Marek Kubica kubica@in.tum.de, Michael Kerscher kerschem@in.tum.de

Punkteverteilung:

1	2	Σ

Aufgabe (1)

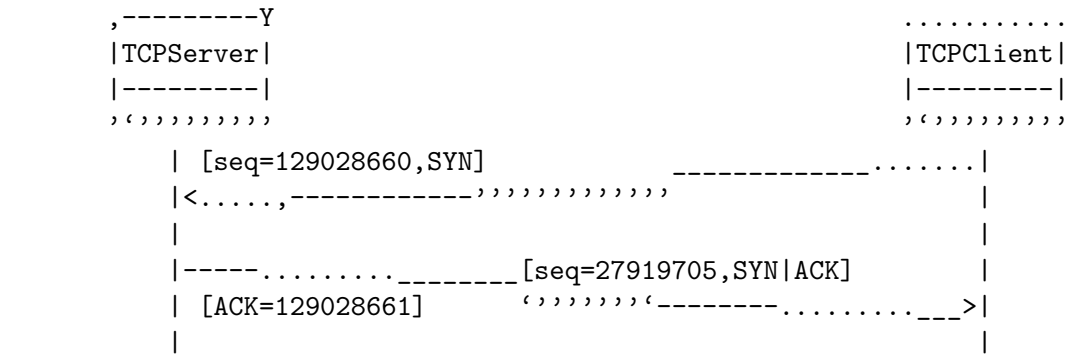
(a) Erledigt.

(b) Erledigt.

(c)
```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.Socket;

public class GrnvsTcpClient {
  private static BufferedReader in;
  private static BufferedWriter out;
  private static Socket socket;

  public static void main(String[] args) {
    if (args.length != 2) {
      System.err.println("Usage: java GrnvsTcpClient <host>
          <port>");
      System.exit(1);
    }
    communicate(args[0], Integer.parseInt(args[1]));
  }

  private static void communicate(String host, int port) {
    try {
      socket = new Socket(host, port);
      in = new BufferedReader(new InputStreamReader(socket.
          getInputStream()));
      out = new BufferedWriter(new OutputStreamWriter(socket
          .getOutputStream()));
      System.out.println(in.readLine());
      say("HELLO");
      hear();
      hear();
      int[] numbers = {0, 0, 0};
      for (int i = 0; i < 3; i++) {
        String line = in.readLine();
        numbers[i] = Integer.parseInt(line.trim());
      }
      int sum = 0;
      for (int i = 0; i < 3; i++) {
        sum += numbers[i];
```

```
            }
            say(String.valueOf(sum));
            if (!hear().equals("Please␣send␣your␣Teamletter..."))
                {
              System.err.println("Sum␣didn't␣match");
              return;
            }
            say("G");
            hear();
            say("Kubica␣Marek");
            out.close();
            in.close();
            socket.close();
          } catch (IOException e) {
            System.err.println(e.toString());
            System.exit(1);
          }
        }

        private static String hear() {
          try {
            String read = in.readLine();
            System.out.println(read);
            return read;
          } catch (IOException e) {}
          return null;
        }

        private static void say(String msg) {
          try {
            System.out.println("—>␣" + msg);
            out.write(msg + "\r\n");
            out.flush();
          } catch (IOException e) {
            System.err.println(e.toString());
          }
        }
    }
```

(d)



(e) Es kommen als Quelladresse die MAC der Netzwerkkarte auf dem das Programm läuft und als Zieladresse die MAC der Netzwerkkarte des Routers, die an das interne Netzwerk angeschlossen ist.

Die Hersteller sind nach der Lookup-Seite Dage-MTI of MC, Inc respektive PC Engines GmbH. Ersteres ist verwunderlich, denn Wireshark würde die MAC-Adresse eher Samsung Electronics zuweisen, was sich auf mit der Aufschrift auf dem Gerät decken würde.

(f) Die TTL der Pakete vom Server ist 54. Da das OS des Servers Linux ist (SUSE-Linux, mit Apache 2.2.8, was auf SUSE-Linux 10.x schließen würde) und die Standard-TTL von Linux 64 ist, ist anzunehmen dass die Pakete mit einer TTL von 64 gestartet sind.

Eine Gegenprobe mit `traceroute` ist leider nicht möglich, da das Paket nach dem Hop bei `nz-net-bb.informatik.tu-muenchen.de` verschwindet und keine Antwort vom Empfänger kommt. Selbst wenn man die TTL auf hohe Werte wie 129 stellt kommt keine Antwort zurück.

(g) Wenn man von der Zahl des ACK-Headers die Länge des vorhergehenden, entsprechenden SEQ-Header abzieht dann kommt immer jeweils die erwartete Größe des Paketes heraus. Als Beispiel der größere Sprung bei dem die "GrnvsTcpServerNachricht übertragen wird: 27919766 - 27818706 = 60, was der erwarteten Länge der übertragenen Daten entspricht (die Länge dieses Paketes variiert mit der Länge der IP-Adresse des Clients, da die Adresse vom Server mitübertragen wird: 100.100.100.100 ist 8 Byte größer als 8.8.8.8).

(h) Es sind ACK und PSH gesetzt. PSH bedeutet, dass das Paket sofort gesendet wird, ohne zu warten bis die Sendepuffer voll sind. Dadurch wird die Kommunikation beschleunigt, da nicht auf eine Bestätigung der Gegenseite gewartet werden muss und dies macht auch für das obige Programm einen Unterschied.

Aufgabe (2)

(a) Der Adressraum enthält bei IPv6 $2^{128} \approx 3.4 \cdot 10^{38}$ Adressen.

Link-Local-Adressen sind Adressen mit dem Format fe80::/10 und beinhalten neben der Kennung des Subnetzes auch die Kennung der Netzwerkkarte. Diese Adressen dürfen von Routern nicht weitergeleitet werden und sind daher nur im gleichen Teilnetz erreichbar.

(b)

Bit	0-3	4-7	8-11	12-15	16-19	20-23	24-27	28-32
0	6	X	X	Y	Y	Y	Y	Y
32	0	3	E	A	0	6	4	0
64	F	E	8	0	0	0	0	0
96	0	0	0	0	0	0	0	0
128	0	0	0	0	0	2	0	7
160	0	0	0	6	3	1	5	2
192	F	E	8	0	0	0	0	0
224	0	0	0	0	0	0	0	0
256	0	0	0	0	0	2	1	7
288	1	1	A	6	0	0	5	2

Das Feld *Hop Limit* gibt an, wieviele Router das Paket durchlaufen darf. Jeder Router auf dem Weg verringert das Feld um eins. Wenn die Zahl 0 erreicht, wird das Paket verworfen. Es ist vergleichbar mit dem TTL-Wert aus IPv4.

(c)

Bit	0-3	4-7	8-11	12-15	16-19	20-23	24-27	28-32
0	6	X	X	Y	Y	Y	Y	Y
32	0	1	F	C	2	C	4	0
64	F	E	8	0	0	0	0	0
96	0	0	0	0	0	0	0	0
128	0	0	0	0	0	2	0	7
160	0	0	0	6	3	1	5	2
192	F	E	8	0	0	0	0	0
224	0	0	0	0	0	0	0	0
256	0	0	0	0	0	2	1	7
288	1	1	A	6	0	0	5	2

Bit	0-3	4-7	8-11	12-15	16-19	20-23	24-27	28-32
0	6	0	0	0	0	0	0	1
32	D	E	A	D	B	E	E	F

(d) Ein IPv6-Multicast-Paket erkennt man daran, dass die ersten 8 Bits der Adresse Einsen sind, also Adressen mit Format ff00::/8.