WYDZIAŁ PODSTAWOWYCH PROBLEMÓW TECHNIKI
POLITECHNIKI WROCŁAWSKIEJ

SPRAWOZDANIE 2 Z LISTY 4

JACEK MUCHA

Technologie sieciowe
prowadzący
dr inż. Łukasz Krzywiecki

WROCŁAW 2010

Spis treści

Cel ćwiczenia

Symulacja przesyłania pakietów. Opracowanie algorytmu, który pozwala retransmitować utracone pakiety i układanie ich we właściwej kolejności.

1 Analiza problemu

Przesyłanie wiadomości przez Internet polega na podzieleniu jej na małe porcje, pakiety, które są wysyłane niezależnie. Po odebraniu każdego pakietu, odbiorca wysyła nadawcy potwierdzenie odbioru. W praktyce pakiety są podczas transmisji tracone, dublowane, dochodzą w niewłaściwej kolejności. Zadanie polega na rozwiązaniu tychże problemów.

2 Projekt systemu

Pomysł polega na stworzeniu w *Z2Sender* listy *memory*, której zadaniem będzie przechowywanie informacji o wszystkich tych wysłanych pakietach, w stosunku do których nie napłynęło jeszcze potwierdzenie odbioru. Elementy tej listy to obiekty o trzech polach:

- numer sekwencyjny pakietu
- czas wysłania pakietu
- kopia wysłanego pakietu

```
class Memory
{
  public int nr;
  public long t;
  public Z2Packet pa;
  public Memory(int ord, long pt, Z2Packet mess)
  {
    this.nr = ord;
    this.t = pt;
    this.pa = mess;
  }
}
```

Procedura dodawania do pamięci wysłanych pakietów realizowana jest w momencie ich wysyłania, to znaczy, wystarczyło nieznacznie zmodyfikować pętlę głównego wątku odpowiedzialnego za wpuszczanie pakietów do medium transmisyjnego.

```
        for(i=0; (x=System.in.read()) >= 0 ; i++)
        {
          ...
          Memory wyslany = new Memory(i,System.currentTimeMillis(),p);
          memory.add(wyslany);
        }
```

Jeśli potwierdzenie odebrania pakietu od odbiorcy nie przyjdzie w ciągu 4 sekund, następuje retransmisja, obsługiwana przez inny wątek:

```
  Thread retransmissionControl = new Thread()
  {
    public void run()
    {
      try
```

```
6         {
7           while (true)
8           {
9             for (int j=0; j<memory.size(); j++)
10            {
11              if (memory.get(j).t-System.currentTimeMillis() < 4000)
12              {
13                Z2Packet retransmitowany = new Z2Packet(1);
14                retransmitowany.data = memory.get(j).pa.data;
15                retransmitowany.setIntAt(memory.get(j).nr, 0);
16                DatagramPacket datagramRetransmitowany = new DatagramPacket(retransmitowany.data
                       , retransmitowany.data.length ,localHost , destinationPort);
17                socket.send(datagramRetransmitowany);
18                memory.get(j).t = System.currentTimeMillis();
19                System.out.println("Retransmisja: "+ memory.get(j).nr);
20              }
21            }
22            sleep(500);
23          }
24        }
25        ...
26      };
27    };
```

Wreszcie, w celu uniknięcia retransmitowania pakietów przesłanych poprawnie, w klasie *ReceiverThread* umieszczono dodatkową pętlę

```
1             for (int i=0; i<memory.size(); i++)
2             {
3               if ( p.getIntAt(0)==memory.get(i).nr)
4               {
5                 memory.remove(i);
6               }
7             }
```

Zmiany w *Z2Receiver* polegają na utworzeniu bufora *received* z przechowywanymi pakietami, które zostały odebrane. Pakiety nie są już wypisywane w kolejności odebrania, lecz według swoich numerów sekwencyjnych.

```
1               received.add(p);    //dodaj do listy otrzymanych
2         for (int i=0; i<received.size(); i++)   //w kazdej iteracji while wypisz otrzymane
3         {
4           if(received.get(i).getIntAt(0) == numerPakietu) //wypisz aktualny pakiet
5           {
6             numerPakietu++;   //ok. mozesz czekac na kolejny pakiet
7             System.out.println("R:"+received.get(i).getIntAt(0)+": "+(char) received.get(i).data
                   [4]);
8             received.remove(received.get(i));
9           }
10        }
```

3 Testowanie

Uruchomienie symulacji polega na wpisaniu w czterech konsolach poniższych poleceń:

```
1 java Z2Receiver 6002 6003>plik2.txt
2 java Z2Forwarder 6001 6002
3 java Z2Forwarder 6003 6000
4 java Z2Sender 6000 6001<plik.txt
```

Rezultatem będzie następująca zawartość pliku wynikowego *plik*2.*txt*:

```
1 R:0: A
2 R:1: l
```

```
R:2: a
R:3:
R:4: m
R:5: a
R:6:
R:7: k
R:8: o
R:9: t
R:10: a
R:11: .
R:12:

R:13: O
R:14: l
R:15: a
R:16:
R:17: n
R:18: i
R:19: e
R:20:
R:21: m
R:22: a
R:23:
R:24: k
R:25: o
R:26: t
R:27: a
R:28: .
R:29:

R:30: E
R:31: l
R:32: a
R:33:
R:34: m
R:35: a
R:36:
R:37: d
R:38: w
R:39: a
R:40:
R:41: k
R:42: o
R:43: t
R:44: y
R:45: .
R:46:
```

4 Podsumowanie

Na podstawie pliku wynikowego można stwierdzić, że wprowadzone do wyjściowych programów zmiany gwarantują poprawne przesyłanie i odbieranie pakietów w warunkach naturalnych. Stwierdzono, że *Z2Sender* zatrzymuje się po odebraniu wszystkich potwierdzeń.