

Netztechnik I

T2INF4201.1

Sicherungsschicht (Data Link Layer)

Markus Götzl
Dipl.-Inform. (FH)
mail@markusgoetzl.de

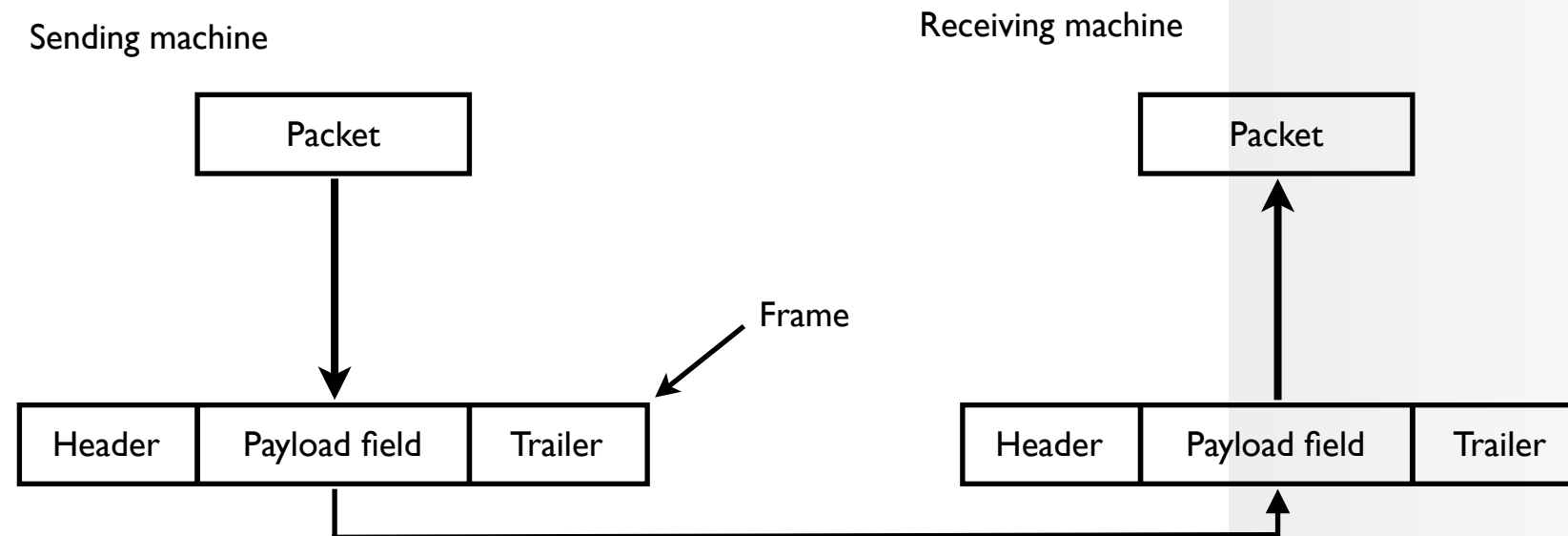
Sicherungsschicht (Data Link Layer)

- ▶ Aufgaben der Sicherungsschicht
 - Framing
- ▶ Fehlererkennung und Fehlerkorrektur
- ▶ Media Access Control (MAC)
- ▶ Ethernet Frame
- ▶ Data Link Layer Switching
 - Virtual LANs

Aufgaben der Sicherungsschicht

- ▶ In der Sicherungsschicht werden folgende Ziele verfolgt:
 - Bereitstellen einer definierten Schnittstelle zur Vermittlungsschicht (Network Layer)
 - Behandlung von Übertragungsfehlern
 - Regulierung des Datenflusses

Datenframes (Kapselung von Paketen aus höheren Schichten):



Computer Networks, Fifth Edition by Andrew Tanenbaum and David Wetherall, © Pearson Education-Prentice Hall, 2011

Grundlegende Dienste der Sicherungsschicht

- ▶ Unbestätigter verbindungsloser Dienst
- ▶ Bestätigter verbindungsloser Dienst
- ▶ Bestätigter verbindungsorientierter Dienst

Grundlegende Dienste der Sicherungsschicht

- ▶ Unbestätigter verbindungsloser Dienst
 - Es wird keine logische Verbindung aufgebaut.
 - Frames werden nicht bestätigt, damit ist ein Verlust eines oder mehrerer Frames nicht feststellbar bzw. korrigierbar.
 - Anwendung: Echtzeitsdienste (z.B. Sprache) über Verbindungskanäle mit kleinen Fehlerraten, wobei die Auswirkungen einer verzögerten Übermittlung gravierender sind, als der Verlust von Informationen.
 - Beispiel: Ethernet, logische Verbindungen und Fehlerkorrektur wird höheren Schichten überlassen.

Grundlegende Dienste der Sicherungsschicht

- ▶ Bestätigter verbindungsloser Dienst
 - Es wird keine logische Verbindung aufgebaut.
 - Jeder gesendete Frame wird einzeln bestätigt. Damit garantiert die Sicherungsschicht, dass jeder gesendete Frame auch empfangen wird.
 - Anwendung: Übertragungen auf unzuverlässigen Verbindungskanälen (z.B. leiterungebundene Übertragungsmedien)
 - Beispiel: 802.11 (WLAN)
 - **Anmerkung:** Empfangsbestätigung (bestätigte Dienste) in höheren Schichten kann u.U. sehr ineffizient sein, da höhere Schichten keine Kenntnis über Parameter - etwa der Framegröße - der Sicherungsschicht haben.
Beispiel: ein Paket wird zur Übertragung, aufgrund seiner Größe, in 10 Frames aufgeteilt. Gehen nur 2 Frames verloren, müssen, um das gesamte Paket zu erhalten, alle 10 Frames neu übertragen werden.

Grundlegende Dienste der Sicherungsschicht

- ▶ Bestätigter verbindungsorientierter Dienst
 - bevor Daten übertragen werden wird eine Verbindung aufgebaut. Jeder Frame der über die Verbindung gesendet wird ist nummeriert. Die Sicherungsschicht garantiert, dass jeder gesendete Frame genau einmal empfangen wird. Zusätzlich garantiert die Sicherungsschicht, dass alle Frames in der richtigen Reihenfolge empfangen werden.
 - Ein bestätigter verbindungsorientierter Dienst auf der Sicherungsschicht versorgt damit die Vermittlungsschicht mit einem zuverlässigem *“Bit-Stream”*
 - Anwendung: Satelliten Verbindungen oder Weitverkehrs-Telefonverbindungen.
 - Ein verbindungsorientierter Dienst durchläuft drei Phasen:
 1. Verbindungsaufbau: Auf beiden Seiten werden alle Variablen (Zähler etc.), welche benötigt werden um zu bestimmen welcher Frame empfangen wurde und welcher nicht, initialisiert.
 2. Übertragung: Frames werden übermittelt.
 3. Verbindungsabbau: Auf beiden Seiten werden alle Variablen (Zähler, etc.), welche für die Verbindung benötigt wurden, freigegeben.

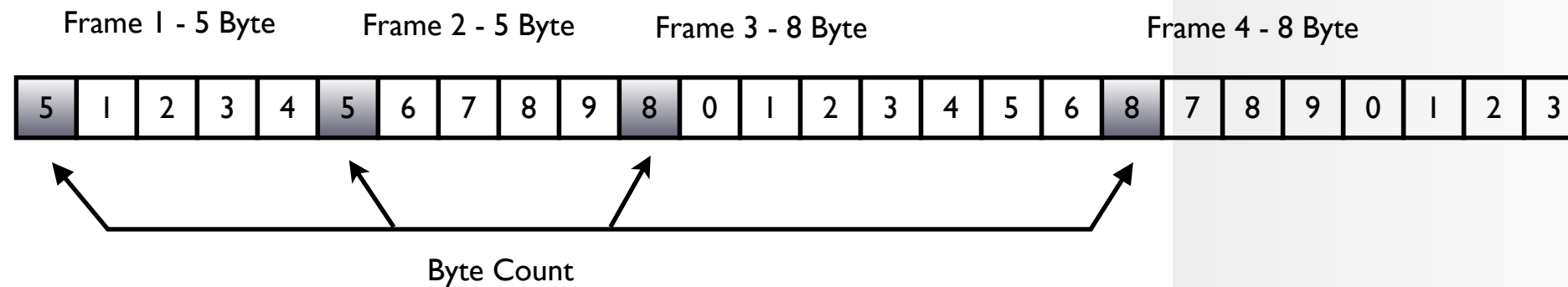
Umsetzung der Frames (Framing):

- ▶ Die Bitübertragungsschicht akzeptiert bzw. liefert einen Folge von Bits (Bit Stream). Die Bitübertragungsschicht fügt u.U. Redundanz zum Bit Stream hinzu, um Übertragungsfehler zu innerhalb akzeptabler Werte zu halten.
- ▶ Es ist aber die Aufgabe der Sicherungsschicht Übertragungsfehler zu erkennen und zu korrigieren.
- ▶ Der übliche Weg das zu bewerkstelligen ist, den Bit Stream in einzelne Frames aufzuteilen und für jeden Frame eine Prüfsumme (checksum) zu errechnen, welche mit dem Frame übertragen wird.
- ▶ Die Prüfsumme kann beim Empfänger neu berechnet werden und mit der gesendeten verglichen werden.

Umsetzung der Frames (Framing):

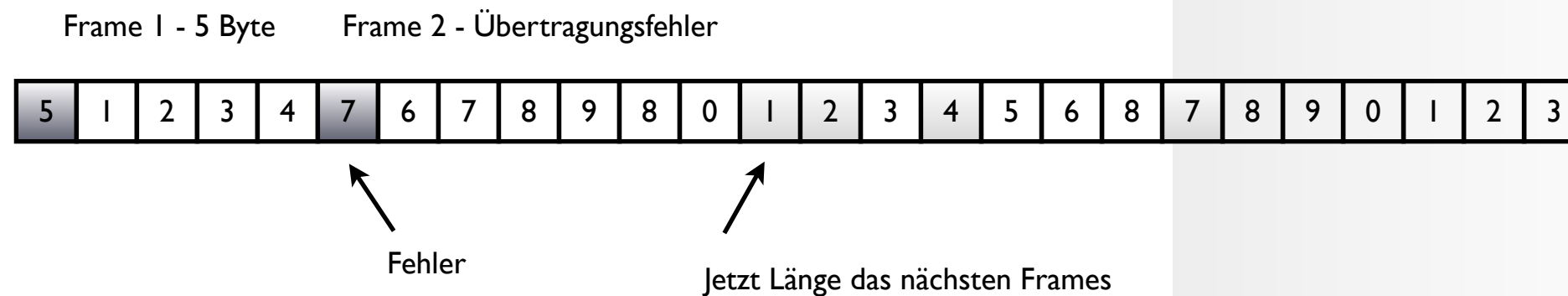
- ▶ Zentrale Aufgabe:
 - Erkennung, wo ein Frame im Bitstrom anfängt und wo er aufhört
 - Framegrenzen müssen im Bitstrom erkennbar sein
- ▶ Verfahren zur Umsetzung von Frames:
 - Byte Count Methode
 - FLAG Byte Methode
 - Byte Stuffing
 - Bit Stuffing
 - Framing durch Codierung

Byte Count Methode:



- In einem speziellen Feld wird die Länge des Frames angegeben. Das Feld ist selbst Teil der Längenangabe.

Byte Count Methode:



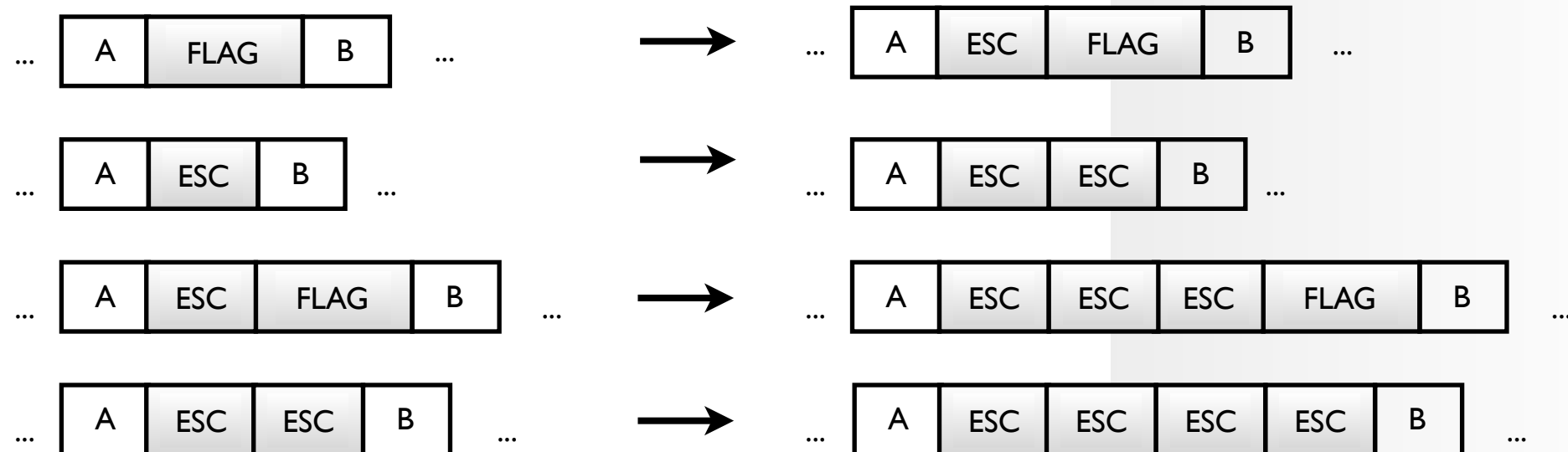
- Bei einem Übertragungsfehler im Byte-Count verliert die gesamte Übertragung die Synchronisation.

FLAG Byte Methode



- ▶ Start und Ende eines Frames werden mit einem FLAG-Byte markiert. Bei Synchronisationsverlust kann die Synchronisation durch Suchen des FLAG-Bytes wieder hergestellt werden.
- ▶ Weiteres Problem: Das FLAG Byte könnte in den Daten enthalten sein. Dies ist bei Binären Daten (Bilder, Musik etc.) nicht auszuschließen.

Byte Stuffing



- ▶ Um zu verhindern das ein FLAG Byte im den Daten die Synchronisation gefährdet wird ein ESC-Byte (Escape) vor das FLAG Byte eingefügt. Damit kann beim Empfänger ein “echtes” FLAG Byte (ohne ESC Byte) von einem “falschen” unterschieden werden.
- ▶ Die ESC-Bytes müssen aus dem Daten entfernt werden bevor sie die Vermittlungsschicht übernimmt.
- ▶ Auftreten von ESC-Bytes in den Daten wird ebenfalls durch ein ESC-Byte gelöst. Hier sind unterschiedliche Kombinationen denkbar. Wichtig ist, dass der original Bit Stream wieder hergestellt werden kann

Bit Stuffing

original Datenstrom: 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

Bit Stuffing : 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0



rekonstruiert : 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

- ▶ Analog zum Byte Stuffing kann die Markierung der Framegrenzen auch auf Bit-Ebene maskiert werden.
- ▶ Hierzu wird ein bestimmtes Bit-Muster zur Markierung der Framegrenzen benutzt (z.B. 01111110,- sechs Einsen!).
- ▶ Sobald im Datenstrom fünf aufeinanderfolgende '1' gefunden werden wird eine '0' eingefügt:
 - '01111110' wird dann als '01111101' übertragen.
- ▶ Beim Empfänger wird die '0' dann wieder aus dem Datenstrom entfernt, sobald fünf mal die '1' und eine '0' gefunden wird: '01111101' -> '01111110'

Framing durch Codierung

- ▶ Bei dieser Methode wird der Start und das Ende eines Frames durch spezielle Codeworte markiert, die im in den Daten nicht vorkommen.
- ▶ Beispiel: 4B/5B Leitungscodierung.
 - Vier Nutzdatenbits werden auf fünf Codebits abgebildet.
 - Mit fünf Bits sind 32 Kodierungen möglich
 - Nur 16 Kodierungen werden für Daten verwendet (0-9 und A-F)
 - Die Übrigen 16 Kodierungen werden für Steuerzwecke verwendet u.a. zur Begrenzung der Frames.

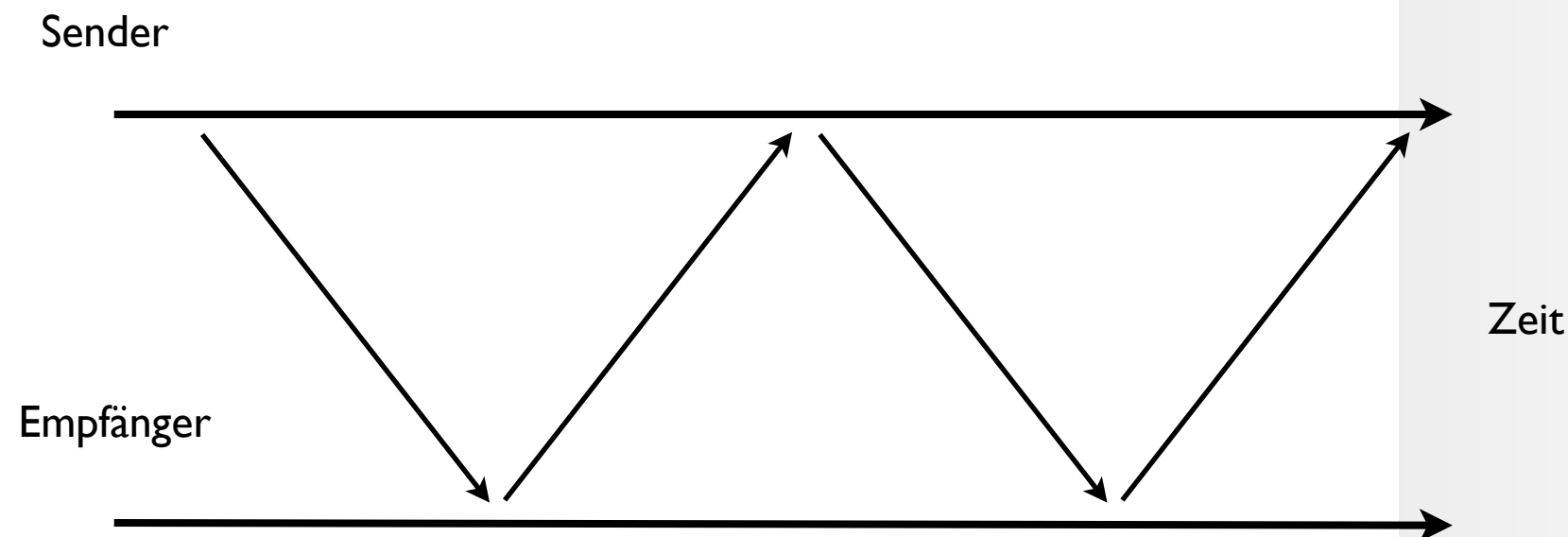
Fehlerkontrolle

- ▶ Sicherstellen das alle gesendeten Frames in der richtigen Reihenfolge ankommen.
- ▶ Bestätigte verbindungslose bzw. verbindungsorientierte Dienste sehen eine Bestätigung jedes empfangen Frames vor.
- ▶ Dieser Ansatz löst folgendes Problem nicht:
 - Ausfall der Übertragungsstrecke (Hardwarefehler) oder Verlust des Bestätigungs-Frames: In diesen Fällen würde der Sender unbestimmte Zeit auf die Bestätigung 'warten', was eine komplett geblockte ('hängende') Kommunikation zur Folge hätte.
 - Lösung: Einführung von Timern (Zählern), wobei nach Ablauf des Timers ein Frames erneut gesendet wird.
 - Bei dieser Lösung muss allerdings noch zusätzlich eine Sequenznummer für gesendete Frames eingeführt werden, da sonst beim Empfänger doppelt empfangene Frames nicht unterschieden werden können.

Flusskontrolle

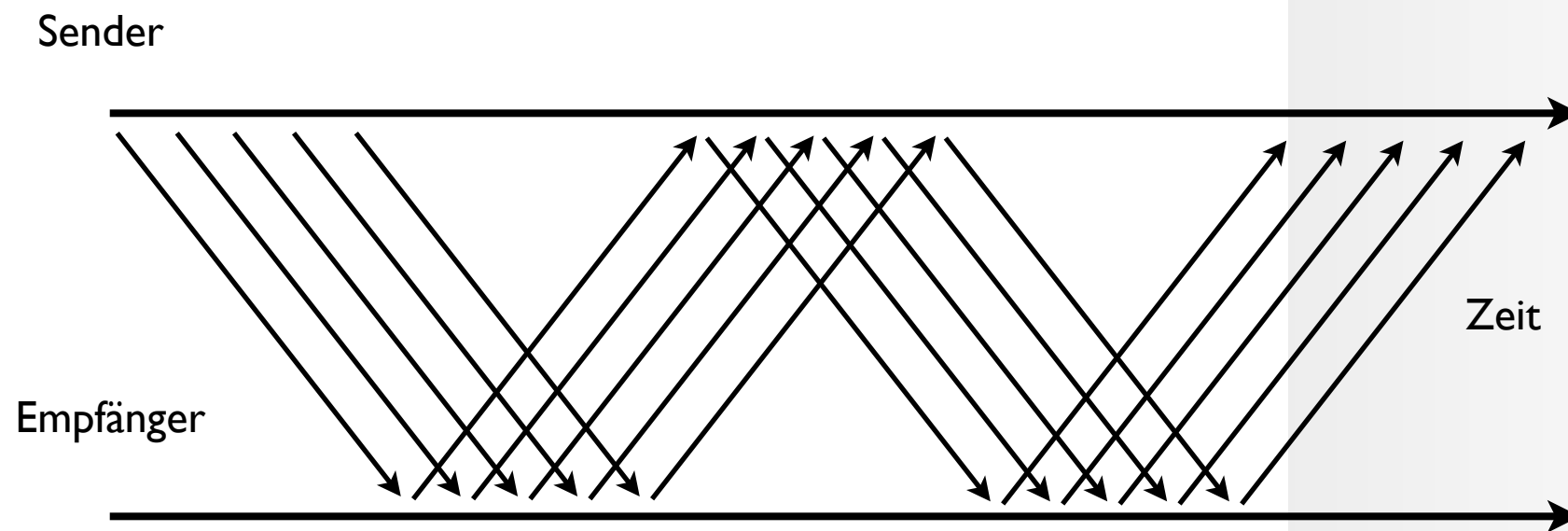
- ▶ Regulierung der Übertragungsrate (Frame-Rate) um zu verhindern, dass z.B. ein schneller (Rechenstarker) Sender einen langsameren (Rechenschwacher) Empfänger überfordert wird.
- ▶ Zwei grundlegende Ansätze werden heute verwendet:
 - Feedback-Based: Es wird erst ein weiterer Block gesendet, wenn der Empfänger den Empfang des letzten Blocks bestätigt hat.
 - Rate-Based: Daten werden generell nur in einer bestimmten (Protokollabhängig) Rate gesendet.
- ▶ Auch die Flusskontrolle wird oft nur in höheren Schichten implementiert.

Datenflusskontrolle: Stop-and-Wait



- Beim Stop-and-Wait Algorithmus wird erst ein neuer Frame gesendet wenn der Empfang des vorherigen bestätigt wurde.

Datenflusskontrolle: Sliding Window



- ▶ Eine bessere Auslastung, als bei Stop-and Wait, lässt sich erreichen, indem der Sender bereits während der Wartezeit weitere Frames schickt.
 - Zu jedem Zeitpunkt sind dann mehrere Rahmen auf der Leitung unterwegs, und der Sender wartet auf entsprechend viele ausstehende Bestätigungen.
 - Der Sender nummeriert die Rahmen, so dass sie individuell bestätigt werden können.

Strategien zum Umgang mit Übertragungsfehlern

- ▶ Fehler Erkennung (Fehler erkennende Codes):
 - Hinzufügen von redundanten Information um Übertragungsfehler zu erkennen. Bei einem erkannten Fehler wird den Frame erneut angefordert.
 - Verwendung meist bei verlässlichen Übertragungen (Übertragungswege mit wenig Störungen): Lichtwellenleiter, gute Kupferkabelverbindungen.
- ▶ Fehler Korrektur (Fehler korrigierende Codes):
 - Hinzufügen von redundanten Informationen um Übertragungsfehler zu korrigieren. Fehler können dann beim Empfänger mit Hilfe der redundanten Informationen selbständig korrigiert werden.
 - Verwendung meist bei nicht verlässlichen Übertragungen (Übertragungswege mit viel Störungen): Funkübertragungen

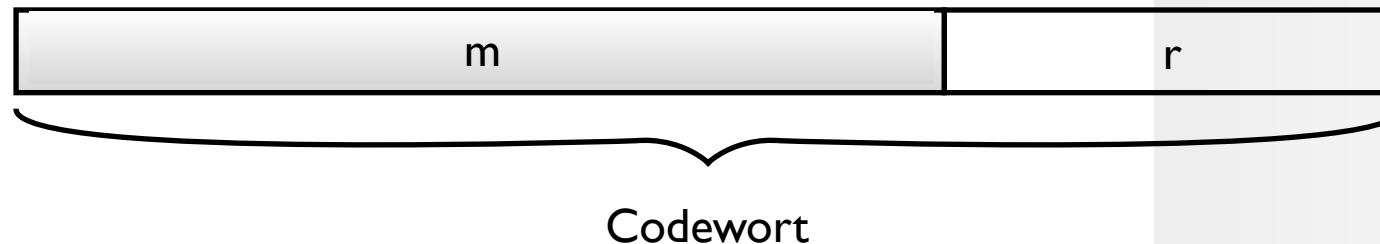
Arten von Übertragungsfehlern

- ▶ Gleichverteilte Bit Fehler:
 - wird in der Regel durch thermische Rauschen oder weißes Rauschen auf dem Übertragungsmedium hervorgerufen.
- ▶ Bündelstörung (Burst Errors):
 - wird durch zeitweilige Störungen hervorgerufen wie plötzliche Frequenz-Überlagerungen.
- ▶ Auslöschungsfehler bzw. symmetrischer Fehler:
 - Auslöschungsfehler: Am Empfänger (Bitübertragungsschicht) kann das gestörte Signal eindeutig als nicht gültig erkannt werden, da es weder der als '0' zugehörigen Signalleistung noch der der '1' zugehörigen entspricht.
 - symmetrischer Fehler: Das Signal am Empfänger entspricht dem entgegengesetzten Signal, so dass eine '0' als '1' bzw. eine '1' als '0' interpretiert wird.

Arten von Übertragungsfehlern

- ▶ Weder die Fehlererkennenden Codes noch die Fehlerkorrigierenden Codes können alle Varianten und Kombinationen von Übertragungsfehlern immer zuverlässig behandeln.
 - Die redundante Information kann ebenfalls von Übertragungsfehlern betroffen sein, da diese Informationen über den gleichen Übertragungskanal gesendet werden muss wie die eigentliche Nutzinformation.
- ▶ Aus diesem Grund werden Fehlererkennenden Codes bzw. Fehlerkorrigierenden Codes oft in unterschiedlichen Schichten parallel verwendet.
- ▶ Zusätzlich werden die Fehlererkennenden Codes bzw. Fehlerkorrigierenden Codes nach den zu erwartenden Fehlerarten ausgesucht bzw. angepasst.

Hamming Abstand



m: Anzahl der Nutzdatenbits

r: Anzahl der Kontrollbits

$n = m + r$: Länge des Codewortes

- ▶ Der Hamming-Abstand benannt nach dem US-amerikanischen Mathematiker Richard Wesley Hamming (1915–1998), sind Maße für die Unterschiedlichkeit von Zeichenketten.
- ▶ Der Hamming-Abstand zweier Blöcke mit fester Länge (Codewörter) ist dabei die Anzahl der unterschiedlichen Stellen.
- ▶ Beispiel: 0 0 1 1 0 und 0 0 1 0 0 → Hamming Abstand = 1

Hamming Abstand eines Codes

- ▶ Unter dem Hamming-Abstand eines kompletten Codes versteht man das Minimum aller Abstände zwischen Wörtern innerhalb des Codes:

Beispiel: Ein Code besteht aus folgenden drei Wörtern:

$x = 0\ 0\ 1\ 1\ 0,$

$y = 0\ 0\ 1\ 0\ 1,$

$z = 0\ 1\ 1\ 1\ 0.$

- ➔ Der Hamming-Abstand zwischen x und y ist 2.
- ➔ Um y zu generieren muss man zwei Bits ändern: $y = x \text{ XOR } 0\ 0\ 0\ 1\ 1.$
- ➔ Der Hamming-Abstand zwischen x und z ist 1.
- ➔ Um z zu generieren muss man ein Bit ändern: $z = x \text{ XOR } 0\ 1\ 0\ 0\ 0.$
- ➔ Der Hamming-Abstand zwischen y und z ist 3.
- ➔ Um z zu generieren muss man drei Bits ändern: $z = y \text{ XOR } 0\ 1\ 0\ 1\ 1.$

Der kleinste der drei Abstände ist 1, also ist der Hamming-Abstand des Codes ebenfalls gleich 1.

Hamming Abstand eines Codes

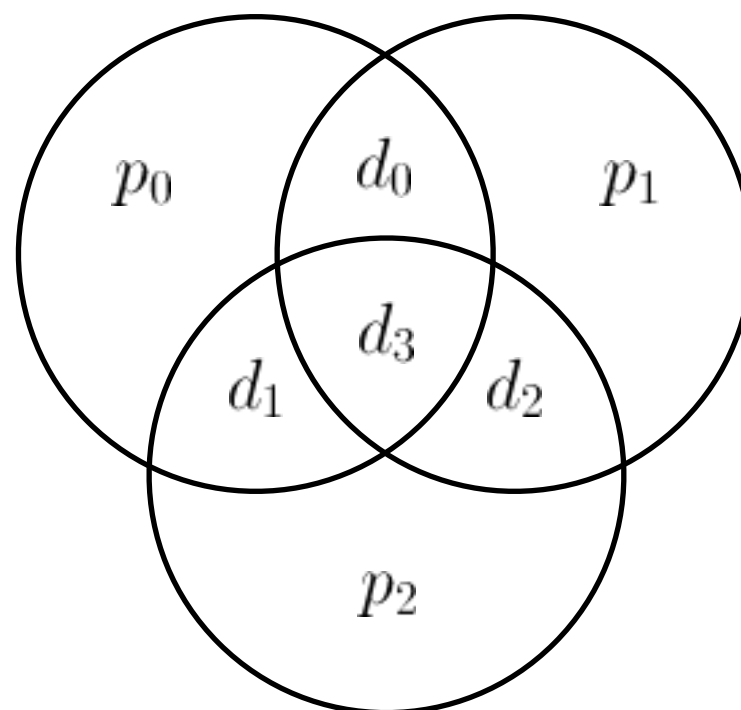
- ➔ Bei Codes mit Hamming-Abstand h können alle $(h-1)$ -Bit-Fehler erkannt werden. In dem Beispiel mit **$h = 1$ kann somit nicht einmal jeder 1-Bit-Fehler erkannt werden** ($x-z$ fällt nicht auf, alle anderen 1-Bit-Fehler erzeugen ungültige Codes, z. B. 00111 aus x oder y).
- ➔ Bei **$h = 2$ können alle 1-Bit-Fehler erkannt werden**. Um die **Fehler auch korrigieren zu können, muss die Hamming-Distanz auf mindestens $2r+1$ vergrößert werden**, wobei r für die Anzahl der korrigierbaren Bit-Fehler steht.
- ➔ Bei **$h = 3$ können alle 1-Bit-Fehler erkannt und korrigiert werden**. Treten 2-Bit-Fehler auf, werden diese unter Umständen falsch „korrigiert“, da das fehlerhafte Wort möglicherweise den Abstand 1 zu einem anderen gültigen Codewort hat.
- ➔ Bei **$h = 4$ können ebenfalls alle 1-Bit-Fehler erkannt und korrigiert werden**. Treten 2-Bit-Fehler auf, können diese zwar erkannt, aber nicht mehr korrigiert werden.
- ➔ Ein Code mit Hamming-Abstand 0 ist nicht möglich, da sich in diesem Fall zwei Codewörter nicht unterscheiden ließen.

Fehlerkorrigierende Codes: Hamming Code

m	r	$n=m+r$
Datenbits	Paritybits	Codewordbits
1	2	3
4	3	7
11	4	15
26	5	31
57	6	63
120	7	127
247	8	255













- ▶ Beim Hamming Code handelt es sich um einen Fehlererkennenden Code mit einem Hamming Abstand von 3. Es können also alle 1-Bit-Fehler erkannt und korrigiert werden.

Fehlerkorrigierende Codes: Hamming Code - Prinzipielle Idee



- ▶ Jeweils 2 Datenbits befinden sich in der Schnittmenge von 2 Paritäts-Bits
- ▶ Jeweils 1 Datenbit befindet sich NICHT in der Schnittmenge von 2 Paritäts-Bits.

Fehlerkorrigierende Codes: Hamming Code - Überprüfung und Korrektur

																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																												
---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Überprüft wird nach Even-Parity (xor = 0) über die Parity-Bits und ihren zugehörigen Datenbits:

$$\gamma_1 : c_1 \vee c_3 \vee c_5 \vee c_7 \vee c_9 \vee \dots \vee c_{31} = 0$$

$$\gamma_2 : c_2 \vee c_3 \vee c_6 \vee c_7 \vee c_{10} \vee \dots \vee c_{31} = 0$$

$$\gamma_3 : c_4 \vee c_5 \vee c_6 \vee c_7 \vee c_{12} \vee \dots \vee c_{31} = 0$$

Hamming Code: Check - Skip Verfahren

Check - Skip nach Position des Paritätsbits:

c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15	c16	c17	c18	...
p1	p2	d1	p3	d2	d3	d4	p4	d6	d7	d8	d9	d10	d11	d12	p5	d13	d14	...

p1 (c1) ⇒ Check 1, Skip 1 ⇒ c1, c3, c5, c7, c9, c11, c13, c15, c17 ...
 p2 (c2) ⇒ Check 2, Skip 2 ⇒ c2, c3, c6, c7, c10, c11, c14, c15, c18, c19 ...
 p3 (c4) ⇒ Check 4, Skip 4 ⇒ c4, c5, c6, c7, c12, c13, c14, c15 ...
 p4 (c8) ⇒ Check 8, Skip 8 ⇒ c8, c9, c10, c11, c12, c13, c14, c15 ...
 : : :

Fehlerkorrigierende Codes: Hamming Code - Beispiel

Überprüfung und Korrektur eines Hamming Codes:

Codeword: 0 1 0 0 0 0 1

A - Bestimmung der Parity-Bits:

c1	c2	c3	c4	c5	c6	c7
p1	p2	d1	p3	d2	d3	d4
0	1	0	0	0	0	1

B - Prüfungsregeln:

(1) : $c1 \text{ xor } c3 \text{ xor } c5 \text{ xor } c7 = 0$
(2) : $c2 \text{ xor } c3 \text{ xor } c6 \text{ xor } c7 = 0$
(3) : $c4 \text{ xor } c5 \text{ xor } c6 \text{ xor } c7 = 0$

C - Überprüfung:

(1) : $c1 \text{ xor } c3 \text{ xor } c5 \text{ xor } c7 = 0 \text{ xor } 0 \text{ xor } 0 \text{ xor } 1 = 0 \text{ xor } 0 \text{ xor } 1 = 0 \text{ xor } 1 = 1 \nleftrightarrow$
(2) : $c2 \text{ xor } c3 \text{ xor } c6 \text{ xor } c7 = 1 \text{ xor } 0 \text{ xor } 0 \text{ xor } 1 = 1 \text{ xor } 0 \text{ xor } 1 = 1 \text{ xor } 1 = 0 \checkmark$
(3) : $c4 \text{ xor } c5 \text{ xor } c6 \text{ xor } c7 = 0 \text{ xor } 0 \text{ xor } 0 \text{ xor } 1 = 0 \text{ xor } 0 \text{ xor } 1 = 0 \text{ xor } 1 = 1 \nleftrightarrow$

D - Korrektur:

Da (2) mit c3, c6 und c7 richtig ist
muss c5 falsch sein: $c5 = 1$

Codeword: 0 1 0 0 1 0 1

Daten: 0 1 0 1

Hamming Code Übungsaufgabe

Fehlerkorrigierende Codes: Weitere Codes

► Faltungscodes

- Durch das mathematische Verfahren der Faltung (jeder Wert von f wird durch das mit g gewichtete Mittel der ihn umgebenden Werte ersetzt) wird der Informationsgehalt der einzelnen Nutzdatenstellen über mehrere Stellen des Codewortes verteilt.
- Durch die Überlagerung von f und g können wie beim Hamming Code Übertragungsfehler erkannt und korrigiert werden.
- Anwendung: GSM, 802.11 (WLAN)

Fehlerkorrigierende Codes: Weitere Codes

► Reed Solomon Codes

- Um Redundanz zur Nachricht hinzuzufügen, werden die Zahlen der Nachricht als Werte eines Polynoms an fest vereinbarten Stützstellen interpretiert.
- Werden bei der Übertragung nun einige wenige Zahlen ausgelöscht, so kann das Polynom wiederum durch Interpolation aus den korrekt übertragenen Zahlen rekonstruiert werden.
- Anwendung: DVB, Audio CDs, DVDs, Voyager 2 (1985)

Fehlerkorrigierende Codes: Weitere Codes

- ▶ LDPC Codes
 - Low-Density-Parity-Check-Codes beschreiben mit Hilfe einer dünn besetzten Kontrollmatrix viele zusammenhängende Paritätsprüfungen für einen ganzen Nutzdatenblock.
 - Anwendung: DVB, 802.11 (neuster IEEE Standard), GBit/s Ethernet

Fehlererkennende Codes: Paritätsbit (Parity)

Even Parity

1	0	1	1	0	1	0	0
---	---	---	---	---	---	---	---

Anzahl Einsen gerade - Paritätsbit = Null

1	0	0	1	0	1	0	1
---	---	---	---	---	---	---	---

Anzahl Einsen ungerade - Paritätsbit = Eins

Odd Parity

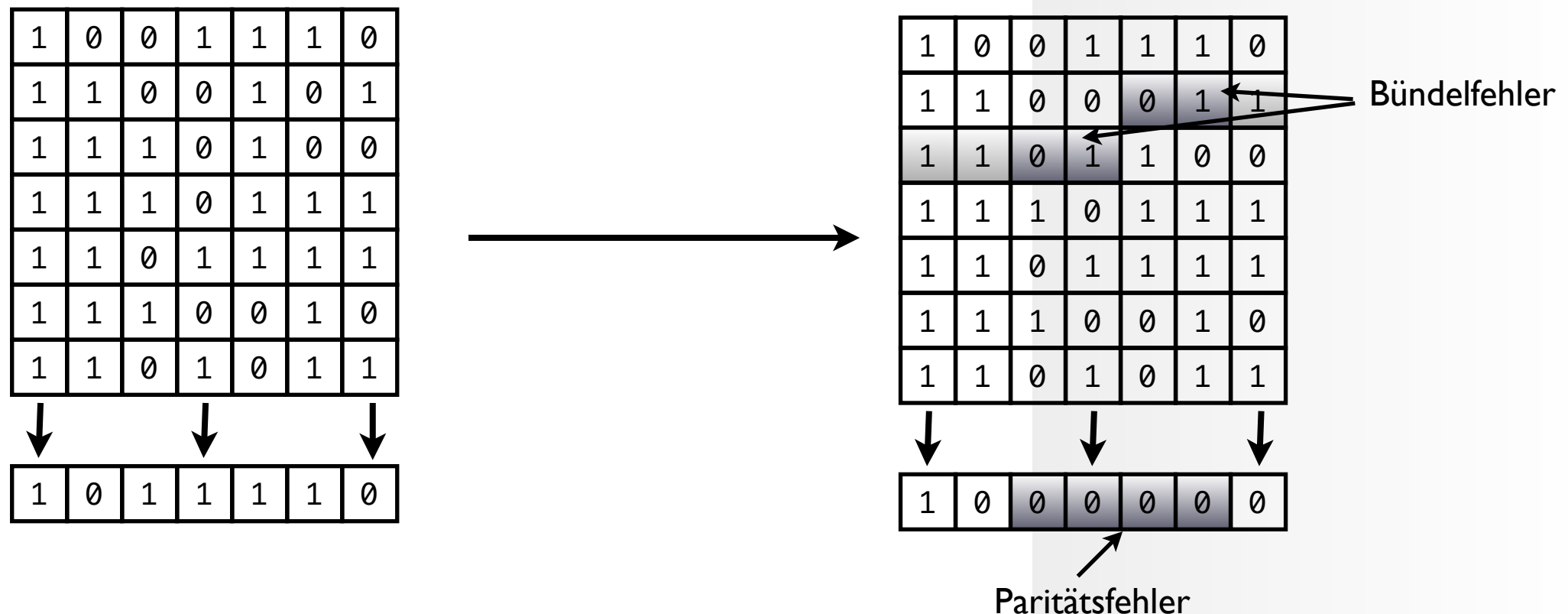
1	0	1	1	0	1	0	1
---	---	---	---	---	---	---	---

Anzahl Einsen gerade - Paritätsbit = Eins

1	0	0	1	0	1	0	0
---	---	---	---	---	---	---	---

Anzahl Einsen ungerade - Paritätsbit = Null

Fehlererkennende Codes: Paritybits Bündelstörung



- ▶ $n \times k$ Parity Bit Interleaving kann eine Bündelstörung der Länge n entdecken. (n =Wortlänge - Zeilen, k =Anzahl Wörter - Spalten)
- ▶ Bündelstörungen der Längen $n+1$ bleiben unentdeckt.

Fehlererkennende Codes: CRCs (Cyclic Redundancy Checks)

- ▶ Die Bitfolge der Daten wird durch ein vorher festzulegendes Generatorpolynom (das CRC-Polynom) Modulo mod(2) geteilt (Polynomdivision), wobei ein Rest bleibt. **Dieser Rest ist der CRC-Wert.** Bei der Übertragung des Datenblocks hängt man den CRC-Wert an den originalen Datenblock an und überträgt ihn mit.
 - Bsp: Bildung eines Polynoms: $110101 \Rightarrow 1 \cdot x^5 + 1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$
- ▶ Um zu verifizieren, dass die Daten keinen Fehler beinhalten, wird der empfangene Datenblock mit angehängtem CRC-Wert als Binärfolge interpretiert, erneut durch das CRC-Polynom Modulo geteilt und der Rest ermittelt. Wenn kein Rest bleibt, ist kein Fehler aufgetreten.

Fehlererkennende Codes: CRC - Beispiel

Daten: 1 1 0 1 1

Generatorpolynom: 1 1 0 1 0 1

A - Konstruktion: Daten mit Anhang

Generatorpolynom ist Polynom 5. Grades (5 Stellen), daher müssen 5 "0" angehängt werden:

Daten mit Anhang: 1 1 0 1 1 0 0 0 0 0

B - CRC Wert:

Daten mit Anhang: 1 1 0 1 1 0 0 0 0 0

Generatorpolynom: 1 1 0 1 0 1

```

-----
                0 0 0 0 1 1 0 0 0 0
                1 1 0 1 0 1
-----
                0 0 0 1 0 1 (Rest)
  
```

CRC Wert: 1 0 1

C - Überprüfung:

Empfange Daten : 1 1 0 1 1 0 0 1 0 1

Generatorpolynom: 1 1 0 1 0 1

```

-----
                0 0 0 0 1 1 0 1 0 1
                1 1 0 1 0 1
-----
                0 0 0 0 0 0 (Rest)
  
```

Rest = 0 -> OK

Fehlererkennende Codes: CRC - Implementierung (C)

```
#include <stdio.h>
#include <stdlib.h>
#include <inttypes.h>
#define CRC32POLY 0x04C11DB7 /* CRC-32 Polynom */

int datastream[] = {1,0,0,0,1,1,0,0};
int databits = 8;
uint32_t crc32 = 0; /* Schieberegister */

int main(void)
{
    int i;
    for (i = 0; i < databits; ++i)
        if (((crc32 & 0x80000000) ? 1 : 0) != datastream[i])
            crc32 = (crc32 << 1) ^ CRC32POLY;
        else
            crc32 <<= 1;
    printf("0x%08X\n", crc32);
    return EXIT_SUCCESS;
}
```

Media Access Control im OSI Modell

ISO/OSI-Schichtenmodell		
7	Anwendungsschicht (Application)	
6	Darstellungsschicht (Presentation)	
5	Kommunikationsschicht (Session)	
4	Transportschicht (Transport)	
3	Vermittlungsschicht (Network)	
2	Sicherungsschicht (Data Link)	2a - Media Access Control (MAC)
		2b - Logical Link Control (LLC)
1	Bitübertragungsschicht (Physical)	

Media Access Control: Aufgaben

- ▶ Die MAC umfasst Netzwerkprotokolle und Bauteile, die regeln, wie sich mehrere Rechner das gemeinsam genutzte physikalische Übertragungsmedium teilen.
 - Sie wird benötigt, weil ein gemeinsames Medium nicht gleichzeitig von mehreren Rechnern verwendet werden kann, ohne dass es zu Datenkollisionen und damit zu Kommunikationsstörungen oder Datenverlust kommt.

Media Access Control: Zugriffsarten

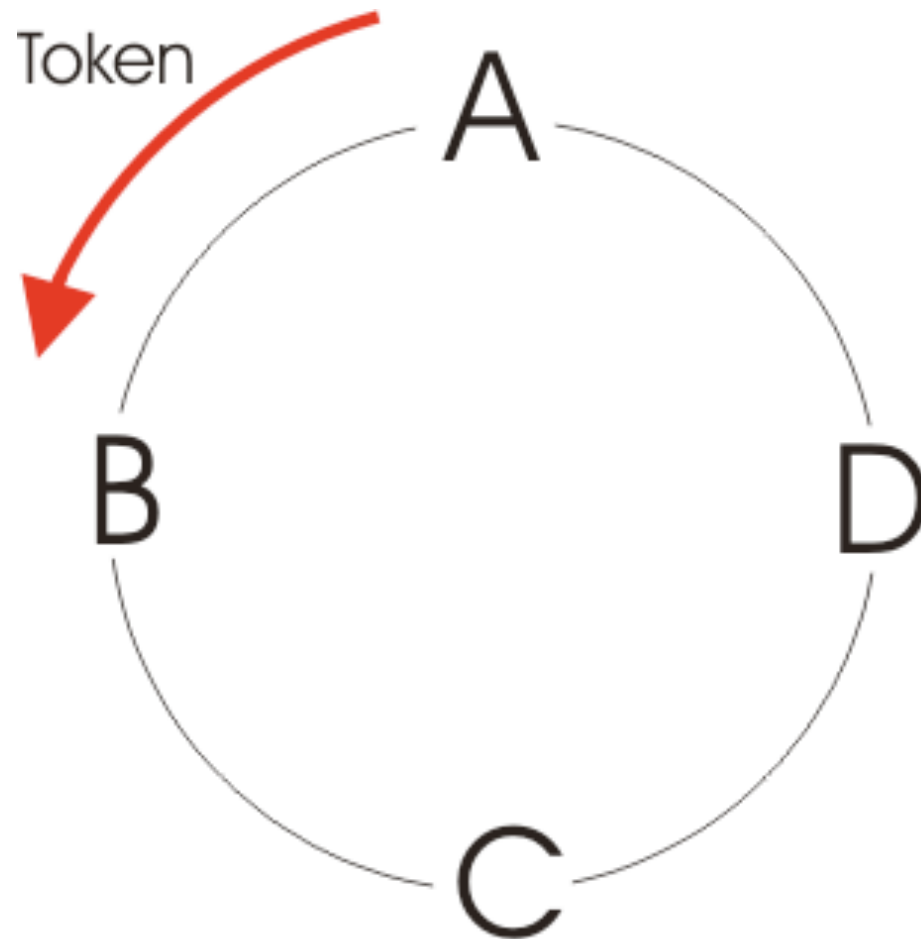
▶ Kontrollierter Zugriff

- der Zugriff auf das Medium so geregelt wird, dass keine Kollisionen auftreten können.
- Anwendungen: Token-Ring

▶ Konkurrierender Zugriff

- jeder kann auf das Medium zugreifen und es gibt Regeln, wie Kollisionen ohne Komplikationen behandelt werden.
- ALOHA, CSMA/CD, CSMA/CA

Kontrollierter Zugriff: Token-Ring



Der Token Ring arbeitet mit dem **Token-Passing-Zugriffsverfahren** und entspricht im Aufbau einer **logischen Ring-Topologie**. Die Anschlussart an das Medium ist damit aktiv (beispielsweise im Gegensatz zum passiven Ethernet), das heißt die Netzwerkstationen beteiligen sich fortwährend aktiv an der Weitergabe des Tokens und werden nicht nur dann aktiv, wenn sie selbst senden wollen.

Das gesamte Paket schickt **Computer A an seinen Nachbar Computer B. Computer B erkennt, dass nicht er der Empfänger des Datenframes ist und sendet es an seinen Nachbar Computer C. Da C als Empfänger eingetragen ist, kopiert er das Datenframe und modifiziert das Token auf empfangen. Dann sendet er den Frame wieder auf den Ring. Da das Token immer noch besetzt ist, kann kein Computer Daten anhängen.** Beim Eintreffen des Frames bei Computer A **überprüft A, ob der Inhalt mit dem versendeten übereinstimmt und die Empfangsmarkierung gesetzt ist.** Ist dies der Fall, so war die **Übertragung erfolgreich.** Der Datenframe wird entfernt und das Token wird wieder auf frei gesetzt. Selbst wenn eine Übertragung fehlgeschlagen ist, muss der Sender nach dem Empfang der Empfangsmarkierung (Quittung) auf jeden Fall ein freies Token senden. So wird gewährleistet, dass nach jeder Datenübertragung ein freies Token im Ring ist.

Konkurrierender Zugriff: ALOHA

► ALOHA

- Beim reinen ALOHA kann jeder Teilnehmer zu einem beliebigen Zeitpunkt sein (stets gleich langen) Frame verschicken.
- Dabei können Frames kollidieren und verstümmeln, diese und müssen neu übertragen werden.
- Die Sender erkennen/erfahren von einer über das ausblenden eine Bestätigung von der empfangenden Station.
- Jeder Sender wartet für die erneute Übertragung eine zufällige Zeitperiode um wiederkehrende Kollisionen zu verhindern.
- **Probleme**
 - Nicht echtzeitfähig, da nicht garantiert werden kann, wann ein Frame tatsächlich übertragen werden kann.

Konkurrierender Zugriff: ALOHA

► Slotted ALOHA

- Im Gegensatz zu ALOHA darf ein Teilnehmer, bei diesem Verfahren nicht zu einem beliebigen Zeitpunkt senden, sondern muss sich an fest vorgegebene Zeitscheiben (Slots) von der Länge eines Paketes halten.
- Jeder Benutzer kann jederzeit in einen dieser Slots schreiben.
- Wenn mehrere gleichzeitig einen Slot benutzen, kommt es zur Kollision aber die Frames können sich jeweils nur voll überdecken.
- Dadurch lässt sich die maximale Datenrate gegenüber ALOHA verdoppeln.
- **Probleme:**
 - Slotted ALOHA ist aufwendiger zu realisieren als ALOHA, da alle Teilnehmer über eine einheitliche und zentrale Zeitquelle synchronisiert werden müssen.

Konkurrierender Zugriff: CSMA - Carrier Sense Multiple Access

- ▶ 1-persistent CSMA
 - **Verfahren:**
 - Bevor eine Station einen Frame sendet überprüft Sie den Sendekanal.
 - Ist der Sendekanal frei, wird der Frame gesendet.
 - Ist der Sendekanal belegt, wartet die Station und sendet den Frame wenn der Kanal frei ist.

Konkurrierender Zugriff: CSMA - Carrier Sense Multiple Access

- ▶ 1-persistent CSMA

- **Probleme:**

- Es kommt zu Kollisionen wenn zwei Stationen gleichzeitig senden. Folgende Situation ist besonders zu beachten:
 - Station A und (u.U. etwas später) Station B sind Sendebereit stellen aber fest, dass der Kanal durch eine Übertragung belegt ist. Beim vorliegenden Verfahren werden beide Stationen warten und gleichzeitig senden sobald der Kanal frei ist, was zu eine Kollision führt.
- Es kann durch die Laufzeitverzögerung zu Kollisionen kommen: Eine Station kann bei einen freien Sendekanal nicht zweifelsfrei feststellen inwieweit nicht doch schon ein Frame von einer entfernten Station gesendet wurde/gesendet wird.

Konkurrierender Zugriff: CSMA - Carrier Sense Multiple Access

- ▶ nonpersistent CSMA

- **Verfahren:**

- Bevor eine Station einen Frame sendet überprüft Sie den Sendekanal.
- Ist der Sendekanal frei, wird der Frame gesendet.
- Ist der Sendekanal belegt, wartet die Station eine zufällige Zeit und überprüft den Sendekanal erneut.
 - Dies verhindert, dass nachdem ein Kanal freigegeben wurde mehrere wartende/sendebereite Stationen gleichzeitig senden!

- **Probleme:**

- Dieses Vorgehen führt zu weniger Kollisionen aber zu längeren Sendeverzögerungen.

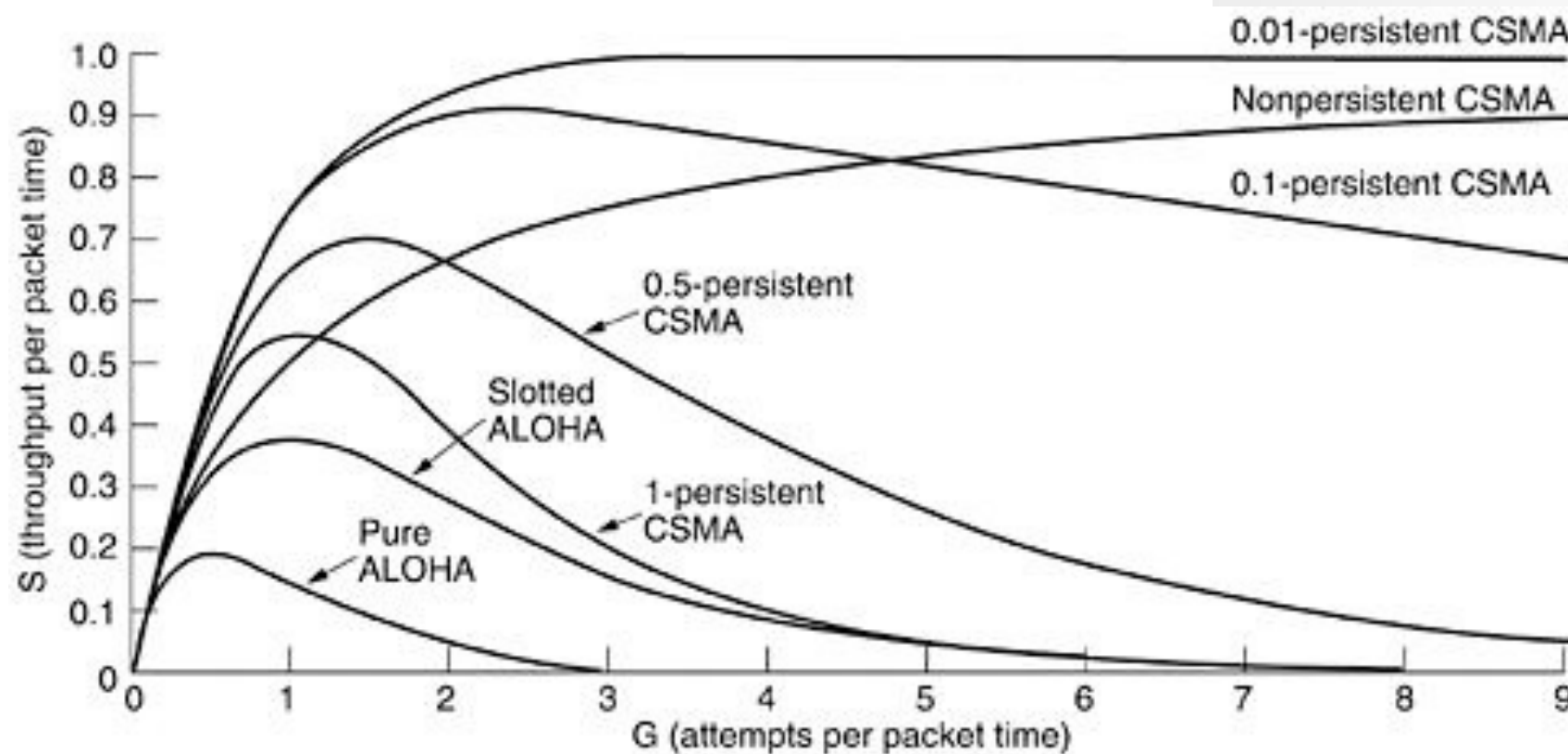
Konkurrierender Zugriff: CSMA - Carrier Sense Multiple Access

► p-persistent CSMA

– Verfahren:

- Wie beim Slotted ALOHA muss sich eine Sendende Station an fest vorgegebene Zeitscheiben (Slots) halten.
- Bekommt eine sendebereite Station einen freien Slot sendet sie mit einer Wahrscheinlichkeit von p oder verschiebt den Sendevorgang mit einer Wahrscheinlichkeit von $q=1-p$ auf den nächsten Slot.
- Ist dieser nächste Slot ebenfalls frei sendet die Station oder verschiebt wieder mit den Wahrscheinlichkeiten p und q .
- Dieser Vorgang wird wiederholt bis die Station die Übertragung abgeschlossen hat oder eine andere Station mit der Übertragung begonnen hat. In letzteren Fall behandelt die Station diese Situation wie eine Kollision und wartet eine Zufällige Zeit bevor sie erneut mit einen Sendeversuch beginnt.
- Trifft eine sendebereite Station von Anfang an auf einen belegten Slot wartet sie auf den nächsten Slot und der Vorgang startet von vorne.

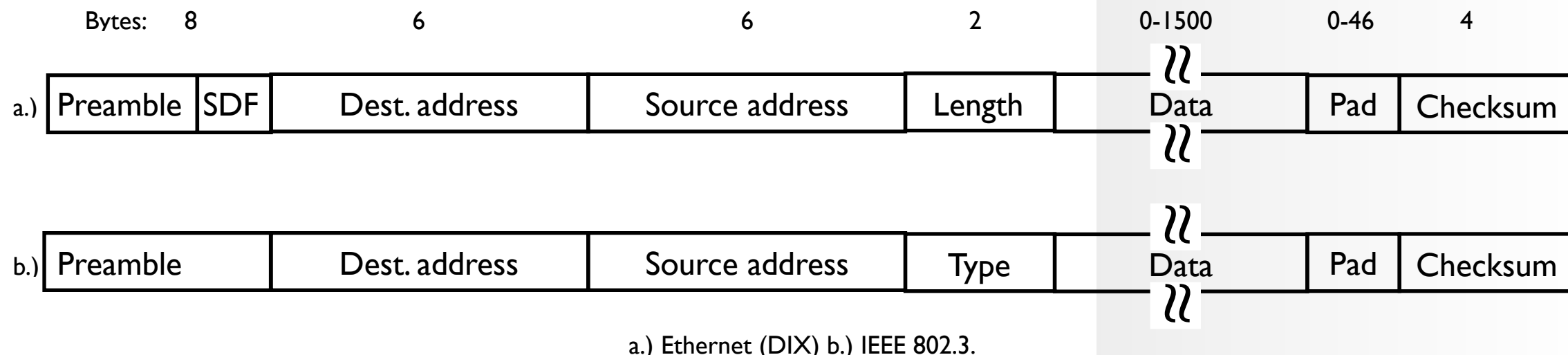
Konkurrierender Zugriff: CSMA - Carrier Sense Multiple Access



Computer Networks, Fifth Edition by Andrew Tanenbaum and David Wetherall, © Pearson Education-Prentice Hall, 2011

- Durchsatz zur Anzahl der Versuche bei verschiedenen “Konkurrierenden Zugriffsverfahren”.

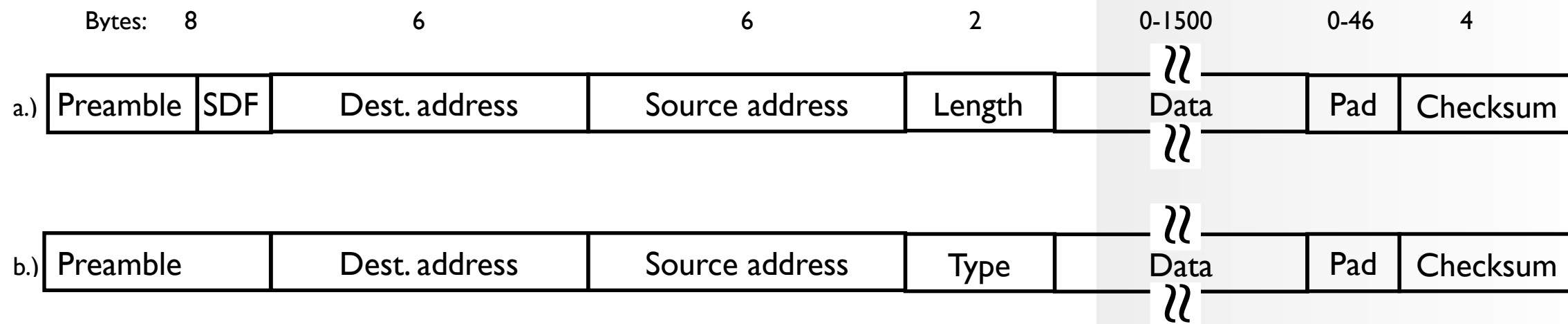
Ethernet Frame



► Preamble:

- 8 Bytes mit je dem Bitmuster: 10101010 (mit Ausnahme der letzten Byte hier sind die letzten 2 Bit auf 1 gesetzt - *Start Delimiter Frame (SDF)*). Dies erzeugt (*Manchester Code*) für 6.4 μ sek eine 10 MHz Welle welche zu Frame-Synchronisation verwendet wird.
- Framegrenzen: Preamble - SDF, Interframe-Gap (9,6 μ sek)
- MAC über 1-persitent CSMA/CD (CD - Collision Detection)

Ethernet Frame

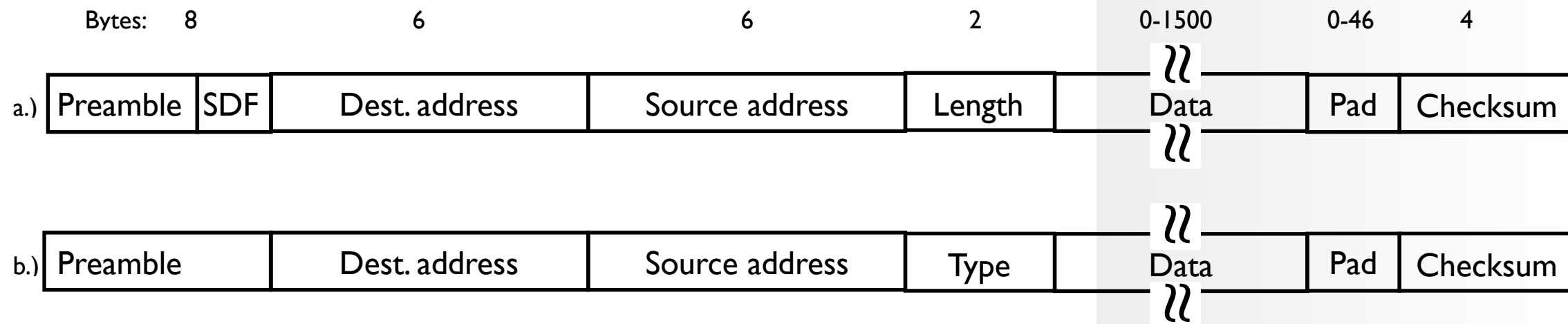


a.) Ethernet (DIX) b.) IEEE 802.3.

► Destination Address:

- Ziel Adressen (MAC-Adresse) um auf dem gemeinsamen Medium die Zielstation zu adressieren.
- Die ersten beiden Bit werden zur Klassifizierung verwendet (nach kanonischer Umwandlung):
 - 1. Bit (I/G Bit) = 0: Unicast Adresse oder 1. Bit = 1 Multicast-Adresse (Gruppe). Eine Ziel Adresse bestehend aus 1'ern ist reserviert für Broadcast (alle Stationen empfangen)
 - 2. Bit (L/G Bit) = 0 Global verwaltet oder 2. Bit = 1 lokal verwaltet.

Ethernet Frame

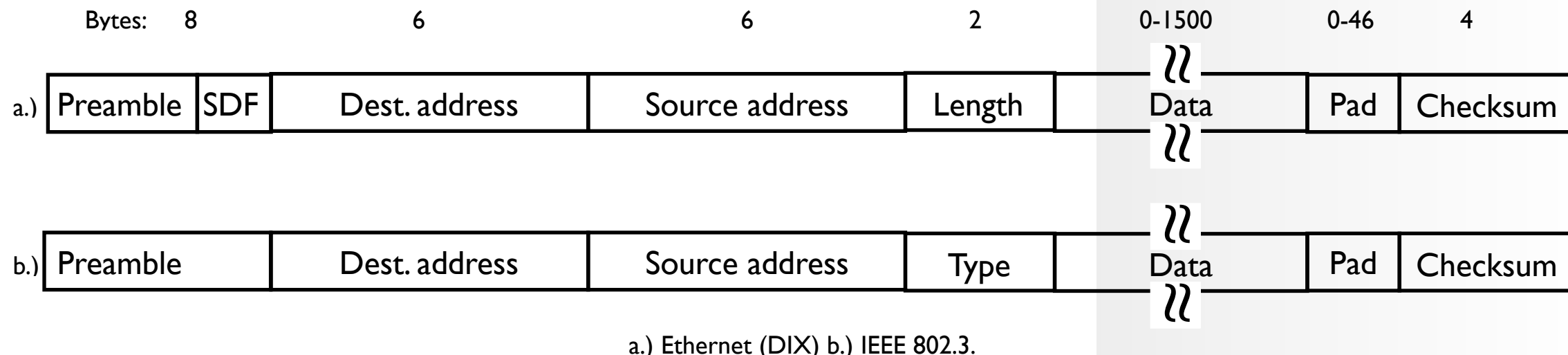


a.) Ethernet (DIX) b.) IEEE 802.3.

► Source Address:

- Quelladressen (MAC-Adresse) werden verwendet um die sendende Station zu identifizieren.
 - Die MAC-Adressen sind global (wenn 2. Bit = 0) von der IEEE vergeben und identifizieren jede Station (z.B. Ethernet Netzwerkkarte) weltweit eindeutig.
 - Hersteller bekommen hierzu einen Teil der MAC-Adresse zugeordnet.
 - MAC-Adressen werden in kanonischer Form geschrieben (z.B. 08:00:01:EA:DE:21). Dabei werden die 48 bit in sechs Oktette eingeteilt und in hexadezimalen Zeichenpaaren dargestellt. Zu beachten ist dabei, dass vor der Umformung von der dualen in die hexadezimale Darstellung das Oktett gespiegelt wird.

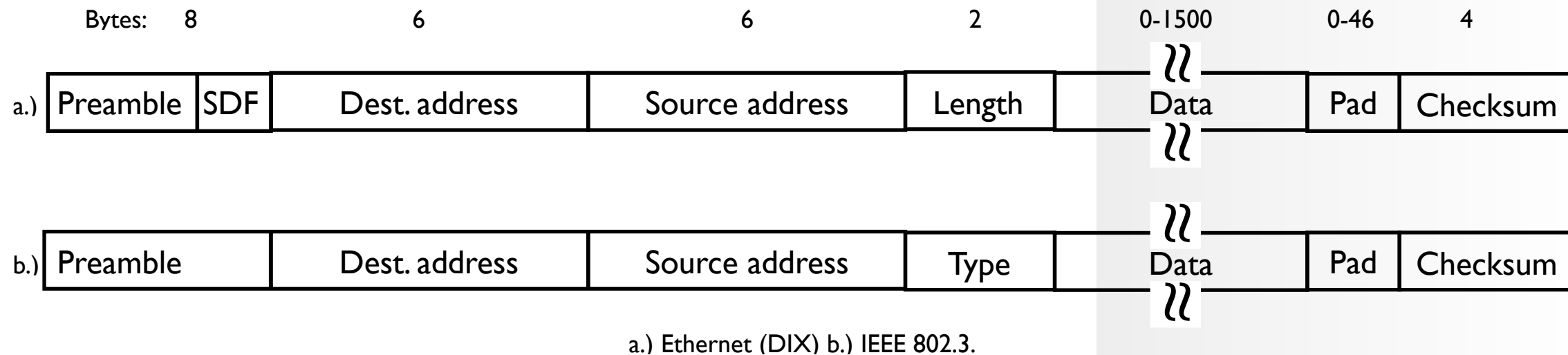
Ethernet Frame



► Type oder Length:

- Je nach Standard a.) Ethernet DIX (DEC, Intel und Xerox) oder b.) IEEE 802.3 wird hier der Typ des transportierten Pakets oder die Länge des Ethernet-Frames angegeben.
 - Typ: alle Werte sind größer als 0x0600 (1536). Beispiel: 0x0800 -> IP Internet Protocol, Version 4 (IPv4)
 - Länge: die maximale länge eines Ethernet-Frames ist 1500 daher, und aus Kompatibilität zum Type Feld, sind alle Werte kleiner gleich 1500

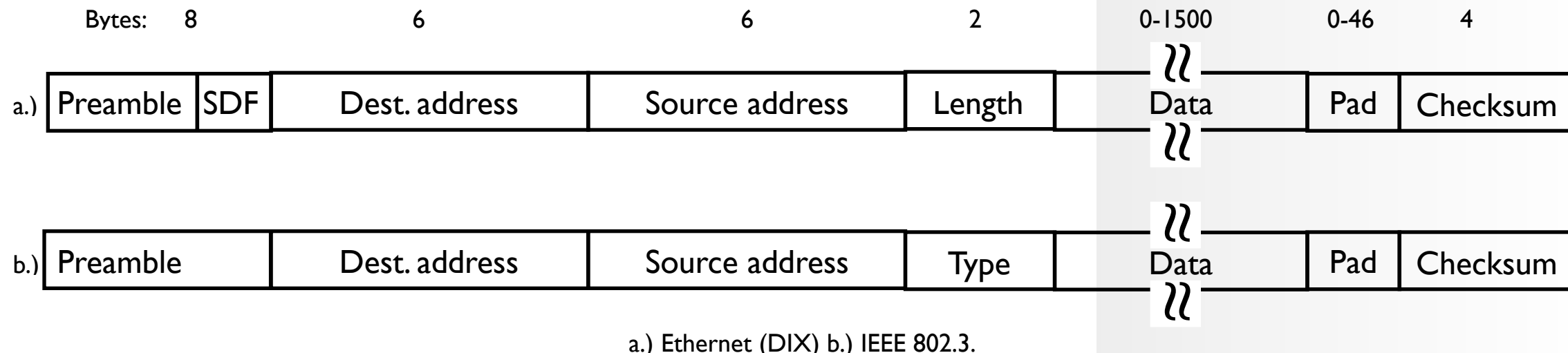
Ethernet Frame



► Data und Pad:

- Das "Data"-Feld enthält die Nutzdaten und kann 0-1500 Byte lang sein.
- Das "Pad"-Feld wird komplett gefüllt falls die Nutzdaten 0 Byte groß sind um die notwendige Mindestlänge eines Ethernet-Frames von 64 Byte zu erreichen. ($6+6+2+0+46+4 = 64$)
 - Eine Mindestlänge von 64 Byte ist notwendig um Kollisionen zu erkennen.

Ethernet Frame

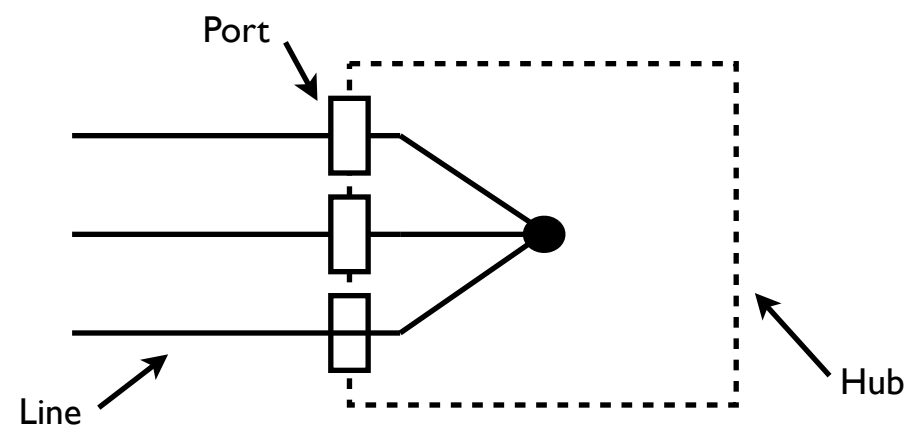


► Checksum:

- Dieses Feld (FCS - Frame Check Sequence) enthält eine 32 Bit CRC Prüfsumme und wird über den eigentlichen Frame berechnet, also beginnend mit der Ziel-MAC-Adresse und endend mit dem Pad-Feld.

Hubs

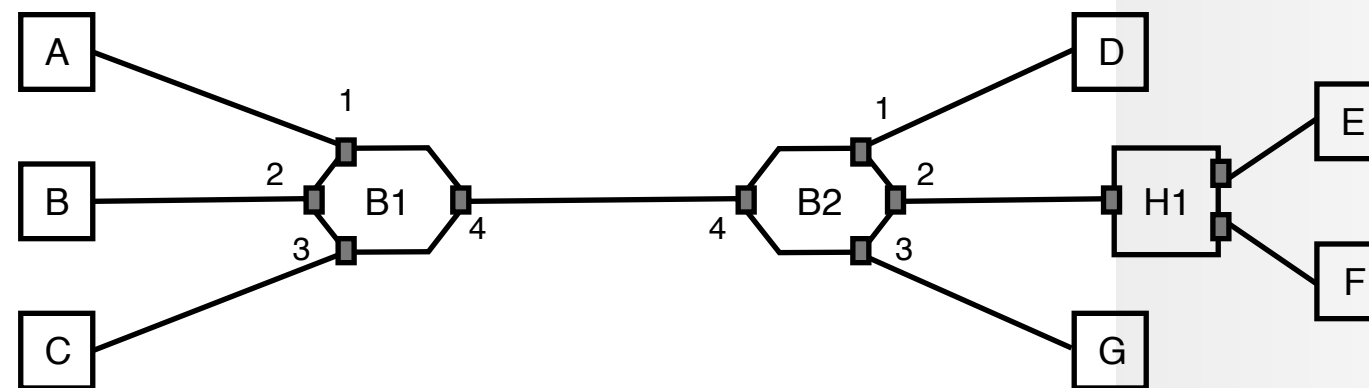
- ▶ Klassisch wurden mehrere Rechner, welche über Ethernet kommunizieren über *Hubs* verbunden.
- Bei Einsatz eines Hubs im Netz wird durch die Verkabelung im physikalischen Sinne eine Stern-Topologie realisiert.
- Der logische Aufbau ähnelt dem einer Bus-Topologie, weil jede gesendete Information alle Teilnehmer erreicht.
- Alle Teilnehmer in einem Netzwerk, die an einen Hub angeschlossen sind, befinden sich in derselben Kollisionsdomäne (vergl. CSMA).



Layer 2 Switch/Bridge

- ▶ Switche oder Bridges arbeiten auf der Sicherungsschicht (Data Link Layer) und bilden eine physikalische Stern Topologie.
 - Ein Switch verarbeitet bei Erhalt eines Pakets die 48 Bit lange MAC-Adresse und legt dazu einen Eintrag in der SAT (Source-Address-Table) an, in der neben der MAC-Adresse auch der physische Port, an dem diese empfangen wurde, gespeichert wird.
 - Im Unterschied zum Hub werden Netzwerkpakete nur noch an den Port weitergeleitet, der für die entsprechende Zieladresse in der SAT gelistet ist.
 - Falls der Weg zur Zieladresse noch unbekannt ist (Lernphase), leitet der Switch das betreffende Paket an alle aktiven Ports.
- ▶ Dadurch kann ein LAN in unterschiedliche Kollisionsdomänen (vergl.CSMA) aufgeteilt werden.
- ▶ Beim Einsatz mehrerer Switches/Bridges kann es zu Schleifen kommen, welche über ein STP (Spanning Tree Protokoll) unterdrückt werden müssen.

Switch/Bridge



SAT B1:

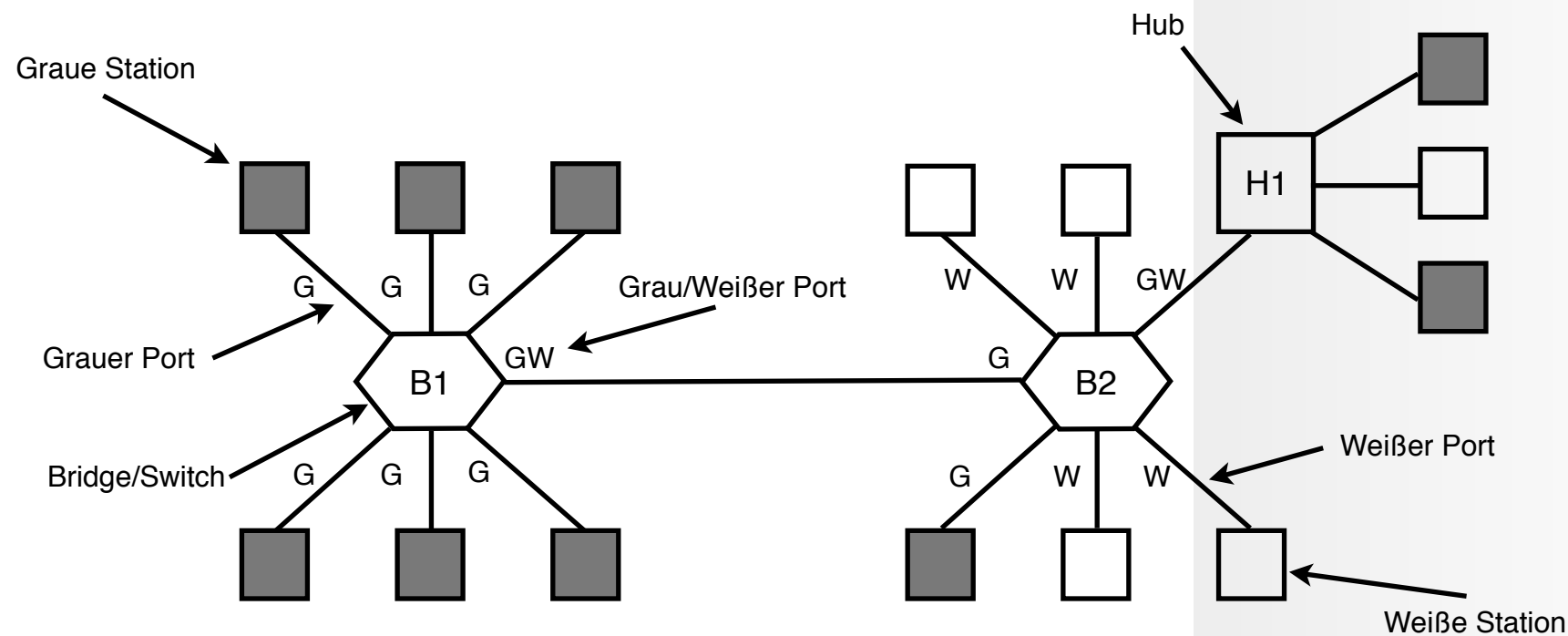
Station	Port
A	1
B	2
C	3
D,E,F,G	4

SAT B2:

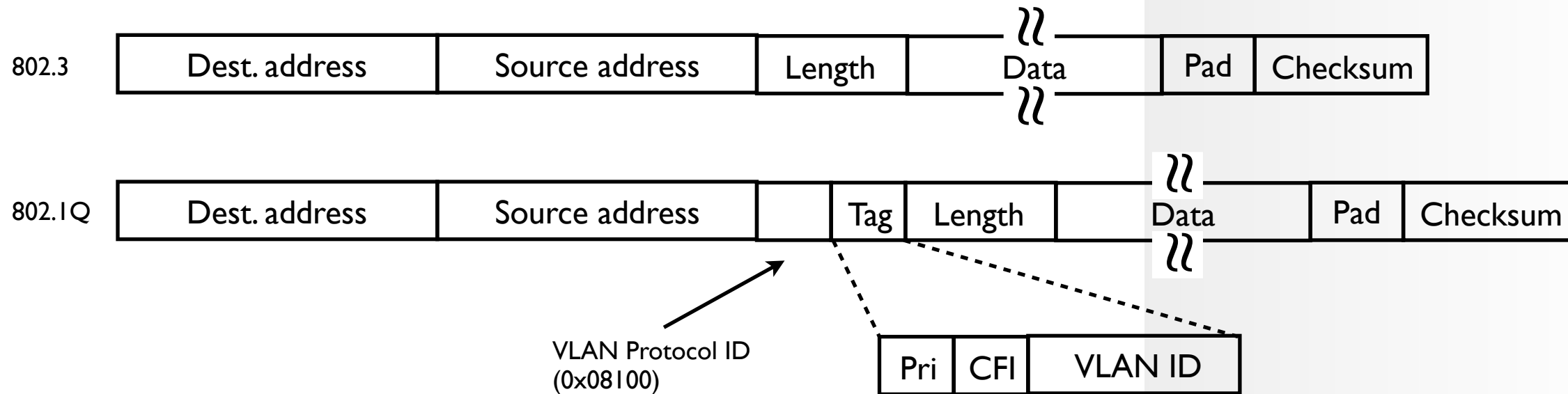
Station	Port
D	1
E	2
F	2
G	3
A,B,C	4

Beispiel: A sendet einen Frame zu D

- B1 empfängt den Frame an Port 1
- B1 sendet den Frame weiter an Port 4 (gelernt wenn einmal ein Frame von D an Port 4 empfangen wurde, sonst an Ports 2,3,4 - Flooding)
- B2 empfängt den Frame an Port 4
- B2 sendet den Frame weiter an Port 1 (gelernt wenn einmal ein Frame von D an Port 1 empfangen wurde, sonst an Ports 1,2,3 - Flooding)

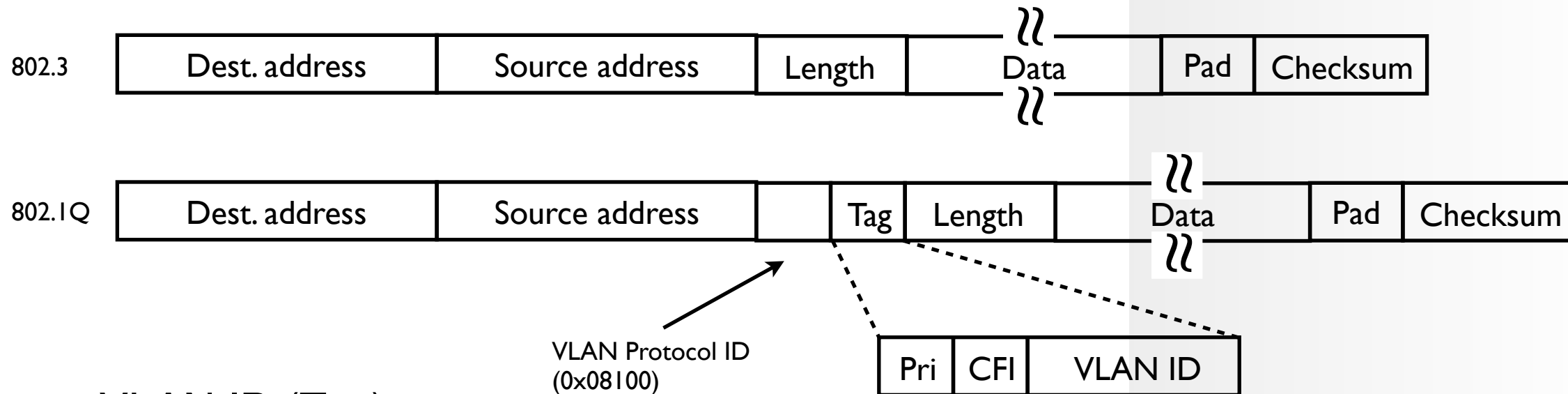


- ▶ Umsetzung über VLAN fähige Switches/Bridges und markierte (tagged) Frames.
- ▶ IEEE Standard 802.1Q
 - Erweiterung des Ethernet-Frames um ein VLAN Tag
 - Rückwärtskompatibel (z.B. alte Netzwerkkarten): Erster Switch/Bridge fügt Tag hinzu - letzter entfernt das Tag wieder.



► VLAN Protocol ID:

- Dieses Feld ist immer 0x08100 und damit größer als 1500 und wird von jeder Ethernet Karte als Typ-Feld erkannt.



► VLAN ID (Tag):

- belegt die unteren 12 Bit des Tag Feldes und bildet die Markierung des VLANs

► Pri und CFI(Tag):

- Pri: 3 Bit Prioritätsfeld (Echtzeit Anwendungen)
- CFI (Canonical Format Indicator): gedacht als Kennzeichner für das Übertragungsformat der MAC Adresse, z.Z. zweckentfremdet als Kennzeichner für 802.5 Frames (Token Ring)
- Beide Felder haben keinen Einfluss auf VLANs und wurden bei der Standardisierung mit spezifiziert.