Technologie sieciowe

sprawozdanie lista 4

Autor: Jakub Duda 236778

1. Cel

Celem zadania była modyfikacja zadanych programów, które symulowały wysyłanie pakietów. Należało to zrobić w taki sposób, aby mimo zakłóceń, które symulowały sieć internetową, odbiorca był w stanie wydrukować wszystkie nadane mu pakiety w kolejności sekwencyjnej.

2. Realizacja

2.1 Wysyłanie

Program wysyłający składa się z dwóch wątków, które są uruchamiane w metodzie `main`. Pierwszy z nich `SenderThread` odpowiada za wysyłanie wiadomości. Drugi z nich `ReceiverThread` odpowiadał za odbieranie potwierdzeń dostarczenia wysyłanych wiadomości. Modyfikacja procesu wysyłania polegała na dodaniu wysłanego elementu do listy `listToSend`.

```
 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);
                    socket.send(packet);
                    toSend.add(packet);
                }

            } catch (Exception e) {
                System.out.println("Z2Sender.SenderThread.run: "
+ e);
            }
        }
    }
```

Listing 1:Wysyłanie wiadomości.

Modyfikacja wątku odbierania potwierdzeń polegała na retransmisji od wiadomości, dla której przyszło potwierdzenie.

```
 class ReceiverThread extends Thread {

        public void run() {
            try {
                while (true) {
                    byte[] data = new byte[datagramSize];
                    DatagramPacket packet = new
DatagramPacket(data, datagramSize);
                    socket.receive(packet);
                    Z2Packet p = new Z2Packet(packet.getData());
                    System.out.println("S:" + p.getIntAt(0) + ":
" + (char) p.data[4]);
                    missing = p.getIntAt(0);
                    if (missing == toSend.size()) {
                        Z2Packet end = new Z2Packet(4 + 1);
                        end.setIntAt(-1, 0);
                        end.data[4] = (byte) 'x';
                        DatagramPacket endPacket = new
DatagramPacket(end.data, end.data.length, localHost,
destinationPort);
                        socket.send(endPacket);
                        toSend.add(endPacket);
                    } else {
                        for (; missing < toSend.size(); missing+
+) {
                            socket.send(toSend.get(missing));
                            sleep(sleepTime);
                        }
                    }
                }
            } catch (Exception e) {
                System.out.println("Z2Sender.ReceiverThread.run:
" + e);
            }
        }
    }
```

Listing 2: Odbieranie potwierdzeń doręczenia.

2.2 Odbieranie

Program nadsłuchujący przychodzących wiadomości posiada dwa wątki uruchamiane w metodzie statycznej `main`. Modyfikacja tego programu polegała na dodaniu wątku który będzie wysyłał wiadomość z wiadomością której brakuje. Dodany został również licznik counter, którego zadaniem było kontrolować przychodzące wiadomości. Dla wiadomości, której numer sekwencyjny był równy z licznikiem, program wypisywał otrzymaną wiadomość, a następnie zwiększał licznik. W przypadku gdy wiadomość została utracona lub opóźniona i nadeszła późniejsza program dodawał ją do mapy `chars`, która składała się z klucza — numeru sekwencyjnego i wartości — znaku.

```
  class ReceiverThread extends Thread {

        public void run() {
            try {
                while (true) {

                    byte[] data = new byte[datagramSize];
                    DatagramPacket packet = new
DatagramPacket(data, datagramSize);
                    socket.receive(packet);
                    Z2Packet p = new Z2Packet(packet.getData());
                    int i = p.getIntAt(0);
                    char c = (char) p.data[4];
                    if (i < 0) {
                        System.out.println("Dostarczono");
                        deliver = true;
                    }
                    if (i > counter) {
                        chars.put(i, c);
                    } else if (i == counter) {
                        System.out.println("R:" + i + ":" + c);
                        counter++;
                    }
                    while (chars.containsKey(counter)) {
                        System.out.println("R:" + counter + ":" +
chars.get(counter));
                        chars.remove(counter);
                        counter++;
                    }
                }
            } catch (Exception e) {

System.out.println("Z2Receiver.ReceiverThread.run: " + e);
            }
        }

    }
```

Listing 3: Odbieranie wiadomości.

Dodany wątek, gdy zostanie przekroczony czas oczekiwania, wysyła wiadomość z numerem sekwencyjnym, którego brakuje.

```
 class SendConfirmationThread extends Thread {

        public void run() {
            try {
                long sTime = System.currentTimeMillis();
                while (!deliver) {
                    long eTime = System.currentTimeMillis() -
sTime;
                    if ((counter + 1) * 500 + 5000 < eTime) {
                        //System.out.println("Brakuje " +
counter);
                        Z2Packet s = new Z2Packet(4 + 1);
                        s.setIntAt(counter, 0);
                        s.data[4] = (byte) 'x';
                        DatagramPacket miss = new
DatagramPacket(s.data, s.data.length, localHost,
destinationPort);
                        socket.send(miss);
                        sTime = System.currentTimeMillis();
                    }
                }

            } catch (Exception e) {

System.out.println("Z2Receiver.SendConfirmationThread.run: " +
e);
            }
        }
    }
```

Listing 4: Wysyłanie zagubionych wiadomości.

3.Wnioski

Mimo iż po drodze wiadomości mogą być tracone, przybywać z rożnymi opóźnieniami, w zmienionej kolejności, a nawet mogą być duplikowane, dzięki modyfikacjom program otrzymujący wiadomość poprawnie jest w stanie ją wyświetlić. Dzieje się tak, dlatego że program odbierający wiadomość wyświetla wiadomości według numerów sekwencyjnych, jeśli któregoś zabraknie, wysyła wiadomość do nadajnika. Ten po otrzymaniu wiadomości zaczyna retransmitować, od którego sygnału brakowało.