Aufgabe 3 [Paketübertragung] (20 Punkte)

Gegeben sei die in Abbildung 1 dargestellte Übertragungsstrecke mit den 3 Netzknoten R_1, R_2 und R_3. Die 3 Quellen Q1, Q2, und Q3 senden Pakete zu dem Zielknoten Z. Die Kapazitäten, Ausbreitungsverzögerungen und Puffergrößen der Links, die die Netzknoten verbinden, sind in der Abbildung gegeben.
In den folgenden Teilaufgaben beträgt die Paketgröße 1500 Bytes. ACKs benötigen 15ms vom Zielknoten Z zu einem Quellknoten Qx.

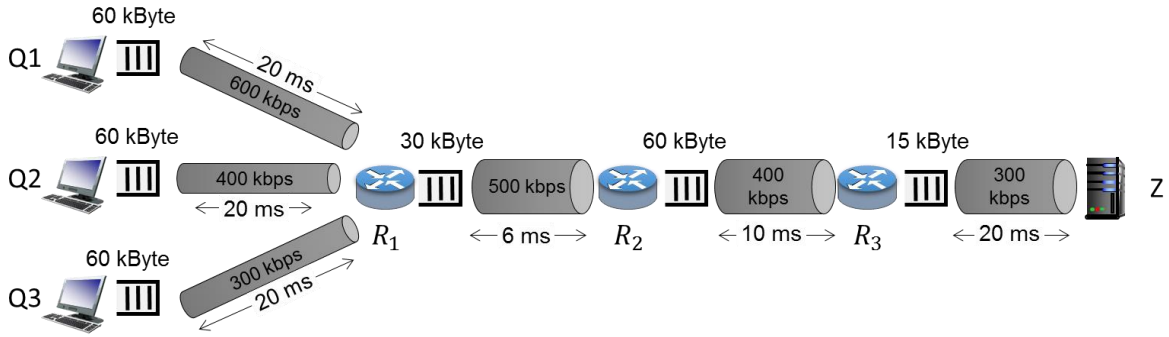

Abbildung 1: Übertragungsstrecke

1) Bestimmen Sie für jede Quelle Q3 die Ende-zu-Ende-Übertragungsdauer eines Pakets. (2P)
2) Quelle Q1 versendet Pakete mit einer Datenrate von 700 kbps. Q2 und Q3 versenden keine Pakete. Bestimmen Sie die ersten 3 Pakete, die verloren gehen. Geben Sie auch an, vor welchem Link die Pakete verloren gehen. (4P)
3) Betrachten Sie nun die Situation, dass zusätzlich zu Quelle Q1 auch die Quellen Q2 mit 600 kbps und Q3 mit 3 Mbps übertragen. Quelle Q1 fängt zuerst an Pakete zu senden, Quelle Q2 startet 1ns später und Quelle Q3 noch 1ns später. Bestimmen Sie das erste Paket, das verloren geht. Geben Sie das Paket in der Form Pq,k an, wobei Pq,k das k-te von Quelle q gesendete Paket ist. Vor welchem Link geht das Paket verloren? (4P)
4) Betrachten Sie nun die Situation, dass zwischen Quelle und Senke auf der Transportschicht ein Go-Back-N-Protokoll mit einem Sendefenster von N=5 Paketen läuft. Gehen Sie von einer saturierten Quelle aus, d.h. an der Quelle sind immer genügend Daten vorhanden, um das Sendefenster voll auszunutzen. Betrachten Sie nicht die Situation am Anfang der Kommunikation sondern nachdem bereits zahlreiche Pakete ausgetauscht wurden. Bestimmen Sie
   a) die Paketverlustrate für die Quellen Q1, Q2 und Q3 (Begründen Sie ihre Antwort) (2P)
   b) die Ende-zu-Ende Übertragungsdauer für ein Paket von Q3 inklusive Wartezeit aber ohne Verarbeitungszeit. (8P)

Aufgabe 4 [HTTP] (17 Punkte)

In dieser Aufgabe wird eine Web-Seite betrachtet, deren Aufbau der Tabelle auf der nächsten Seite zu entnehmen ist. Im Browser und auf dem Web-Server läuft die http-Version „Persistent http ohne Pipelining“. Hinweise zur Notation und Parameter der TCP-Verbindungen finden Sie ebenfalls auf der nächsten Seite, auf der Sie die Aufgabe bearbeiten.

1) Skizzieren Sie in Abbildung 1 den Verlauf der Seitenübertragung **beginnend mit dem ersten Request bis zur vollständigen Übertragung des dritten Bildes**. Tragen Sie pro RTT die Anzahl und den Inhalt der vom Server bzw. Client übertragenen Segmente ein. Gehen Sie davon aus, dass alle Pakete, die gleichzeitig gesendet werden auch gleichzeitig und in der richtigen Reihenfolge ankommen. (8P)

2) Skizzieren Sie in Abbildung 2 die Übertragung des siebten und achten Bildes, wenn Segmente 12 bis 15 des siebten Bildes verloren gehen. Wie groß ist das Sendefenster des Web-Servers nach Empfang des letzten ACKs für ein Segment von Bild 8? (9P)

Name: Matrikelnummer:

Verwenden Sie folgende Notation

- Requests: `MO:Req:Sx-y`, `IOn:Req:Sx-y`
- Segmente: `MO:Sx-y`, `IOn:Sx-y`
- ACKs: `n x ACK` (die Anzahl genügt)

Aufbau der Web-Seite:

Objekt	Größe Request [Bytes]	Größe Objekt [Bytes]	
HTML Code	300 Bytes	1600 Bytes	
Bild 1	1700 Bytes	1200 Bytes	
Bild 2-10	700 Bytes	10000 Bytes	

Übersicht weiterer Parameter:

MSS	500 Bytes
IW	2 MSS

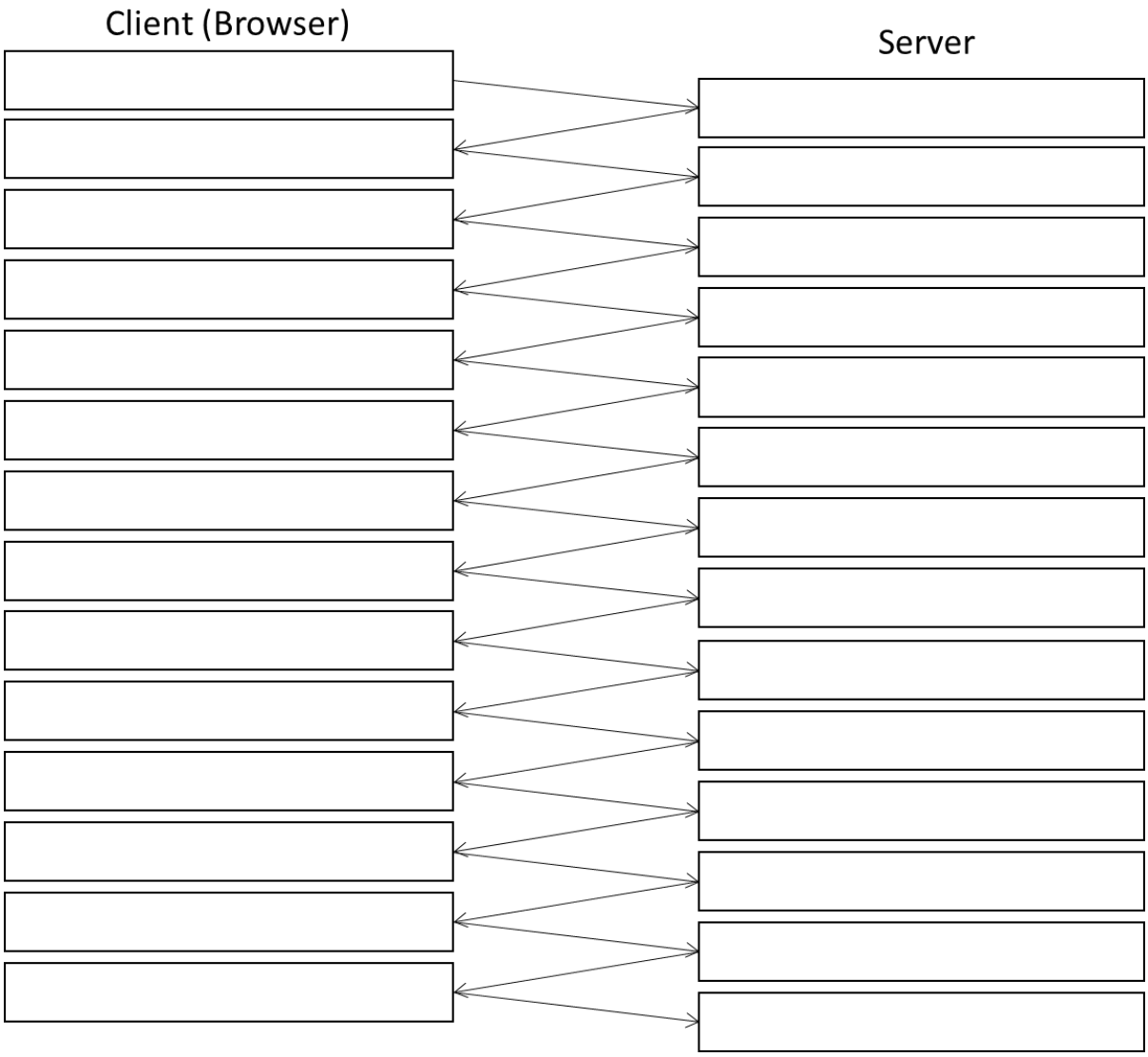

Abbildung 1: Vorlage für Skizze in Aufgabe 2.1)

Name: Matrikelnummer:

Verwenden Sie folgende Notation:

- Requests: `IO`**`n:`**`Req`
- Segmente: `IO`**`n:`**`S`**`x-y`**
- ACKs: `IO`**`n:`**`A`**`x-y`** `bzw. IO`**`n:k`**`*A`**`x`**

Kurzfassung der Aufgabe:

- Übertragung von Bild 7 und 8, wenn Segmente 12 bis 15 von Bild 7 verloren gehen.
- Bild 7 und Bild 8 haben jeweils eine Größe von 10000 Bytes. MSS ist 500 Bytes
- Gefragt ist auch das cwnd nach Empfang des letzten ACKs zu Bild 8.

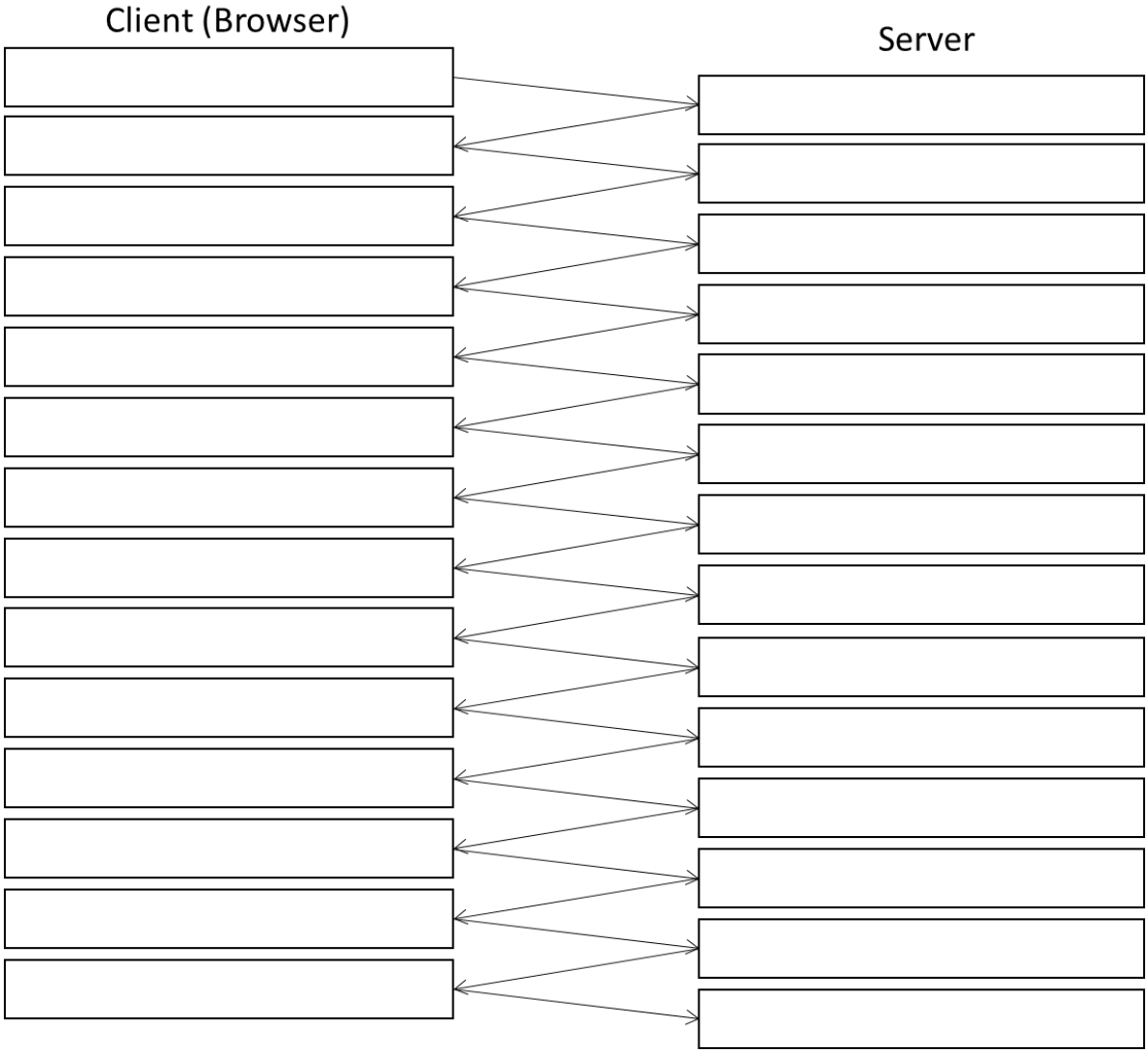

Abbildung 2: Vorlage für Skizze in Aufgabe 2.2)

Name: Matrikelnummer:

Aufgabe 5 [Python: Nachrichten und Segmente] (23 Punkte)

In Abbildung 2 ist der Python-Code „code.py“ gegeben. Zunächst wird der Code mit dem Aufruf „python code.py S 1.1.1.1 3000 Alpha“ gestartet. Eine Sekunde später wird der Code mit dem Befehl „python code.py C 1.1.1.1 3000 Beta“ auf einem anderen Rechner ausgeführt.

1) Erklären Sie die Funktion rl(sock). (3P)

2) Geben Sie die Sequenz der Nachrichten an, die auf der Anwendungsschicht zwischen Client und Server übertragen werden. Führen Sie für die Nachrichten die Bezeichnungen C1,C2,… und S1,S2,… ein und geben Sie die Größe der Nachrichten an. (8P)

3) Skizzieren Sie die Sequenz der ausgetauschten Segmente auf der Transportschicht. **Geben Sie präzise an, welche Segmente bei Empfang eines Segments gesendet werden**. Verwenden Sie für die Segmente die Bezeichnungen, die Sie in der vorigen Teilaufgabe verwendet haben und geben Sie an, welche Bytes der Nachricht im Segment transportiert werden (z.B. C3:700-950). Bei den Segmenten ohne Payload (SYN. ACK, FIN) genügt die Angabe des Typs. Sie können die Vorlage in **Abbildung 1** verwenden oder auch ein freies Format wählen. (12P)

   Gehen Sie von folgenden Parametern aus:

   - Maximum Segment Size: 1500 Bytes
   - Initial Window: 2 Segmente
   - Bei der Verarbeitung auf Anwendungsebene vergeht keine Zeit. Segmente auf Transportebene sollen also inklusive der durch sie ausgelösten Befehle im Programm-Code in der Reihenfolge ihres Eintreffens behandelt werden.

Tipps zur Python-Syntax:

1) Der Befehl `b'Hallo'` wandelt den String ’Hallo’ in ein Byte-Array um.

2) Der Befehl *`bytearray(n)`* erzeugt ein Byte-Array mit n Nullen

Beispiel: *`a=bytearray(2)`* ➔*`a=b'\x00\x00'`*

1) Der Befehl `a=n**b` steht für a=“n hoch b“

2) Der Befehl `int(x)` macht aus x (z.B. String, Float) einen Integer-Wert

   Beispiele: `int('5')`➔`5`, `int(5.7)`➔`5`

3) Der Befehl `float(x)` macht aus x (z.B. String, Integer) eine Gleitkommazahl

   Beispiele: `float('1.5')`➔`1.5`, `float(5)`➔`5.0`

4) Der Befehl *`for item in list:`* entspricht einer FOR-Schleife, in der über die Liste `list` iteriert wird

5) Der Befehl *`break`* unterbricht die Ausführung einer Schleife. Das Programm wird mit dem ersten Befehl nach der Schleife fortgesetzt.

6) Wird ein Python-Skript von der Kommandzeile aufgerufen, so stehen die Kommandozeilenparameter als Strings in der Liste `sys.argv`. Der Name des Skripts steht in `sys.argv[0]`.

   Beispiel: `python code.py S 1.1.1.1 3000 Alpha`

   ➔`sys.argv = ['code.py' 'S' '1.1.1.1' '3000' 'Alpha']`

Name: Matrikelnummer:

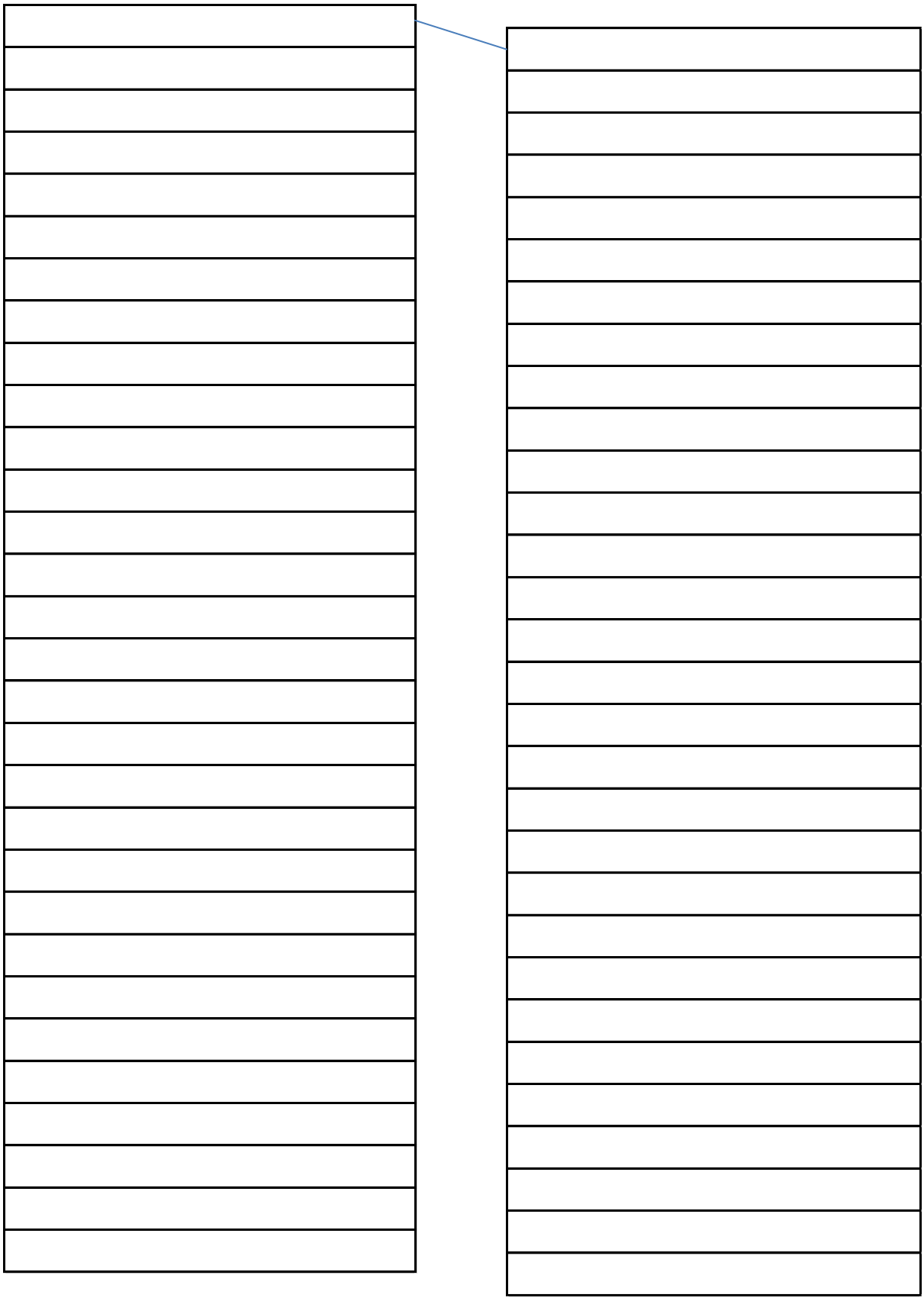

Abbildung 1: Vorlage für Skizze der Paketsequenz
(Die „schräge“ Linie kennzeichnet die erste Übertragung von Client zu Server)

Name: Matrikelnummer:

Abbildung 2: Python Code „code.py“ zu Aufgabe 5

```
import socket
import sys
import time

socket.setdefaulttimeout(10)

def main(client_or_server,adress,name):
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    if client_or_server=='S':
        start_server(sock,adress,name)
    elif client_or_server=='C':
        start_client(sock,adress,name)

def first_contact(sock,name):
    sock.send(b"Hello, I'm "+name.encode())
    try:
        if sock.recv(11)==b"Hello, I'm ":
            return sock.recv(100).decode('utf-8')
        else:
            return ''
    except socket.timeout:
        return ''

def start_client(sock,adress,name):
    try:
        sock.connect(adress)
    except socket.timeout:
        return
    if not first_contact(sock,name):
        sock.close()
        return
    cmdlist=[(b'0.5+',3600),(b'1.0+',2500), (b'2.0+',70)]
    start_task(sock,cmdlist)
    sock.close()

def start_server(sock,adress,name):
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.bind(adress)
    sock.listen(1)
    try:
        conn, addr = sock.accept()
    except socket.timeout:
        return
    if not first_contact(conn,name):
        conn.close()
        return
    start_service(conn)
    conn.close()
    sock.close()
```

Name: Matrikelnummer:

```
def start_task(sock, commands):
    for cmd in commands:
        # bytearray(n) liefert ein Bytearray mit n Nullen
        sock.send(cmd[0]+bytearray(cmd[1]-5)+b'\n')
    s=[]
    while True:
        try:
            s.append(rl(sock)+1)
        except:
            return s
        if not s[-1]:
            return s[:-1]

def start_service(sock):
    while True:
        try:
            x=float(sock.recv(3)) # Bsp: float('1.5') ==> 1.5
            sock.recv(1)
            n=5+rl(sock)
            # bytearray(n) liefert ein Bytearray mit n Nullen
            # a**b heißt a hoch b; Bsp: int(3600**0.5-1)➔ 59
            sock.send(bytearray(int(n**x-1))+b'\n')
        except:
            return

def rl(sock):
    n=0
    c=1
    while c:
        c=sock.recv(1)
        if c==b'\n':
            return n
        n+=1
    return ''

if __name__=='__main__':
    # in sys.argv stehen die Kommandozeilenparameter
    # sys.argv[0] ist der Name des aufgerufenen Python-Skripts
    client_or_server,adress,port,name=sys.argv[1:]
    # int(port) wandelt String port in Integer um
    main(client_or_server,(adress,int(port)),name)
```

Aufgabe 6 [TCP] (15 Punkte)

In den folgenden beiden Teilaufgaben ist jeweils der Zustand einer bereits geöffneten TCP Verbindung gegeben. Weitere Segmente werden übertragen und es können Paketverluste auftreten. Vervollständigen Sie die angegebenen Tabellen, bis alle Segmente und ACKs angekommen sind. Verwenden Sie die folgenden Parameter:

- Initial Window=3 Segmente, IW=3*MSS
- Retransmission Timeout: 10*RTT
- ssthresh wird mit 500 Segmenten initialisiert
- Die RTT wird von der Ausbreitungsverzögerung und nicht der Bottleneck-Bandbreite dominiert. Es kann davon ausgegangen werden, dass alle Segmente eines Sendefensters gleichzeitig aber nacheinander am Empfänger ankommen und auch die daraufhin gesendeten Acknowledgements wieder gleichzeitig aber nacheinander am Sender ankommen.

1) Vervollständigen Sie in Tabelle 1 den Ablauf der Übertragung von 40 Segmenten. Nach den 40 Segmenten werden keine weiteren Segmente versendet. Die Segmente 5 bis 10 gehen verloren. (15P)

Name: Matrikelnummer:

Tabelle 1: Lösungsvorlage TCP

Sender (Segmente: 1-40, verloren: 5-10)									Empfänger		Kommentare (nicht notwendig)
RTT	ACKs An	FlightSize 1	DupACK	ssthresh	cwnd	Segmente Ab	FlightSize 2	CA Segment	Segmente AN	ACKs Ab	
0	x	0	0	10	25						

Aufgabe 7 [Netzwerksegmente] (30 Punkte)

1) In einem Netz befinden sich 4 Netzwerksegmente, in denen 3, 6, 14 und 31 IP-Adressen für Hosts und Router-Interfaces benötigt werden. Ihnen stehen die IP-Adressen 100.100.100.80-100.100.100.191 zur Verfügung. Teilen Sie den Netzwerksegmenten jeweils einen Adressbereich zu und geben Sie die Subnetzmaske an. (3P)

2) Sie wollen die Anzahl benötigter IP-Adressen für drei Netzwerksegmente A, B und C minimieren. Als Vorgabe haben Sie die Anzahl benötigter IP-Adressen und auch, dass einige dieser IP-Adressen bereits fest vergeben sind. Bestimmen Sie für die drei Netzwerksegmente das kleinstmögliche Subnetz. (3P)

Netzwerksegment	Anzahl IP-Adressen	Fest vergebene IP-Adressen
A	4	100.100.100.9-100.100.100.12
B	7	100.100.100.92, 100.100.100.98
C	10	100.100.100.41,100.100.100.47

3) Bestimmen Sie für das Netz in Abbildung 2 alle Netzwerksegmente und geben Sie die Liste der Netzwerk-Interfaces sowie die Anzahl der benötigten IP-Adressen (ohne Netzwerk- und Broadcastadresse) an, die in diesen Netzwerksegmenten benötigt werden. (9P)

   Verwenden Sie die Notation `R/I` für das Interface, das von einem Router `R` zu einem Netzknoten `I` führt. Falls über dieses physikalische Interface mehrere virtuelle Interfaces (VLANs) laufen, spezifizieren Sie das virtuelle Interface als `R/I[V]`, wobei V die VLAN ID ist.

Netzwerksegment	Liste der Interfaces	Anzahl IPs
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		

Name: Matrikelnummer:

4) Stellen Sie für den Router R_2 in Abbildung 2 eine Routing-Tabelle auf. Verwenden Sie wenn möglich die Nummerierung der Netzwerksegmente aus der vorigen Aufgabe als Adressbereiche in der Routing-Tabelle. (7P)

Adressbereich	Interface	Next Hop

5) Betrachten Sie das Netz in Abbildung 1. Die beiden Hosts A und B senden jeweils ein IP-Paket an den Host C. Tragen Sie in Tabelle 1 alle Ethernet-Frames ein, die versendet werden, um das IP Paket von A nach C zu übertragen. Verfahren Sie analog für das IP Paket von B nach C in Tabelle 2. Gehen Sie davon aus, dass alle Ziel-MAC Adressen bereits bekannt sind und in den ARP-Tabellen stehen. (4P)

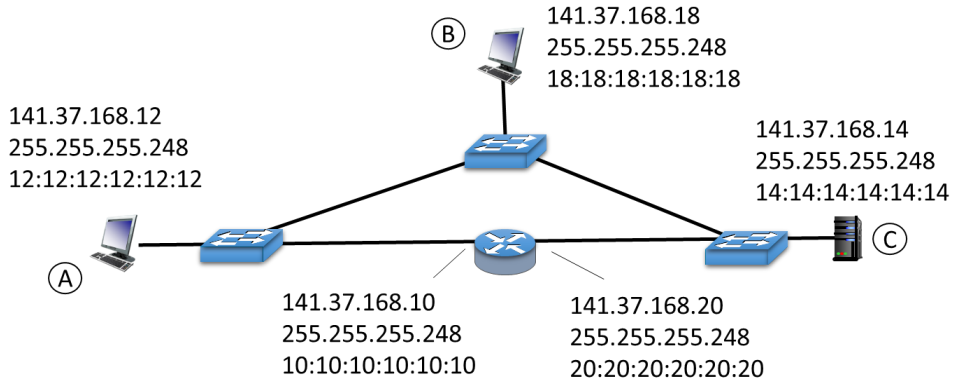

Abbildung 1: Netzwerk zu Aufgabe 8-5)

Tabelle 1: Ethernet Pakete zu IP Paket von A nach C

Src MAC	Dst MAC	Src IP	Dst IP

Tabelle 2: Ethernet Pakete zu IP Paket von B nach C

Src MAC	Dst MAC	Src IP	Dst IP

Name: Matrikelnummer:

6) Ein Router R habe die in der linken Hälfte von Tabelle 3 dargestellte Routing-Tabelle. Bestimmen Sie für die IP-Adressen auf der rechten Seite der Tabelle, auf welcher Route der Router R ein Datagramm mit dieser Ziel-IP-Adresse weiterleiten würde. (4P)
Hinweis: Die Spalte „IP Adressen“ muss nicht ausgefüllt werden.

Tabelle 3: Routing-Tabelle

Route	Adress-Bereich	Subnetzmaske	IP Adressen	Liste der Ziel IP-Adressen	Route
1	0.0.0.0	0.0.0.0		121.17.64.177	
2	121.0.0.0	255.240.0.0		121.17.164.177	
3	121.16.0.0	255.252.0.0		121.20.164.177	
4	121.17.0.0	255.255.192.0		121.223.0.240	
5	121.17.0.0	255.255.128.0			
6	121.17.64.0	255.255.255.128			
7	121.17.64.160	255.255.255.240			
8	121.192.0.0	255.224.0.0			
9	121.223.0.0	255.255.224.0			
10	121.223.16.0	255.255.248.0			

Dezimal	Binär	Dezimal	Binär	Dezimal	Binär
128	1000 0000	1	0000 0001	121	0111 1001
192	1100 0000	16	0001 0000	160	1010 0000
224	1110 0000	17	0001 0001	164	1010 0100
240	1111 0000	20	0001 0100	175	1010 1111
248	1111 1000	32	0010 0000	177	1011 0001
252	1111 1100	34	0010 0010	223	1101 1111
254	1111 1110	63	0011 1111		
255	1111 1111	64	0100 0000		

Name: Matrikelnummer:

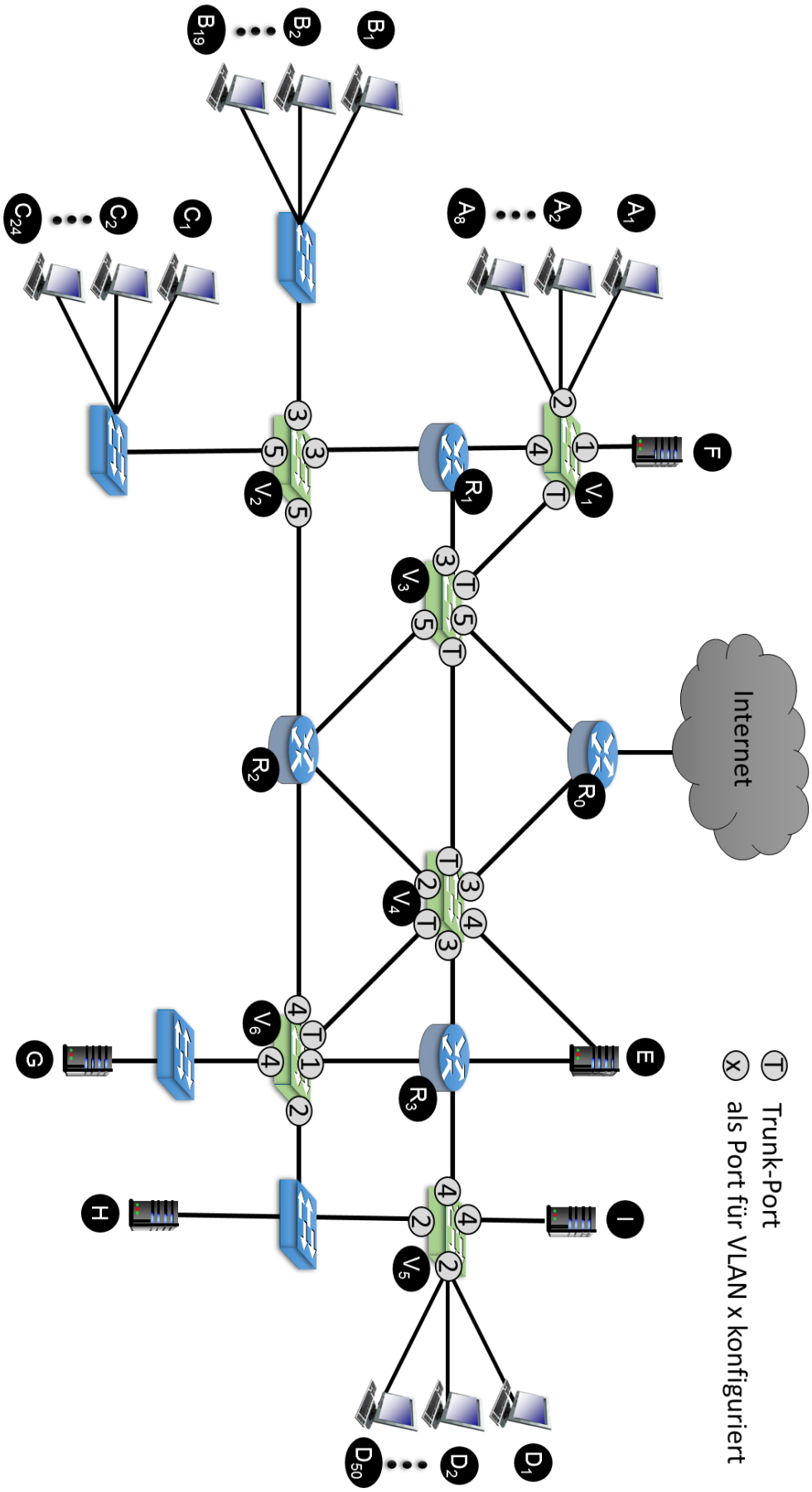

Abbildung 2: Netz mit mehreren Netzwerksegmenten zu Aufgabe 8-3) und 8-4)