Lehrstuhl für Netzarchitekturen und Netzdienste
School of Computation, Information and Technology
Technische Universität München

Grundlagen Rechnernetze und Verteilte Systeme (IN0010)

Übungsblatt 6

25. Mai – 5. Juni 2026

Hinweis: *Da wegen Pfingsten und Fronleichnam mehrere Übungsgruppen ausfallen, wird dieses Blatt vom 27. bis 29. Mai (Mi – Fr) sowie am 1. und 2. Juni (Mo / Di) in den Übungen behandelt. Die Übungsgruppen vom Mittwoch, den 3. Juni, bis Freitag, den 5. Juni, entfallen.*

Aufgabe 1 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 grad($r(x)$) = 3 bit.

b) Bestimmen Sie die Checksumme für die gegebene Nachricht.

Zunächst werden an die Nachricht grad($r(x)$) = 3 Nullen angehängt: `10101100`**`000`**. Anschließend wird durch $r(x)$ dividiert:

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

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 B[1] 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.

[1] Jumbo-Frames werden in GRNVS nicht betrachtet

Aufgabe 2 Hubs, Bridges, Switches und WLAN

Gegeben ist das Netzwerk aus Abbildung 2.1 mit verschiedenen Netzwerkkomponenten. Die Notebooks sind bereits mit dem Access Point AP assoziiert. Die ARP-Caches und Switching-Tabellen aller Geräte sind jedoch leer.

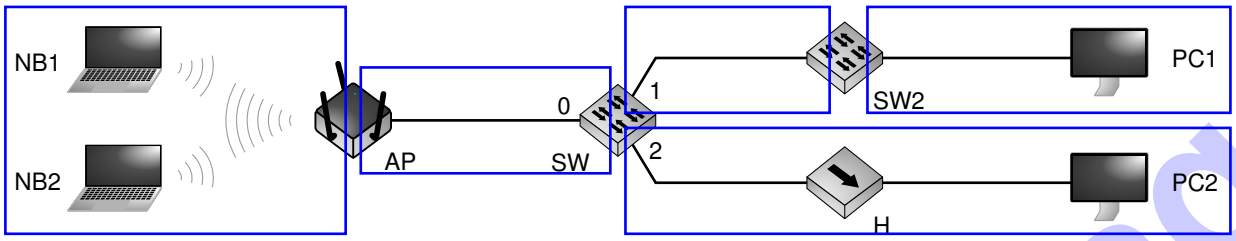


Abbildung 2.1: Topologie

a)* Markieren Sie alle Kollisionsdomänen in Abbildung 2.1. Stellen Sie sicher, dass die Interfaces eindeutig der jeweiligen Kollisionsdomäne zugeordnet sind.
WLAN nutzt mehrere Kanäle, die jeweils andere Frequenzbänder nutzen und somit eigene Kollisionsdomänen bilden, um z B. unterschiedliche Netzwerke störungsfrei voneinander zu trennen. Üblicherweise ist aber davon auszugehen, dass in einem Funknetz, das von einem Access Point aufgespannt wird, nur ein Kanal genutzt wird und es sich daher auch nur um eine Kollisionsdomäne handelt.

b)* Begründen Sie für den Switch SW, den Access Point AP und den Hub H jeweils, ob sie eine MAC-Adresse besitzen oder nicht.
Hinweis: Überlegen Sie, ob das Gerät für die Kommunikation transparent ist oder nicht.

Ein Switch ist transparent und leitet Rahmen lediglich anhand der Zieladresse weiter, wird selbst nicht adressiert und besitzt daher keine MAC-Adresse. Es gibt zwar sogenannte Managed-Switches, die zu Konfigurationszwecken auch IP- und MAC-Adressen haben; für die Grundfunktionalität des Weiterleitens auf Schicht 2 ist dies aber nicht notwendig!
Ein Access Point wird von den assoziierten WLAN-Clients immer direkt adressiert und benötigt daher eine MAC-Adresse.
Ein Hub arbeitet auf Schicht 1 und hat daher allgemein keine Kenntnis von MAC-Adressen und folglich auch keine.

NB1 möchte eine Nachricht **an PC2** senden. Nehmen Sie an, dass NB1 die MAC-Adresse von PC2 kennt.

c) Nennen Sie die Geräte, deren MAC-Adressen im Rahmen auf dem Link **von NB1 zu AP** enthalten sind. Geben Sie außerdem für jede Adresse ihre jeweilige(n) Bedeutung(en) an. Die Bedeutungen der Adressen bei WLAN sind:

- SA: Quelladresse (Source Address)
- DA: Zieladresse (Destination Address)
- TA: Senderadresse (Transmitter Address)
- RA: Empfängeradresse (Receiver Address)

Adresse/Gerät	Bedeutung(en)
NB1	SA, TA
AP	RA
PC2	DA

Der Rahmen erreicht nun den Switch SW.

d)* Begründen Sie, an welche Ports (0,1,2) der Rahmen von dem Switch weitergeleitet wird.

An alle Ports außer dem eingehenden Port, da die Switching-Tabelle leer ist.

e)* Aktualisieren Sie die Switching-Tabelle in Tabelle 2.1 so, dass sie den Zustand nach dem Durchlaufen des Rahmens durch den Switch widerspiegelt.

Teilaufgabe	Adresse/Gerät	Port
e)	NB1	0
h)	PC2	2

Tabelle 2.1: Switching-Tabelle des Switches SW

f) Nennen Sie die Geräte, deren MAC-Adressen im Rahmen auf dem Link **von SW zu PC2** enthalten sind. Geben Sie außerdem für jede Adresse ihre jeweilige(n) Bedeutung(en) an.

Adresse/Gerät	Bedeutung(en)
NB1	SA
PC2	DA

PC2 antwortet auf die Anfrage mit einem weiteren Rahmen.

g) Begründen Sie, an welche Ports (0,1,2) der Switch den Rahmen **der Antwort** weiterleitet.

An Port 0, da er weiß, dass NB1 über Port 0 erreichbar ist.

h) Aktualisieren Sie die Switching-Tabelle in Tabelle 2.1 so, dass sie den Zustand nach dem Durchlaufen des Rahmens **der Antwort** durch den Switch widerspiegelt.

Aufgabe 3 Das Address Resolution Protocol

Abbildung 3.1 zeigt ein kleines Direktverbindungsnetz mit einem PC und einem Server und den jeweiligen MAC- und IP-Adressen. Der Router R ist bei PC und Server als Default Gateway konfiguriert und ermöglicht so die Erreichbarkeit anderer Netze.
Der PC möchte nun eine Anfrage an den Server S stellen. Aus einer höheren Schicht ist dem PC die IP-Adresse des Servers bereits bekannt[2]. Gehen Sie davon aus, dass die ARP-Caches aller beteiligten Netzwerkkomponenten leer sind.

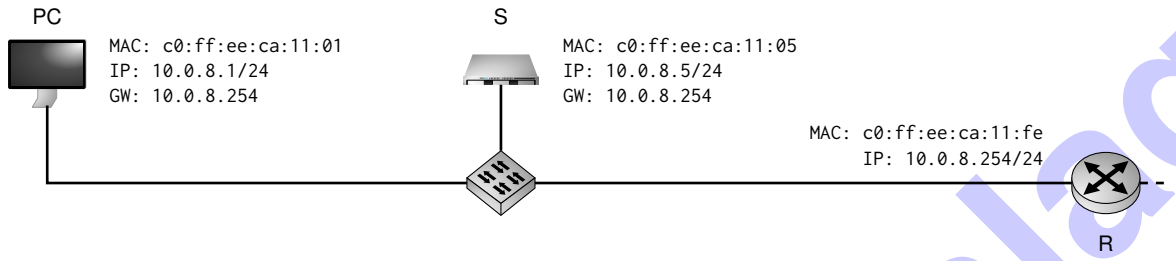


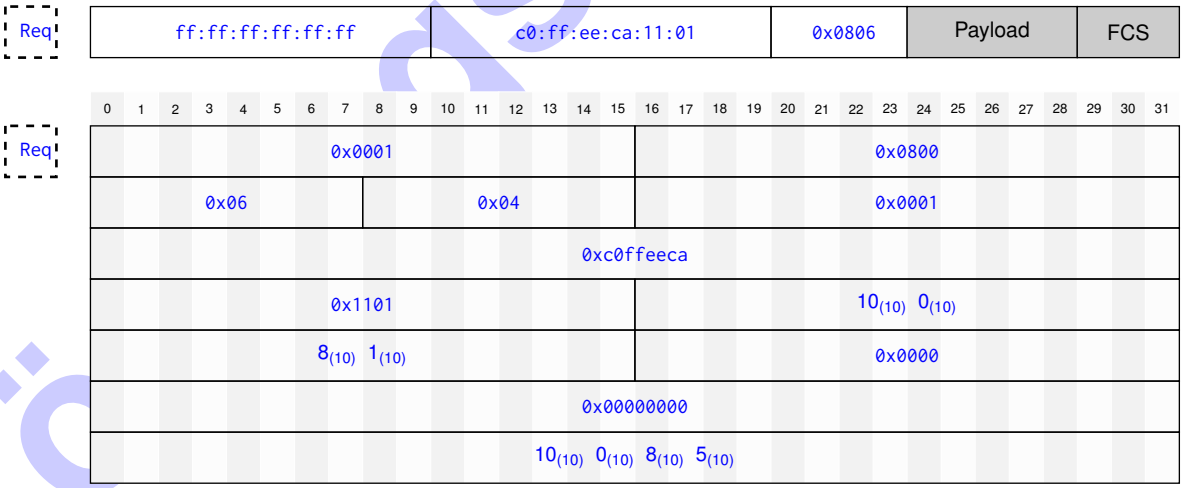
Abbildung 3.1: Netztopologie

Um das von Schicht 3 erhaltene IP-Paket weiter zu versenden, muss der PC zunächst die MAC-Adresse des Servers auflösen. Dies geschieht mittels ARP.

a)* An welche Ziel-MAC-Adresse wird der PC das ARP-Request adressieren?

> Die Ziel-MAC-Adresse ist die Broadcastadresse `ff:ff:ff:ff:ff:ff`, da das ARP-Request an alle Knoten im Direktverbindungsnetz gesendet wird.

b) Füllen Sie für das ARP-Request den Ethernet- und ARP-Header aus.
Hinweis: Es ist nicht notwendig, den Header binär auszufüllen. Achten Sie aber darauf, dass Sie die Zahlenbasis deutlich kennzeichnen, z. B. `0x10` für hexadezimal oder $63_{(10)}$ für dezimal. Die Namen der Headerfelder und einige konstante Werte sind auf dem Cheatsheet abgedruckt.



c)* Erhält Router R das ARP-Request ebenfalls, obwohl PC nur die MAC-Adresse des Servers benötigt? Wenn ja, wie wird der Router reagieren?

> Nachdem das ARP-Request per Broadcast versendet wird, wird es auch der Router erhalten. Er wird allerdings anhand der *Target Protocol Address* im Request feststellen, dass er nicht angesprochen ist und daher die Nachricht verwerfen.

[2] z.B. durch manuelle Eingabe oder durch die Auflösung eines Domain-Names über DNS; siehe später in Kapitel 5

Der Server erhält das ARP-Request und antwortet entsprechend mit einem ARP-Reply.

d) Füllen Sie für das ARP-Reply den Ethernet- und ARP-Header aus.

Rep

c0:ff:ee:ca:11:01	c0:ff:ee:ca:11:05	0x0806	Payload	FCS

Rep

0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31
0x0001		0x0800	
0x06	0x04	0x0002	
0xc0ffeeca			
0x1105		$10_{(10)}$ $0_{(10)}$	
$8_{(10)}$ $5_{(10)}$		0xc0ff	
0xeeca1101			
$10_{(10)}$ $0_{(10)}$ $8_{(10)}$ $1_{(10)}$			

e)* Angenommen, der PC möchte nun einen Server außerhalb des Direktverbindungsnetzes unter der IP-Adresse 185.86.232.164 kontaktieren. Welche Headerfelder sind nun anders im Vergleich zum ARP-Request aus Aufgabe b)?

Der PC stellt zunächst fest, ob das Ziel im gleichen Netz, also lokal erreichbar ist. Ist das – so wie hier – nicht der Fall, muss das Default Gateway kontaktiert werden, sodass die Nachricht über die Netzgrenze hinweg weitergeleitet werden kann. Daher wird der PC als *Target Protocol Address* nun die IP-Adresse seines Default Gateways, also 10.0.8.254, wählen.
Insbesondere ist es **nicht** möglich, direkt 185.86.232.164 als *Target Protocol Address* zu nutzen, da der Zielhost nun, im Gegensatz zu unserem Server, nicht mehr im lokalen Netz liegt und daher von einem ARP-Request auch nicht erreicht werden würde.

Aufgabe 4 Bit-Stuffing (Zusatzaufgabe)

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 4.1 dargestellt (vgl. Ethernet-Rahmen in den Vorlesungsfolien).

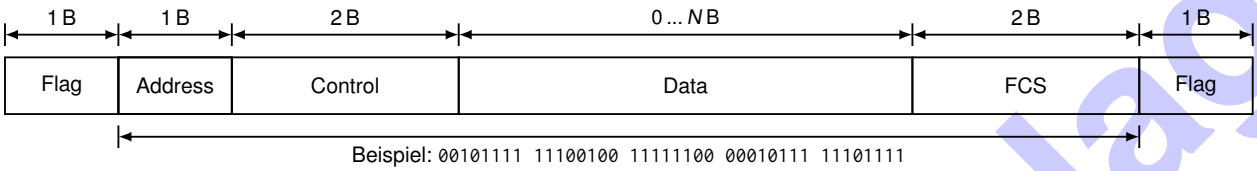


Abbildung 4.1: Aufbau eines HDLC-Rahmens

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 konstanter[3] Länge. Die Länge der Nutzdaten sei stets ein ganzzahliges[4] Vielfaches von 8 bit.

a)* In Abbildung 4.1 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. Ein möglicher Vorteil der HDLC-Version besteht darin, dass lange Eins-Folgen verhindert werden. Je nach Leitungscode 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 00101111101100100111111010000010111111010111 01111110`

[3] Das ist eine Vereinfachung. Bei HDLC können einige Header-Felder unterschiedliche Längen haben.
[4] Obwohl dies eine übliche Konvention ist, handelt es sich auch hier um eine vereinfachende Annahme.

c)* Modifizieren Sie den in Abbildung 4.1 dargestellten HDLC-Header, so dass anstelle von Bit-Stuffing Längenangaben verwendet werden können. Nehmen Sie dabei an, dass maximal $N = 255\,\mathrm{B}$ 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.