POLITECHNIKA POZNAŃSKA
WYDZIAŁ ELEKTRYCZNY
Instytut Automatyki, Robotyki i Inżynierii Informatycznej

Jacek Eichler
Tomasz Adamczyk

Sprawozdanie z zajęć laboratoryjnych

Programowanie sieciowe - protokoły tekstowe

8 grudnia 2018

1. TREŚĆ ZADANIA

Temat: Komunikacja pomiędzy klientem a serwerem (1:1), w oparciu o autorski protokół tekstowy.

Protokół:

- połączeniowy,
- wszystkie dane przesyłane w postaci tekstowej (sekwencja znaków ASCII),
- każdy komunikat opatrzony znacznikiem czasu,
- nazwy pól o określonej długości: **2** znaki,
- struktura elementów nagłówka zdefiniowana jako klucz=wartość$ ○ (przykład) Operacja=dodaj$
- wymagane pola:
  - pole operacji – „OP”, ○ pole statusu – „ST”, ○ pole identyfikatora – „ID”.
- dodatkowe pola zdefiniowane przez programistę.

Funkcje oprogramowania:

- nawiązanie połączenia,
- uzgodnienie identyfikatora sesji, • wykonywanie operacji matematycznych na dwóch argumentach:
  - „poteguj” – potęgowanie, ○ „logarytmuj” – logarytmowanie, ○ 2 inne, zdefiniowane przez programistę.
- przeglądanie historii wykonywanych obliczeń:
  - po stronie klienta:
    - ✦ poprzez podanie identyfikatora sesji,
    - ✦ poprzez podanie identyfikatora obliczeń.
- po stronie serwera:
  - poprzez podanie identyfikatora sesji, ○ poprzez podanie identyfikatora obliczeń, ○ wyświetlenie wszystkich dotychczas wykonanych obliczeń.
- zakończenie połączenia.

Inne:

- gdy wartość wyniku wykracza poza zakres zmiennej, powinien zostać zwrócony kod
- błędu,
- każde obliczenia powinny posiadać unikalny identyfikator,
- identyfikator sesji powinien być przesyłany w trakcie komunikacji i powiązany • z obliczeniami,

- odwołanie się do nieistniejących obliczeń lub obliczeń wykonanych przez innego użytkownika, powinno skutkować przesłaniem odpowiedniego statusu.

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

Budowa pakietu

Operacja	Status	Liczba 1	Liczba 2	ID operacji	ID sesji	Znacznik czasu

Komendy:

- ***addition***
- ***subtract***
- ***power***
- ***logarithm***
- ***!exit***
- ***!disconnect***
- ***!history***

Kody operacji:

- AD – operacja dodawania
- SB - operacja odejmowania
- PW - operacja potęgowania
- LG - operacja logarytmowania
- IN – wysłanie pakietu inicjalizującego przez serwer z numerem ID
- EX – zakończenie pracy klienta oraz serwera
- DC – rozłączenie klienta
- HS – historia na podstawie ID sesji
- HO – historia na podstawie ID operacji

Kody statusu:

- WR – oczekiwanie na wynik operacji
- RS – przesłanie wyniku operacji
- NO – status pusty – przesyłany m.in. przy wysyłaniu pakietu inicjalizującego
- WH – oczekiwanie na przesłanie historii
- AD – typ operacji wysyłanej historii
- SB - typ operacji wysyłanej historii
- PW - typ operacji wysyłanej historii
- LG - typ operacji wysyłanej historii
- DT – brak historii dla podanego ID operacji

- DC – brak historii dla podanego ID sesji

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

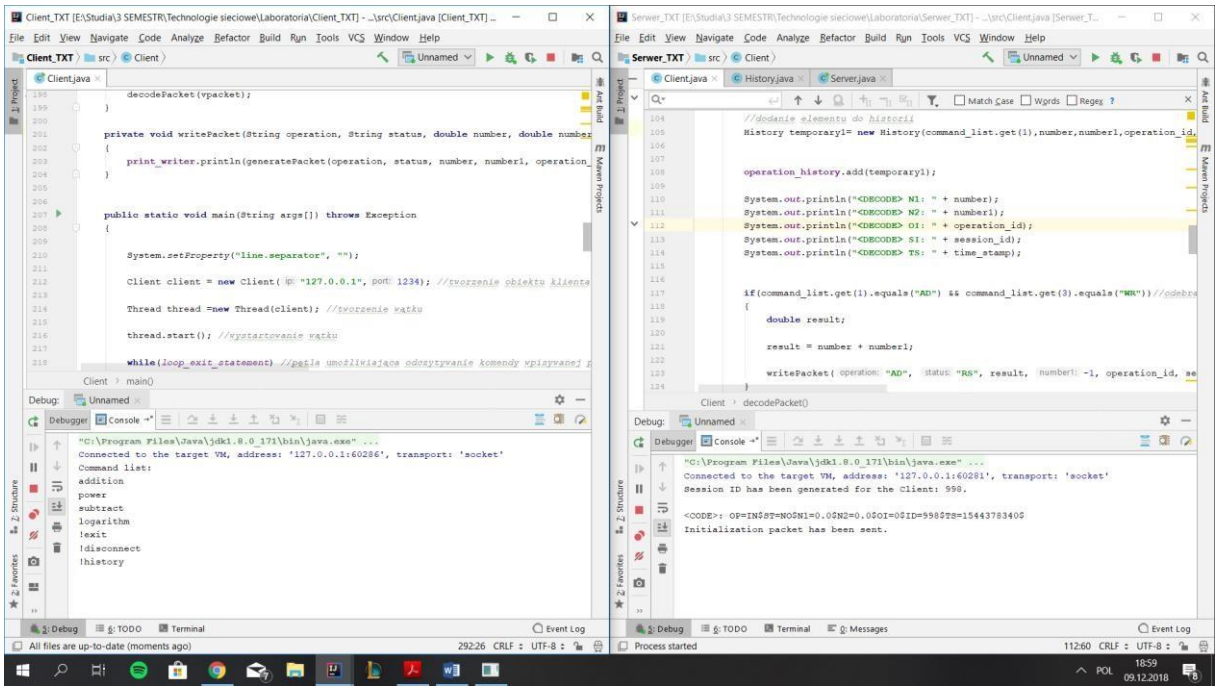
Aplikacja klienta:

- **https://github.com/WangHoHan/calculator-with-text-protocol/blob/master/Calculator%20With%20Text%20Protocol/Client/src/Client.java**

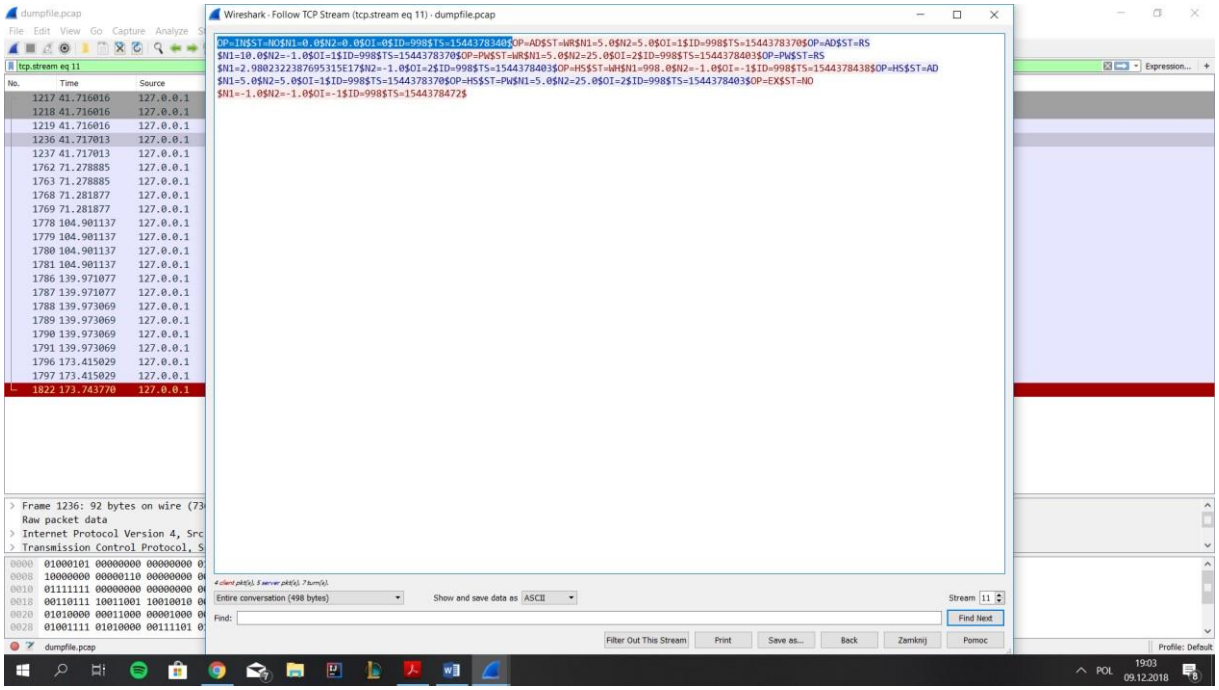
Aplikacja serwera (zawiera 3 pliki):

- **https://github.com/WangHoHan/calculator-with-text-protocol/blob/master/Calculator%20With%20Text%20Protocol/Server/src/Client.java**
- **https://github.com/WangHoHan/calculator-with-text-protocol/blob/master/Calculator%20With%20Text%20Protocol/Server/src/History.java**
- **https://github.com/WangHoHan/calculator-with-text-protocol/blob/master/Calculator%20With%20Text%20Protocol/Server/src/Server.java**

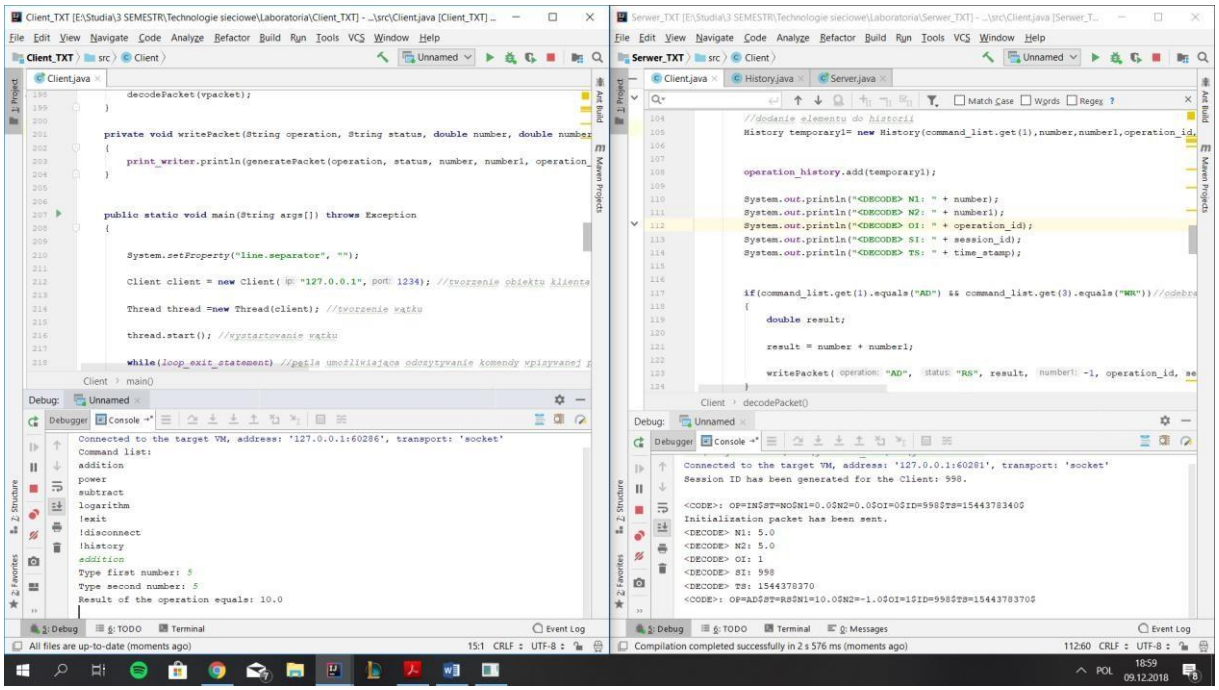
4. PRZEBIEG PRZYKŁADOWEJ SESJI KOMUNIKACYJNEJ – OPIS SŁOWNY ORAZ OBRAZ SESJI ZAREJESTROWANY PRZEZ PROGRAM *WIRESHARK*, WRAZ ZE STOSOWNYMI OBJAŚNIENIAMI.



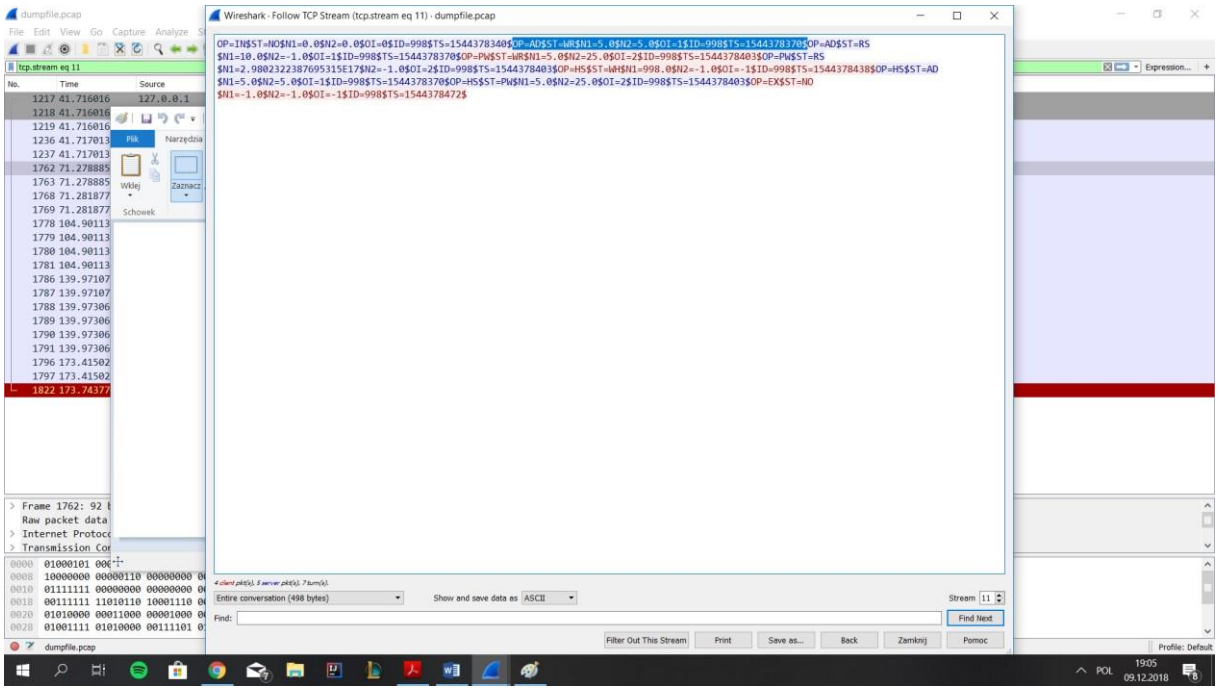
Rysunek 1: Wysłanie pakietu inicjalizującego przez serwer do podłączającego się klienta



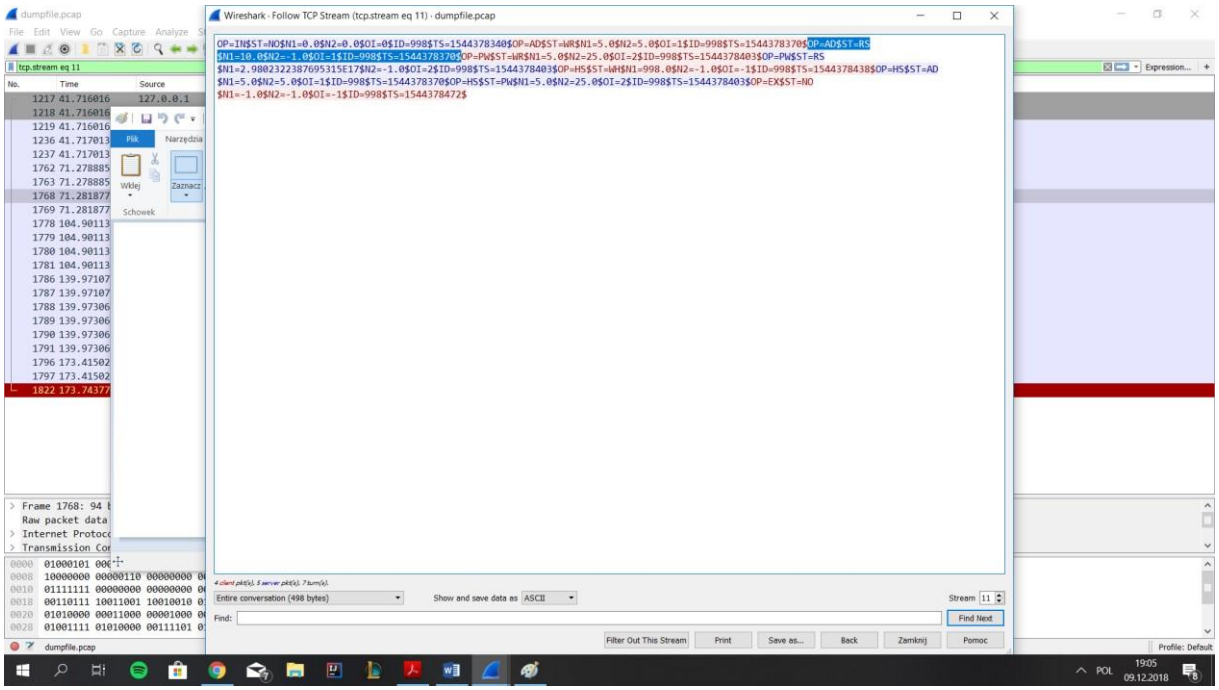
Rysunek 2: Klient w polu operacji zawiera komunikat IN, który sygnalizuje przesłanie numeru ID sesji dla nowo podłączonego klienta.



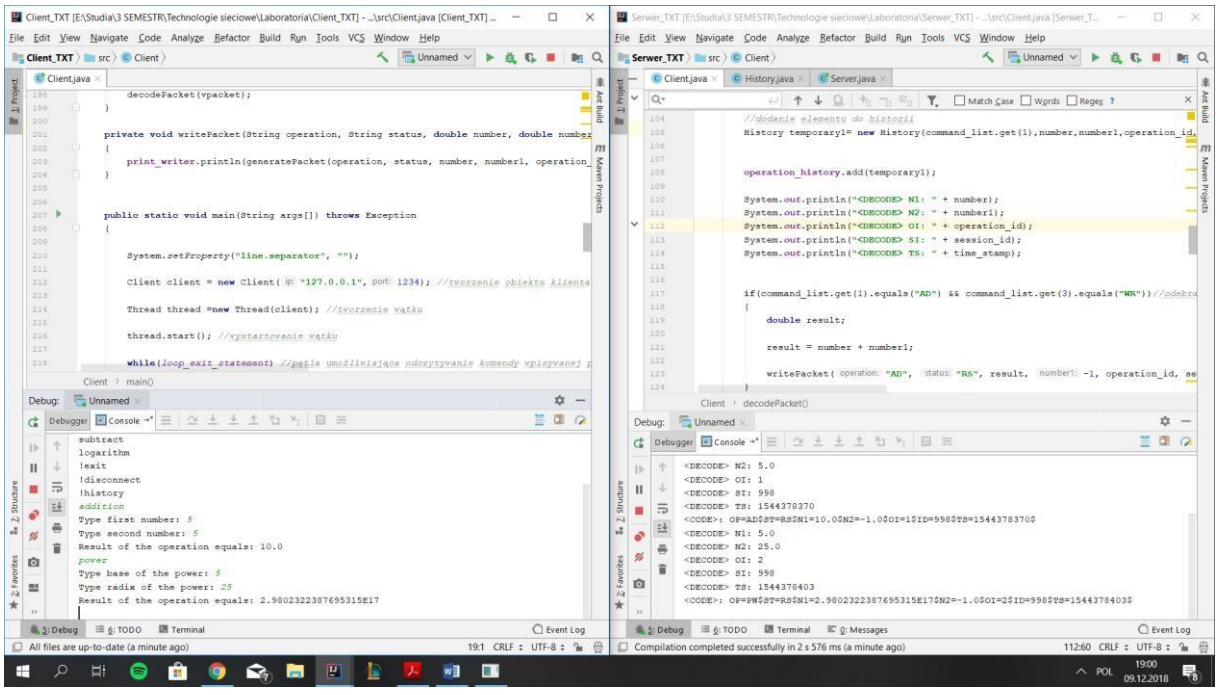
Rysunek 3: Użytkownik wykonuję operację dodawania liczb, serwer wysyła wynik.



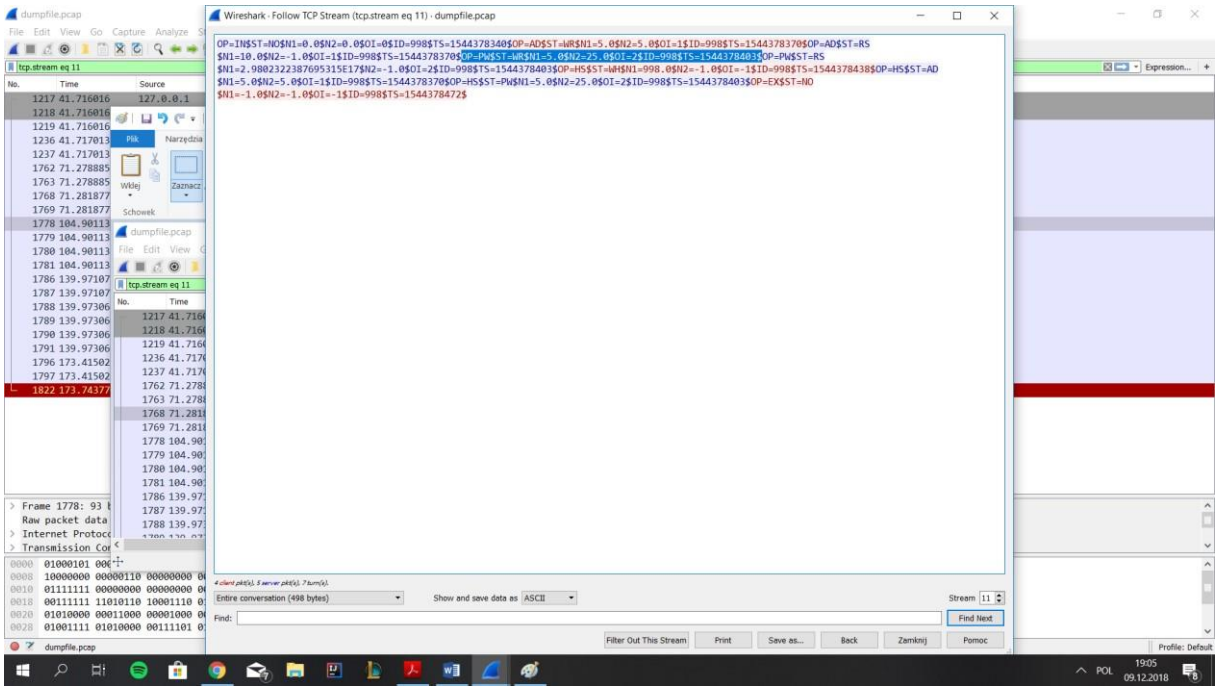
Rysunek 4: Klient w polu operacji umieszcza AD, co sygnalizuję operację dodawania. Status ustawiony jest na WR, co oznacza, iż klient po wysłaniu liczb będzie oczekiwał na zwrócenie wyniku przez aplikację serwera. W polach N1 oraz N2 zapisane są liczby wpisane przez użytkownika. Przesyłany jest również ID sesji, ID operacji oraz znacznik czasu.



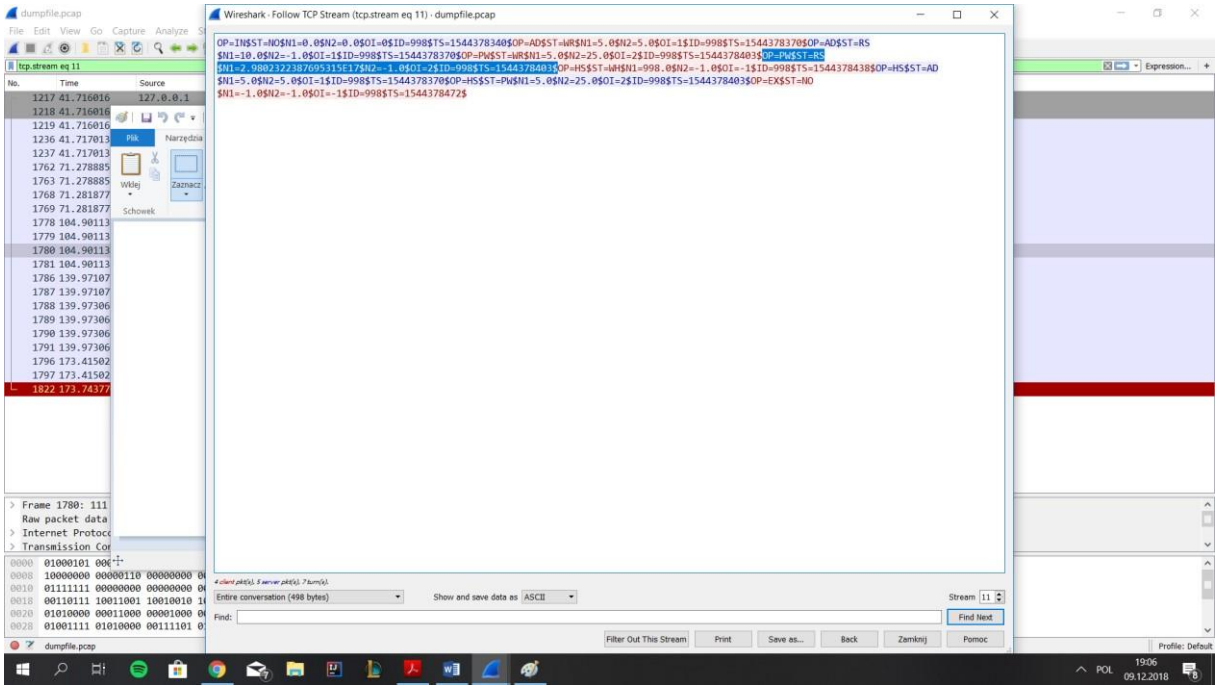
Rysunek 5: Serwer przesyła wynik dodawania, na co wskazuję pole operacji ustawione na AD oraz status RS (result send). W polu N1 znajduję się wynik operacji dodawania, natomiast w polu N2 przesyłana jest liczba -1, która wskazuje, iż pakiet przesyła jedynie jedną istotną liczbę zapisana w polu N1.



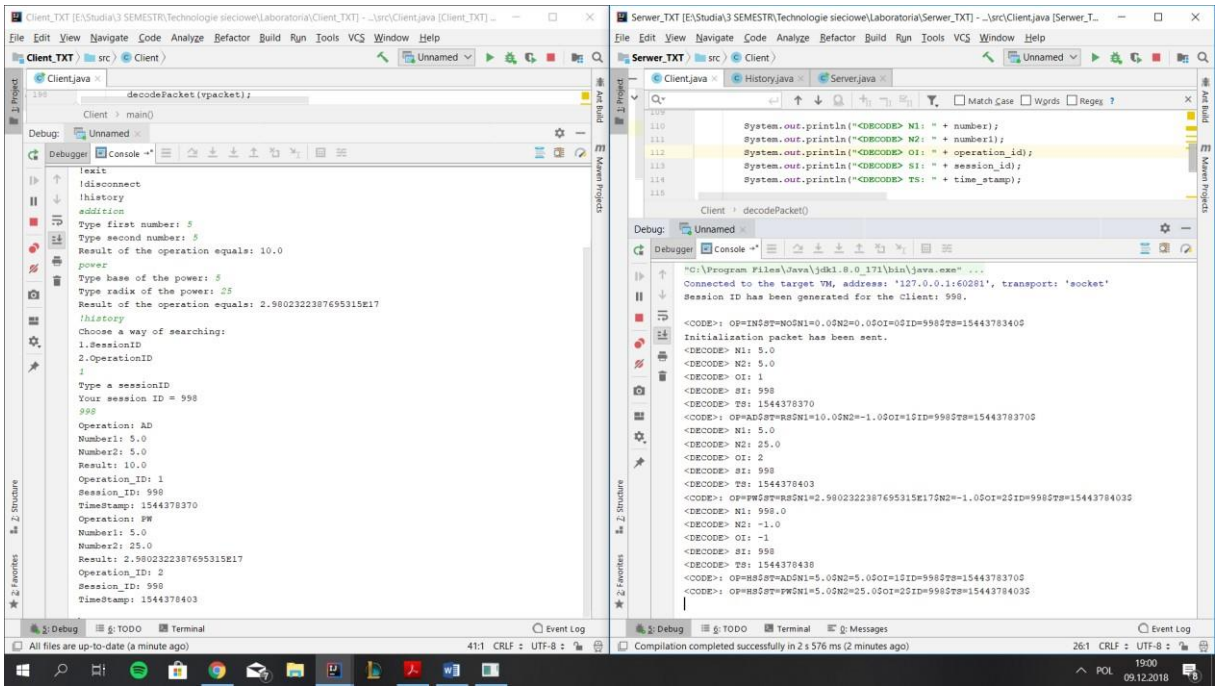
Rysunek 6: Użytkownik wybiera operację potęgowania, wpisuje dwie liczby, wysyła je do serwera i na otrzymuje wynik operacji.



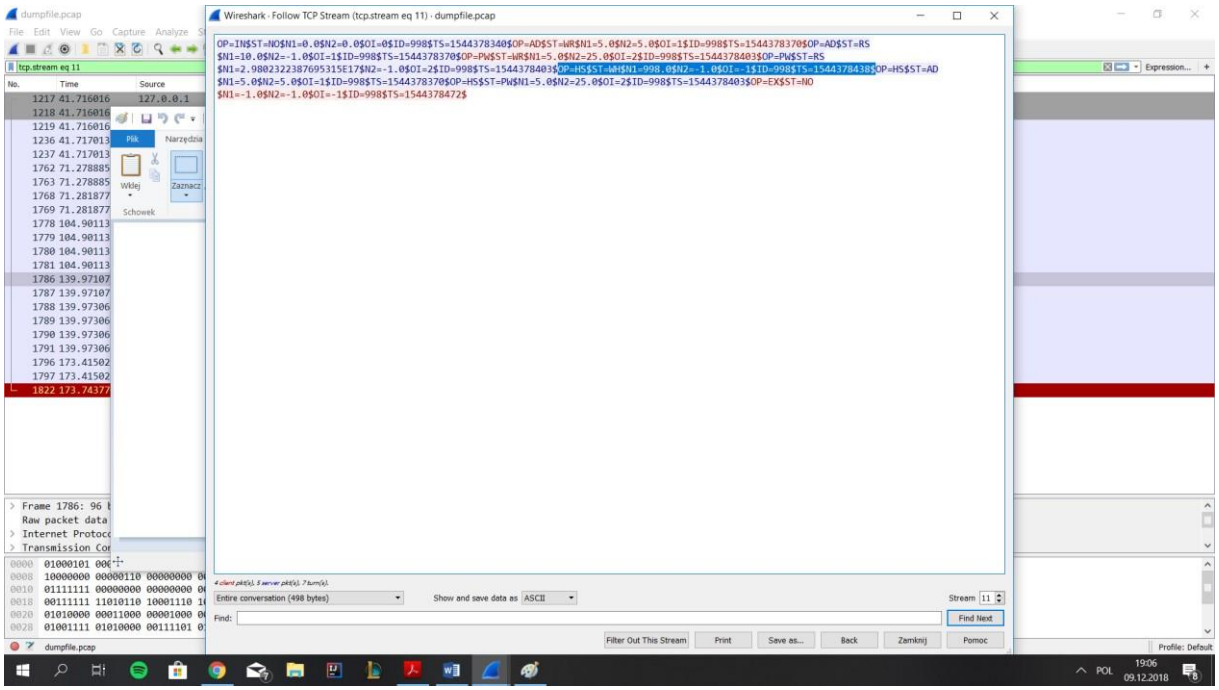
Rysunek 7: Klient w polu operacji umieszcza PW, co sygnalizuję operację potęgowania. Status ustawiony jest na WR, co oznacza, iż klient po wysłaniu liczb będzie oczekiwał na zwrócenie wyniku przez aplikację serwera. W polach N1 oraz N2 zapisane są liczby wpisane przez użytkownika. Przesyłany jest również ID sesji, ID operacji oraz znacznik czasu.



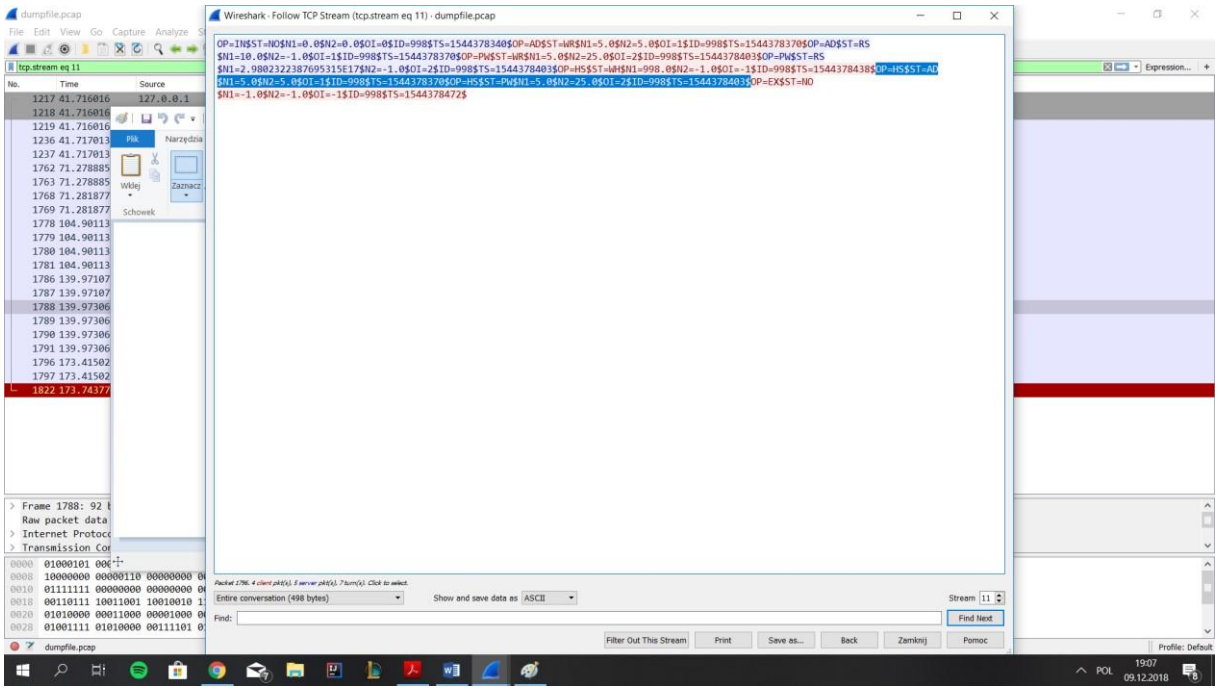
Rysunek 8: Serwer przesyła wynik potęgowania, na co wskazuję pole operacji ustawione na PW oraz status RS (result send). W polu N1 znajduję się wynik operacji dodawania, natomiast w polu N2 przesyłana jest liczba -1, która wskazuje, iż pakiet przesyła jedynie jedną istotną liczbę zapisana w polu N1.



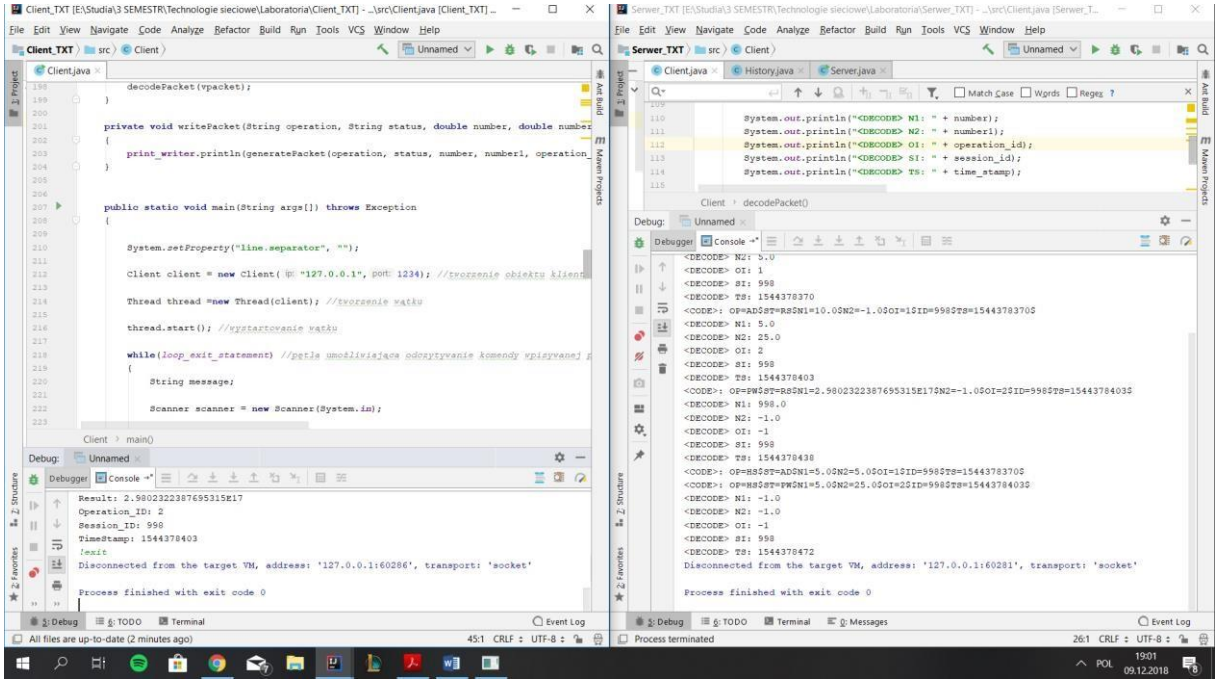
Rysunek 9: Użytkownik wpisuję komendę by wyświetlić historię operacji. Następnie wybiera, aby przesłana historia była identyfikowana po ID sesji. Po wpisaniu odpowiedniego ID, serwer przesyła historię dotychczas wykonanych operacji.



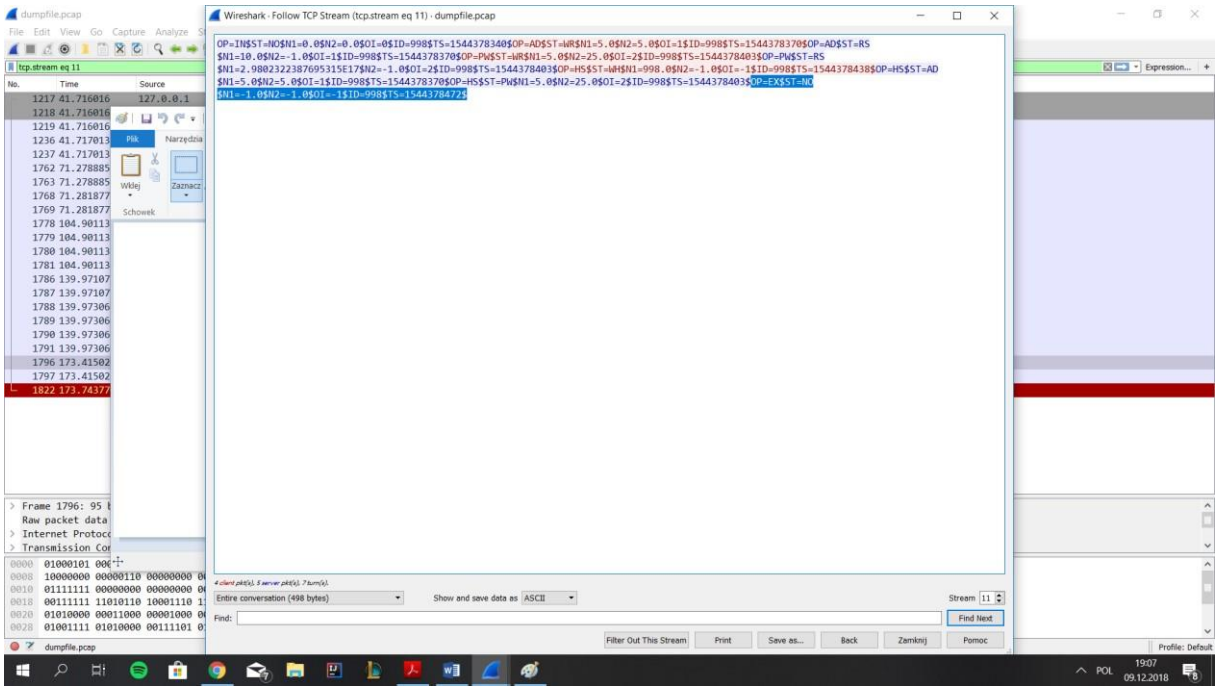
Rysunek 10: Klient w polu operacji umieszcza HS, co sygnalizuję rozkaz pobrania historii po numerze ID sesji. Status ustawiony jest na WH, co oznacza, iż klient oczekuje na przesłanie historii.. W polach N1 oraz N2 zapisane są liczby -1, które sygnalizują iż to pole nie jest istotne w tym pakiecie.. ID operacji również posiada wartość -1 by nie wpływać na rzeczywisty licznik.



Rysunek 11: Serwer przesyła 2 pakiety, gdyż dla podanego ID sesji wykonane zostały 2 operacje. W polu operacji zapisane jest HS, co wskazuję na przesyłanie historii, natomiast w polu statusu zawarta jest informacja o typie przesyłanej operacji. W polach N1 oraz N2 zawarte są liczby, wynik obliczany jest po stronie klienta.



Rysunek 12: Użytkownik zakończa pracę aplikacji klienta jak i serwera poprzez wpisanie komendy !exit.



Rysunek 13: W polu operacji zawarty jest komunikat EX, który przekazuje serwerowi rozkaz wyłączenia aplikacji. Pole statusu zawiera NO, co sygnalizuje brak istotności pola w przesyłanym pakiecie.

5. ODPOWIEDZI NA PYTANIA POSTAWIONE W SEKCJI *„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*. Na stronie czwartej znajduję się link do wersji źródłowej wraz z komentarzami.

2. **Przetestuj połączenie pomiędzy programami, rejestrując całość transmisji. Przeanalizuj przechwycone dane. Czy przesłane dane są w pełni tekstowe?** Transmisja została przechwycona w całości, umieszczona została w pliku o nazwie *dumpfile.pcap*.(ip: 127.0.0.1, port: 1234) Link do pliku znajduję się również na czwartej stronie sprawozdania. Przesyłane dane są w pełni tekstowe.

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

Wielkość pakietu zależy od wielkości przesyłanych liczb, gdyż jak wiemy na 1 bajcie maksymalnie możemy zapisać liczbę 255. Istnieje możliwość łatwej rozbudowy protokołu, gdyż wymagałoby to niedużych zmian w funkcji kodującej, dekodującej oraz wysyłającej pakiet. Należałoby dodać kolejne pole, które chcemy przesyłać i dopisać je w odpowiednich funkcjach.