

# Grundlagen Rechnernetze und Verteilte Systeme (IN0010)

## Übungsblatt 10

11. Juli – 15. Juli 2022

### Aufgabe 1 Schiebefensterprotokolle

Wir betrachten ein Sliding-Window-Verfahren, dessen Sende- und Empfangsfenster  $w_s = w_r = 2$  beträgt. Der Sequenznummernraum sei  $\mathcal{S} = \{0, 1\}$ . Die Fehlerbehandlung erfolge analog zu Go-Back-N. Abbildung 1.1 zeigt eine Datenübertragung, wobei die Blitze für durch Störungen verlorengegangene Segmente stehen. Die beiden ersten ACKs erreichen also nicht den Sender.

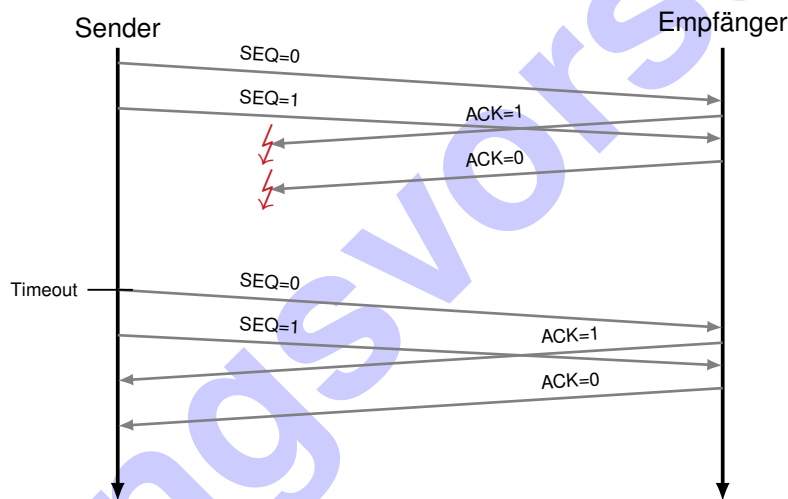


Abbildung 1.1: Modifiziertes Alternating-Bit-Protocol

a)\* Welches Problem tritt in dem Beispiel bei der Übertragung auf?

Der Empfänger leitet nach Erhalt der ersten beiden Segmente diese an die nächsthöhere Schicht weiter. Er weiß nach dem Senden der ersten beiden ACKs nicht, dass diese den Sender nicht erreicht haben. Der Sender wird nach einem Timeout diese ersten beiden Segmente wiederholen. Diese tragen dieselben Sequenznummern wie die ersten beiden Segmente. Leider erwartet der Empfänger aber zwei **neue** Segmente mit eben diesen Sequenznummern. Der Empfänger ist also nicht in der Lage zu unterscheiden, ob es sich um eine Wiederholung oder um zwei neue Segmente handelt. Er wird daher auch diese beiden Segmente an die nächsthöhere Schicht weiterleiten, welche nun Daten **doppelt** erhalten hat.

b) Passen Sie  $\mathcal{S}$  an, so dass das Verfahren korrekt funktionieren kann. Begründen Sie Ihre Antwort.

Da das Wiederholungsverfahren Go-Back-N ist, akzeptiert der Empfänger jeweils nur das nächste erwartete Segment (unabhängig davon, dass sein Empfangsfenster größer ist). In diesem Fall reicht bereits der Sequenznummernraum  $\mathcal{S} = \{0, 1, 2\}$  aus, da auf diese Weise stets ein „Schutzabstand“ von einer Sequenznummer besteht.

Im Folgenden betrachten wir die beiden Verfahren Go-Back-N und Selective Repeat. Die Sequenznummern  $s \in \mathcal{S}$  haben eine Länge von 4 bit. Beantworten Sie die folgenden Fragen **sowohl für Go-Back-N als auch Selective Repeat**.

c)\* Wie viele unbestätigte Segmente darf der Sender jeweils senden, um eine gesicherte Verbindung zu realisieren? Begründen Sie Ihre Antwort anhand von Beispielen. (Hinweis: Denken Sie an in möglichst ungünstigen Momenten verlorene Bestätigungen)

Bezeichne  $w_s$  das Sendefenster. Dann gilt allgemein:

- **Go-Back-N:**

Der Empfänger akzeptiert bei Go-Back-N grundsätzlich immer nur das nächste erwartete Segment. Segmente, die *out-of-order* ankommen, werden ignoriert. Der ungünstigste Fall für Go-Back-N ist der, dass alle Segmente erfolgreich übertragen werden, dann aber alle Bestätigungen auf dem Weg zum Sender verloren gehen. Dieser Fall ist in Abbildung 1.1 dargestellt. Abhilfe kann geschaffen werden, indem stets ein Segment weniger gesendet wird als insgesamt Sequenznummern zur Verfügung stehen. Aus diesem Grund muss für das Sendefenster bei Go-Back-N stets  $w_s \leq |\mathcal{S}| - 1$  gelten.

- **Selective Repeat:**

Der ungünstigste Fall für Selective Repeat besteht darin, dass  $w_s$  Segmente erfolgreich übertragen werden, anschließend aber alle Bestätigungen verloren gehen. Sei  $x$  die Sequenznummer des ersten Segments. In diesem Fall wird der Empfänger – da er ja alle Segmente erfolgreich erhalten hat – sein Empfangsfenster um  $w_s$  weiterschieben. Der Sender denkt aber, dass alle Segmente verlorengegangen sind. Er wird aus diesem Grund die Segmente mit den Sequenznummern  $x, x + 1, \dots, x + w_s - 1$  wiederholen. Die Bedingung ist nun, dass keine der Sequenznummern der **wiederholten** Segmente in das aktuelle Empfangsfenster des Empfängers fallen darf. Andernfalls würde der Empfänger eine Wiederholung als neues Segment akzeptieren.

Für  $w_s = 4$  und  $|\mathcal{S}| = 8$  sieht man nun an einem Beispiel schnell, dass alles funktioniert. Wählt man hingegen  $w_s = 5$ , tritt ein Konflikt auf.

Für  $w_s = 3$  und  $|\mathcal{S}| = 7$  ist ebenfalls alles in Ordnung. Wählt man hier jedoch  $w_s = 4$ , so tritt wieder ein Konflikt auf.

Allgemein gilt also für das Sendefenster  $w_s \leq \left\lfloor \frac{|\mathcal{S}|}{2} \right\rfloor$ .

**Hinweis:**

Nimmt man an, dass es keine kumulativen Bestätigungen gibt, so gibt es einen weiteren Fall, der zum selben Ergebnis führt: Es geht die Bestätigung des ersten Segments verloren, die übrigen erreichen jedoch den Sender. Beispiel:  $|\mathcal{S}| = 7$ ,  $w_s = 4$ . Der Sender sendet Segmente mit den Sequenznummern 0, 1, 2, 3. Der Empfänger erwartet also als nächstes die Sequenznummern 4, 5, 6, 0. Der Sender wiederholt das Segment mit Sequenznummer 0, was vom Empfänger aber fälschlicherweise als neues Segment interpretiert wird.

d)\* Begründen Sie, welche oberen und unteren Grenzen für das Empfangsfenster des Empfängers bei den beiden Verfahren jeweils sinnvoll sind.

Bei Go-Back-N reicht prinzipiell ein Empfangsfenster der Größe  $w_r = 1$ , da stets nur das nächste erwartete Segment akzeptiert wird.

Bei Selective Repeat hingegen muss das Empfangsfenster mind. so groß wie das Sendefenster sein und darf natürlich nicht größer als etwa die Hälfte des Sequenznummernraums sein, also  $w_s \leq w_r \leq \lfloor \frac{|S|}{2} \rfloor$ . Andernfalls verwirft der Empfänger u. U. Segmente, die nicht in der richtigen Reihenfolge eintreffen und infolge der zu geringen Größe des Empfangsfensters nicht in selbiges hineinfließen.

e)\* Für eine praktische Implementierung benötigt der Empfänger einen Empfangspuffer. Wie groß sollte dieser bei den beiden Verfahren jeweils gewählt werden?

Unabhängig vom Verfahren sollte der Empfangspuffer stets die Größe des maximal erlaubten Sendefensters sein.

Bei Selective Repeat leuchtet das ein, da hier Segmente tatsächlich auf der Transportschicht zwischengespeichert werden müssen, bis fehlende Segmente eingetroffen sind.

Bei Go-Back-N hingegen könnte man argumentieren, dass Segmente ohnehin in der richtigen Reihenfolge eintreffen müssen und diese daher auch sofort an höhere Schichten weitergeleitet werden können. Dies trifft nur bedingt zu, denn die Verarbeitungsgeschwindigkeit des Empfängers könnte nicht ausreichen, um eintreffende Segmente schnell genug weiterzuleiten.

Unabhängig von der (etwas philosophischen) Frage, wie groß Puffer bei konkreten Implementierung zu wählen sind, sollten Sie sich den Unterschied zwischen dem Empfangsfenster und einem etwaigen Empfangspuffer klarmachen. Beispielsweise könnte die Größe des Empfangsfensters stets dem noch freien Speicher im Empfangspuffer entsprechen während das Sendefenster durch das Empfangsfenster nach oben beschränkt wird.

## Aufgabe 2 Fluss- und Staukontrolle bei TCP

Das im Internet am weitesten verbreitete Transportprotokoll ist TCP. Dieses implementiert Mechanismen zur Fluss- und Staukontrolle.

a)\* Diskutieren Sie die Unterschiede zwischen Fluss- und Staukontrolle. Welche Ziele werden mit dem jeweiligen Mechanismus verfolgt?

- **Flusskontrolle:**  
Verhinderung von Überlastsituation beim Empfänger
- **Staukontrolle:**  
Reaktion von Überlastsituation im Netz

b) Ordnen Sie die folgenden Begriffe jeweils der TCP-Fluss- bzw. Staukontrolle zu:

- Slow-Start
- Empfangsfenster
- Congestion-Avoidance
- Multiplicative-Decrease

Zur Flusskontrolle gehört lediglich das Empfangsfenster, da über dieses der Empfänger dem Sender mitteilen kann, wie viel Daten er maximal auf einmal senden darf.

Die übrigen Begriffe gehören alle zur Staukontrolle, wobei Slow-Start und Congestion-Avoidance die beiden Staukontrollphasen einer TCP-Verbindung sind. Als Multiplicative-Decrease hingegen bezeichnet man das Halbieren des Staukontrollfensters bei Verlust eines Segments.

Zur Analyse der mit TCP erzielbaren Datenrate betrachten wir den Verlauf einer zusammenhängenden Datenübertragung, bei der die Slow-Start-Phase bereits abgeschlossen ist. TCP befindet sich also in der Congestion-Avoidance-Phase. Wir bezeichnen die einzelnen Fenster wie folgt:

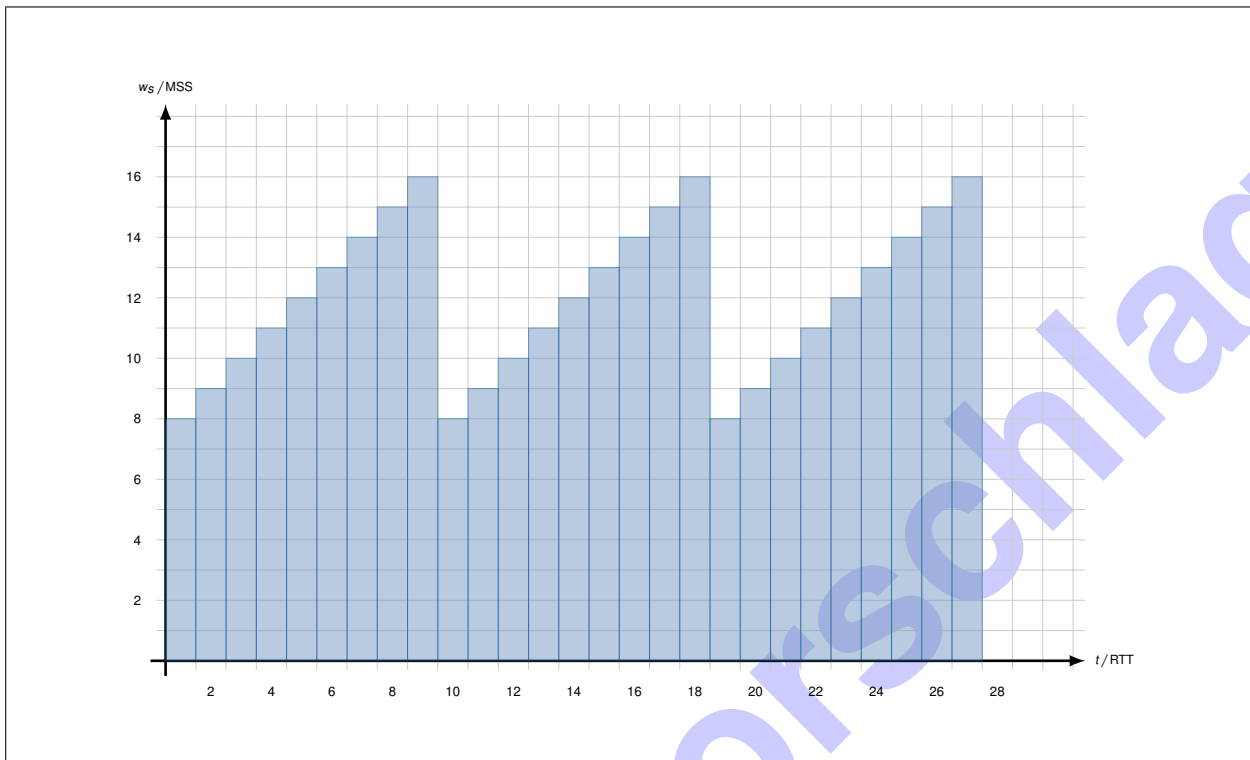
- Sendefenster  $W_s$ ,  $|W_s| = w_s$
- Empfangsfenster  $W_r$ ,  $|W_r| = w_r$
- Staukontrollfenster  $W_c$ ,  $|W_c| = w_c$

Wir gehen davon aus, dass das Empfangsfenster beliebig groß ist, so dass das Sendefenster allein durch das Staukontrollfenster bestimmt wird, d. h.  $W_s = W_c$ . Es treten keinerlei Verluste auf, solange das Sendefenster kleiner als ein Maximalwert  $x$  ist, also  $w_s < x$ .

Wird ein vollständiges Sendefenster bestätigt, so vergrößert sich das aktuell genutzte Fenster um genau 1 MSS. Hat das Sendefenster den Wert  $x$  erreicht, so geht genau eines der versendeten TCP-Segmente verloren. Den Verlust erkennt der Sender durch mehrfachen Erhalt derselben ACK-Nummer. Daraufhin halbiert der Sender das Staukontrollfenster, bleibt aber nach wie vor in der Congestion-Avoidance-Phase, d. h. es findet kein erneuter Slow-Start statt. Diese Vorgehensweise entspricht einer vereinfachten Variante von TCP-Reno (vgl. Vorlesung).

Als konkrete Zahlenwerte nehmen wir an, dass die maximale TCP-Segmentgröße (MSS) 1460 B und die RTT 200 ms beträgt. Die Serialisierungszeit von Segmenten sei gegenüber der Ausbreitungsverzögerung vernachlässigbar klein. Segmentverlust trete ab einer Sendefenstergröße von  $w_s \geq x = 16 \text{ MSS}$  auf.

c)\* Erstellen Sie ein Schaubild, in dem die aktuelle Größe des Sendefenster  $w_s$  gemessen in MSS über der Zeitachse  $t$  gemessen in RTT aufgetragen ist. In Ihrem Diagramm soll zum Zeitpunkt  $t_0 = 0$  s gerade die Sendefenstergröße halbiert worden sein, also  $w_s = x/2$  gelten. Zeichnen Sie das Diagramm im Zeitintervall  $t = \{0, \dots, 27\}$ .



d)\* Wieviel Zeit vergeht, bis nach einem Segmentverlust das Staukontrollfenster infolge eines weiteren Segmentverlusts wieder reduziert wird?

Nach einem Segmentverlust wird  $w_c$  auf  $x/2$  reduziert und anschließend pro vollständig bestätigtem Fenster wieder um 1 MSS vergrößert. Da die Serialisierungszeit vernachlässigbar klein ist, können also zu einem Zeitpunkt  $t_0$  insgesamt  $w_c$  Segmente gesendet werden, welche zum Zeitpunkt  $t_0 + RTT$  bestätigt werden. Folglich erhalten wir für die Zeit bis zum erneuten Erreichen des Maximalwerts

$$T = \left(\frac{x}{2} + 1\right) \cdot RTT = 9 \cdot 200 \text{ ms} = 1.8 \text{ s.}$$

e)\* Bestimmen Sie allgemein die durchschnittliche Verlustrate  $\theta$ . Hinweis: Da das Verhalten von TCP in diesem idealisierten Modell periodisch ist, reicht es aus, lediglich eine Periode zu betrachten. Setzen Sie die Gesamtzahl übertragener Segmente in Relation zur Anzahl verlorener Segmente (Angabe als gekürzter Bruch ist ausreichend).

Zunächst bestimmen wir die Anzahl  $n$  an Segmenten, welche während eines „Sägezahns“ übertragen werden:

$$\begin{aligned} n &= \sum_{i=x/2}^x i = \sum_{i=1}^x i - \sum_{i=1}^{x/2-1} i = \frac{x \cdot (x+1)}{2} - \frac{(\frac{x}{2}-1) \cdot \frac{x}{2}}{2} \\ &= \frac{x^2 + x}{2} - \frac{x^2}{8} + \frac{x}{4} \\ &= \frac{3}{8}x^2 + \frac{3}{4}x \\ &\stackrel{x=16}{=} 108 \end{aligned}$$

Pro „Sägezahn“ geht genau ein Segment verloren. Wir erhalten also für die Verlustrate

$$\theta = \frac{1}{\frac{3}{8}x^2 + \frac{3}{4}x} = \frac{1}{108} \approx 9.30 \cdot 10^{-3}$$

f) Bestimmen Sie mit Hilfe der Ergebnisse aus den Teilaufgaben (c) und (e) die in der betrachteten TCP-Übertragungsphase durchschnittlich erzielbare Übertragungsrate in kB/s.

**Hinweis:** Verwenden Sie den exakten Wert (Bruch) aus Teilaufgabe e).

Für die Datenrate erhalten wir

$$\begin{aligned} r_{TCP} &= \frac{n \cdot MSS}{T} \cdot (1 - \theta) \\ &= \frac{108 \cdot 1460 \text{ B}}{1.8 \text{ s}} \cdot \frac{107}{108} \\ &= \frac{107 \cdot 1460 \text{ B}}{1.8 \text{ s}} \\ &= \frac{1562200}{18} \text{ B/s} \\ &\approx \frac{1562}{18} \text{ kB/s} \approx 86.79 \text{ kB/s.} \end{aligned}$$

g)\* Bis zu welcher Übertragungsrate könnten Sie mit UDP maximal über den Kanal senden, ohne einen Stau zu erzeugen? Berücksichtigen Sie, dass der UDP-Header 12 B kleiner als der TCP-Header ohne Optionen ist.

Offenbar lassen sich 15 MSS verlässlich übertragen. Zusätzlich trägt ein Segment bei UDP 12 B mehr Nutzdaten als bei TCP. Wir erhalten also

$$\begin{aligned} r_{UDP} &= \frac{15 \cdot (MSS + 12 \text{ B})}{RTT} \\ &= \frac{15 \cdot (1460 \text{ B} + 12 \text{ B})}{0.2 \text{ s}} \\ &= \frac{15 \cdot 1472 \text{ B}}{0.2 \text{ s}} \\ &\approx 110.40 \text{ kB/s.} \end{aligned}$$

### Aufgabe 3 Network Address Translation

In dieser Aufgabe soll die Weiterleitung von IP-Paketen (IPv4) bei Verwendung eines NAT-fähigen Routers betrachtet werden. Für die Zuordnung zwischen öffentlichen und privaten IP-Adressen verfügt ein NAT-fähiger Router über eine Abbildungstabelle, die die Beziehung zwischen lokalem und globalem Port speichert. Viele NAT-fähige Geräte speichern zusätzlich noch weitere Informationen wie die entfernte IP-Adresse oder die eigene globale IP-Adresse (z. B. wenn der Router mehr als eine globale IP besitzt). Davon wollen wir hier absehen.

Abbildung 3.1 zeigt die Netztopologie. Router R1 habe NAT aktiviert, wobei auf IF1 eine private und auf IF2 eine öffentliche IP-Adresse verwendet werde. Router R2 nutze kein NAT. PC2 habe bereits mit Server 2 kommuniziert, wodurch der Eintrag in der NAT-Tabelle von R1 entstanden ist (siehe Abbildung 3.1). Wählen Sie dort, wo Sie die Freiheit haben, sinnvolle Werte für die IP-Adressen und Portnummern.

a)\* Geben Sie PC1 und Interface 1 von R1 eine passende IP-Adresse. Das Subnetz ist 10.0.0.0/24.

Möglich sind zum Beispiel:

- PC1: 10.0.0.1
- R1 IF1: 10.0.0.254

b)\* PC1 baue nun eine HTTP-Verbindung zu Server 2 auf. Geben Sie die Felder für die Quell-IP, Ziel-IP, Quell-Port, Ziel-Port und TTL des IP- bzw. TCP-Headers für die Pakete an den drei markierten Stellen in Abbildung 3.1 an. Geben Sie außerdem neu entstehende Einträge in der NAT-Tabelle von R1 an.

Siehe Abbildung 3.1.

- **Zwischen PC1 und R1:** TTL = 64  
Wichtig ist beim Quell-Port, dass dieser größer als 1023 ist (da Nummern kleiner 1024 Well-Known-Ports repräsentieren und nicht als Quell-Ports verwendet werden). Außerdem sollte er nicht größer sein als 65535, da Portnummern 16 bit lang sind. Der Zielport ist mit TCP 80 (HTTP) vorgegeben.
- **R1 und R2** TTL = 63  
R1 tauscht die private Quell-IP durch seine eigene öffentliche IP-Adresse aus. Der Quell-Port wird (wenn nicht schon anderweitig belegt) für gewöhnlich beibehalten. Andernfalls wird auch dieser geändert, z. B. inkrementiert. Die genaue Wahl der Portnummer hängt vom jeweiligen NAT-Typ ab. Wir behalten die Portnummern sofern möglich einfach bei. An dieser Stelle wird auch ein neuer Eintrag in der NAT-Tabelle erzeugt: [10.0.0.1, 3627, 3627].
- **Zwischen R2 und Server 2** TTL = 62  
Keine Änderung, da ein gewöhnlicher Router IP-Adressen und Portnummern nicht verändert. Die TTL wird aber natürlich dekrementiert.

c) Server 2 antworte nun PC1. Geben Sie in Abbildung 3.2 analog zur vorherigen Teilaufgabe die Header-Felder an den drei benannten Stellen sowie neu entstehende Einträge in der NAT-Tabelle von R1 an.

Wir nehmen an, dass der Server Pakete mit TTL = 64 versendet.

- **Zwischen Server 2 und R2** TTL = 64  
Der Server adressiert die Antwort zunächst an R1 (wohin auch sonst?).
- **Zwischen R2 und R1** TTL = 63  
R2 ändert (außer der TTL) nichts.
- **Zwischen R1 und PC1:** TTL = 62  
R1 nutzt den Eintrag in der seiner NAT-Tabelle um die private IP-Adresse des tatsächlichen Empfängers zu ermitteln. Anschließend werden Ziel-IP und Ziel-Port (wenn nötig) ausgetauscht und das Paket weitergeleitet.

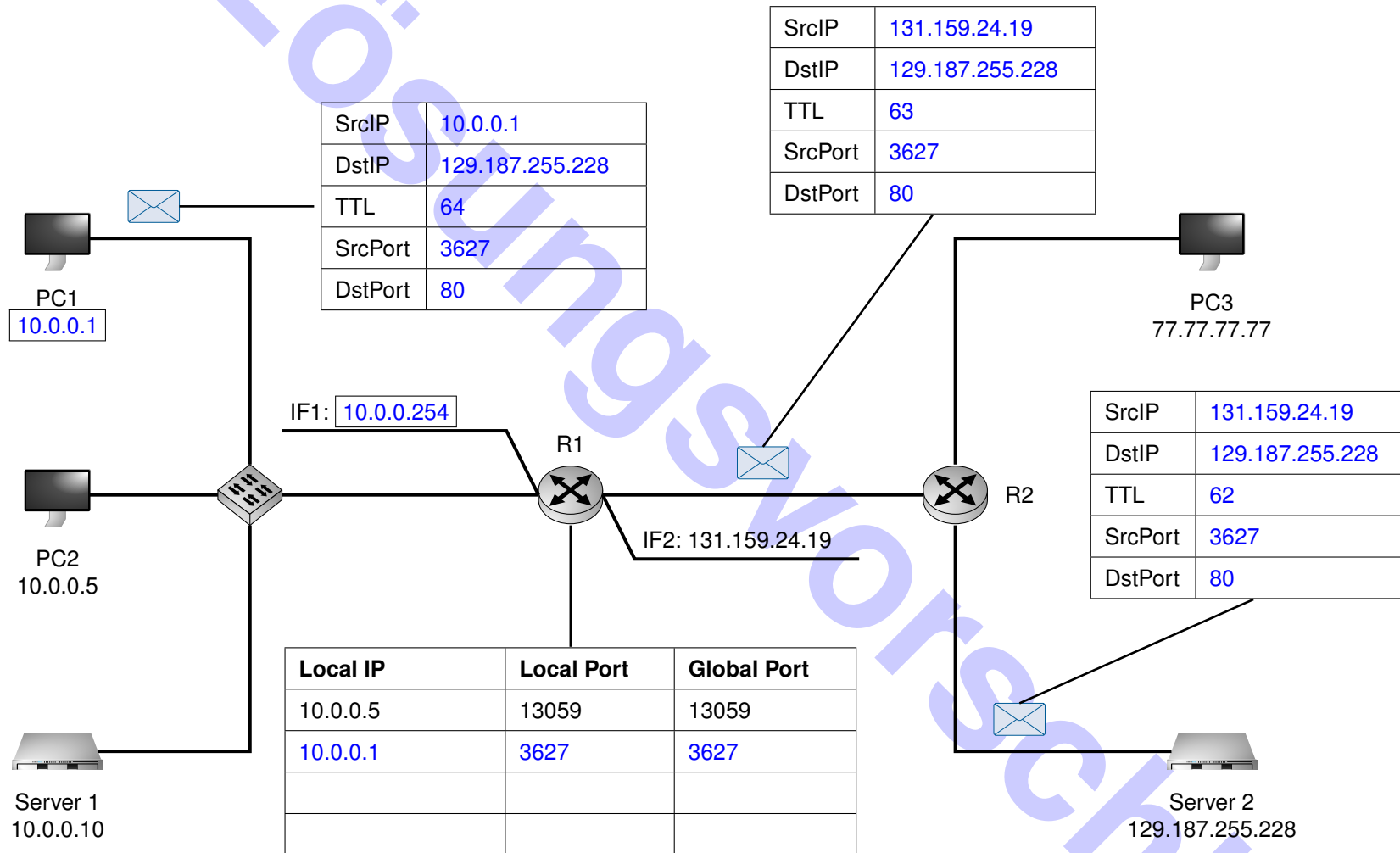


Abbildung 3.1: Lösungsblatt für Aufgabe 3a)/b)



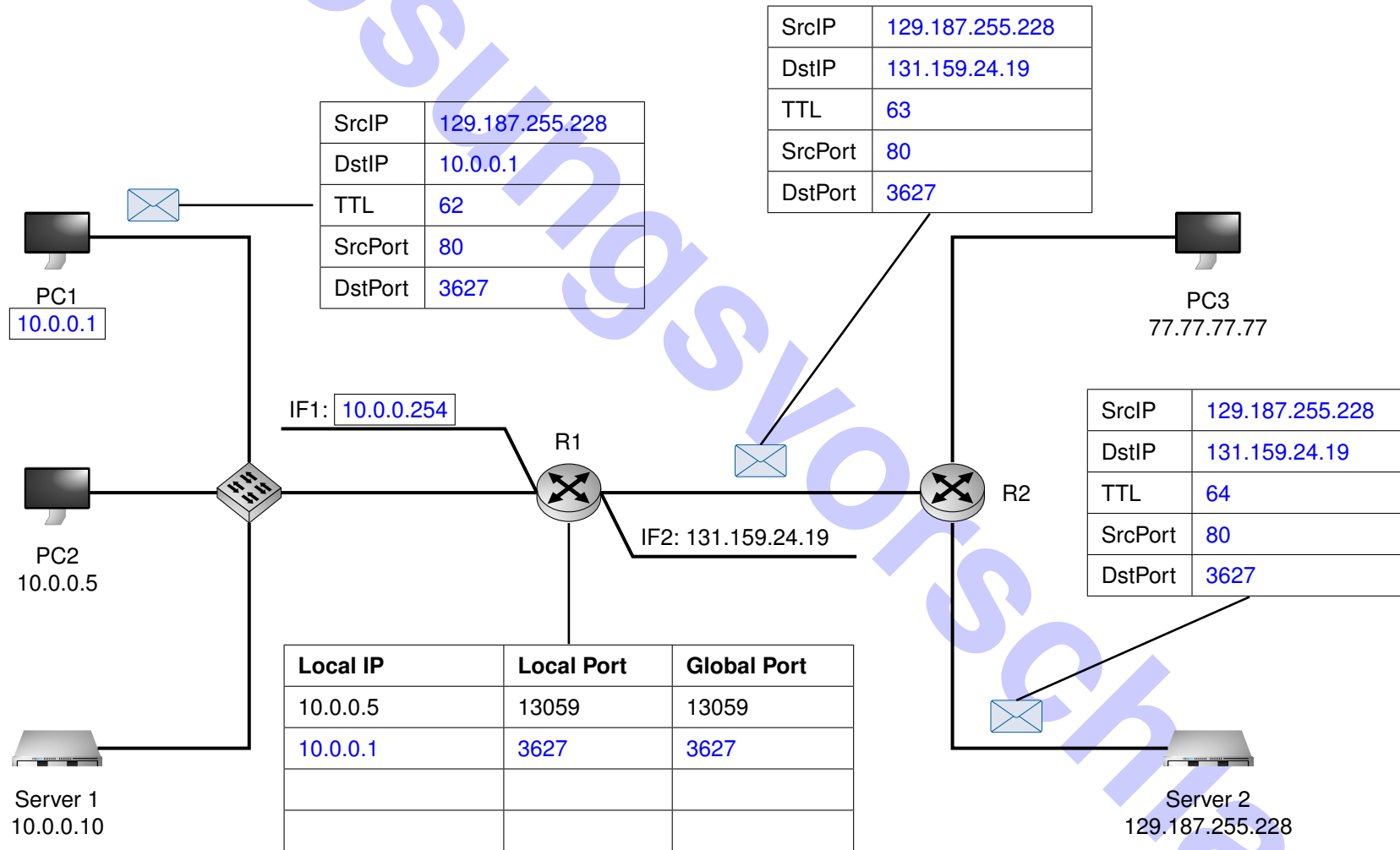


Abbildung 3.2: Lösungsblatt für Aufgabe 3c)

d)\* Server 1 baut nun ebenfalls eine TCP-Verbindung zu Server 2 auf Port 80 auf. Dabei wählt er zufällig den Absender-Port 13059. Beschreiben Sie das am NAT auftretende Problem und wie dieses gelöst wird.

Es gibt eine Kollision mit dem ersten Eintrag in der NAT-Tabelle: Der NAT-Router kann bei Antworten von Server 2 nicht mehr unterscheiden, ob diese für PC1 oder Server 2 bestimmt sind, da als einziges Unterscheidungsmerkmal die globale Portnummer existiert.  
Die Lösung besteht darin, dass der NAT-Router vor der Erzeugung neuer Einträge prüft, ob der jeweilige Port bereits in Verwendung ist. Ist dies der Fall, wählt der NAT-Server eine zufällige Portnummer aus dem Bereich der Ephemeral Ports (oder inkrementiert die Portnummer) und speichert sowohl die lokale als auch die neue globale Portnummer ab. Bei eingehenden Paketen wird in den L4-PDUs die Portnummer zurückübersetzt.

e)\* R1 erhält von PC3 ein an 131.159.24.19:13059 adressiertes Paket. Wie wird R1 mit diesem Paket verfahren? Welche Probleme können sich daraus ergeben?

R1 wird die Zieladresse des Pakets gemäß der NAT Tabelle übersetzen und an PC2 weiterleiten, obwohl der ursprüngliche Eintrag für Server2 angelegt wurde. PC2 erhält ein „unerwartetes“ Packet und muss damit umgehen können. Die fälschlicherweise oft angenommene Firewallfunktion des NAT kann hierbei nicht ermöglicht werden.

f) Ergibt sich für PC2 ein Problem, wenn dieser ein „zufälliges“ Paket mit TCP-Payload auf einem Port mit einer bestehenden Verbindung erhält?

Das Paket besitzt wahrscheinlich eine andere Absender-IP und einen anderen Source Port und wird somit nicht der bestehenden Verbindung zugeordnet. Wenn Absender-IP und Source Port „zufällig“ übereinstimmen, so fällt die Sequenznummer des Pakets (mit hoher Wahrscheinlichkeit) nicht in das aktuell gültige Empfangsfenster und wird somit verworfen.

g)\* Welche weiteren Unterscheidungskriterien könnten von einem NAT-Router verwendet werden?

Globale IP (wenn mehrere Interfaces/IP Adressen am Router konfiguriert sind), remote IP, remote Portsowie die Protokollnummer(TCP oder UDP).

h)\* Welches Problem tritt auf, wenn PC1 einen Echo Request an Server 2 sendet?

Da ICMP keine Portnummern verwendet, kann der NAT-Router keinen Eintrag erzeugen. Die Antwort wird daher verworfen.

i) Beschreiben Sie eine mögliche Lösung für das in der vorherigen Teilaufgabe aufgetretene Problem.

Der NAT-Router könnte im Falle von ICMP Paketen zusätzlich zur Protokollnummer den ICMP-Identifizier als Ersatz für die fehlenden Portnummern verwenden. In diesem Fall muss der NAT-Router aber in jedem Fall auch zwischen den IP-Protokollen (TCP, UDP, ICMP usw.) unterscheiden.

j) Welches Problem ergibt sich, wenn ein NAT-Router ICMP TTL-Exceeded Nachrichten empfängt und an den Empfänger (Absender des auslösenden Pakets) weiterleiten möchte? Wie kann dieses Problem umgangen werden?

TTL-Exceeded Nachrichten sind eigene ICMP Nachrichten, deren Identifizier nicht im NAT eingetragen wurde (Nachrichten werden nicht im eigenen Netzwerk generiert, sondern von Rechnern außerhalb). Eine Zuordnung zum Empfänger ist somit nicht möglich. ICMP TTL Exceeded enthalten neben dem ICMP Header auch den IP Header und die ersten 8 Payload Bytes des auslösenden Pakets<sup>1</sup>. Darüber kann das NAT die auslösende Verbindung identifizieren. Bei TCP und UDP sind hier die Portnummern zu finden, bei ICMP Nachrichten der ursprüngliche Identifizier.

k)\* Nun möchte PC3 eine HTTP-Verbindung zu Server 1 aufbauen. Kann dies unter den gegebenen Umständen funktionieren? (Begründung!)

PC3 kann das Paket nicht direkt an die Adresse 10.0.0.10 adressieren, da es sich hierbei um eine private IP-Adresse handelt, welche im Internet nicht geroutet wird. Wenn PC3 die öffentliche IP von R1 kennt, hinter dem sich Server 1 befindet, so kann er das Paket zwar an die IP-Adresse von R1 und TCP80 adressieren. R1 hat jedoch (soweit aus der Aufgabenstellung hervorgeht) keinen passenden Eintrag in der NAT-Tabelle und kann daher den Empfänger des Pakets nicht ermitteln.

l) Wie könnte das Problem unter Beibehaltung des NATs umgangen werden?

Im NAT kann eine statische Weiterleitung, ein sogenanntes Portforwarding, eingetragen werden. Beispiel: 10.0.0.10 80 80. Darüber kann Server 1 auf der IP-Adresse von R1 über den Router R1 von außen auf Port 80 erreicht werden.

## Aufgabe 4 TCP und Long Fat Networks (Hausaufgabe)

In dieser Aufgabe betrachten wir sog. *Long Fat Networks*. Darunter versteht man Verbindungen, welche zwar eine hohe Übertragungsrate aber insbesondere auch eine hohe Verzögerung aufweisen. Beispiele dafür sind u. a. Satellitenverbindungen in Folge der hohen Ausbreitungsverzögerungen. Wir wollen insbesondere die Auswirkungen auf die TCP-Staukontrolle untersuchen.

a)\* Bei TCP wird das Sendefenster in Abhängigkeit des Empfangsfensters sowie des Staukontrollfensters gewählt. Wie lautet der genaue Zusammenhang?

$$w_s = \min(w_r, w_c)$$

Zwei Nutzer seien nun über einen geostationären Satelliten an das Internet mit hoher Übertragungsrate angebunden. Die RTT zwischen beiden Nutzern betrage 800 ms, die Übertragungsrate sei  $r = 24 \text{ Mbit/s}$ .

b)\* Wie groß muss das Sendefenster (gemessen in Byte) gewählt werden, damit kontinuierlich gesendet werden kann?

Das erste ACK kann frühestens nach einer RTT eintreffen, sofern man die Serialisierungszeiten vernachlässigt. Es ergibt sich also für das Sendefenster

$$w_s \geq \text{RTT} \cdot r = 800 \cdot 10^{-3} \text{ s} \cdot 24 \cdot 10^6 \frac{\text{bit}}{\text{s}} = 100 \cdot 24 \cdot 10^3 \text{ B/s} = 2.4 \text{ MB.}$$

c)\* Warum ist die Situation in Teilaufgabe b) ein Problem für die TCP-Flusskontrolle?

Da das Sendefenster als Minimum aus Empfangs- und Staukontrollfenster gewählt wird und der Empfänger dem Sender sein Empfangsfenster über das Receive-Window-Feld mitteilt, welches auf 16 bit beschränkt ist, ist auch das Sendefenster auf einen Maximalwert von  $(2^{16} - 1) \text{ B} = 65535 \text{ B}$  beschränkt. Wir bräuchten laut Teilaufgabe b) allerdings ein Sendefenster der Größe  $2,4 \cdot 10^6 \text{ B}$ .

d)\* Lesen Sie Sektion 2 von RFC 1323 (<http://www.ietf.org/rfc/rfc1323.txt>, siehe Anhang). Beschreiben Sie die Lösung für das Problem aus Teilaufgabe c).

Wir benötigen die Option *TCP-Window-Scaling*, welche dafür sorgt, dass das Receive-Window mit  $2^x$  skaliert wird. Das Feld „shift.cnt“ der TCP-Window-Scaling-Option gibt den Exponenten  $x$  an.

e) Bestimmen Sie den minimalen Wert für das `shift_cnt`-Feld der TCP-Window-Scaling-Option.

$$(2^{16} - 1) \cdot 2^x \geq 2,4 \cdot 10^6$$
$$x \geq \text{ld} \left( \frac{2,4 \cdot 10^6}{2^{16} - 1} \right)$$
$$= \text{ld}(36,62) \approx 5,19 \Rightarrow x = 6$$

**Erklärung:** Wir suchen den kleinsten Exponenten  $x$ , so dass das maximale Receive-Window größer als der in Teilaufgabe b) berechnete Wert von  $2,4 \cdot 10^6$  B ist. Das Receive-Window ist 16 bit breit, kann also maximal den Wert  $0xffff = 2^{16} - 1$  annehmen. Dieser Wert muss also nun mit  $2^x$  skaliert werden.

Ein kurzer Blick auf die Größe des Sequenznummernraums von TCP ( $|S| = 2^{32}$ , da SEQ- und ACK-Nummern 32 bit lange Felder sind) zeigt, dass wir kein Problem wie in der Aufgabe "Schiebefensterprotokolle" bekommen.

f) Geben Sie den Header des ersten TCP-SYN-Pakets an, welches die Verbindung aufbaut. Verwenden Sie dazu die konkreten Zahlenwerte aus der Angabe. Ein TCP-Header ist zur Erinnerung nochmals in Abbildung 4.1 dargestellt. Dort finden sich auch zwei Vordrucke zur Lösung.

**Hinweis:** Es ist nicht notwendig, den Header binär auszufüllen. Machen Sie aber bitte deutlich, ob es sich um hexadezimale, dezimale oder binäre Darstellung der Zahlen handelt.

Angenommen die Größe des Staukontrollfensters betrage derzeit die Hälfte des in Teilaufgabe b) berechneten Werts. Die MSS betrage 1200 B und die TCP-Verbindung befinde sich derzeit in der Congestion-Avoidance-Phase.

g) Wie lange dauert es, bis das Fenster die Leitung komplett ausnutzen kann?

**Hinweis:** Das Staukontrollfenster wird durch TCP-Window-Scaling nicht beeinflusst.

Das Fenster wird pro RTT um 1 MSS vergrößert. Folglich werden

$$\frac{1,2 \cdot 10^6 \text{ B}}{1200 \text{ B}} \cdot 0,8 \text{ s} = \frac{96 \cdot 10^2}{12} \text{ s} = 800 \text{ s}$$

benötigt.

h) Ergibt sich aus dem Ergebnis von Teilaufgabe g) ein Problem?

Ja. Es dauert mehr als 10 min bis TCP das Receive-Window wieder vollständig ausnutzt – viel zu lange.

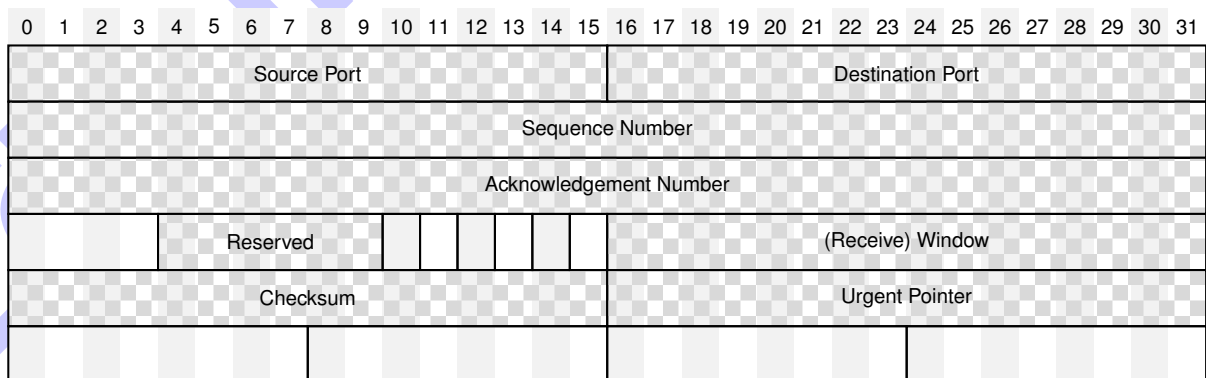
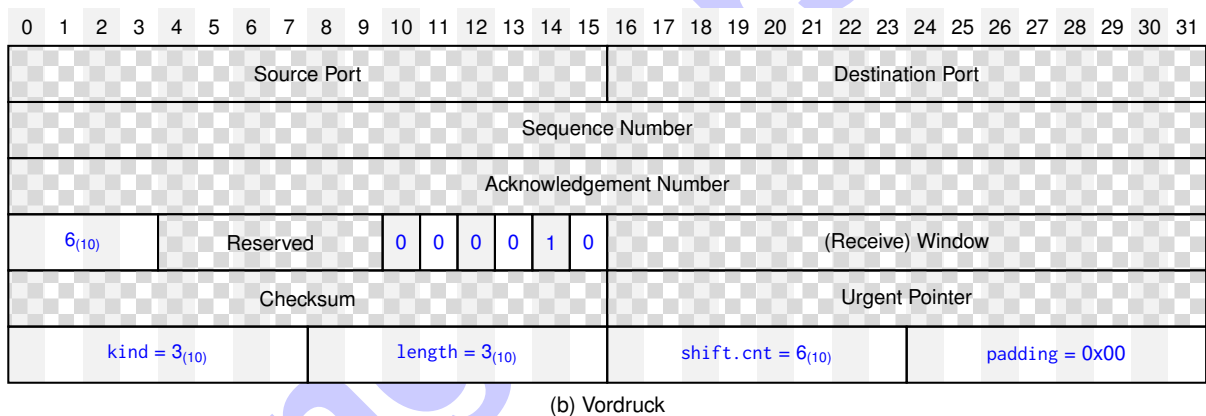
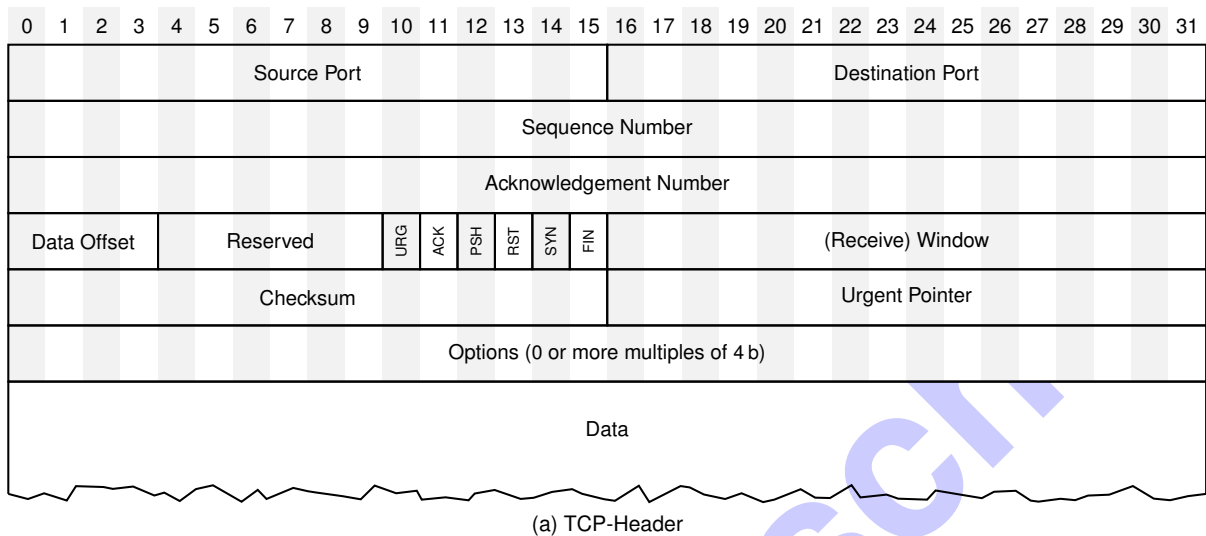


Abbildung 4.1: TCP-Header und Vordrucke zur Lösung von Aufgabe 4

## 2. TCP WINDOW SCALE OPTION

### 2.1 Introduction

The window scale extension expands the definition of the TCP window to 32 bits and then uses a scale factor to carry this 32-bit value in the 16-bit Window field of the TCP header (SEG.WND in RFC-793). The scale factor is carried in a new TCP option, Window Scale. This option is sent only in a SYN segment (a segment with the SYN bit on), hence the window scale is fixed in each direction when a connection is opened. (Another design choice would be to specify the window scale in every TCP segment. It would be incorrect to send a window scale option only when the scale factor changed, since a TCP option in an acknowledgement segment will not be delivered reliably (unless the ACK happens to be piggy-backed on data in the other direction). Fixing the scale when the connection is opened has the advantage of lower overhead but the disadvantage that the scale factor cannot be changed during the connection.)

The maximum receive window, and therefore the scale factor, is determined by the maximum receive buffer space. In a typical modern implementation, this maximum buffer space is set by default but can be overridden by a user program before a TCP connection is opened. This determines the scale factor, and therefore no new user interface is needed for window scaling.

### 2.2 Window Scale Option

The three-byte Window Scale option may be sent in a SYN segment by a TCP. It has two purposes: (1) indicate that the TCP is prepared to do both send and receive window scaling, and (2) communicate a scale factor to be applied to its receive window. Thus, a TCP that is prepared to scale windows should send the option, even if its own scale factor is 1. The scale factor is limited to a power of two and encoded logarithmically, so it may be implemented by binary shift operations.

TCP Window Scale Option (WSopt):

Kind: 3 Length: 3 bytes

```
+-----+-----+-----+
| Kind=3 |Length=3 |shift.cnt|
+-----+-----+-----+
```

This option is an offer, not a promise; both sides must send Window Scale options in their SYN segments to enable window scaling in either direction. If window scaling is enabled, then the TCP that sent this option will right-shift its true receive-window values by 'shift.cnt' bits for transmission in SEG.WND. The value 'shift.cnt' may be zero (offering to scale, while applying a scale factor of 1 to the receive window).

This option may be sent in an initial <SYN> segment (i.e., a segment with the SYN bit on and the ACK bit off). It may also be sent in a <SYN,ACK> segment, but only if a Window Scale option was received in the initial <SYN> segment. A Window Scale option in a segment without a SYN bit should be ignored.

The Window field in a SYN (i.e., a <SYN> or <SYN,ACK>) segment itself is never scaled.

Zusätzlicher Platz für Lösungen. Markieren Sie deutlich die Zuordnung zur jeweiligen Teilaufgabe. Vergessen Sie nicht, ungültige Lösungen zu streichen.

