

## Opis projektu 3 – UDP over TCP Tunnel

Autor: Robert Grochowski

Indeks: s17124

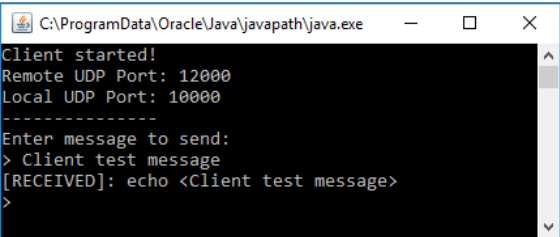
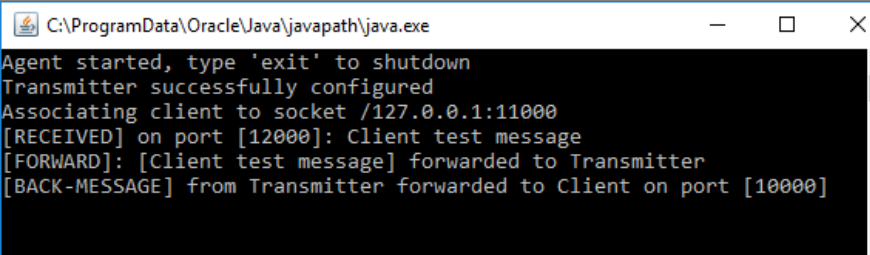
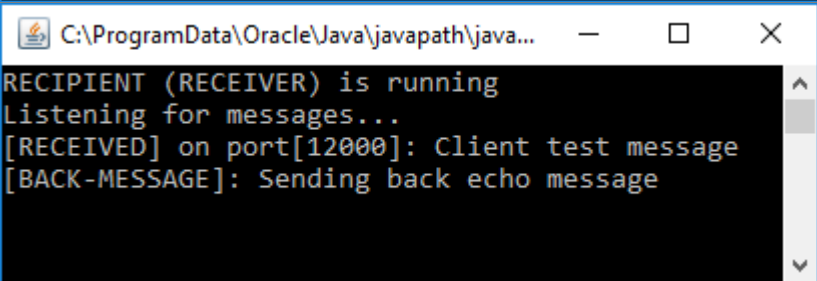
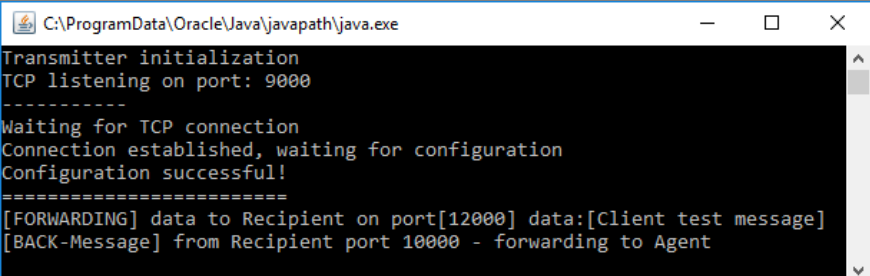
### Podstawowe uruchomienie

Za kompilację odpowiedzialny jest plik **compile.bat**.

Do uruchomienia projektu przygotowałem dwa pliki startowe:

1. **Run\_oneclient.bat** – przypadek, w którym uruchamiamy cały tunel z jednym klientem po stronie agenta
2. **Run\_fewblicents.bat** – przypadek, w którym uruchamiany jest tunel z trzema klientami po stronie agenta

Po uruchomieniu jednego z wariantów i po wpisaniu czegoś do Klienta powinniśmy dostać wiadomość zwrótną. Przykład działania:

Klient	 <pre>C:\ProgramData\Oracle\Java\javapath\java.exe Client started! Remote UDP Port: 12000 Local UDP Port: 10000 ----- Enter message to send: &gt; Client test message [RECEIVED]: echo &lt;Client test message&gt; &gt;</pre>
Agent	 <pre>C:\ProgramData\Oracle\Java\javapath\java.exe Agent started, type 'exit' to shutdown Transmitter successfully configured Associating client to socket /127.0.0.1:11000 [RECEIVED] on port [12000]: Client test message [FORWARD]: [Client test message] forwarded to Transmitter [BACK-MESSAGE] from Transmitter forwarded to Client on port [10000]</pre>
Odbiorca	 <pre>C:\ProgramData\Oracle\Java\javapath\java... RECIPIENT (RECEIVER) is running Listening for messages... [RECEIVED] on port[12000]: Client test message [BACK-MESSAGE]: Sending back echo message</pre>
Przeказник	 <pre>C:\ProgramData\Oracle\Java\javapath\java.exe Transmitter initialization TCP listening on port: 9000 ----- Waiting for TCP connection Connection established, waiting for configuration Configuration successful! ===== [FORWARDING] data to Recipient on port[12000] data:[Client test message] [BACK-Message] from Recipient port 10000 - forwarding to Agent</pre>

## Opis projektu

Aplikacja składa się z 4 głównych klas:

- **skj.pro3.agent.Agent** – klasa odpowiedzialna za Agent; Posiada klasę-wątek do komunikacji TCP z Przekaznikiem (*TransmitterCommunicator*) oraz otwiera n portów UDP na których nasłuchuje od klientów (*ClientCommunicator*)
- **skj.pro3.transmitter.Transmitter** – klasa odpowiedzialna za Przekaznik. Posiada ona dwie wewnętrzne klasy-wątki odpowiedzialne za komunikację z Agentem oraz z odbiorcą nazywające się odpowiednio *AgentCommunicator* oraz *RecipientCommunicator*
- **skj.pro3.user.Client** – Klasa klienta; nasłuchuje na wiadomości z konsoli oraz wypisuje komunikaty odebrane od odbiorcy (precyzyjniej – od agenta). Za asynchroniczne pobieranie wiadomości odpowiedzialna jest metoda z zaimplementowanym wątkiem (*receiveMesages()*)
- **skj.pro3.user.Recipient** – Klasa odbiorcy; Posiada n-klas-wątków odpowiedzialnych za nasłuchiwanie (*MessageReceiver*). Ponadto, otwiera n-portów wysyłających. Każdy port nasłuchujący ma przypisany sobie port wysyłający. Po odebraniu wiadomości od Klienta (precyzyjniej - od - przekaznika) wysyła wiadomość powrotną „echo” tzn. tą samą wiadomość co został z przedrostkiem „echo”.

## Opis parametrów każdego procesu

Składnia została opisana w następujący sposób: nawiasy kwadratowe należy zastąpić odpowiednią wartością (z pominięciem symboli nawiasów)

- **Proces Agent:**  
[adres IP przekaznika] [port TCP przekaznika] [adres IP odbiorcy] [port nasłuchujący odbiorcy] [kolejny port nasłuchujący odbiorcy] [...]  
np. 127.0.0.1 9000 127.0.0.1 12000 12001 12002
- **Proces Przekaznika:**  
[port TCP przekaznika]  
np. 9000
- **Proces Klienta:**  
[Adres IP Agent] [port nasłuchu Agent/Odbiorcy] [port wysyłki Odbiorcy (który jest zarówno portem nasłuchującym tego klienta)]  
np. 127.0.0.1 12000 10000
- **Proces odbiorcy:**  
-l [porty nasłuchujący 1] [port nasłuchujący 2] [...] -p [port wysyłający 1] [...]  
np. -l 12000 12001 12002 -s 10000 10001 10002  
Uwaga: Portów nasłuchujących musi być tyle samo, ile portów wysyłających, ponieważ port pierwszy nasłuchujący odpowiada portowi pierwszemu wysyłającemu itd.

## Zdalne wyłączenie transmitera

Zgodnie z wymogami zawartymi w skrypcie, zaimplementowałem zdalne zamykanie pracy transmitera oraz agenta poprzez wpisanie przez konsolę komendy 'exit'

Przykładowe wyjście konsoli Agent'a po wpisaniu komendy 'exit':

```
Agent started, type 'exit' to shutdown
Transmitter successfully configured
Associating client to socket /127.0.0.1:11000
[RECEIVED] on port [12000]: test client A
[FORWARD]: [test client A] forwarded to Transmitter
[BACK-MESSAGE] from Transmitter forwarded to Client on port [10000]
Associating client to socket /127.0.0.1:11001
[RECEIVED] on port [12001]: test client B
[FORWARD]: [test client B] forwarded to Transmitter
[BACK-MESSAGE] from Transmitter forwarded to Client on port [10001]
exit
Shutting down transmitter...
Shutting down transmitter...OK
Shutting down Agent...
Shutting down Agent...OK

Process finished with exit code 0
```

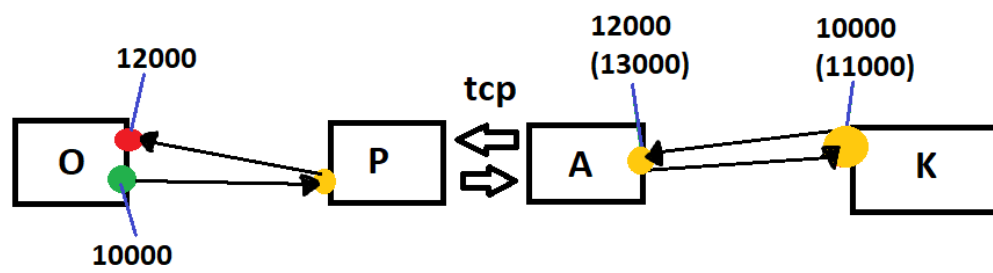
Transmitter:

```
Transmitter initialization
TCP listening on port: 9000
-----
Waiting for TCP connection
Connection established, waiting for configuration
Configuration successful!
=====
[FORWARDING] data to Recipient on port[12000] data:[test client A]
[BACK-Message] from Recipient port 10000 - forwarding to Agent
[FORWARDING] data to Recipient on port[12001] data:[test client B]
[BACK-Message] from Recipient port 10001 - forwarding to Agent
Shutting down agentCommunicator...
Shutting down agentCommunicator...OK
Shutting down recipientCommunicator...
Shutting down recipientCommunicator...OK

Process finished with exit code 0
```

## Napotkane problemy i ich rozwiązanie

Aplikację tworzyłem wedle wymogów zawartych w skrypcie. Jednak wymagana była drobna modyfikacja projektu, aby było możliwe testowanie w jednej sieci. Problem wyglądał następująco: Aplikacja wymaga otwarcia dwóch (lub więcej w przypadku wielu klientów) takich samych portów Odbiorcy i Agent'a. Przez to, uruchomienie tych dwóch procesów rzucił wyjątek. Rozwiązaniem tego problemu była translacja portów Agent'a oraz Klienta na porty o 1000 większe. Jednak ten zabieg jest ukryty we na wyjściu konsol lub w podawanych parametrach. Należy jedynie pamiętać, żeby różnica między portami nasłuchującymi a wysyłającymi była większa lub równa 1000. Problem ten opisuje na poniższej grafice:



- gniazdo nasłuchujące
- gniazdo wysyłające
- gniazdo nasłuchująco-wysyłające

numer w nawiasie to  
rzeczywisty numer portu