

Grundlagen Rechnernetze und Verteilte Systeme

IN0010, SoSe 2019

Übungsblatt 5

27. Mai – 31. Mai 2019

Hinweis: Mit * gekennzeichnete Teilaufgaben sind ohne Lösung vorhergehender Teilaufgaben lösbar.

Aufgabe 1 Medienzugriffsverfahren

a)* Erläutern Sie kurz das Prinzip von *ALOHA*.

Eine Station sendet, sobald Daten anliegen. Übertragungen werden Out-of-Band bestätigt (andere Frequenz).

b) Wie werden Kollisionen in *ALOHA* erkannt?

Nicht direkt, sondern über das Ausbleiben der Out-of-Band Bestätigung.

c) Erläutern Sie kurz das Prinzip von **Slotted ALOHA**.

Station beginnt im nächsten Zeitslot zu senden, ungeachtet dessen, ob bereits eine Übertragung stattfindet.

d) Worin besteht der Vorteil von *Slotted ALOHA* gegenüber normalem *ALOHA*?

Durch die Aufteilung in Zeitslots verringert sich die Wahrscheinlichkeit für Kollisionen, da Stationen nicht mehr zu jedem beliebigen Zeitpunkt mit einer Übertragung beginnen können.

Sofern die Zeitslots der Übertragungsdauer einer kompletten Nachricht entsprechen und die Knoten ausreichend genau miteinander synchronisiert sind (was bei derart langen Zeitslots möglich ist), tritt eine Kollision entweder zu Beginn eines Zeitslots auf oder es ist garantiert, dass höchstens eine Station sendet. (siehe Vorlesung)

e)* Erläutern Sie kurz das Prinzip von *CSMA*.

Medium wird vor dem Senden abgehört. Wenn das Medium im aktuellen Zeitslot frei ist, kann im nächsten begonnen werden zu senden.

f) Erläutern Sie kurz, welche Ergänzungen *CSMA/CD* gegenüber reinem *CSMA* hat.

Kollisionen werden erkannt und betroffene Rahmen erneut übertragen.

g) Wie werden erfolgreiche Übertragungen bei *CSMA/CD* bei Ethernet erkannt?

Eine Übertragung wird als erfolgreich angenommen, wenn während der Übertragung keine Kollision festgestellt wurde bzw. kein JAM-Signal empfangen wurde.

h) Erläutern Sie kurz, welche Ergänzungen *CSMA/CA* gegenüber reinem *CSMA* hat.

Kollisionen können i. A. nicht erkannt werden. Stattdessen wird deren Auftrittswahrscheinlichkeit durch Randomisierung des Sendebeginns verringert.

i)* Was versteht man unter *Binary Exponential Backoff*?

CSMA (sowohl *CD* als auch *CA*) warten nach einer Kollision bzw. nicht erfolgreichen Übertragung eine zufällige Anzahl an Slotzeiten. Diese Anzahl wird aus dem Backoff-Window zufällig und gleichverteilt gezogen. Mit jeder Kollision bzw. nicht erfolgreichen Übertragung wird dieses Fenster verdoppelt (binary exponential) bis ein bestimmter Maximalwert erreicht ist. Nach einer erfolgreichen Übertragung wird das Fenster zurückgesetzt.

Aufgabe 2 ALOHA und CSMA/CD

Gegeben sei ein Netzwerk (s. Abbildung 1) bestehend aus drei Computern, welche über ein Hub miteinander verbunden sind. Die Distanzen zwischen den Computern betragen näherungsweise $d_{12} = 1 \text{ km}$ bzw. $d_{23} = 500 \text{ m}$. Etwaige indirekte Kabelführung darf vernachlässigt werden. Die Übertragungsrate betrage $r = 100 \text{ Mbit/s}$. Die

relative Ausbreitungsgeschwindigkeit betrage wie üblich $\nu = 2/3$. Die Lichtgeschwindigkeit sei mit $c_0 = 3 \cdot 10^8 \text{ m/s}$ gegeben.

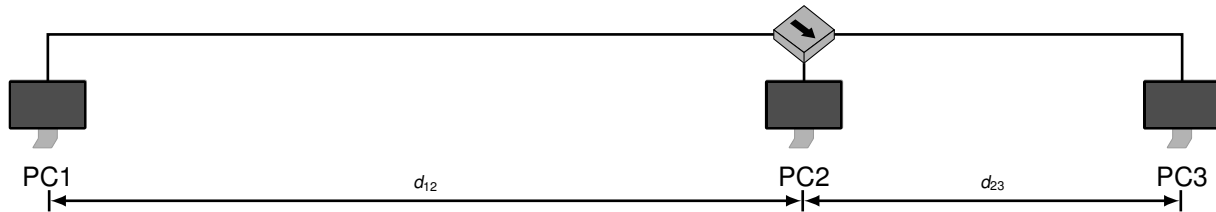


Abbildung 1

Zum Zeitpunkt

- $t_0 = 0 \text{ s}$ findet keine Übertragung statt und keiner der Rechner hat Daten zu versenden,
- $t_1 = 5 \mu\text{s}$ beginnt PC1,
- $t_2 = 15 \mu\text{s}$ beginnt PC2 und
- $t_3 = 10 \mu\text{s}$ Beginn PC3

jeweils einen Rahmen der Länge 94 B zu senden.

a)* Berechnen Sie die Serialisierungszeit t_s für eine Nachricht.

$$t_s = \frac{l}{r} = \frac{94 \cdot 8 \text{ bit}}{100 \cdot 10^6 \text{ bit/s}} = 7,52 \mu\text{s}$$

b)* Berechnen Sie die Ausbreitungsverzögerungen $t_p(1,2)$ und $t_p(2,3)$ auf den beiden Streckenabschnitten.

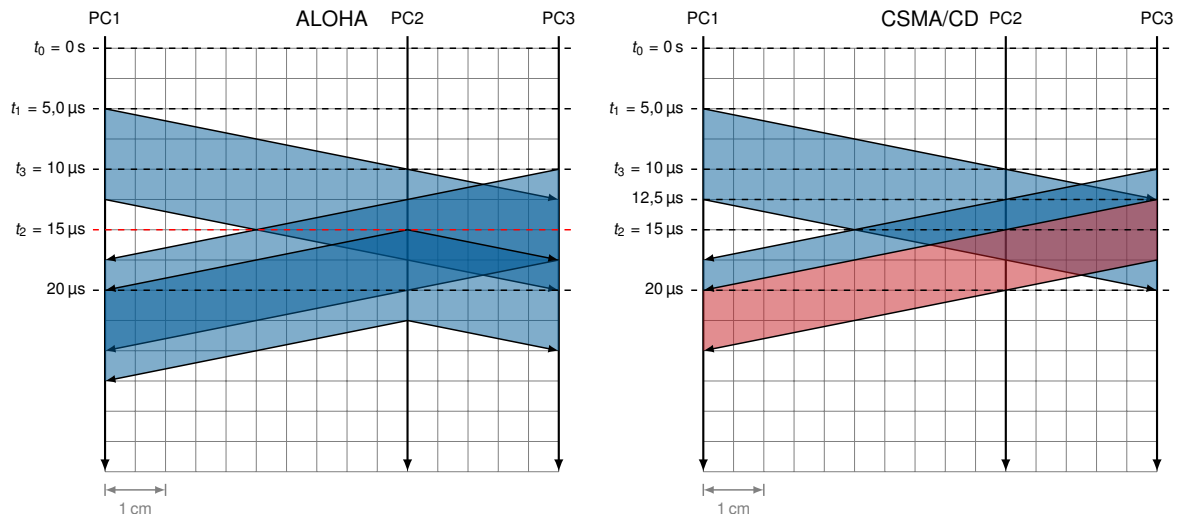
$$t_p(1,2) = \frac{d_{12}}{\nu c_0} = \frac{1000 \text{ m}}{\frac{2}{3} \cdot 3 \cdot 10^8 \text{ m/s}} = 5,0 \mu\text{s}$$

$$t_p(2,3) = \frac{d_{23}}{\nu c_0} = \frac{500 \text{ m}}{\frac{2}{3} \cdot 3 \cdot 10^8 \text{ m/s}} = 2,5 \mu\text{s}$$

c) Zeichnen Sie für ALOHA und 1-persistentes CSMA/CD jeweils ein Weg-Zeit-Diagramm, das den Sendevorgang im Zeitintervall $t \in [t_0, t_0 + 30 \mu\text{s})$ darstellt. Maßstab: $100 \text{ m} \triangleq 5 \text{ mm}$ bzw. $2,5 \mu\text{s} \triangleq 5 \text{ mm}$, Slotzeit: $\approx 5 \mu\text{s}$

Bei ALOHA wird das Medium nicht abgehört. Dies bedeutet, dass zum Zeitpunkt t_2 PC2 zu senden beginnt, obwohl er die Übertragung von PC1 und PC3 bereits detektieren könnte. Im Gegensatz dazu wird bei CSMA/CD das Medium abgehört. Aus diesem Grund beginnt PC2 nicht zu senden. PC3 allerdings kann infolge der endlichen Signalausbreitungsgeschwindigkeit noch nicht wissen, dass PC1 bereits sendet. Es kommt also zur Kollision.

Bei $t = 12,5 \mu\text{s}$ erkennt PC3 die Kollision und bricht die eigene Übertragung ab. Um sicherzugehen, dass alle Stationen, die an das gemeinsame Medium angeschlossen sind, über die Kollision informiert werden, sendet PC3 ein *JAM-Signal*. Dieses wäre bei Ethernet ein 4 B langes alternierendes Bitmuster (in der Aufgabe geht es bis jetzt nicht konkret um Ethernet, sondern lediglich um das zugrundeliegende Medienzugriffsverfahren CSMA/CD – eine Andeutung des JAM-Signals reicht aus).



d) Aus der vorhergehenden Teilaufgabe ist zu erkennen, dass bei beiden Verfahren Kollisionen auftreten. Im Gegensatz zu ALOHA funktioniert CSMA/CD aber unter den gegebenen Umständen nicht. Warum?

Bei ALOHA wird der Verlust eines Rahmens dadurch erkannt, dass der Absender keine Bestätigung erhält. Ein derartiges Quittungsverfahren gibt es bei CSMA/CD nicht. Stattdessen nimmt ein Sender bei CSMA/CD an, dass ein Rahmen erfolgreich übertragen wurde, falls während der Übertragung keine Kollision aufgetreten ist. Im konkreten Fall hat PC1 allerdings die Übertragung abgeschlossen, bevor ihn die Übertragung bzw. das JAM-Signal von PC3 erreicht. PC1 erkennt daher die Kollision nicht und geht fälschlicherweise von einer erfolgreichen Übertragung aus.

e) Wie lautet für CSMA/CD die Bedingung, dass ein Knoten eine Kollision rechtzeitig erkennen kann?

Die Serialisierungszeit muss mind. der doppelt so lang sein, wie die maximal mögliche Ausbreitungsverzögerung zwischen den beiden am weitesten voneinander entfernten Knoten. Nur so ist sichergestellt, dass ein Knoten noch sendet, wenn er das „Störsignal“ des am weitesten von ihm entfernten Knoten empfängt, der seinerseits unmittelbar vor der „Ankunft des ersten Bits“ selbst angefangen hat zu senden.

f) Berechnen Sie für CSMA/CD die maximale Entfernung zweier Rechner innerhalb einer Kollisionsdomäne in Abhängigkeit der minimalen Rahmenlänge. Setzen Sie die Werte für FastEthernet ein ($r = 100 \text{ Mbit/s}$, $l_{\min} = 64 \text{ B}$).

Im Falle einer Kollision darf keiner der sendenden Knoten seinen Sendevorgang beenden, bevor er die Kollision bemerkt hat. Ansonsten würde er davon ausgehen, dass die Übertragung erfolgreich war. Dies bedeutet, die minimale Serialisierungszeit $t_{s,\min}$ eines Rahmens muss zweimal der Ausbreitungsverzögerung zwischen den beiden am weitesten entfernten Stationen entsprechen:

$$\begin{aligned}
 t_{s,\min} &= 2 \cdot t_{p,\max} \\
 \frac{l_{\min}}{r} &= 2 \cdot \frac{d_{\max}}{\nu c} \\
 d_{\max} &= \frac{1}{2} \cdot \nu c \cdot \frac{l_{\min}}{r} \\
 d_{\max} &= \frac{1}{2} \cdot \frac{2}{3} \cdot 3 \cdot 10^8 \frac{\text{m}}{\text{s}} \cdot \frac{64 \cdot 8 \text{ bit}}{100 \cdot 10^6 \frac{\text{bit}}{\text{s}}} \\
 d_{\max} &= 10^8 \frac{\text{m}}{\text{s}} \cdot \frac{64 \cdot 8 \text{ bit}}{100 \cdot 10^6 \frac{\text{bit}}{\text{s}}} = 512 \text{ m}
 \end{aligned}$$

g)* Welchen Einfluss haben Hubs, Brücken und Switches auf die Kollisionsdomäne?

Hubs verbinden Netzsegmente auf der physikalischen Schicht, wodurch eine gemeinsame Kollisionsdomäne entsteht (Bus).

Brücken unterbrechen Kollisionsdomänen, indem Rahmen nur dann in das jeweils andere Netzsegment weitergeleitet werden, wenn sich der jeweilige Empfänger in diesem Segment befindet. (Die Weiterleitungsentcheidungen werden auf Basis der Ziel-MAC-Adresse getroffen. Durch Beobachtung des Datenverkehrs kann eine Brücke mit der Zeit lernen, welche Knoten sich auf welcher Seite befinden.)

Switches sind im Wesentlichen Brücken mit mehr als zwei Ports. Sie unterbrechen daher ebenfalls Kollisionsdomänen.

Aufgabe 3 Cyclic Redundancy Check (CRC)

Die Nachricht 10101100 werde mittels CRC, wie in der Vorlesung eingeführt, gesichert. Als Reduktionspolynom sei $r(x) = x^3 + 1$ gegeben.

a)* Wie lang ist die Checksumme?

Die Länge der Checksumme in Bit entspricht dem Grad des Reduktionspolynoms, hier also $\text{grad}(r(x)) = 3$ bit.

b) Bestimmen Sie die Checksumme für die gegebene Nachricht.

Zunächst werden an die Nachricht $\text{grad}(r(x)) = 3$ Nullen angehängt: 10101100000. Anschließend wird durch $r(x)$ dividiert:

```

10101100000 : 1001 = 10111011 Rest 011
1001|
----|
001111|
1001|
----|
01100|
1001|
----|
01010|
1001|
----|
001100|
1001|
----|
01010
1001
----
0011

```

c) Geben Sie die übertragene Bitfolge an.

Die übertragene Bitfolge besteht aus der ursprünglichen Nachricht konkateniert mit der eben berechneten Prüfsumme: 10101100 011.

Bei der Übertragung trete nun das Fehlermuster 00100000000 auf.

d)* Wie lautet die empfangene Bitfolge?

Die empfangene Bitfolge ist das XOR aus der übertragenen Bitfolge und dem Fehlermuster:

```

10101100011
XOR 00100000000
-----
10001100011

```

e) Zeigen Sie, dass der Übertragungsfehler erkannt wird.

Die empfangene Bitfolge wird wieder durch $r(x)$ dividiert:

```

10001100011 : 1001 = 10011111 Rest 100
1001|
----|
0001110|
  1001|
  ----|
  01110|
   1001|
   ----|
   01110|
    1001|
    ----|
    01111|
     1001|
     ----|
     01101|
      1001|
      ----|
      0100

```

Es bleibt der Rest 100. Bei einer fehlerfreien Übertragung hätte hingegen kein Rest bleiben dürfen.

f)* Geben Sie ein Fehlermuster an, welches nicht erkannt werden kann.

Vielfache des Reduktionspolynoms können nicht erkannt werden, z. B. 10010000000:

```

  10101100011
XOR 10010000000
-----
  00111100011

```

Division durch $r(x)$ ergibt:

```

(war: 00110111 Rest 000)
00111100011 : 1001 = 00111011 Rest 000
001001|
----|
  01100|
   1001|
   ----|
   01010|
    1001|
    ----|
    001101|
     1001|
     ----|
     01001|
      1001|
      ----|
      0000

```

g) CRC wurde in der Vorlesung ausdrücklich als fehlererkennender, nicht aber als fehlerkorrigierender Code eingeführt. Zeigen Sie, dass mittels CRC selbst 1 bit-Fehler im konkreten Beispiel dieser Aufgabe nicht korrigierbar sind.

Argumentativ Die übertragene Nachricht ist 11 bit lang, d. h. es gibt insgesamt elf mögliche 1 bit-Fehler. Die Checksumme ist jedoch nur 3 bit lang, d. h. es könnten maximal sieben Bitfehler unterschieden werden, da nur

sieben von Null verschiedene Reste existieren. Damit ist eine eindeutige Zuordnung von einem Rest auf einen konkreten Bitfehler nicht möglich.

Beweis durch Gegenbeispiel Es reicht, zwei unterschiedliche Fehlermuster zu finden, die denselben Rest erzeugen, denn dann kann von diesem Rest nicht eindeutig auf einen der beiden Fehler geschlossen werden. Die Fehlermuster 00001000000 und 00000001000 liefern beide den Rest 001.

Diskussion: Was ist mit längeren Checksummen?

Ein Ethernet-Rahmen hat eine Größe von bis zu 1518 B¹ inkl. Checksumme. Dies entspricht $1518 \cdot 8 = 12144$ möglichen 1 bit-Fehlern. Die Checksumme ist 32 bit lang, womit sich $2^{32} - 1$ von Null verschiedene Reste ergeben. Da nun $2^{32} - 1 > 12118$ ist, könnte nach obiger Darstellung eine Korrektur möglich sein. Allerdings muss dazu etwas mehr Mathematik betrachtet werden, da die Anzahl möglicher Reste und deren eindeutige Zuordbarkeit zu 1 bit-Fehlern vom Aufbau des Reduktionspolynoms abhängen.

Tatsächlich ist es nun so, dass mittels des bei Ethernet verwendeten Polynoms 1 bit-Fehler wirklich zu eindeutigen Resten führen. Eine Korrektur wäre damit zwar möglich, wird in der Praxis bei Ethernet aber nicht verwendet.

Grundsätzlich hängt es also von

- der Wahl des Reduktionspolynoms und
- dem Größenverhältnis zwischen Nutzdaten und Checksumme

ab, ob 1 bit-Fehler mittels CRC korrigierbar sind.

¹Jumbo-Frames werden in GRNVS nicht betrachtet

Aufgabe 4 Bit-Stuffing (Hausaufgabe)

In der Vorlesung wurden Verfahren zum Erkennen von Rahmengrenzen vorgestellt. Bei FastEthernet beispielsweise werden Rahmengrenzen mit Hilfe von Steuerzeichen erkannt, was durch die Verwendung der 4B5B-Kodierung möglich wird. Dabei stellt der 4B5B-Code bereits sicher, dass diese Steuerzeichen nie zufällig durch die Konkatenation von Codewörtern auftreten können.

Im Folgenden wollen wir untersuchen, wie das *High Level Data Link Control (HDLC) Protocol* diesem Problem begegnet. Bei HDLC handelt es sich um ein alternatives Schicht-2 Protokoll, welches beispielsweise auf seriellen Schnittstellen zwischen Cisco-Routern eingesetzt wird. Ein Beispiel für einen HDLC-Rahmen ist in Abbildung 2 dargestellt (vgl. Ethernet-Rahmen in den Vorlesungsfolien).

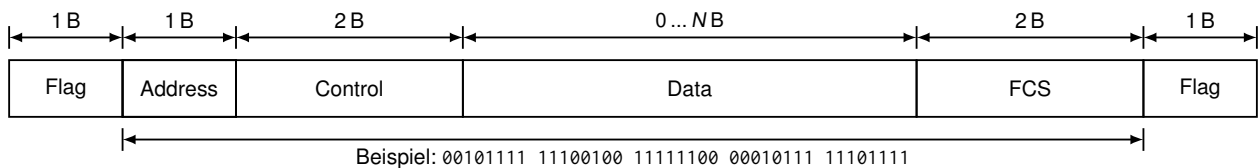


Abbildung 2: Aufbau eines HDLC-Rahmens

HDLC sieht am Anfang und Ende eines jeden Rahmens ein spezielles Begrenzungsfeld (Flag) mit dem Wert $0x7E$ vor. Es ist also nicht auf die Verwendung zusätzlicher Codes zur Erkennung von Rahmengrenzen angewiesen. Allerdings muss nun sichergestellt werden, dass das Begrenzungsfeld nicht zufällig in einem Rahmen auftaucht. Alle Headerfelder seien von konstanter² Länge. Die Länge der Nutzdaten sei stets ein ganzzahliges³ Vielfaches von 8 bit.

a)* In Abbildung 2 ist eine beispielhafte Bitsequenz für den Rahmen ohne Begrenzungsfelder abgebildet. Diese soll nun mittels HDLC übertragen werden. Um zu verhindern, dass das Begrenzungsfeld $0x7E$ in den Daten auftaucht, soll Bit-Stuffing verwendet werden. Überlegen Sie sich ein möglichst effizientes Verfahren. Beschreiben Sie dieses in Worten.

Nach jeweils fünf konsekutiven Einsen wird eine Null eingefügt. Dies kann vom Empfänger leicht rückgängig gemacht werden, indem er nach jeweils fünf konsekutiven Einsen die darauf folgende Null wieder entfernt.

Würde erst nach sechs Einsen eine Null eingefügt, so würde die Bitfolge 0111111 nach dem Stuffing genau das Begrenzungsfeld ergeben. Bereits nach vier oder weniger Einsen zu stopfen erhält zwar die Codetransparenz, ist jedoch weniger effizient.

Anmerkung: Warum nicht nach jeder sechsten Eins eine Siebte stopfen? Aus Sicht der Codetransparenz spricht nichts dagegen. Für HDLC wurde aber die oben beschriebene Variante gewählt. Der einzige Vorteil der HDLC-Version besteht in meinen Augen darin, dass lange Eins-Folgen verhindert werden. Je nach Leitungscodierung kann dies die Taktrückgewinnung ermöglichen, z. B. bei Einsatz einer „inversen“ MLT-3 Kodierung, bei der eine logische Null durch einen Pegelwechsel und eine logische Eins durch einen ausbleibenden Pegelwechsel dargestellt wird. Bei anderen Leitungscodes, wie z. B. NRZ, wird die Taktrückgewinnung nicht ermöglicht, da nach wie vor beliebig lange Nullfolgen auftreten können.

b) Geben Sie die gesamte zu übertragende Bitsequenz gemäß dem in Teilaufgabe a) entwickelten Verfahren an. Markieren Sie die eingefügten Bitstellen.

01111110 00101111 01100100 11111101 00000010 11111101 111111 01111110

²Das ist eine Vereinfachung. Bei HDLC können einige Header-Felder unterschiedliche Längen haben.

³Obwohl dies eine übliche Konvention ist, handelt es sich auch hier um eine vereinfachende Annahme.

c)* Modifizieren Sie den in Abbildung 2 dargestellten HDLC-Header, so dass anstelle von Bit-Stuffing Längenangaben verwendet werden können. Nehmen Sie dabei an, dass maximal $N = 255$ B Daten pro Frame übertragen werden können.

Es reicht, zwischen Flag- und Address-Feld ein neues Feld „Length“ einzufügen. Dieses habe eine Länge von 1 B und ermöglicht damit die Angabe der Werte 0, ..., 255. Das Datenfeld trägt stets ein ganzzahliges Vielfaches von 8 bit. Damit werden Daten bis zu 255 B möglich. Die übrigen Headerfelder haben (da nicht anders angegeben) feste Längen.

