Kapitel 2

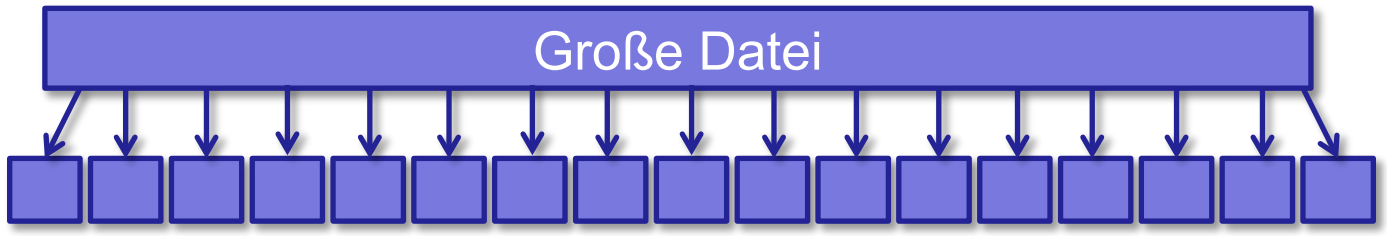
Pakete, Rahmen, Fehlererkennung

2. Pakete, Rahmen, Fehlererkennung –
2.1 Einleitung

- Dieses Kapitel
  - beschreibt eine fundamentale Idee in Netzwerken (NW)
  - das Konzept von Paketen
  - erklärt, wie sich Sender und Empfänger koordinieren um ein Paket zu übertragen
  - zeigt, wie Pakete im Ethernet-NW definiert sind
  - diskutiert Mechanismen um Übertragungsfehler entdecken zu können

2.2 Das Konzept der Pakete

- Fast alle NWe senden Daten nicht als durchgängigen Strom von Bits
  - NWe zerteilen Daten in kleine Blöcke, die ***Pakete***



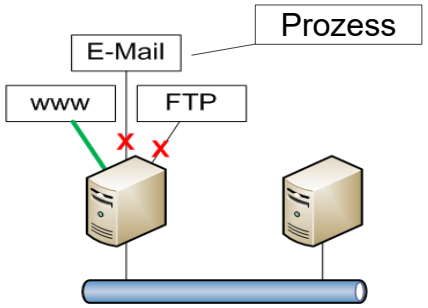
  - Pakete werden unabhängig voneinander übertragen

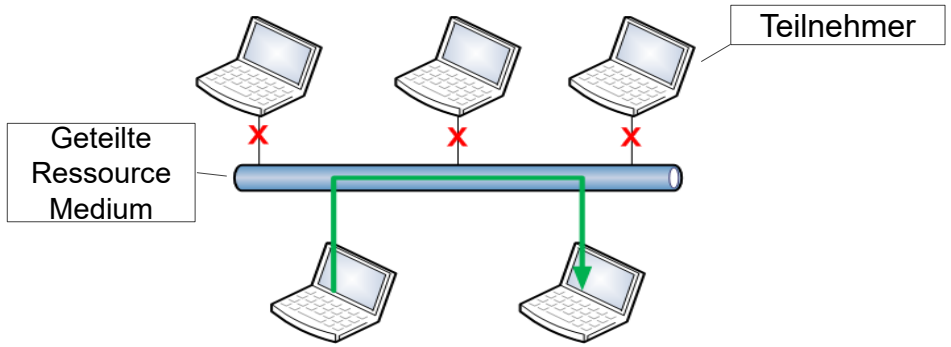
- Computer NWe werden oft Paket-NW oder paketvermittelnde (packet-switched) NW genannt, da sie auf Paket basierter Technologie beruhen.

2.2 Das Konzept der Pakete

Drei Gründe motivieren den Einsatz von Paketen:

1. Sender und Empfänger müssen sich koordinieren um Daten korrekt zu übertragen
   - kommt ein Paket fehlerhaft an, so muss nur dieses neu übertragen werden und nicht die komplette Datei
2. Faire Ressourcenverteilung
   - mehrere Prozesse/Computer möchten Datei senden/empfangen

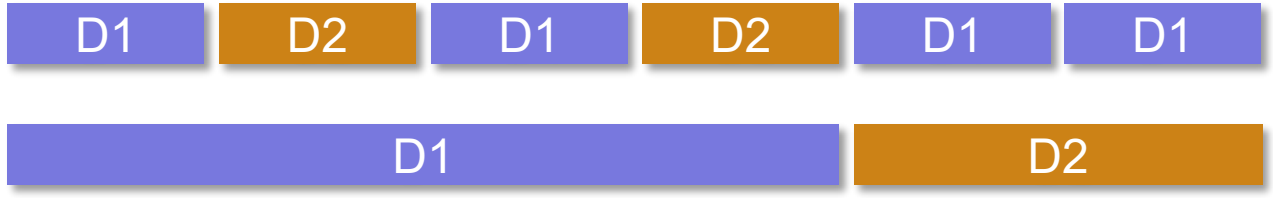




   - Würde Datei am Stück übertragen, müssten andere Teilnehmer inakzeptabel lange warten

2.2 Das Konzept der Pakete

- Wie lange dauert ein Filetransfer?
- Filegröße N = 200MB, R = 1Mbps
  → Dauer = 27 Minuten
  - Exklusiver Zugriff aufs NW würde andere Prozesse unerträglich lange blockieren
- Zum Vergleich: Rechner zerlegt Daten in Pakete zu je 1000 Bytes
  → Dauer/Paket = 8 Millisekunden
  - Nach 8 Millisekunden darf ein Anderer senden
  - Ist kein Anderer da, dauert Filetransfer auch nur ca. 27 Minuten
- Beispiel: 2 Dateien, eine sehr groß
  - Übertragung der kleinen Datei dauert nicht unverhältnismäßig lange (Fairness, Responsivität, Interaktivität)



mit Paketen --> kleine Datei D2 wird “sofort” übertragen

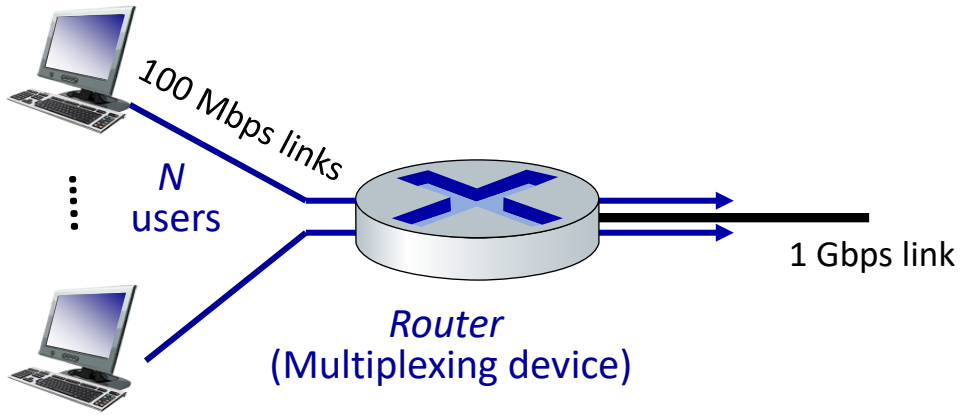
ohne Pakete → kleine Datei D2 muss evtl. lange Warten, bis große Datei fertig ist

2.2 Das Konzept der Pakete

➢ Fortsetzung zu: Drei Gründe motivieren den Einsatz von Paketen

3. Effiziente Nutzung der Übertragungsressource

- Nicht jeder sendet ständig Daten
- Auch geringe Leerzeiten eines Senders können von anderen Sendern genutzt werden
- Höhere Auslastung der gemeinsamen Ressource möglich, mehr Teilnehmer möglich
- Geringere Kosten für den Einzelnen



Beispiel:

- 1 Gb/s Link
- Jeder Nutzer:
  - 100 Mb/s falls "aktiv"
  - aktiv 10% der Zeit
- *Circuit-switching (ohne Pakete, ala klassisches Telefon):* 10 Nutzer
- *Packet switching*: mit 35 Nutzer, Wahrscheinlichkeit > 10 gleichzeitig aktive ist kleiner als 0.0004 *
- *Q*: Wie kommt man auf 0.0004?
- *Q*: Was passiert bei > 35 Nutzer?

* Mehre interaktive Aufgaben online auf:
http://gaia.cs.umass.edu/kurose_ross/interactive

2.4 Pakete und Frames

- Aber: Empfänger muss Start und Ende eines Datenblocks erkennen können
  - Sender versieht Datenblock mit einem „Rahmen“
  - Frame (eng.) bedeutet Rahmen
  - Hier: Frame = Header + Datenblock + Trailer
    - ***Header*** (= Kopf): kennzeichnet Start eines Datenblocks
    - ***Trailer*** (= Anhänger): kennzeichnet Ende eines Datenblocks

  - Rahmenformat hängt von der Technologie ab

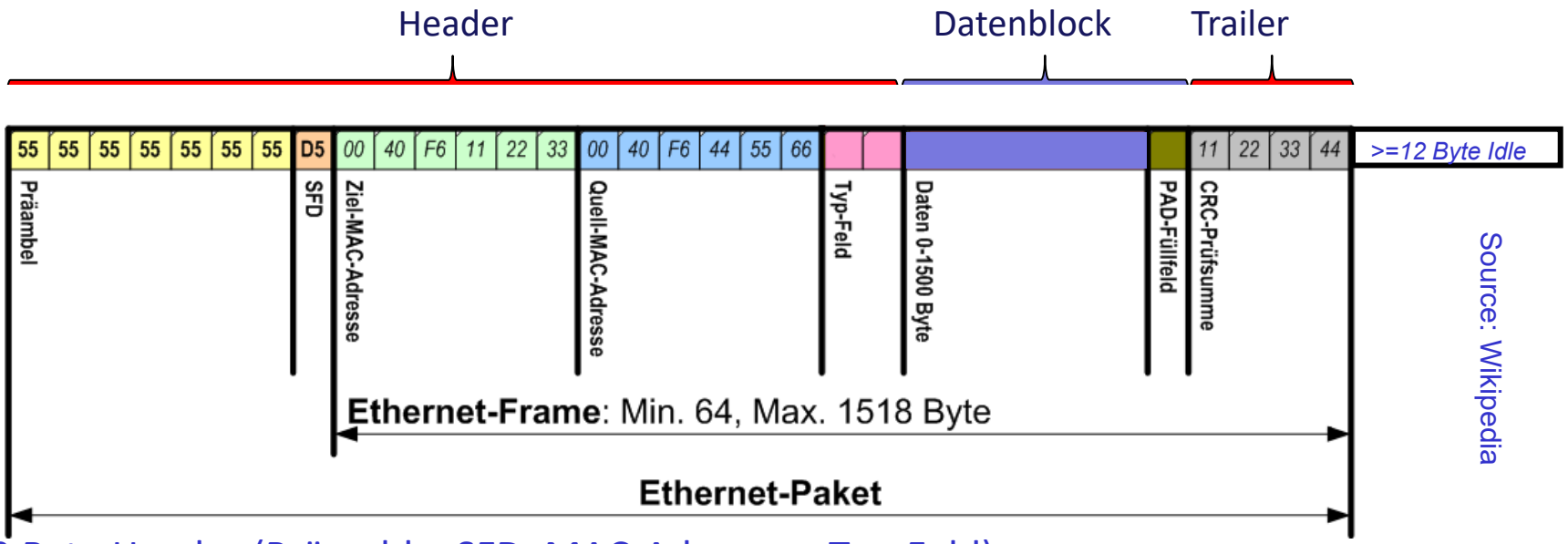
Terminologie:
Gebrauch der Begriffe Frame und Paket sind schicht- und technologie-abhängig.

Im Allgemeinen:
Frame = Header + Nutzdaten + Trailer
Paket = Header + Nutzdaten (ohne Trailer)
Datagramm = Paket, das unzuverlässig übertragen wird
Segment = Paket, das bei Fehler wiederholt übertragen wird

2.4 Pakete und Frames
Beispiel: Ethernet

später mehr

➢ Beispiel: IEEE 802.3 (Ethernet-II Typ) – Paket bzw. Frame



– 22 Byte Header (Präamble, SFD, MAC-Adressen, Typ-Feld)
– 0-1500 Byte Daten inkl. Auffüllzeichen (“PAD”)
– 4 (+12) Byte Trailer (CRC, Idle)

Bem: Begriffe sind oft kontextabhängig. So ist ein Ethernet-Frame nicht exakt ein Ethernet-Paket und ein Ethernet-Paket hat einen Trailer. Für uns ist jetzt nur die „Idee” wichtig und nicht die exakte Terminologie.

Einschub: Effizienz

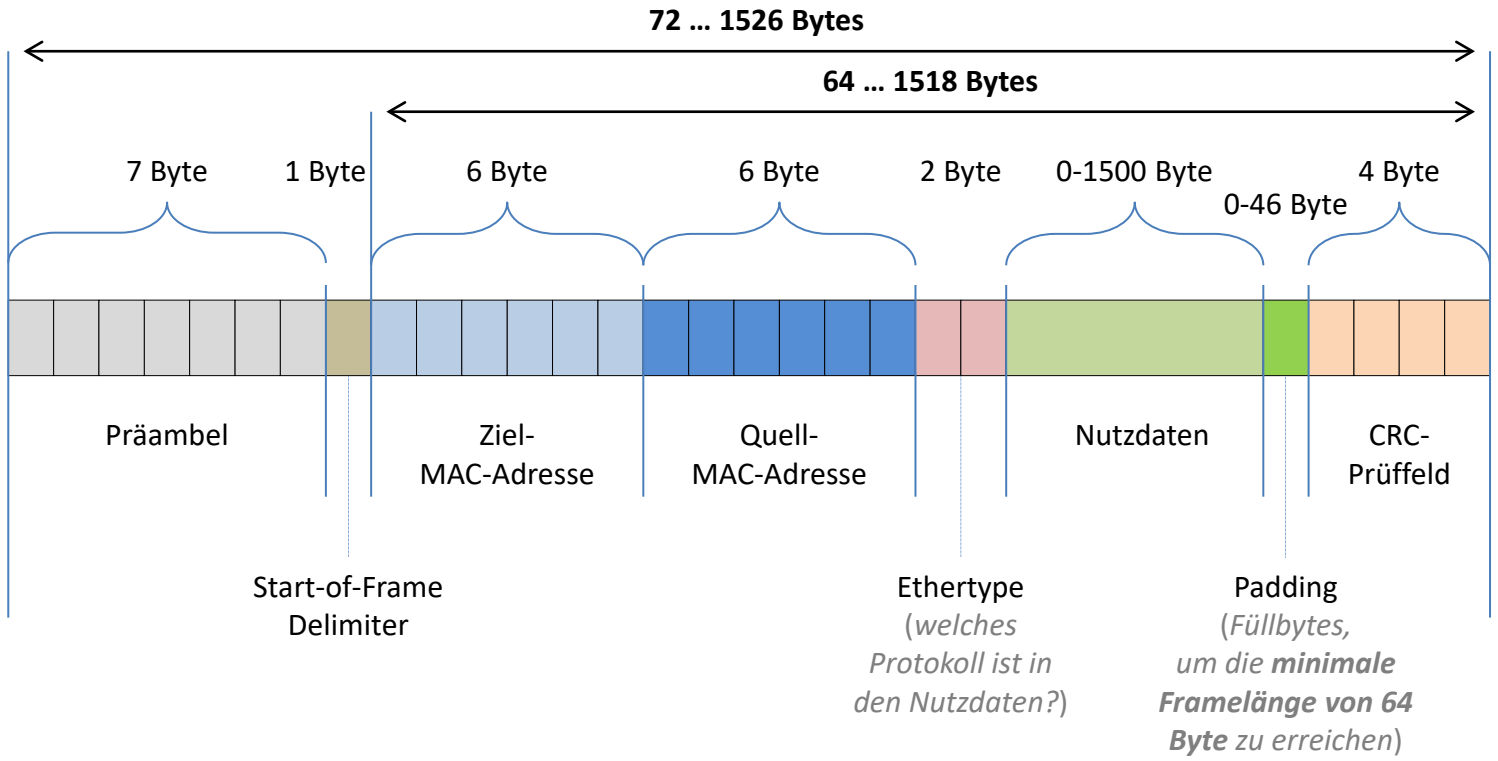
- Ziel eines Netzwerks: Übertragung von Nutzdaten.
- Header (und Trailer) sind notwendig, damit das Netzwerk funktioniert, zählen aber nicht zu den unmittelbaren Nutzdaten.
- Headerdaten sind somit ein gewisser (notwendiger) Overhead und verringern damit die *Effizienz* des Netzwerks.

- ***Effizienz*:** Zu welchem Anteil wird die verfügbare Kapazität des Netzwerks zur Übertragung der Eigentlichen Nutzdaten genutzt?

$$\eta = \frac{N_{nutz}}{N_{total}}$$

Einschub: Effizienz

➢ Beispiel: IEEE 802.3 Ethernet-II



- 18 (+8) Byte Header und Trailer
- 12 Byte IDLE (Interframe Gap)
- 0-1500 Byte Nutzdaten

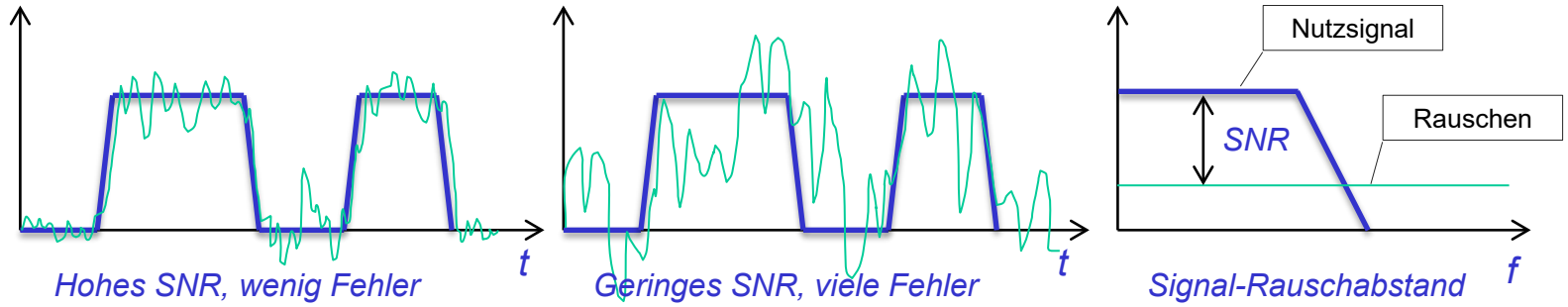
$$\eta_{max} = \frac{1500}{1526+12} = 97{,}53\%$$

Maximale Effizienz

Q: Was ist die minimale Effizienz?

2.5 Übertragungsfehler

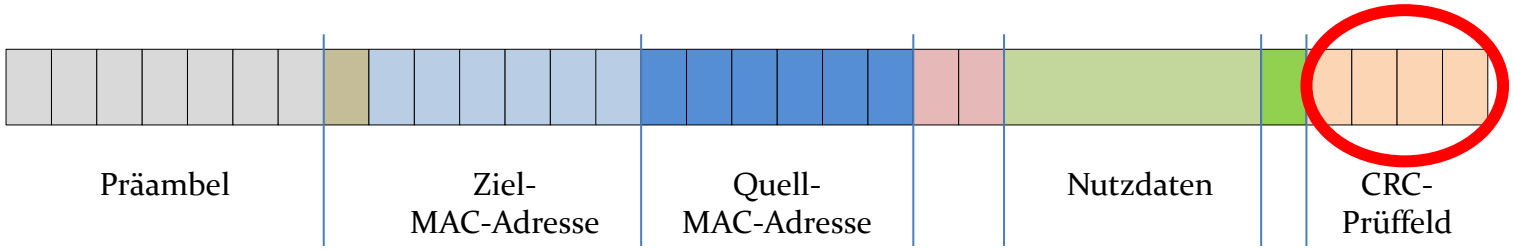
- Elektromagnetische Störungen können ungewollte elektrische Ströme erzeugen
- **Rauschen**: „Hintergrundstörungen“ auf dem Medium, welche nicht zum Nutzsignal gehören
  → *Signal-Rauschabstand, SNR*
  (vgl. Lärm in einem Raum)



- Störungen und Rauschen können zu einer Fehlinterpretation eines oder mehrer Bits im Empfänger führen.
- Interferenzen
  - können sogar dazu führen, dass ein Empfänger scheinbar Daten empfängt obwohl gar keine gesendet werden.

2.6 Fehlererkennung

- Benötigen Mechanismen um Übertragungsfehler zu erkennen
- Es gibt mehrere Verfahren:
  - Parity Bits
  - Checksums
  - CRC: Cyclic Redundancy Checks
  - …
- Grundsätzliche Idee:
  - Sender fügt dem Rahmen redundante Information zu, die die Daten charakterisieren
  - Empfänger überprüft, ob die zusätzliche Info noch mit den Daten übereinstimmt.
- Beispiel: CRC bei Ethernet



2.6.0 Fehlererkennung bei Menschen

- Auch die menschliche Sprache/Schrift hat viele redundante Informationen, die der Fehlererkennung und –behebung dienen:

Können sie das lesen?

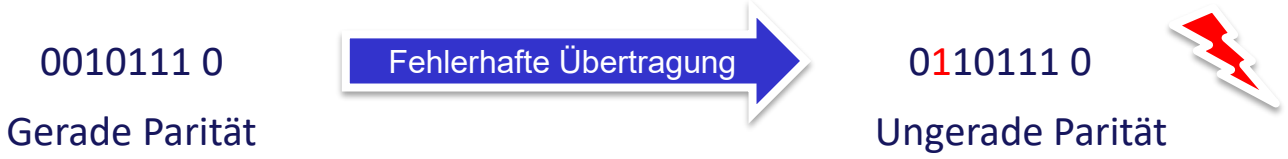
Den Frsochren zfuloge speilt es für das Gihern kiene Rlole wie die Bachustben angoerdnet snid, salnoge der esrte und der ltezte Bachustbe an der rihctigen Stlele shteen.

2.6.1 Parity Check

- redundante Information: 1 Bit
- zwei Paritätsformen
  - gerade oder ungerade (even or odd)
  - müssen beim Sender und Empfänger übereinstimmen
- Das Paritätsbit wird so gesetzt, dass die gewählte Paritätsform erhalten wird

Datum	Even Parity Bit (Anzahl der 1er Bits ist gerade)	Odd Parity Bit (Anzahl der 1er Bits ist ungerade)
0010111	0	1
1011101	1	0
0000001	1	0

- Wenn ein Bitfehler auftritt, stimmt die Paritätsform nicht mehr → Empfänger erkennt Übertragungsfehler

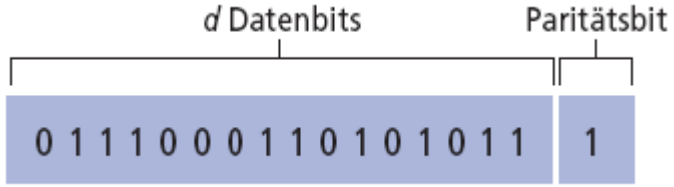


2.6.1 Parity Check

Ein-Bit-Parität:

–Erkennt Ein-Bit-Fehler

–Versagt bei Zwei-Bit-Fehler

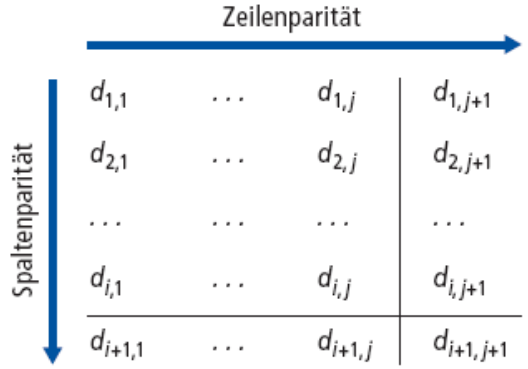


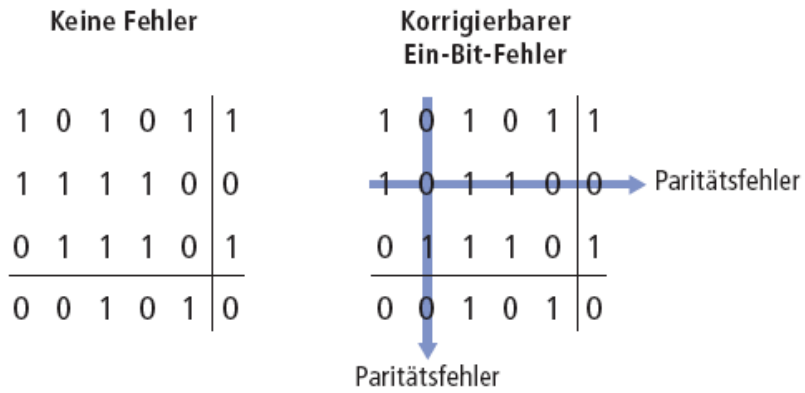
Gerade Parität!

Interaktive Übungen unter:
http://gaia.cs.umass.edu/kurose_ross/interactive/

Zweidimensionale Parität:

- Erkennt und **korrigiert** Ein-Bit-Fehler





Aus: Kurose

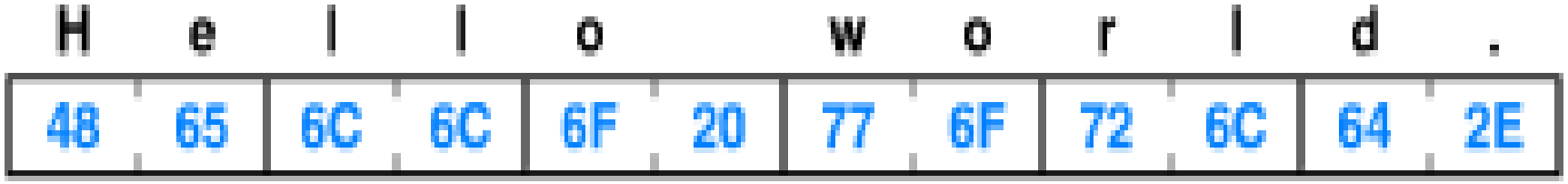
2.6.1 Parity Check

- Alternativen zur Fehlererkennung unterscheiden sich in 3 Weisen:
  - Die Größe der zusätzlichen Informationen
  - Der Berechnungsaufwand
  - Anzahl der Bitfehler, die entdeckt werden

2.6.2 Checksum

- Wird in vielen NW verwendet (z.B. IP, UDP, TCP)
- Um die Prüfsumme (=checksum) zu berechnen,
  - behandle die Daten als eine Sequenz von binären Ganzzahlen und berechne deren Summe
  - in Wirklichkeit können die Daten Bilder, Musik etc. sein
- Figure 7.6 zeigt eine 16-bit Checksum-Berechnung
- Der Sender
  - betrachtet jedes Zeichenpaar als eine 16-bit Ganzzahl
  - und berechnet die Summe
    - falls die Summer größer als 16 Bits wird, wird der Übertrag zur endgültigen Summe addiert.

2.6.2 Checksumme



4865 + 6C6C + 6F20 + 776F + 726C + 642E + carry = 71FC

Aus: Comer

Figure 7.6 An example 16-bit checksum computation for a string of 12 ASCII characters. Characters are grouped into 16-bit quantities, added together using 16-bit arithmetic, and the carry bits are added to the result.

2.6.2 Checksumme

- Hauptvorteil ist die kleine Größe und die einfache Berechnung
  - viele NW verwenden eine 16-Bit oder 32-bit Checksum
  - es wird eine einzige Prüfsumme für ein ganzes Packet berechnet → wenig Overhead
- Nachteilig ist, dass nicht alle üblichen Fehler entdeckt werden
  - Beispiel:

H	**e**	**l**	**l**	**o**		**w**	**o**	**r**	**l**	**d**	**.**	Checksum	
48h	65h	6Ch	6Ch	6Fh	20h	77h	6Fh	72h	6Ch	64h	2Eh	71	FC

Übertragung mit ↓ 2 Bitänderungen

I	**e**	**l**	**l**	**o**		**v**	**o**	**r**	**l**	**d**	**.**	Checksum	
49h	65h	6Ch	6Ch	6Fh	20h	**76h**	6Fh	72h	6Ch	64h	2Eh	71	FC

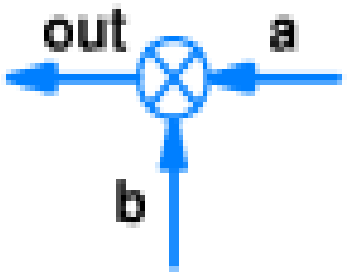
- Gleiche Checksumme
- wird vom Empfänger als fehlerfrei akzeptiert!

2.6.3 CRC

- Wie kann ein NW mehr Fehler finden ohne den zusätzlich benötigten Platzbedarf in einem Paket zu erhöhen?
  - die Antwort liegt in der CRC (Cyclic Redundancy check)-Technik
  - Findet breiten Einsatz in physikalischen Netzwerken (z.B. Ethernet, WLAN, ATM)

- Figure 7.8 zeigt einige HW Komponenten
- Figure 7.9 illustriert, wie sich Bits während einer Schiebeoperation bewegen und wie sich die Ausgabe ändert

2.6.3 CRC



(a)

a	b	out
0	0	0
0	1	1
1	0	1
1	1	0

(b)

Figure 7.8 (a) A diagram of hardware that computes an *exclusive or*, and (b) the output value for each of the four combinations of input values. Such hardware units are used to calculate a CRC.

2.6.3 CRC

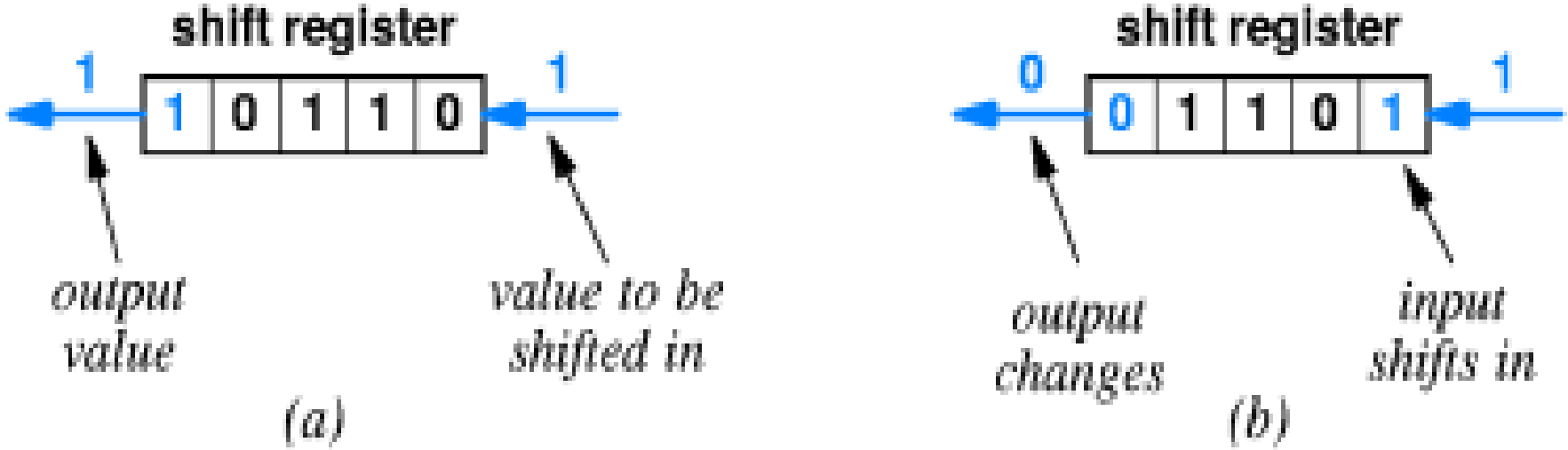


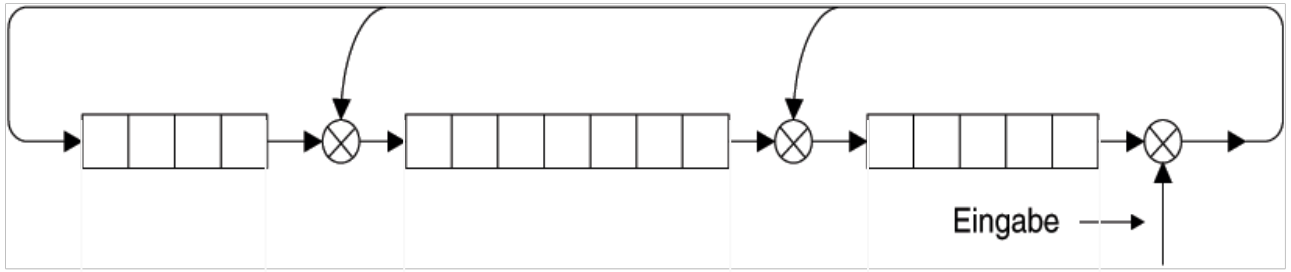
Figure 7.9 A shift register (a) before and (b) after a shift operation. During a shift, each bit moves left one position, and the output becomes equal to the leftmost bit.

2.6.3 CRC

- Figure 7.10 illustriert die Berechnung einer 16-bit CRC
- Es gibt mehrere CRC Algorithmen
  - Unterscheiden sich in der Anzahl und Bit-Größe der Schieberegister und der Initialisierungswerte
- Anstatt die CRC eines empfangenen Paketes zu berechnen,
  - berechnet man die CRC über das empfangene Paket und die empfangene CRC
  - Falls alle Bits korrekt empfangen wurden, wird der berechnete Wert Null sein.

2.6.3 CRC

Figure 7.10 Diagramm einer Hardware zur CRC-Berechnung. Nachdem die Bits einer Nachricht in die Einheit geschoben wurden, enthalten die Versatzregister das 16-Bit-CRC der Nachricht.



Beispiel: Rahmen 11011 wird eingegeben. Folgender Zeitverlauf ergibt sich

1)		0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	Inhalt der Schiebereg nach Initialisierung
2)	(1)					(1)								(1)						Input 1 (am Schieberegister anliegender Wert)
3)		1	0	0	0		1	0	0	0	0	0	0		1	0	0	0	0	Inhalt der Schiebereg. nach Schift-Operation
4)	(1)					(1)								(1)						Input 1
5)		1	1	0	0		1	1	0	0	0	0	0		1	1	0	0	0	Inhalt der Schiebereg. nach Shift-Operation
6)	(0)					(0)								(0)						Input 0
7)		0	1	1	0		0	1	1	0	0	0	0		0	1	1	0	0	Inhalt der Schiebereg. nach Shift-Operation
8)	(1)					(1)								(1)						Input 1
9)		1	0	1	1		1	0	1	1	0	0	0		1	0	1	1	0	Inhalt der Schiebereg. nach Shift-Operation
10)	(1)					**(0)**								(1)						Input 1
11)		1	1	0	1		0	1	0	1	1	0	0		1	1	0	1	1	=CRC

2.6.3 CRC

- Zwei Gründe warum CRC mehr Fehler findet als eine Prüfsumme
  1. Ein Input-Bit bewegt sich durch alle drei Register,
     - ein einzelnes Bit des Pakets beeinflusst den CRC-Wert dramatisch
  2. Rückkopplung, die Ausgabe des rechten Schieberegisters beeinflusst jede XOR-Einheit
     - der Effekt eines einzelnen Bits wandert durch die Schieberegister nicht nur einmal
- Mathematisch wird die CRC durch eine Polynomdivision berechnet

2.6 Fehlererkennung

Figure 7.11 A modification of the frame format from Figure 7.3 that includes a *16*-bit CRC.

2.7 Fehlerbehebung

Frage: Empfänger erkennt Fehler. Wie kann er behoben werden?

- Zwei Möglichkeiten
  1. es werden noch mehr redundante Informationen gesendet anhand der Empfänger den Fehler beheben kann (siehe z.B. zweidimensionale Parität)
  2. Der Empfänger fordert das fehlerhafte Paket nochmals vom Sender an
- Für geringe BER ist die zweite Möglichkeit effizienter
  - nur im unwahrscheinlichen Fehlerfall werden Fehlerkorrekturdaten übertragen

2.7 Fehlerbehebung

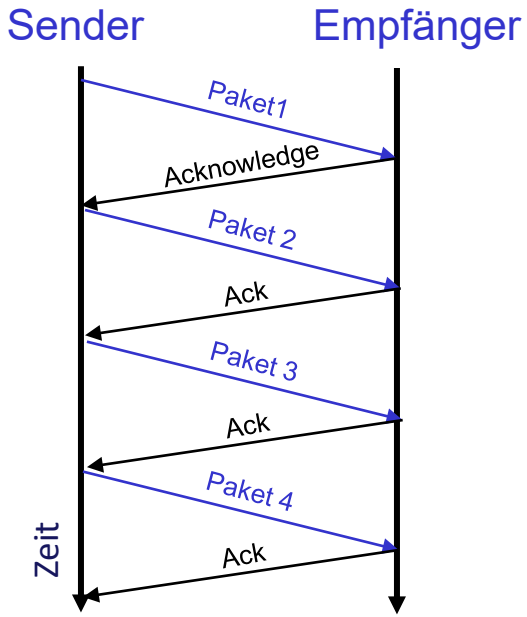
- Es gibt zwei Anforderungsverfahren
  1. Empfänger schickt speziellen Anforderungsrahmen an den Sender
  2. Empfänger bestätigt dem Sender alle fehlerfrei empfangenen Pakete. Erhält der Sender keine Bestätigung, so sendet er das letzte Paket nach einiger Zeit nochmals

2.7 Fehlerbehebung

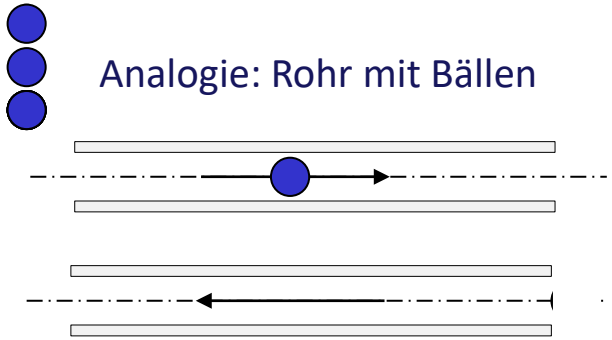
- Meist wird Verfahren 2 verwendet
  - beinhaltet Flusskontrolle (schneller Sender kann langsamen Empfänger nicht mit Daten überfluten)
  - Empfänger sendet eh meistens Daten an den Sender zurück. Die Bestätigung kann „*huckepack*“ mit gesendet werden.
- Mehrere Unterverfahren
  - Stop and Wait
    - Sender darf weiteres Paket nur senden, falls vorhergehendes Paket bestätigt wurde
  - Sliding Window
    - Sender darf n Pakete senden bevor er auf eine Bestätigung warten muss (n = Window size)

2.7 Fehlerbehandlung

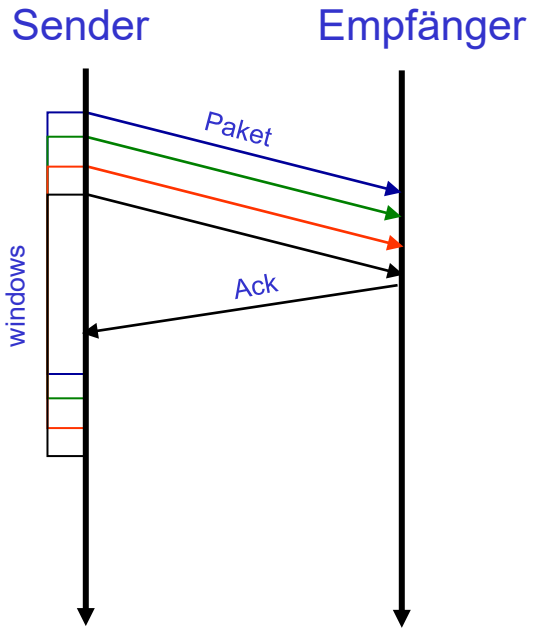
Stop and Wait



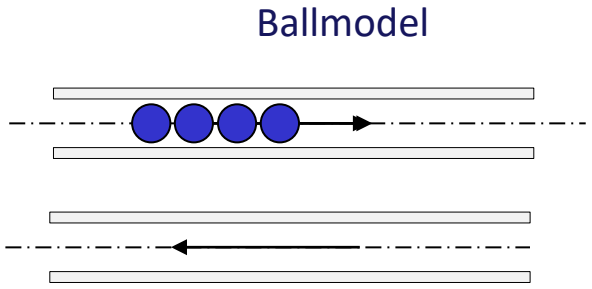
- einfache "Buchhaltung“
- geringe Netto-Datenrate, wenn viele Rahmen in die Leitung „passen“



Sliding Window



- komplizierte "Buchhaltung"
- hohe Netto-Datenrate



Ausblick

In diesem Kapitel:

- Strukturierung von Daten in Pakete und Frames
- Möglichkeiten der Fehlererkennung
- Möglichkeiten zur Fehlerbehandlung

Nächster Schritt: Wie funktionieren lokale Netzwerke (LANs) mit mehreren Teilnehmern?

- Wie erfolgt der Zugriff auf die gemeinsam genutzten Ressourcen?
- Welche Netzwerktopologien gibt es?
- Welche Netzwerktechnologien gibt es?