Sprawozdanie nr 4

Technologie sieciowe

Laboratorium piątek 9:15

Weronika Jasiak
236733

1. Cele

Celem ćwiczenia jest modyfikacja programów by symulowały one przesyłanie danych przy pomocy protokołu.

2. Realizacja

2.1. Definicje

TCP (ang. Transmisją Control Protokół) jest to protokół służący do przesyłania danych między hostami. Protokół ten gwarantuje dostarczenie wszystkich pakietów w całości, z zachowaniem kolejności i bez duplikatów.

TCP/IP jest to Protokół TCP korzystający z usług protokołu IP do wysyłania i odbierania danych oraz ich fragmentacji.

2.2. Algorytm wykorzystany w zadaniu.

1. Dane wraz z numerem sekwencyjny zostają wysyłane z hosta nadającego (dalej nazywanym X) do hosta odbierającego (dalej nazywanym Y).

2. Po odebraniu danych przez Y, wysyłane jest potwierdzeniu odbioru wraz z numerem sekwencyjnym do X.

3. Numer sekwencyjny umożliwia odpowiednia kolejność odczytywania wiadomości przez Y.

4. W przypadku gdy X nie otrzyma potwierdzenia otrzymania pakietu od Y w ustalonym czasie, następuje retransmisja tego pakietu czyli ponowne wysłanie.

2.3. Przebieg ćwiczenia

Grupa programów pomagająca w symulacji protokołu TCP/IP składa się z następujących plików:

- Z2Forwarder - program symulujący medium transmisyjne oraz sytuacje w których pakiety nie docierając do hostów.
- Z2Packet - klasa imitująca pakiet w datagramie.
- Z2Sender - program symulujący host wysyłający dane.
- Z2Receiver - program symulujący host odbierający dane.

```
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());
                        buffer[p.getIntAt(0)] = p;
                        // WYSLANIE POTWIERDZENIA
                        packet.setPort(destinationPort);
                        socket.send(packet);
                  }
            } catch (Exception e) {
                  System.out.println("Z2Receiver.ReceiverThread"
                  + ".run: " + e);
            }

      }

}
```

Kod. 1: Metoda klasy Z2Receiver służąca do odbierania wiadomości.

Metoda odbiera pakiet z gniazda sieciowego, a następnie zapisuje pakiet do komórki tablicy o indeksie równym numerze sekwencyjnym pakietu. Na koniec odsyła potwierdzenie otrzymania pakietu.

```
class PrintThread extends Thread {
	@Override
	public void run() {
		try {
		    while (true) {
		        if (buffer[nextItem] != null)
		            System.out.println("Received: " + (char)
		                buffer[nextItem].data[4]);
		            nextItem++;
		        } else {
		            Thread.sleep(50);
		        }
		    }
		} catch (Exception e) {
		    e.printStackTrace();
		}
	}
}
```

Kod. 2: Metoda klasy Z2Receiver służąca do drukowania wiadomości.

Metoda ta drukuje w odpowiedniej kolejności na standardowe wyjście znaki przekazane w pakiecie.

```
class SenderThread extends Thread {
	@Override
	public void run() {
		int i, x;
		try {
		      for (i = 0; (x = in.read()) >= 0; i++) {
		            Z2Packet p = new Z2Packet(4 + 1);
		            p.setIntAt(i, 0);
		            p.data[4] = (byte) x;
		            p.received = false;
		            synchronized (packetsBuffer) {
		                  packetsBuffer.add(p);
		            }
		            sendPacket(p);
		            System.out.println("Send: " + (char)
		                  p.data[4]);
		            sleep(sleepTime);
		      }
		} catch (Exception e) {
		      System.out.println("Z2Sender.SenderThread.run: "
		            + e);
		}
	}
}
```

Kod. 3: Metoda w Klasie Z2Sender służąca do wysyłania pakietów.

Metoda ta wczytuje znaki ze standardowego wejścia, tworzy z nich pakiety, dodaje do listy pakietów i wysyła, następnie odczekuje określony czas do wysłania kolejnego pakietu.

```
class ReceiverThread extends Thread {
      @Override
        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());
                        p.received = true;
                        synchronized (packetsBuffer) {
                              packetsBuffer.set(p.getIntAt(0), p);
                        }
                  }
            } catch (Exception e) {
                  System.out.println("Z2Sender.ReceiverThread.run”
                        + e);
            }
      }
}
```

Kod. 4: Metoda w Klasie Z2Sender służąca do odbierania potwierdzeń dotarcia pakietu.
Metoda ta służy do odbierania potwierdzeń dostarczenia pakietu do hostu docelowego i oznaczania pakietów jako
dostarczone.

```
class RetransmitThread extends Thread {
      @Override
      public void run() {
            try {
            while (true) {
            Thread.sleep(1000);
            synchronized (packetsBuffer) {
                  for (Z2Packet packet : packetsBuffer) {
                        if (!packet.received) {
                              System.out.println("Retransmit: " +
                                    (char) packet.data[4]);
                              sendPacket(packet);
                         }
                  }
            }
            }
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }
}
```

Kod. 5: Metoda w Klasie Z2Sender służąca do retransmisji pakietu.

Po określonym czasie oczekiwania (1000) nadawana jest retransmisja pakietów dla których nie została zanotowana odpowiedź.

3. Przykładowy przebieg

```
Retransmit: i
Retransmit: e
Retransmit: m
Retransmit: a
Retransmit:
Send: k
Send: o
Received: k
Received: o
Received: t
Received: a
```

Output. 6: Część wywołania dla danych z pliku plik.txt

Pakiet wysyłany jest minimum dwukrotnie (pierwszy raz jako wiadomość, drugi raz jako potwierdzenie dotarcia pakietu). Kolejne pakiety drukowane są tylko wtedy gdy dotrą do hosta.

4. 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 czyli mogą występować w nim zakłócenia powodujące gubienie pakietów.