W tym pliku zamieszczony jest opis działania opracowanego przeze mnie protokołu.
Do funkcjonowania protokołu potrzebne są 2 pliki.
SKJServer.jar
SKJClient.jar

Instrukcja Uruchamiania

Serwer uruchamiany jest podając jako argumenty po kolei porty które mają byd „obpukane" w celu nawiązania połączenia.
Przykładowa komenda włączająca Serwer:
java –jar SKJServer.jar 45662 45662 45773 48891

Klient uruchamiany jest podając jako argumenty: [nr.portu] [wiadomosc] [lista portów]
Podanie jako pierwszy argument ciągu znaków „localhost" ustawi adres lokalny urządzenia jako adres docelowy.
Przykładowa komenda uruchamiająca Klienta:
java –jar SKJClient.jar localhost Wyraz 45662 45662 45773 48891

Opis Działania Serwera

Serwer w momencie uruchomienia wyszukuje wszystkie unikatowe porty na liscie podanych portów i następnie otwiera na nich sockety UDP i zaczyna na nich równolegle nasłuchiwać .
Jeśli którykolwiek socket jest zajęty program kooczy działanie.
W momencie otrzymania datagramu socket rozpoczyna niezalezny wątek odpowiedzialny za dodanie nr. Portu do listy portów odwiedzonych przez klienta i sprawdzenie czy odwiedził on wszystkie potrzebne porty w dobrej kolejnosci. Wykonywanie tych dzialan w niezaleznym wątku umozliwia jak najszybszy powrót socketa do nasłuchiwania co pozwala na równloeglą obsługę wielu klientów.
Jeśli wątek uzna że klient zapukał po kolei do wszystkich wymaganych portów. Otwiera na losowym Niezajętym porcie socket TCP i na ostatnim odwiedzonym przez klienta porcie UDP wysyła komunikat zawierający nr. Portu TCP na którym należy nawiązad połączenie. W momencie otrzymania komunikatu TCP serwer odsyła odpowiedź zawierającą podwojony komunikat.
np.
Jeśli komunikat wejściowy to „Gruszka" to w odpowiedzi znajdzie się komunikat „Gruszka Gruszka"
Po otrzymaniu przez klienta odpowiedzi kanał TCP jest zamykany.

Opis Działania Klienta

Klient w momencie uruchomienia wysyła po kolei na podany adres datagramy na podane w liście argumentów porty UDP. Wysłane datagramy zawierają jako treść nr. Komunikatów numerowane od 0. Jest to mechanizm zapobiegawczy przeciw błędom związanym z tym że pakiety przyjdą w innej kolejności niż zostały wysłane.
Po wysłaniu ostatniego zapukania Klient oczekuje na ostatnim odwiedzonym kanale UDP odpowiedzi od serwera zawierającą nr. Portu na którym ma mieć miejsce właściwa komunikacja TCP. jeśli nie otrzyma jej przez 2 sekundy kończy działanie programu.
W momenci otrzymania odpowiedzi Klient otwiera na otrzymanym porcie socket TCP którym wysyła na serwer wiadomośd będącą drogim elementem z listy argumentów. Następnie oczekuje na odpowiedzi od serwera na sockecie TCP po otrzymaniu ich kończy działanie programu.