

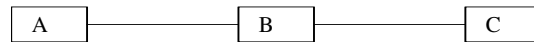


## 4. Übung zur Vorlesung Rechnernetze (Musterlösung)

Abgabe: 07.12.2004

### Aufgabe 1: Effizienz von Transportprotokollen (6 Punkte, 3+1+2)

- a) Auf dem letzten Übungsblatt hatten wir die Effizienz des Stop-And-Wait Protokolls betrachtet, wobei wir einige vereinfachende Annahmen getroffen hatten. Dieses Mal wollen wir die Acknowledge-Pakete nicht vernachlässigen, und auch die Möglichkeit mit berücksichtigen, dass Übertragungsfehler auftreten können.  $p$  sei die Wahrscheinlichkeit, dass ein Datenpaket oder das dazugehörige Acknowledge verloren geht (d.h. die Wahrscheinlichkeit, dass ein gesendetes Paket wiederholt werden muss). Leiten Sie analog zu dem in der Übung verwendeten Verfahren eine Formel zur Berechnung der Effizienz des Stop-and-Wait Protokolls her! Der senderseitige Timeout soll dabei auf  $5 * (TRANS\_A + 2 * PROP)$  festgelegt werden. Vereinfachend gehen wir davon aus, dass die Übertragungszeit für ein Acknowledge ( $TRANS\_A$ ) gleich der Übertragungszeit für ein Datenpaket ( $TRANS\_P$ ) ist.
- b) Betrachten Sie zwei hintereinandergeschaltete Übertragungsstrecken AB und BC. Es wird auf jeder Teilstrecke separat das Stop-and-Wait Protokoll angewandt.



Beide Verbindungen haben eine Kapazität von 1,5 Mbps full-duplex. Die Paketlänge betrage jeweils 512 Bit. Die Signallaufzeit  $Prop$  betrage 5 ms zwischen A und B und 3 ms zwischen B und C.  $p$  betrage 0.02 für die Strecke AB und 0.01 für die Strecke BC. Berechnen Sie mit Hilfe der Formel aus Teilaufgabe a) die Effizienz auf jeder Teilstrecke bei den gegebenen Parametern und geben Sie den Durchsatz von A nach C an!

- c) Berechnen Sie mit Hilfe der in der Übung vorgestellten Formel die Effizienz auf jeder Teilstrecke bei Anwendung des Go-Back-N Verfahrens in Abhängigkeit von der Fenstergröße  $w$ ! Verwenden Sie die in Aufgabe b) angegebenen Parameter (die Übertragungszeit für Acknowledge-Pakete sei dieses Mal jedoch zu vernachlässigen)! Welche Effizienz und welcher Durchsatz auf der Strecke von A nach C ergibt sich bei Fenstergröße 20 und bei Fenstergröße 200?

Lösungsvorschlag:

a)

Sei  $p$  = Wahrscheinlichkeit, dass ein Datenpaket oder das dazugehörige Ack verloren geht. Ein Paket wird im Mittel  $N_x$  mal übertragen.

→ Zeit zwischen 2 Paketen im Mittel:

$$S = (N_x - 1) * (Transp + Timeout) + 1 * (Transp + Transa + 2 * Prop)$$

Der Timeout beim Sender wird auf  $5 * (Transa + 2 * Prop)$  gesetzt. Außerdem sei  $Transa = Transp$ .

→

$$\begin{aligned} S &= N_x * (Transp + 5 * (Transp + 2 * Prop)) - 1 * (Transp + 5 * (Transp + 2 * Prop)) + 1 * (Transp + Transp + 2 * Prop) \\ &= N_x * (6 * Transp + 10 * Prop) - 4 * Transp - 8 * Prop \end{aligned}$$

$N_x$  ist der Erwartungswert einer Zufallsvariablen. Diesen können wir wie folgt berechnen:

$$\begin{aligned} N_x &= \sum_{i \geq 1} i * P(i \text{ Übertragungen}) = \sum_{i \geq 1} i * (p^{i-1} * (1-p)) = (1-p) * \sum_{i \geq 1} i * p^{i-1} = (1-p) * \sum_{i \geq 0} (i+1) * p^i \\ &= (1-p) * \left( \sum_{i \geq 0} i * p^i + \sum_{i \geq 0} p^i \right) = (1-p) * \left( \frac{p}{(1-p)^2} + \frac{1}{1-p} \right) = (1-p) * \frac{p+1-p}{(1-p)^2} = (1-p) * \frac{1}{(1-p)^2} = \frac{1}{1-p} \end{aligned}$$

$$\begin{aligned}
S &= \frac{1}{1-p} * (6 * Transp + 10 * Prop) - 4 * Transp - 8 * Prop \\
&= \frac{6 * Transp + 10 * Prop}{1-p} - (4 * Transp + 8 * Prop) = \frac{6 * Transp + 10 * Prop - (1-p) * (4 * Transp + 8 * Prop)}{1-p} \\
&= \frac{6 * Transp + 10 * Prop - 4 * Transp - 8 * Prop + 4 * p * Transp + 8 * p * Prop}{1-p} \\
&= \frac{(2 + 4 * p) * Transp + (2 + 8 * p) * Prop}{1-p} \\
U &= \frac{Transp}{S} = \frac{Transp}{\frac{(2 + 4 * p) * Transp + (2 + 8 * p) * Prop}{1-p}} = \frac{Transp * (1-p)}{(2 + 4 * p) * Transp + (2 + 8 * p) * Prop}
\end{aligned}$$

b)

$$\text{Übertragungsdauer für ein Datenpaket } TRANSP = \frac{512 \text{ Bit}}{1.500.000 \text{ Bit / s}} = 0,341 \text{ ms}$$

$$\text{Übertragungsdauer für ein Ack } TRANSACK = TRANSP = 0,341 \text{ ms}$$

Für Link AB gilt:

Verlustwahrscheinlichkeit =  $p = 0.02$

Signallaufzeit = PROP = 5ms.

$$U = (0,341 * 0,98) / (2,08 * 0,341 + 2,16 * 5) = 0,02903$$

Somit ergibt sich ein effektiver Durchsatz von 1,5Mbps mal 0,02903 = 0,04355 Mbps = 43,55 Kbps.

Für Link BC gilt:

Verlustwahrscheinlichkeit =  $p = 0.01$

Signallaufzeit = PROP = 3ms.

$$U = (0,341 * 0,99) / (2,04 * 0,341 + 2,08 * 3) = 0,04867$$

Somit ergibt sich ein effektiver Ende-zu-Ende Durchsatz von 43,55 Kbps mal 0,04867 = 2,11958 Kbps.

c)

Für Link AB gilt:

Falls  $w \geq 1 + 2 * (PROP / TRANSP) = 1 + 2 * (5 / 0,341) = 31$ :

$$U = (1-p) / (1 + 2 * p * (PROP / TRANSP)) = 0,98 / (1 + 0,04 * (5 / 0,341)) = 0,61770$$

Falls  $w < 31$ :

$$\begin{aligned}
U &= ((1-p) * w) / ((2 * (PROP / TRANSP) + 1) * (1-p + w * p)) \\
&= (0,98 * w) / ((2 * (5 / 0,341) + 1) * (0,98 + 0,02 * w)) = (0,98 * w) / (30,32551 * (0,98 + 0,02 * w)) \\
&= (0,98 * w) / (29,71900 + 0,60651 * w)
\end{aligned}$$

Bei Fenstergröße 20 ergibt sich:  $U = (0,98 * 20) / (29,71900 + 0,60651 * 20) = 0,46834$

Somit ergibt sich ein effektiver Durchsatz von 1,5 Mbps mal 0,46834 = 0,70252 Mbps.

Bei Fenstergröße 200 ergibt sich:  $U = 0,61770$

Somit ergibt sich ein effektiver Durchsatz von 1,5 Mbps mal 0,61770 = 0,92655 Mbps.

Für Link BC gilt:

Falls  $w \geq 1 + 2 * (PROP / TRANSP) = 1 + 2 * (3 / 0,341) = 19$ :

$$U = (1-p) / (1 + 2 * p * (PROP / TRANSP)) = 0,99 / (1 + 0,02 * (3 / 0,341)) = 0,84187$$

Falls  $w < 19$ :

$$\begin{aligned}
U &= ((1-p) * w) / ((2 * (PROP / TRANSP) + 1) * (1-p + w * p)) \\
&= (0,99 * w) / ((2 * (3 / 0,341) + 1) * (0,99 + 0,01 * w)) = (0,99 * w) / (18,59530 * (0,99 + 0,01 * w)) \\
&= (0,99 * w) / (18,40935 + 0,18595 * w)
\end{aligned}$$

Bei beiden Fenstergrößen ergibt sich:  $U = 0,84187$

Somit ergibt sich ein Ende-zu-Ende Durchsatz bei Fenstergröße 20 von 0,70252 Mbps mal 0,84187 = 0,59143 Mbps.

Somit ergibt sich ein Ende-zu-Ende Durchsatz bei Fenstergröße 200 von 0,92655 Mbps mal 0,84187 = 0,78003 Mbps.

## Aufgabe 2: Implementierung eines einfachen File-Transfer Protokolls (4 Punkte)

Wir wollen die Programmieraufgabe vom vergangenen Übungsblatt erweitern. Laden Sie sich zunächst den Java-Code aus folgendem Archiv herunter: [http://www-vs.informatik.uni-ulm.de/teach/ws04/rn1/OSI\\_Aufgabe2.zip](http://www-vs.informatik.uni-ulm.de/teach/ws04/rn1/OSI_Aufgabe2.zip).

(Achtung: Wir haben die Implementierung der Layer4 Klasse etwas verändert, sodass nun auch der Verlust von Acknowledge Paketen simuliert wird).

Zunächst sollen Sie das Rahmenformat der Layer7-Nachrichten erweitern. Zusätzlich zu der Sequenznummer soll der Header noch ein weiteres Byte enthalten, welches den Typ des Paketes definiert. Die möglichen Pakettypen werden folgendermaßen codiert: 1 = *Start\_of\_File*, 2 = *Datenpaket*, 3 = *Acknowledge*, 4 = *End\_of\_File*. Anstelle eines einfachen Byte-Arrays soll die Klasse *Layer7* nun das Versenden und Empfangen einer kompletten ASCII-Datei ermöglichen (verwenden Sie die Datei *File.xy*, die sie ebenfalls in dem Archiv finden). Dazu soll zunächst in einem *Start\_of\_File* Paket der Dateiname übertragen werden. In den nachfolgenden *Datenpaketen* (die eine Maximallänge von 100 Byte haben sollen), soll dann der Inhalt der Datei übertragen werden. Die Übertragung wird mit einem *End\_of\_File* Paket abgeschlossen. Nach Erhalt dieses Pakets soll der Empfänger die erhaltene Datei auf der lokalen Festplatte abspeichern. Jedes Paket wird vom Empfänger mit einem *Acknowledge* bestätigt. Wie auf dem letzten Übungsblatt sollen Sie sicherstellen, dass verloren gegangene Pakete automatisch wiederholt werden (die Klasse *Layer4* simuliert einen zufälligen Paketverlust).

Geben Sie bitte den kompletten Quelltext ausgedruckt oder per Email ab! Schicken Sie den gesamten Quelltext gezippt an: [schorr@informatik.uni-ulm.de](mailto:schorr@informatik.uni-ulm.de) mit dem Betreff „RN1 – Übung 4“.

*Lösungsvorschlag:*

*Siehe Quellcode: [http://www-vs.informatik.uni-ulm.de/teach/ws04/rn1/OSI\\_Aufgabe2\\_Musterloesung.zip](http://www-vs.informatik.uni-ulm.de/teach/ws04/rn1/OSI_Aufgabe2_Musterloesung.zip)*

*Achtung! Die erste Version der Musterlösung enthielt noch einen Bug. Die Acknowledge-Pakete wurden ohne das Typfeld im Header verschickt. Der Fehler wurde in der Musterlösung inzwischen korrigiert.*