Michał Budnik – Sprawozdanie 4

# 1. Zadanie 1

## 1.1 Opis zagadnienia

Przy próbie transmitowania wiadomości z użyciem programów **Z2Sender** i **Z2Reciever** programy te komunikują się bez jakichkolwiek zakłóceń, co nie jest wiarygodnym odwzorowaniem sposobu, w jaki przesyłane są pakiety np. przez Internet. Dlatego też do dokładniejszej symulacji używany jest program **Z2Forwarder**, który ze zmiennym opóźnieniem przekazuje dalej otrzymane pakiety, mogąc je również kompletnie zgubić. Nie jest to jeszcze idealne odwzorowanie połączenia internetowego, gdyż niektóre pakiety są też zniekształcane (co było symulowane na poprzedniej liście i weryfikowane z użyciem kodu CRC). Niemniej jednak zwraca to już uwagę na pewien fundamentalny problem – co jeśli treść komunikatu dotrze do odbiorcy w zmienionym szyku? Co, jeśli niektóre pakiety dotrą więcej niż raz, a niektóre wcale?

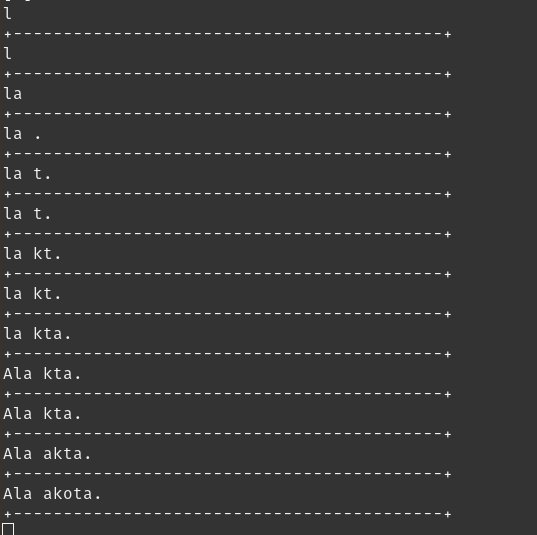
## 1.2 Założenia

W celu rozwiązania tego problemu zostały użyte następujące założenia:

- pakiety wysłane później mogą być dostarczone wcześniej,  
- niektóre pakiety mogą być w pełni utracone – należy wtedy przesłać je ponownie,  
- pakiety są 5-cio bajtowe, z czego pierwsze cztery bajty są zarezerwowane na numer sekwencyjny, a pozostały bajt to faktyczna wiadomość (char).

## 1.3 Realizacja

Zagadnienie tak naprawdę dzieli się na dwa mniejsze podproblemy. Pierwszy to ponowne złożenie odebranych pakietów w logiczną całość. Rozwiązanie tego problemu jest dosyć trywialne dzięki temu, że każdy pakiet posiada swój numer sekwencyjny. Jeżeli wiadomość jest wysyłana jako ciąg pakietów odczytanych po kolei ze strumienia, to złożenie wiadomości według numerów sekwencyjnych gwarantuje jej poprawność. Ponieważ jednak pakiety nie muszą być odebrane w tej samej kolejności, w której zostały wysłane, zaimplementowany został prosty bufor. Jest on inicjowany jako tablica rozmiaru 4096 bajtów zawierająca puste znaki. Po odebraniu pakietu jest on po prostu wpisywany w miejsce tablicy według jego numeru sekwencyjnego (pakiet z numerem 16 zostanie wpisany do tablicy pod indeksem 15). Rozwiązanie to jest proste i efektywne – istnieje możliwość podglądu odebranych już pakietów bez posiadania pełnej lub poprawnie sekwencyjnie wiadomości, w przypadku odebrania pakietu kilkukrotnie nie trzeba nic zmieniać w samym buforze, gdy przychodzi pakiet z numerem większym niż wielkość buforu, bufor jest powiększany.



Output 1 - Przykładowe działanie programu

Drugim podproblemem jest fakt, że niektóre pakiety (lub potwierdzenia otrzymania pakietu) mogą się kompletnie zgubić. Rozwiązanie zostało zaimplementowane po strony nadawcy. Każdy pakiet, który zostaje wysłany, jest również wpisywany do globalnej tablicy pakietów (inicjowanej jako pusta tablica). W ten sposób gwarantowany jest prosty dostęp do wszystkich pakietów, które są wysyłane. Gdy otrzymywany jest komunikat o odebraniu pakietu przez odbiorcę, dany pakiet zostaje usuwany z tablicy pakietów wysłanych – mamy w tym momencie gwarancję, że odbiorca jest już w posiadaniu tego pakietu, więc nie będzie potrzebne jego ponowne przesyłanie. Dodatkowo uruchamiany jest nowy wątek **ResenderThread,** który po każdych ‘n’ sekundach nieaktywności ponownie rozsyła wszystkie pakiety z tablicy pakietów wysłanych. Zakładane jest, że jeżeli informacja zwrotna o otrzymaniu pakietu nie dotarła z powrotem do wysyłającego w ciągu ‘n’ sekund, to prawdopodobnie pakiet ten (lub potwierdzenie odbioru) został zgubiony. W takim przypadku pakiet wciąż znajduje się w tablicy pakietów wysłanych, więc **ResenderThread** ponownie prześle pakiet. Rozwiązanie to również jest proste i efektywne – koniec końców gwarantuje odebranie wszystkich pakietów przez adresata, w zależności od ustawionego parametru ‘n’ nie tworzy zbyt dużego spamu w sieci, jeżeli jakiś pakiet zostanie dwa razy odebrany, dzięki poprzednio zaimplementowanemu rozwiązaniu nie następuje żaden konflikt.

class ResenderThread extends Thread{

public void run(){

try{

while(true){

for (Z2Packet p: Z2Sender.*toSend*){

DatagramPacket packet = new DatagramPacket(p.data, p.data.length, localHost, destinationPort);

socket.send(packet);

}

*sleep*(5000);

}

} catch(Exception e){

System.*out*.println("Z2Sender.SenderThread.run: "+e);

}

}

}

Kod 1 - ResenderThread

## 1.4 Wnioski

Wysyłanie komunikatów przez Internet (lub dowolną inną sieć) wiąże się z ryzykiem zniekształcenia przekazu lub zagubienia jego części. Dlatego też niezbędne są sposoby na rozwiązanie tych problemów stosunkowo niskim kosztem. Zaprezentowany w tej liście sposób radzenia sobie ze zgubionymi pakietami prezentuje dwie ważne cechy prawidłowego protokołu transmisyjnego – po pierwsze, część przekazu musi zawierać unikatowy identyfikator, który umożliwi odbiorcy właściwe złożenie wszystkich otrzymanych pakietów, a po drugie odbiorca musi wiedzieć która część pakietu jest przekazem, a która identyfikatorem.

Tę pierwszą cechę można osiągnąć na kilka sposobów, w zależności od potrzeb. W zaprezentowanych rozwiązaniach używane są czterobajtowe liczby całkowite dodatnie, co daje dokładnie 2,147,483,647 identyfikatorów. Jest to ładnym i prostym do odczytania kodem identyfikacyjnym pakietu, jednakże odbiorca musi wiedzieć o tym standardzie.

Dodatkowym problemem, który został zauważony w otrzymanych programach to brak jakiejkolwiek równowagi ilości danych potrzebnych na identyfikator a ilości faktycznie ważnych danych w pakiecie. Każdy pakiet posiada 5 bajtów danych, z czego 4 to metadane identyfikacyjne. Oznacza to, że transmisja pliku rozmiaru 1 GB wymagałaby przesłania 5 GB danych (nie licząc dosyłania pakietów, które nie dotarły). Można łatwo zauważyć, że nie jest to idealne rozwiązanie (szczególnie że taka transmisja wymagałaby ponad 200 razy więcej identyfikatorów, niż jesteśmy zmieścić w 4 bajtach, nawet jeśli będą to liczby bez znaku).

W obliczu takiego przypadku ciężko nie zauważyć, że przykładowo pakiet 58B danych plus 6B metadanych wypadłby znacznie lepiej, wymagając około 17 milionów (zamiast ponad miliarda) pakietów i tyleż samo identyfikatorów (kilkukrotnie więcej niż zmieściłoby się w 4 bajtach, ale 16 milionów razy mniej niż pomieściłoby 6 bajtów). W takim przypadku transfer pliku o rozmiarze 1 GB wymagałby przesłania 1.1 GB danych.