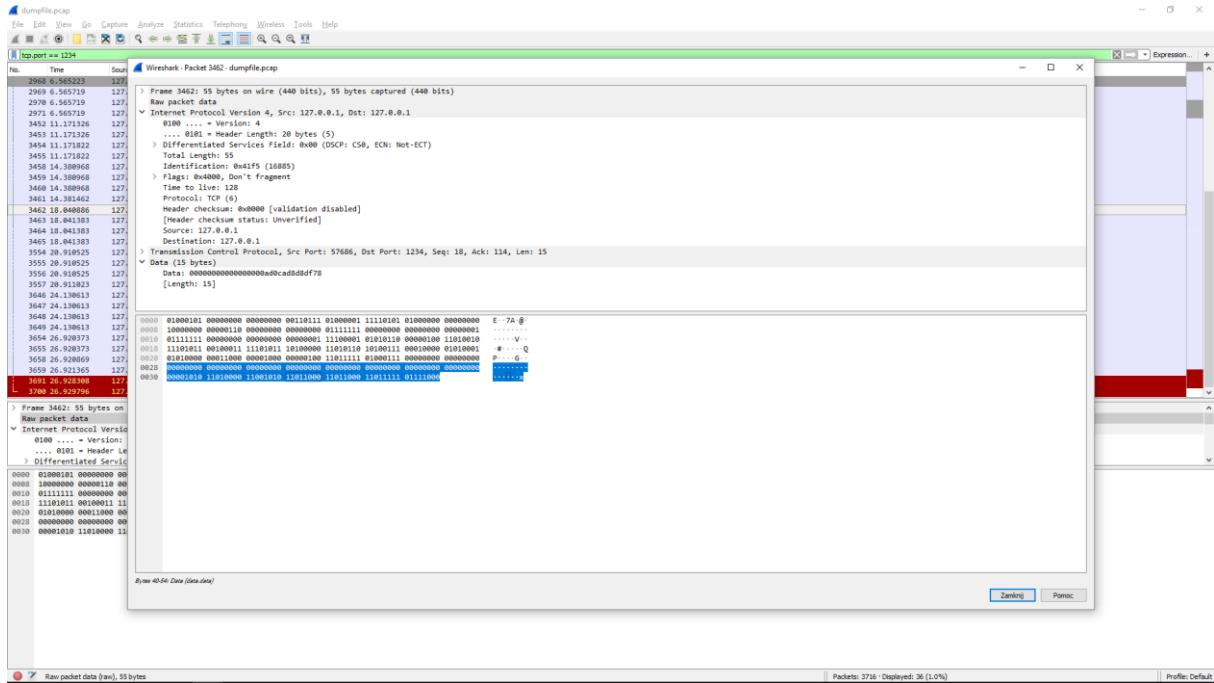
Rysunek 23: Serwer dostaje wiadomość od klienta numer 1 (!invite) i wysyła do klienta numer 2 wiadomość: „SERVER: Second user have invited you to the chat.”, pole operacji nie zmienia się w stosunku do wiadomości od klienta 1. Pole odpowiedzi ustawia się na 6 (110).



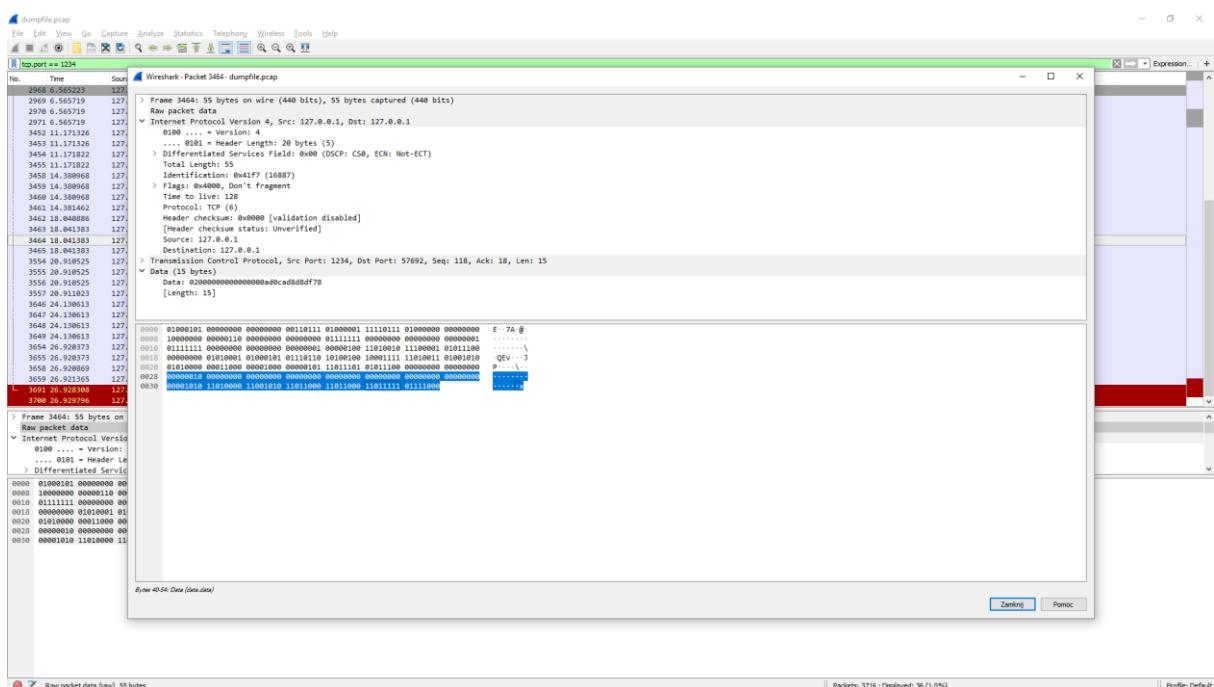
Rysunek 24: Klient akceptuje zaproszenie od klienta 1 i wysyła do serwera wiadomość: „!accept”. Pole operacji ustawia na 1 (0001), pole odpowiedzi 0 (000).



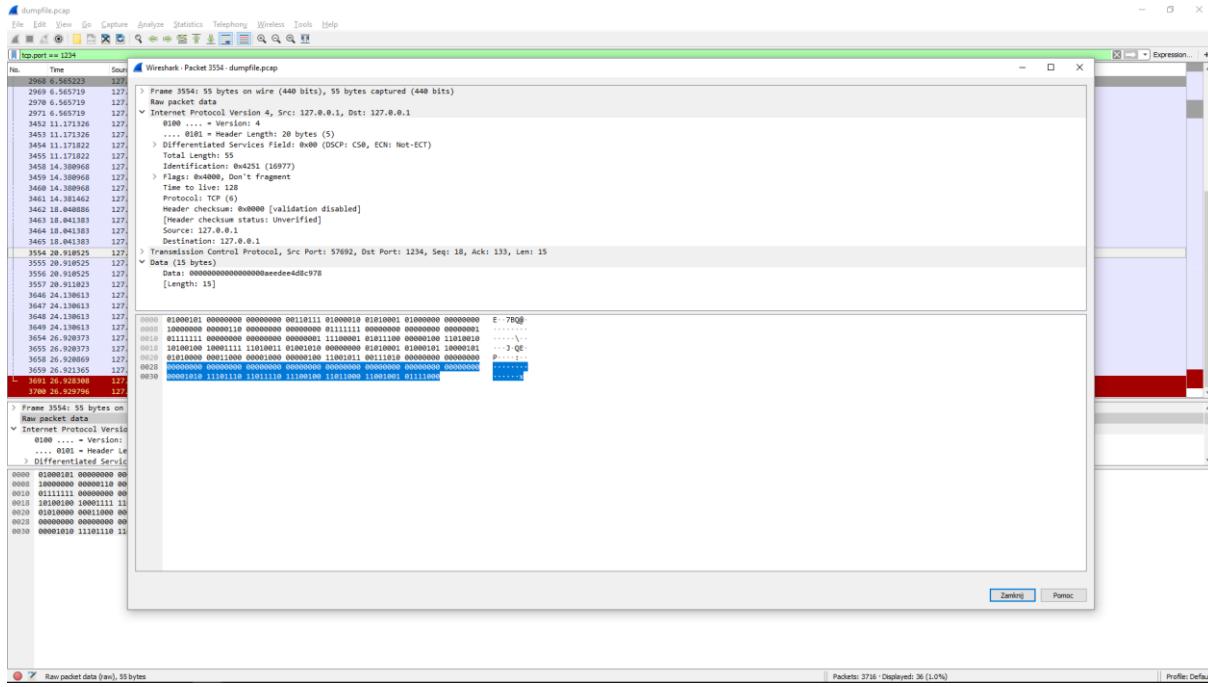
Rysunek 25: Serwer otrzymuje informację o akceptacji od klienta numer 2. Serwer wysyła pakiet z wiadomością: „ **SERVER: Second user had accepted your invite.**”. Pole operacji się nie zmienia w stosunku do poprzedniego pakietu otrzymanego od klienta numer 2, pole odpowiedzi wynosi 2 (010).



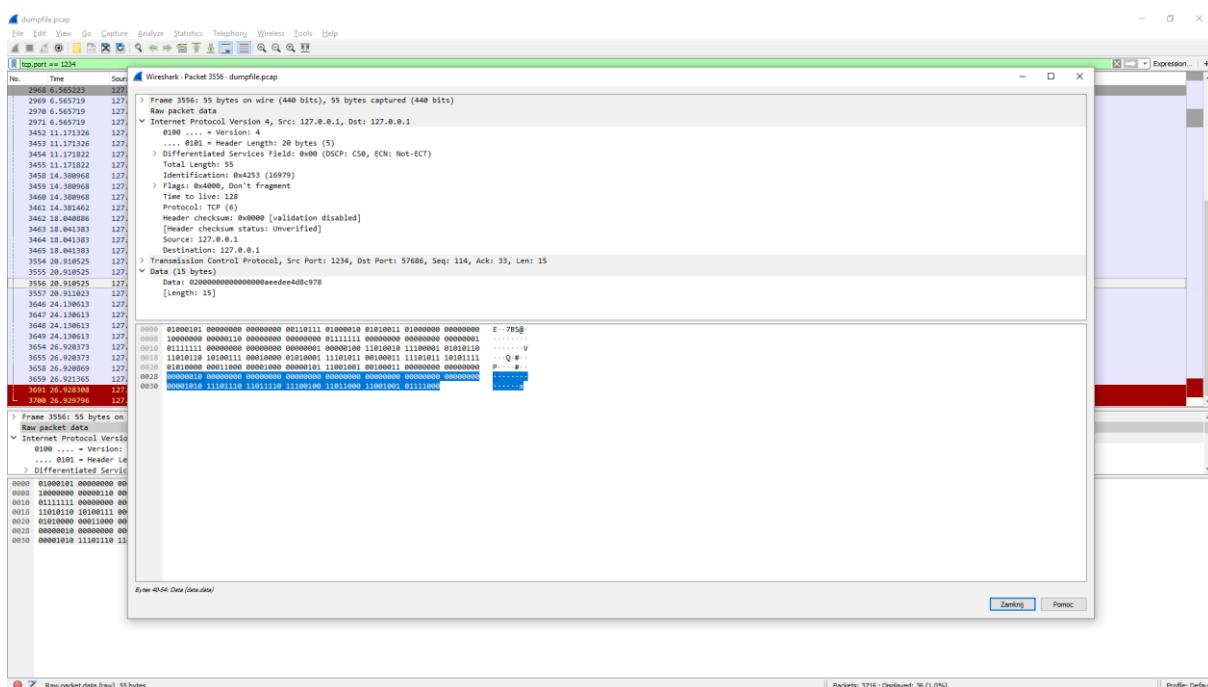
Rysunek 26: Klient wysyła wiadomość: „hello” do drugiego klienta. Pole operacji wynosi 0 (0000) oraz pole odpowiedzi wynosi 0 (000).



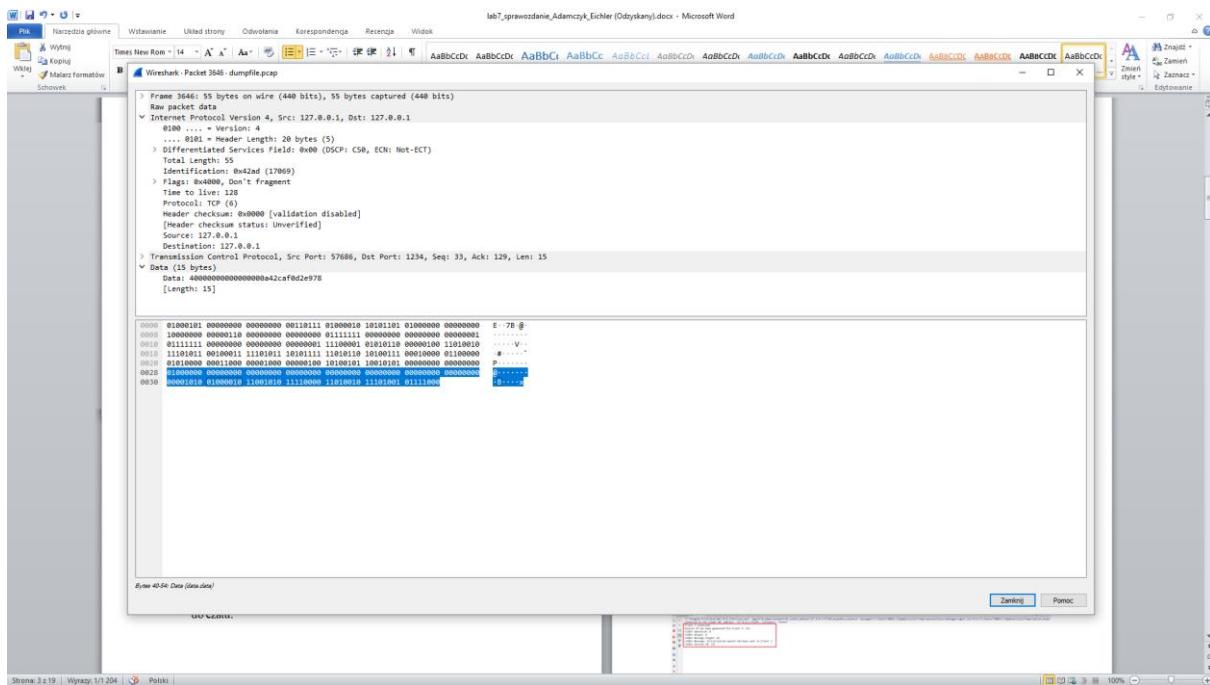
Rysunek 27: Serwer otrzymuje wiadomość hello od klienta 1 i wysyła ją do klienta 2. Pole odpowiedzi ustala na 1 (001). Pole operacji się nie zmienia i wynosi 0 (0000).



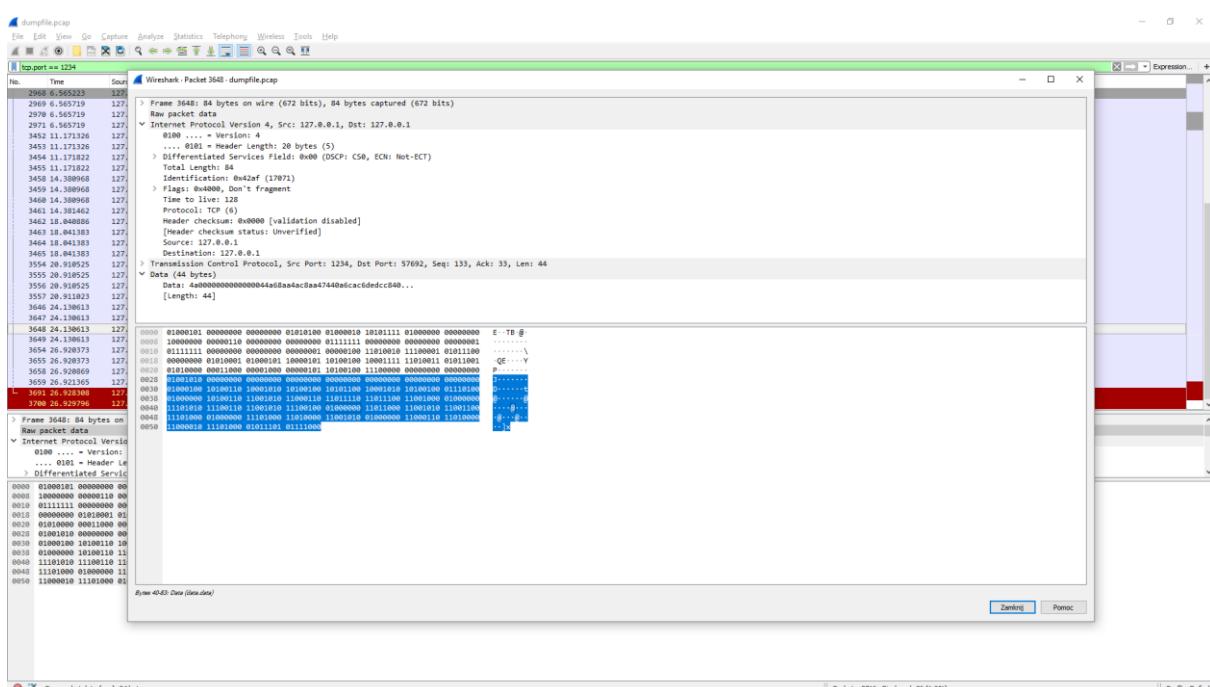
Rysunek 28: Klient numer 2 wysyła wiadomość: „world” do klienta numer 1. Pole operacji ustawia na 0 (0000), pole odpowiedzi również wynosi 0 (000).



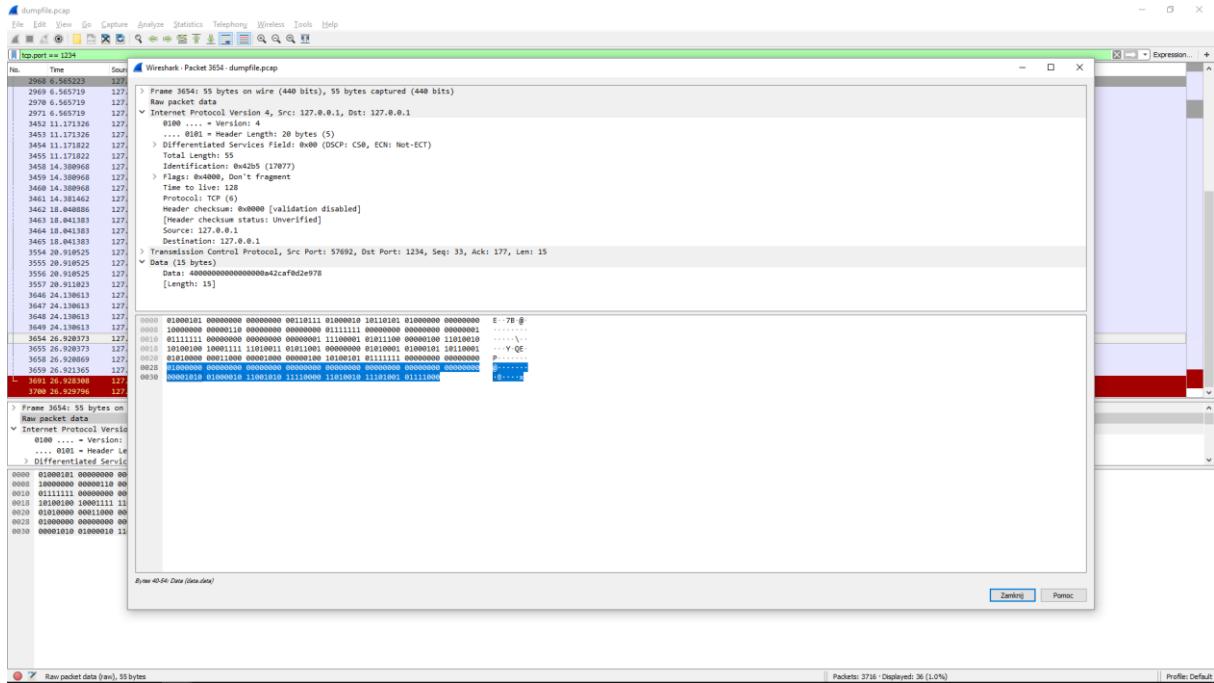
Rysunek 29: Serwer otrzymuje od klienta 2 wiadomość „world” i przesyła ją do klienta 1 ustawiając pole odpowiedzi na 1 (001). Pole operacji się nie zmienia i wynosi 0 (0000).



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 operacji nie zmienia się w stosunku do pakietu wysłanego do serwera od klienta 1 o rozłączeniu.



Rysunek 32: Drugi klient wysyła do serwera wiadomość „!exit” tym samym kończąc 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 przesyłana 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 wiążałaby 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.