Technische Universität München
Lehrstuhl Informatik VIII
Prof. Dr.-Ing. Georg Carle
Dipl.-Ing. Stephan Günther, M.Sc.
Nadine Herold, M.Sc.
Dipl.-Inf. Stephan Posselt

Tutorübung zur Vorlesung Grundlagen Rechnernetze und Verteilte Systeme
Übungsblatt 8 (10. Juni – 17. Juni 2013)

Hinweis: Die mit * gekennzeichneten Teilaufgaben sind ohne Kenntnis der Ergebnisse vorhergehender Teilaufgaben lösbar.

Aufgabe 1 Schiebefensterprotokolle

Angelehnt an das Alternating Bit Protocol könnte man das folgende Sliding-Window-Verfahren definieren: Das Protokoll verwende ein Sende- und Empfangsfenster der Größe $w_s = w_r = 2$ sowie den Sequenznummernraum $\mathcal{S} = \{0, 1\}$. Die Fehlerbehandlung erfolge analog zu Go-Back-N. Abbildung 1 zeigt eine Datenübertragung. Die Blitze stehen für durch Störungen verlorengegangene Segmente. Die beiden ersten ACKs erreichen also nicht den Sender.

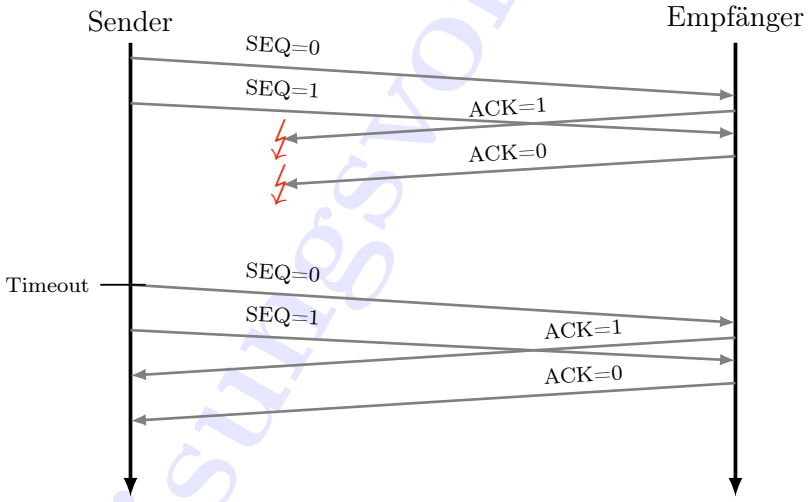


Abbildung 1: Modifiziertes Alternating-Bit-Protocol

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.

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 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 Abbilung 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 $x, x+1, \ldots, x+w_s-1$ 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 kummulativen 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 Sequenznummer $0, 1, 2, 3$. 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, wie groß das Empfangsfenster des Empfängers bei den beiden Verfahren jeweils gewählt werden sollte.

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

$$w_s \leq w_r \leq \left\lfloor \frac{|\mathcal{S}|}{2} \right\rfloor .$$

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 hineinfallen.

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. Stau-kontrolle 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 TCP-Datenrate betrachten wir den Verlauf einer zusamenhängenden Datenübertragung, bei der die Slow-Start-Phase bereits abgeschlossen ist. TCP befinde 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 Empfänger 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 TCP-Reno (vgl. Vorlesungsfolien).

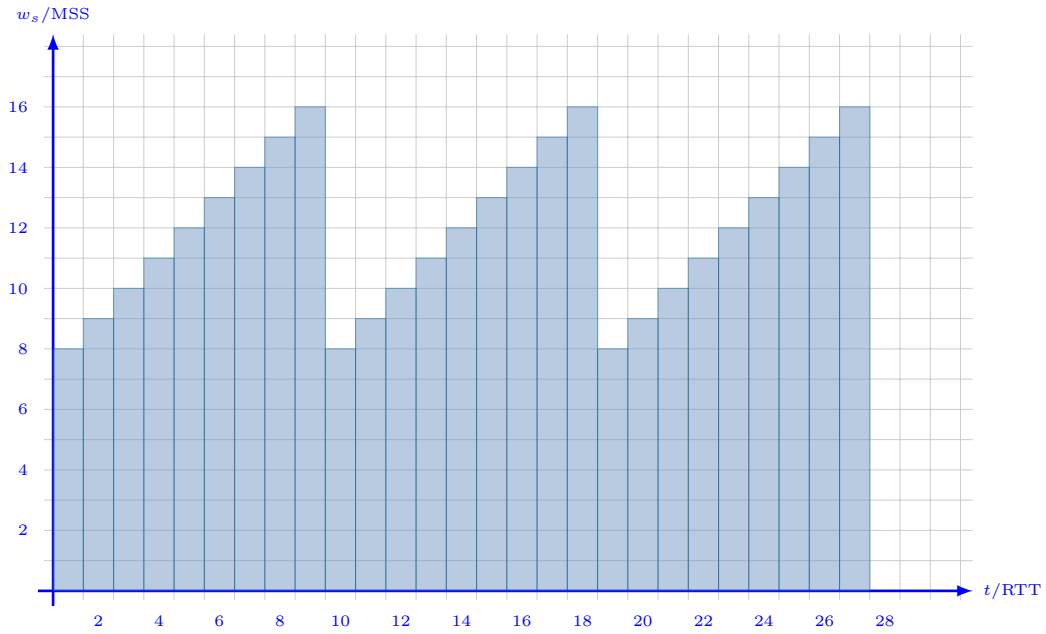
Als konkrete Zahlenwerte nehmen wir an, dass die maximale TCP-Segmentgröße (MSS) 1460 B und die RTT 200 ms beträgt. Die Serialisierungszeit von Segmenten sei gegenüber der Ausbreitungsverzögerung vernachlässigbar klein. Segmentverlust trete ab einer Sendefenstergröße von $w_s \geq x = 16\,\text{MSS}$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

c)* Wieviel Zeit vergeht, bis nach einem Segmentverlust das Staukontrollfenster einfolge 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 $t_0 + \text{RTT}$ bestätigt werden. Folglich erhalten wir für die Zeit bis zum erneuten Erreichen des Maximalwerts

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

d)* 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 $t_0 = 0\,\text{s}$ gerade die Sendefenstergröße halbiert worden sein, also $w_s = x/2$ gelten. Zeichnen Sie das Diagramm im Zeitintervall $t = \{0, \ldots, 27\}$.



e)* Bestimmen Sie allgemein die durchschnittliche Verlustrate θ. 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.

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

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

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

$$\theta = \frac{1}{\frac{3}{8}x^2 + \frac{3}{4}x} \approx 9.26 \cdot 10^{-3}.$$

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

Für die Datenrate erhalten wir

$$r_{TCP} = \frac{n}{T} \cdot (1-\theta) = \frac{108 \cdot 1460 \cdot 8\,\text{bit}}{1.8\,\text{s}} \cdot (1-\theta) \approx 694\,\text{kbit/s}.$$

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

$$r_{UDP} = \frac{15 \cdot (\mathrm{MSS} + 12\,\mathrm{B})}{\mathrm{RTT}} = \frac{15 \cdot (1460\,\mathrm{B} + 12\,\mathrm{B})}{0.2\,\mathrm{s}} \approx 883\,\mathrm{kbit/s}.$$