CN Task5

Julian Bertol

November 21, 2024

1

1.1 Entwickeln Sie zuerst den Sender

```java
import java.io.IOException;
import java.io.Serializable;
import java.net.Socket;
import java.io.ObjectOutputStream;

public class Auto implements Serializable {
    private String farbe, marke;
    private int ps;
    private double gewichtInTonnen;

    public Auto(String farbe, int ps, double gewichtInTonnen, String marke) {
        this.farbe = farbe;
        this.ps = ps;
        this.gewichtInTonnen = gewichtInTonnen;
        this.marke = marke;
    }

    public String holeBeschreibung() {
        return "Ich bin ein " + farbe + "er " + marke;
    }

    public String holeSpassfaktor() {
        String result = "Ich wiege " + gewichtInTonnen + " Tonnen mit " + ps + " PS.";
        if (ps / gewichtInTonnen > 130) {
            result += " Ich bin sehr schnell!";
        } else if (ps / gewichtInTonnen > 80) {
            result += " Ich bin flott!";
        }
        return result;
    }

    public static void Sender() throws IOException {
        Auto myAuto = new Auto("Silbern", 86, 1.2, "Opel");
        Socket socket = new Socket("127.0.0.1", 1337);
        ObjectOutputStream outputStream = new ObjectOutputStream(socket.
            getOutputStream());

        outputStream.writeObject(myAuto);
        outputStream.flush();

        System.out.println("Auto wurde erfolgreich gesendet!");
    }

    private static void startReceiver() {
        Thread receiverThread = new Thread(() -> {
            try {
                Receive.Server();
            } catch (Exception e) {
                System.err.println("Fehler beim Starten des Servers:");
            }
        });

```

```
        receiverThread.start();
    }

    public static void main(String[] args) throws IOException {
        startReceiver();
        Sender();
    }
}
}
```

1.2 Empfänger des Autos

```
import java.io.IOException;
import java.io.ObjectInputStream;
import java.net.ServerSocket;
import java.net.Socket;

public class Receive {
    public static void main(String[] args) throws IOException, ClassNotFoundException
        {
        ServerSocket serverSocket = new ServerSocket(1337);
        System.out.println("Server l uft und wartet auf eine Verbindung...");
        Socket socket = serverSocket.accept();
        System.out.println("Verbindung von " + socket.getInetAddress() + " hergestellt
            .");
        ObjectInputStream objectInputStream = new ObjectInputStream(socket.
            getInputStream());
        Auto myAuto = (Auto) objectInputStream.readObject();
        System.out.println("Beschreibung: " + myAuto.holeBeschreibung());
        System.out.println("Spa faktor: " + myAuto.holeSpassfaktor());
    }
}
```

1.3 Tests

```
Server läuft und wartet auf eine Verbindung...
Verbindung von /127.0.0.1 hergestellt.
Auto wurde erfolgreich gesendet!
Beschreibung: Ich bin ein Silverner Opel
Spaßfaktor: Ich wiege 0.9 Tonnen mit 500 PS. Ich bin sehr schnell!
```

```
julian@UbuntuJulian:~/send_test$ java Receive
Server läuft und wartet auf eine Verbindung...
Verbindung von /10.100.0.124 hergestellt.
Beschreibung: Ich bin ein Silberner Opel
Spaßfaktor: Ich wiege 1.2 Tonnen mit 95 PS.
```

2 Stop-and-Wait Berechnung

- Gegeben:
  - L = 1500 bytes (12000 bit) Paketübertragung
  - D = 6400km (Distanz)
  - V = $3 * 10^8 m/s$
  - R = $1Gbit/s(10^9 bit/s)$ Bandbreite
- Gesucht:
  - RTT
  - U: Auslastung

2.1 Wie hoch wäre die Auslastung durch einen einzelnen Sender für ein Überseekabel von der HS Furtwangen nach New York an der Ostküste in USA mit dem Stop-and-Wait-Verfahren?

- U = $\frac{L/R}{RTT+L/R}$
- RTT = $\frac{2*D}{v}$
- RTT = $\frac{2*6400000m}{3*10^8 m/s}$
- RTT = 0.04267 S = 4.27 ms
- U = $\frac{12000bit/10^9 bit/s}{4.27ms+12000bit/10^9 bit/s}$
- U = 0.0281%

2.2 Wie lange darf eine Verbindungsstrecke höchstens sein, damit Daten noch 500Mbit/s über das Stop-and-Wait-Verfahren übertragen werden können?

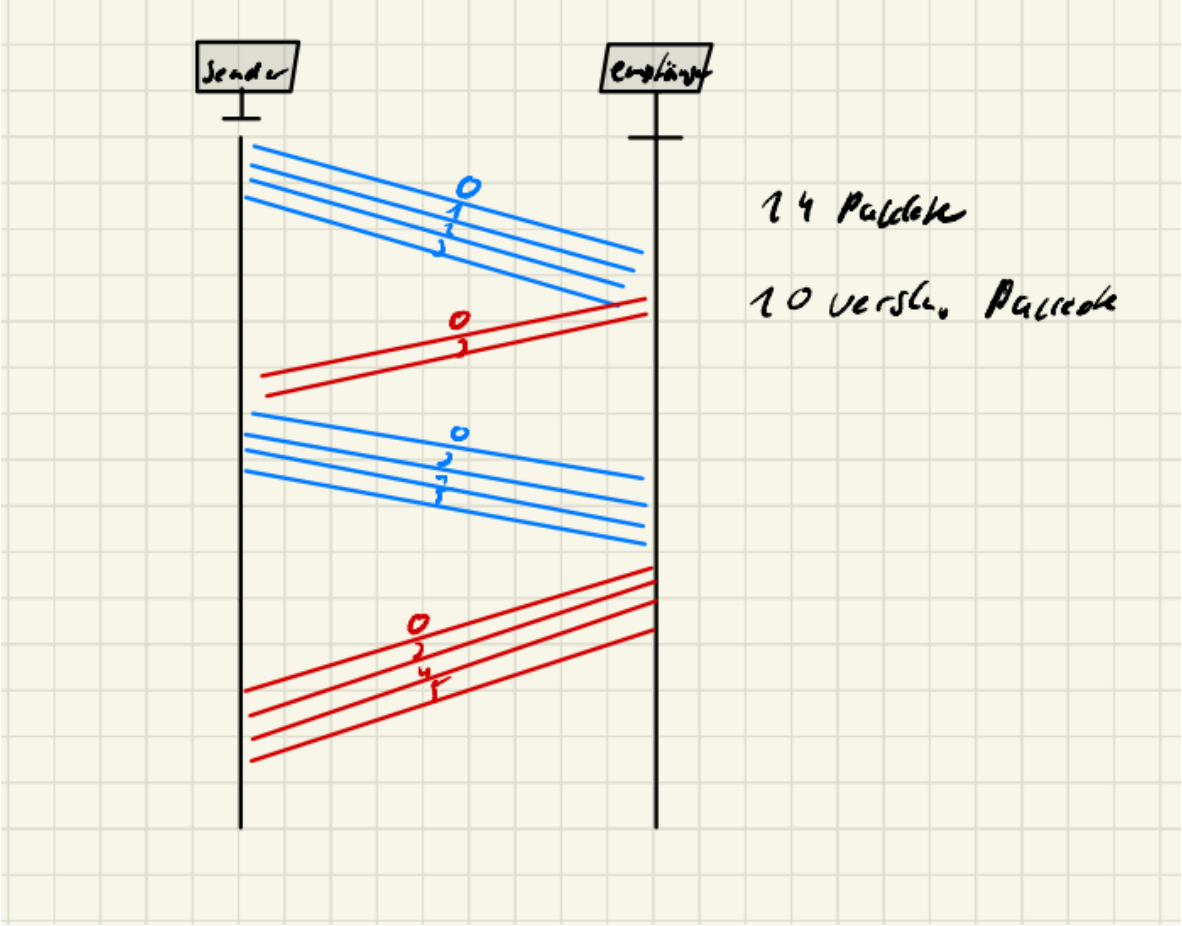
- Gegeben:
  - V = $500Mbit/s = 500000000bit/s = 500 * 10^6 bit/s$ (Übertragungsrate)
  - P = 12000bit (Packetgröße)
  - R = $1Gbit/s(10^9 bit/s)$ Bandbreite
- Gesucht:
  - RTT(max)
  - D(max)
- RTT(max) = $\frac{P}{V} - \frac{P}{R}$
- RTT(max) = $\frac{12000bit}{500000000bit/s} - \frac{12000bit}{10000000000bit/s}$
- RTT(max) = 0,000012s
- D(max) = $\frac{RTT(max)*v}{2}$
- D(max) = $\frac{0,000012s*3*10^8 m/s}{2}$
- D(max) = 1800m = 1.8km

2.3 Hätte ein Stop-and-Wait-Verfahren eine negative Auswirkung auf die Geschwindigkeit Ihrer WLAN oder Mobilfunkverbindungen? Betrachten Sie nur die Strecke zwischen Ihrem Endgerät und der Funkstation.

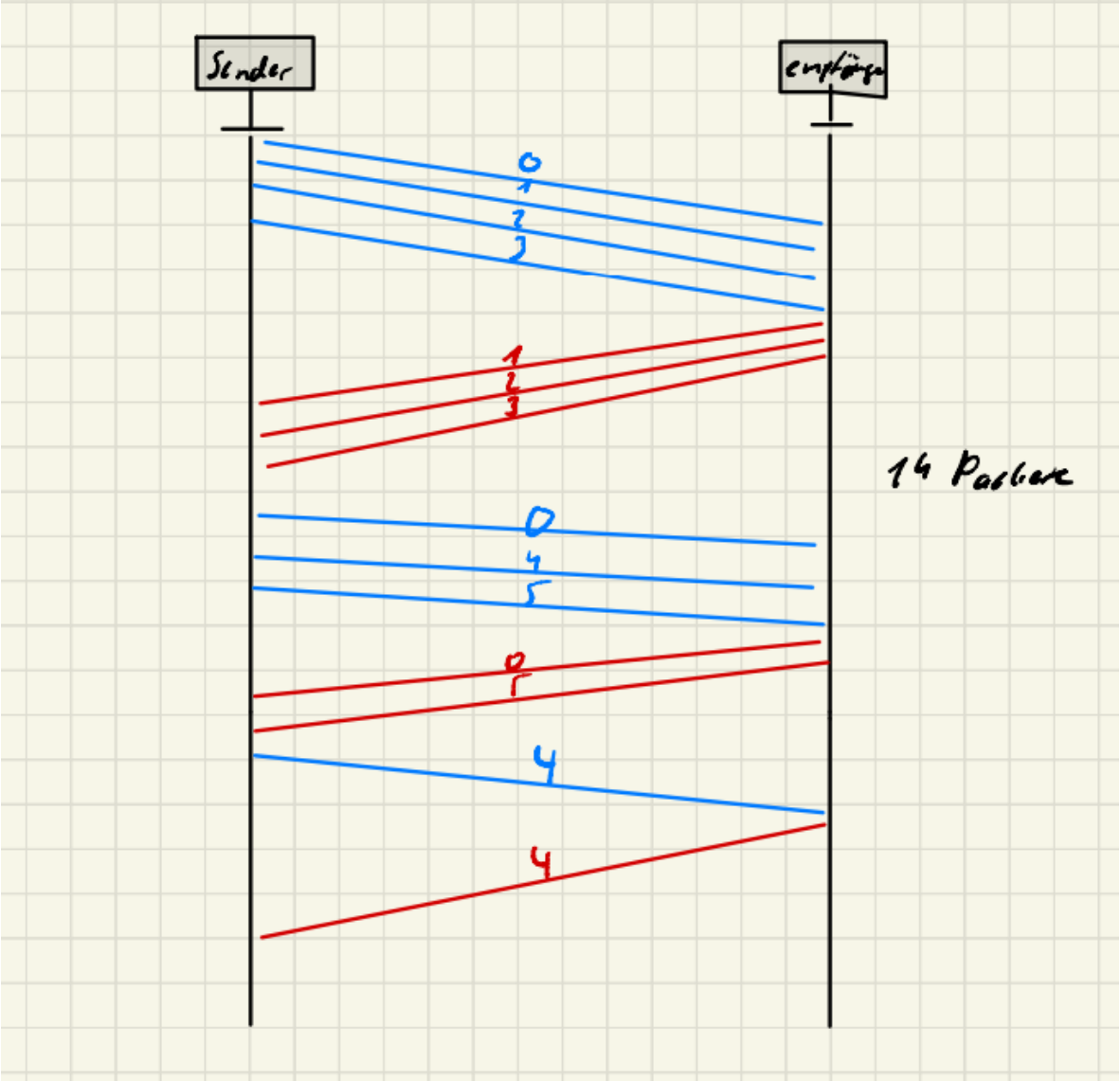
- Ja, da Verbindungen über Funk vor allem größere Strecken eine hohe Latenz nach sich ziehen
- Da man jedes mal auf die Bestätigung warten muss, kann das einiges länger dauern

3 Go-Back-N Verfahren

3.1 Stellen Sie den Vorgang grafisch als Ablaufdiagramm dar, wenn bei der ersten Übertragung von Paket 1 und Paket 2 ein Verlust auftritt (das Paket erreicht den Empfänger nicht). Wie viele Pakete wurden insgesamt übertragen?



3.2 **Stellen sie den Vorgang dar, wenn beim Senden von Paket 0 und Paket 4 ein Verlust auftritt. Wie viele Pakete wurden insgesamt übertragen?**



4

4.1 Warum kann es notwendig sein, bei der Verwendung von Text-Streams die Kodierung einzustellen?

- Weil sost eventuel manche Sonderzeichen nicht korrekt angezeigt werden.

4.2 Wieso ist das bei der Übertragung von Daten über einen Bitstream nicht der Fall?

- In einem Bitstream werden rohe Binärdaten übertragen, die keine spezifische Zeichenkodierung voraussetzen. Da hier keine Interpretation als Text erfolgt, spielt die Kodierung keine Rolle.

4.3 Suchen Sie im Internet (z.B. auf Wikipedia) nach der Funktionsweise von Base64. Wieso werden binäre Daten mit Base64 codiert, falls Sie über ein textbasiertes Protokoll übertragen werden?

- Binäre Daten enthalten oft nicht-druckbare Zeichen oder solche, die von textbasierten Protokollen (z. B. E-Mail, HTTP) nicht unterstützt werden. Base64 codiert Binärdaten in ein ASCII-Format, das ausschließlich druckbare Zeichen verwendet.

4.4 Erinnern Sie sich an die GPG-Aufgaben letztes Mal. Was bewirkte der Schalter –armor beim Exportieren von Schlüsseln? Wieso könnte dies sinnvoll sein?

- Der Schalter –armor in GPG wandelt die Binärdaten des Schlüssels in ein lesbares ASCII-Format (ähnlich Base64) um.