Übung 8 zur Vorlesung "Vernetzte Systeme" WS 1999/2000

Prof. Dr. F. Mattern

Ausgabedatum: 8. Dez. 1999
Abgabedatum: 15. Dez. 1999

32. Aufgabe (Ethernet)

(3 Punkte) Erläutern Sie, inwiefern es ein Problem ist, wenn bei Ethernet eine Kollision erst dann erkannt werden würde, wenn das Datenpaket bereits abgesendet wurde (wieso also Mindestpaketlängen sinnvoll bzw. notwendig sind).

33. Aufgabe (CSMA/CD und Binary Exponential Backoff)

Wenn im CSMA/CD-Protokoll zwei Stationen ein freies Übertragungsmedium erkennen und "fast" gleichzeitig mit dem Senden beginnen, kommt es trotz *carrier sense* zu einer Kollision. Sobald dies von einer der Stationen erkannt wird, sendet diese ein kurzes *Jam*-Signal und beginnt nach einer gewissen Wartezeit einen erneuten Sendeversuch.

In der offiziellen Ethernet-Spezifikation (IEEE 802.3) wird diese Wartezeit durch den *binary exponential backoff algorithm* wie folgt definiert:

> The delay is an integral multiple of slot time. The number of slot times to delay before the n-th retransmission attempt is chosen as a uniformly distributed random integer r in the range $0 \leq r < 2^K$, where $K = \min(n, 10)$.

a) (3 Punkte) Was ist der Vorteil des *binary exponential backoff algorithm* gegenüber dem Bestimmen der Wartezeit innerhalb fester Grenzen (also K=const, z.B. K=10)?

b) (4 Punkte) In einem IEEE 802.3 (Ethernet) LAN mit 4 Stationen gebe es folgende Sendewünsche:[1]

- Station A: Slot 1, Slot 3, Slot 12
- Station B: Slot 1, Slot 7, Slot 8
- Station C: Slot 5
- Station D: Slot 5

Sendewünsche werden gepuffert, bis sie erfüllt werden können. Die Stationen verwenden eine Zufallszahlenfunktion, die Zufallszahlen zwischen 0 und 16383 liefert, sowie die Modulo-Division, um den Bereich der Zufallszahlen auf das jeweils gültige Intervall einzuschränken. Um z.B. zu bestimmen, wann eine Station ihren 2. *retransmission attempt* durchführt, wird mit *modulo 4* gerechnet; beim 3. Mal mit *modulo 8*, usw.

Die Reihen der Zufallszahlen seien wie folgt (nicht alle Zahlen werden tatsächlich benötigt):

- Station A: 10, 3277, 13815, 3386, 2371, 5378
- Station B: 777, 2023, 8575, 989, 5034, 99
- Station C: 746, 2523, 733, 432, 9234, 2487
- Station D: 944, 258, 6403, 3922, 2348, 4287

Vervollständigen Sie die folgende Tabelle, indem Sie einen erfolgreichen Sendeversuch mit **S** markieren; einen erfolglosen Sendeversuch (d.h. eine Kollision) mit **X**, eine wartende Station mit **W** und eine inaktive Station mit – markieren:

Zeitschlitz	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Station A																				
Station B																				
Station C																				
Station D																				

[1] Zur Vereinfachung nehmen wir an, dass Sendewünschen im LAN nur jeweils zu Beginn eines Slots auftreten.

34. Aufgabe (Sliding-Window-Protokoll mit Ringpuffer)

Ringpuffer erlauben das Speichern einer fortlaufenden Reihe von Werten in einer endlichen Datenstruktur. Ein solcher Puffer kann als einfacher Cachespeicher benutzt werden, in dem die letzten `MAX_BUF` Einträge zwischengespeichert werden. Auch als Undo-Speicher in einem Editor kann ein solcher Ringpuffer Einsatz finden. Auch im Sliding-Window-Protokoll bietet sich eine Verwendung von Ringpuffern an.

Nehmen Sie an, dass für die Realisierung eines Empfangsfensters der Grösse `MAX_BUF` als Ringpuffer ein Array *Buf*[0:`MAX_BUF-1`] verwendet wird. Eine Integer-Variable *Nfe* enthalte die Sequenznummer des NFE (*next frame expected*). Der Puffer sei eingebettet zwischen zwei Kommunikationsschichten, die jeweils neue Pakete anliefern bzw. gepufferte Pakete anfordern. Beachten Sie, dass in diesem Szenario das Empfangsfenster nur dann weiterrücken kann, wenn von der darüberliegenden Schicht auch wirklich ein Paket angefordert wurde – erfolgt dies nicht, füllt sich irgendwann der Puffer und neu angelieferte Pakete müssen abgelehnt werden.

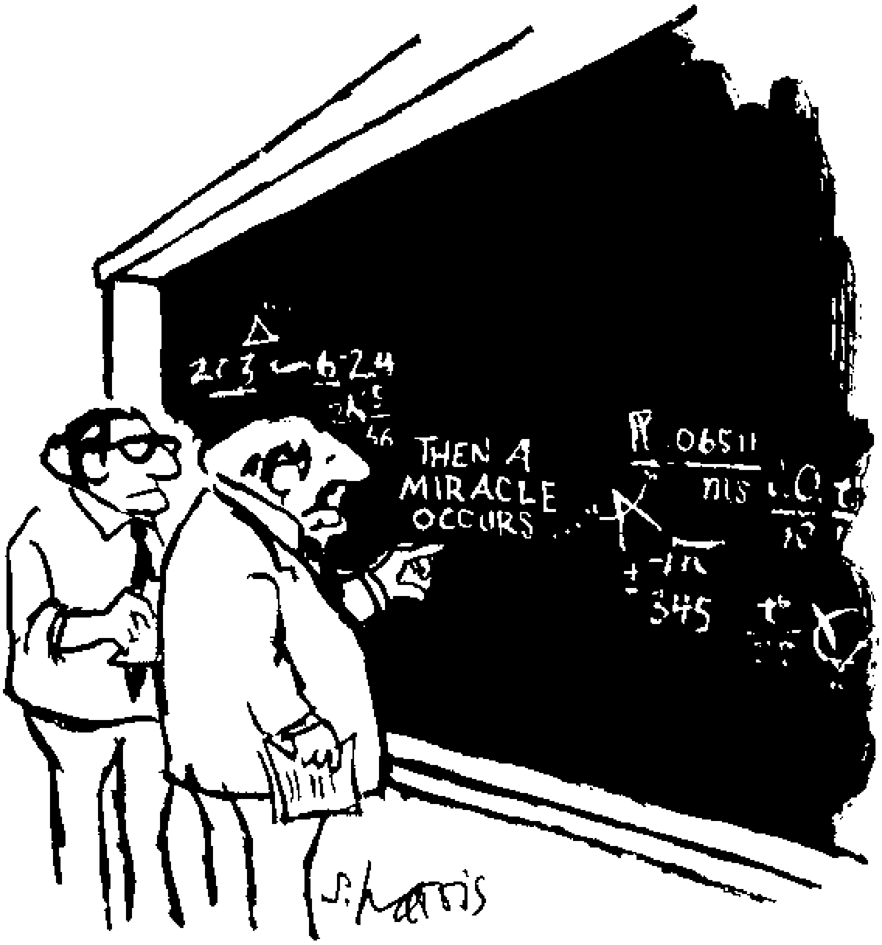
Schildern Sie (in Pseudocode), was geschieht (also welche Variablen wie verändert werden, welche Werte zurückgegeben werden bzw. welche Acknowledgements wann versendet werden), wenn

a) (4 Punkte) die tieferliegende Kommunikationsschicht ein Datenpaket mit der Sequenznummer s anbietet,

b) (4 Punkte) die darüberliegende Kommunikationsschicht ein Datenpaket anfordert.

Beachten Sie, dass gepufferte Pakete immer in der richtigen Reihenfolge weitergeleitet werden müssen, dass der Puffer voll sein kann und dass Datenpakete mit zu kleiner oder zu grosser Sequenznummer geeignet behandelt werden müssen. (Müssen im letzten Falle auch Acknowledgements versandt werden?)

c) (2 Punkte) Wenn der Puffer leer ist, die darüberliegende Kommunikationsschicht aber ein Datenpaket anfordert, sollte dieser Aufruf an den Puffer blockieren. Schildern Sie, was bei einer Implementierung beachtet werden muss, wenn der Puffer aber dennoch während dieser Blockade Datenpakete der tieferliegenden Kommunikationsschicht entgegennehmen können soll.



"I THINK YOU SHOULD BE MORE EXPLICIT HERE IN STEP TWO."

$Id: uebung8.lyx,v 1.9 1999/12/08 11:39:36 langhein Exp $