Technologie sieciowe

Sprawozdanie nr 4

TCP/IP

Mateusz Laskowski

236618

1. Cel

Celem zadania jest modyfikacja podanych klas, tak aby symulowały one przesyłanie danych przy pomocy protokołów sieciowych (TCP).

2. Realizacja zadania

2.1. Definicje

TCP (ang. Transmission Control Protocol - Protokół sterowania transmisją) jest to protokół służący do przesyłania danych między hostami. Protokół ten gwarantuje dostarczenie wszystkich pakietów w całości, z zachowaniem kolejności i bez duplikatów. **TCP** jest protokołem działającym w trybie klient–serwer. Serwer oczekuje na nawiązanie połączenia na określonym porcie. Klient inicjuje połączenie do serwera.
TCP/IP jest to Protokół TCP (rozszerzenie) korzystający z usług protokołu IP do wysyłania i odbierania danych oraz ich fragmentacji.

Datagram - podstawowa jednostka przekazu powiązana z siecią komutacyjną pakietów. Datagramy zwykle są zbudowane z sekcji nagłówka i ładunku. Datagramy dostarczają możliwość bezpołączeniowej komunikacji w sieci komutacyjnej pakietów.

2.2. Programy

Z2Forwarder – program symulujący medium transmisyjne oraz sytuacje, w których pakiety nie docierają do hostów.

Z2Packet – klasa imitująca pakiet w datagramie.

Z2Sender – program symulujący host wysyłający dane.

Z2Receiver – program symulujący host odbierający dane.

2.3. Rozwiązanie problemu

```
try
{
    while(true)
    {
        byte[] data = new byte[datagramSize];
        DatagramPacket packet = new DatagramPacket(data, datagramSize);
        socket.receive(packet);
        Z2Packet p = new Z2Packet(packet.getData());
        int idx = p.getIntAt(0);
        if (received[idx][0] == idx) {
            packet.setPort(destinationPort);
            socket.send(packet);
            //System.out.println("Receiver - Duplikat | " + p.getIntAt(0) +
" - " + (char) p.data[4] + " || lastPrinted = " + lastPrinted);
        }
        else {
            received[idx][0] = (byte)((idx)&0xFF);
            for (int i = 1; i <= 4; i++) {
                received[idx][i] = p.data[i];
            }
            if (p.getIntAt(0) == 0) {
                lastPrinted = 0;
                //System.out.println("Receiver - Otrzymano poczatkowy
pakiet | " + received[idx][0] + " = " +(char) received[idx][4]);
                int i = 0;
                System.out.print("R: ");
                while (received[i][0] != -1) {
                    System.out.print((char) received[i][4] + " ");
                    lastPrinted = i;
                    i++;
                }
                System.out.println("");
                //System.out.println("Receiver - lastPrinted zwiekszone do
" + lastPrinted);
            }
        …
        …
    }
    // WYSLANIE POTWIERDZENIA
    packet.setPort(destinationPort);
    socket.send(packet);
}
catch(Exception e)
{
    System.out.println("Z2Receiver.ReceiverThread.run: "+e);
}
```

Kod 1.: Fragment metody klasy Z2Receiver służąca do odbierania wiadomości. Ten fragment chroni przed duplikacją wiadomości, jest jeszcze wiele warunków w metodzie, które chronią odebraną wiadomość, aby była poprawnie odczytana. Na końcu można zauważyć, że po udanym odbiorze host wysyła wiadomość potwierdzającą o odebraniu pakietu. W razie niepowodzenia jest przesyłana wiadomość do klasy Z2Sender, gdzie tam są opisane jak ma postępować program na wszelkie błędy.

```
class SenderThread extends Thread
{
    public void run()
    {
        int i, x;
        try
        {
            for(i=0; (x=System.in.read()) >= 0 ; i++)
            {
                Z2Packet p=new Z2Packet(4+1);
                p.setIntAt(i,0);
                p.data[4]= (byte) x;
                DatagramPacket packet =
                        new DatagramPacket(p.data, p.data.length,
                                localHost, destinationPort);
                sended[i][0] = (byte)((i)&0xFF);
                sended[i][4] = (byte) x;
                socket.send(packet);
                sendedPackages++;
                sleep(sleepTime);
            }
        }
        catch(Exception e)
        {
            System.out.println("Z2Sender.SenderThread.run: "+e);
        }
    }

}
```

Kod 2.: Metoda w klasie Z2Sender służąca do wysyłania pakietów. Metoda wczytuje znaki ze standardowego wejścia, tworząc z nich pakiety, dodaje do listy pakietów i wysyła jeden pakiet z listy. Po wysłaniu następuje przerwa, a po krótkiej przerwie powtarza wysyłanie kolejnego pakietu.

```
try
{
    int idx;
    while(true)
    {
        if (receivedConfirmations < sendedPackages) {
            byte[] data=new byte[datagramSize];
            DatagramPacket packet=
                    new DatagramPacket(data, datagramSize);
            try {
                socket.receive(packet);
            }
            catch (SocketTimeoutException e) {
                System.out.println("Socket czekał ponad 10 sekund");
                //lastConfirmationTime = System.nanoTime();
                for (int i = 0; i < sendedPackages; i++) {
                    if (received[i][0] == -1) {
                        Z2Packet p=new Z2Packet(4+1);
                        for (int j = 1; j <= 4; j++) {
                            p.data[j]= sended[i][j];
                        }
                        p.setIntAt(i,0);
                        packet = new DatagramPacket(p.data, p.data.length,
localHost, destinationPort);
                        socket.send(packet);
                    }
                }
            }
            Z2Packet p=new Z2Packet(packet.getData());
            //System.out.println("S[potwierdzenie]:"+p.getIntAt(0)+ ":
"+(char) p.data[4]);

            //firstConfirmation = true;
            //lastConfirmationTime = System.nanoTime();

            idx = p.getIntAt(0);
            if (received[idx][0] == -1) {
                received[idx][0] = (byte)((idx)&0xFF);
                for (int i = 1; i <= 4; i++) {
                    received[idx][i] = p.data[i];
                }
                receivedConfirmations++;
            }
            //Wysyłanie duplikatow pakietów, bo nie dostaliśmy ich
wcześniejszych potwierdzeń
            for (int i = 0; i <= idx; i++) {
                if (received[i][0] == -1) {
                    p=new Z2Packet(4+1);
                    for (int j = 1; j <= 4; j++) {
                        p.data[j]= sended[i][j];
                    }
                    p.setIntAt(i,0);
                    packet = new DatagramPacket(p.data, p.data.length,
localHost, destinationPort);
                    socket.send(packet);
                }
            }
        }
        else {
            System.out.println("Otrzymano wszystkie potwierdzenia");
            break;
```

Kod3.: Metoda w klasie Z2Sender służąca do odbierania potwierdzeń dotarcia pakietów i wszelkich uwag od odbierającego wiadomości. Odbiera potwierdzenia dotyczące dotarcia pakietów do hostu docelowego. W razie jakiegoś błędu metoda odbierze powiadomienie o errorze z klasy Z2Receiver i o jego typie. Dzięki informacji w razie błędu wyśle raz jeszcze dany pakiet, który utrudnił odczyt, zduplikował lub nie powiódł.

3. Wnioski

Pomimo kilkukrotnego przesłania tych samych danych mamy pewność, że dane będą drukowane w tej samej kolejności i nie będą się dublować. Protokół TCP/IP pomaga w transmisji danych, ponieważ sprawdza on czy dotarły wszystkie pakiety, a medium transmisyjne nie musi być stałe, a to znaczy, że mogą występować w nim zakłócenia powodujące gubienie pakietów podczas przesyłania danych.