

# Master Studiengang

---



**h\_da**

HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES

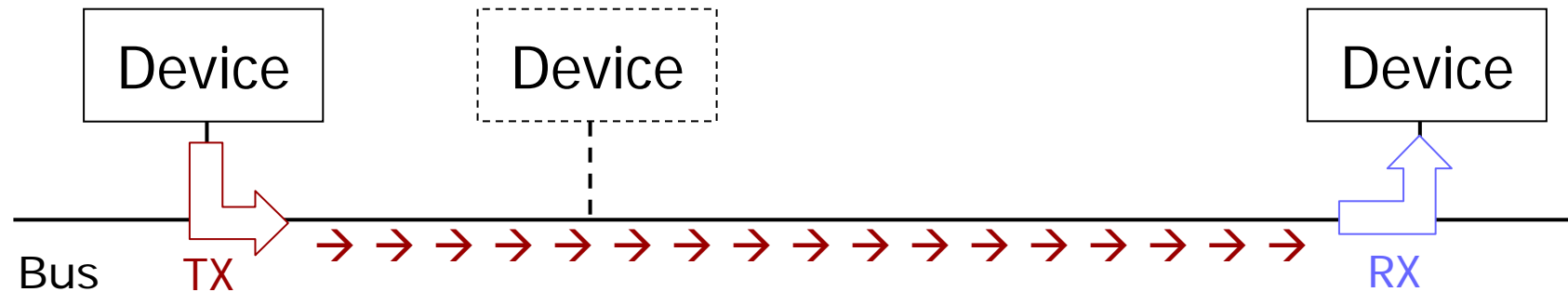
**fbi**  
FACHBEREICH INFORMATIK

## 5. Bus-, verteilte Systeme u. Eigenschaften

Prof. Dr. Ralf S. Mayer,  
Prof. Dr. Peter Altenbernd  
Fachbereich Informatik, h\_da

- Eigenschaften verteilter Systeme
  - Signallaufzeit
  - Buszuteilung
  - Bandbreite
  - Priorisierung
  - Zeitverhalten
- Bussysteme
  - Standards
  - Eigenschaften bezüglich OSI
  - Einsatz

# Verteilte Systeme und Bussysteme



- Bei der Verbindung von Geräten über ein Medium muss die Signallaufzeit berücksichtigt werden
- Maximale mögliche Ausbreitungsgeschwindigkeit:
  - Lichtgeschwindigkeit  $c_0 = 299.792.458 \text{ m/s} \approx 3 \times 10^8 \text{ m/s}$  im Vakuum
  - Geschwindigkeit im Medium:  $c_m < c_0$ 
    - In elektrischen Leitern (Kabeln) geht man von  $c_m = 17 - 20 \text{ cm/ns}$  aus.  $\rightarrow c_m \approx 2/3 c_0$
    - Geschwindigkeit im Medium in der Regel frequenzabhängig:  $c_m = c_m(f)$   $\rightarrow$  Dispersion
    - Dämpfung im Medium ist frequenzabhängig
    - Reflexion an Kabelenden ist möglich.  $\rightarrow$  Abschluss-Widerstände (Beispiel: Entspiegelung einer Brille)

# Signalausbreitung im Medium

- Ein Rechtecksignal (binäre Übertragung) kann aus einer Überlagerung von Sinusfunktionen dargestellt werden, wobei
  - $f_0$  die Frequenz des Rechteck-Signals ist und (unendlich viele) ungerade Harmonische  $f_{n+1}$  dies Frequenz in geeigneter Amplitude  $a_{n+1}$  hinzuaddiert werden: **harmonics: 1** → Fourier-Synthese

$$x_{\text{rechteck}}(t) = \frac{4}{\pi} \sum_{n=0}^{\infty} \frac{\sin((2n+1)t)}{2n+1}$$



- Bei einem realen Übertragungsmedium (Kabel, Lichtleiter etc) muss man aufgrund der Dispersion, Dämpfung und Reflexion davon ausgehen, dass ein ausgesandtes Rechtecksignal nicht mehr als Rechteck beim Empfänger ankommt.
- Auswirkung auf das System
  - sorgfältiges Design (elektrisch, physikalisch) notwendig
  - Reale Bandbreite kleiner als theoretische
  - Störeinflüsse von außen minimieren
  - Störung der Umgebung beachten (**ElektroMagnetische Verträglichkeit**)

# Bus-Zugriff (Arbitration)



Buszuteilung – Erkennung/ Vermeidung von Kollisionen:

- Master kontrollierte Netzwerke
- Token-basierter Zugriff: Token-Passing in Ringnetzen. Token wird von Knoten zu Knoten weitergeleitet. Knoten der Frei-Token erhält, darf senden. Token-Ring, ARCNET, Profibus, ...
- Feste / auf Anfrage zugeteilte Bandbreite
  - TDM(A): Zeitschlitz in Funknetzen (GSM), Kabelnetzen (ISDN)
  - FDM(A): Frequenzteilung (Rundfunk: Stereo), Kabel: ISDN/ADSL
- Multimaster-Netzwerke: mehrere Teilnehmer können „beliebig“ senden. → Gefahr von Kollisionen
  - Ethernet: Kollisionserkennung
  - CAN: Kollisionsvermeidung

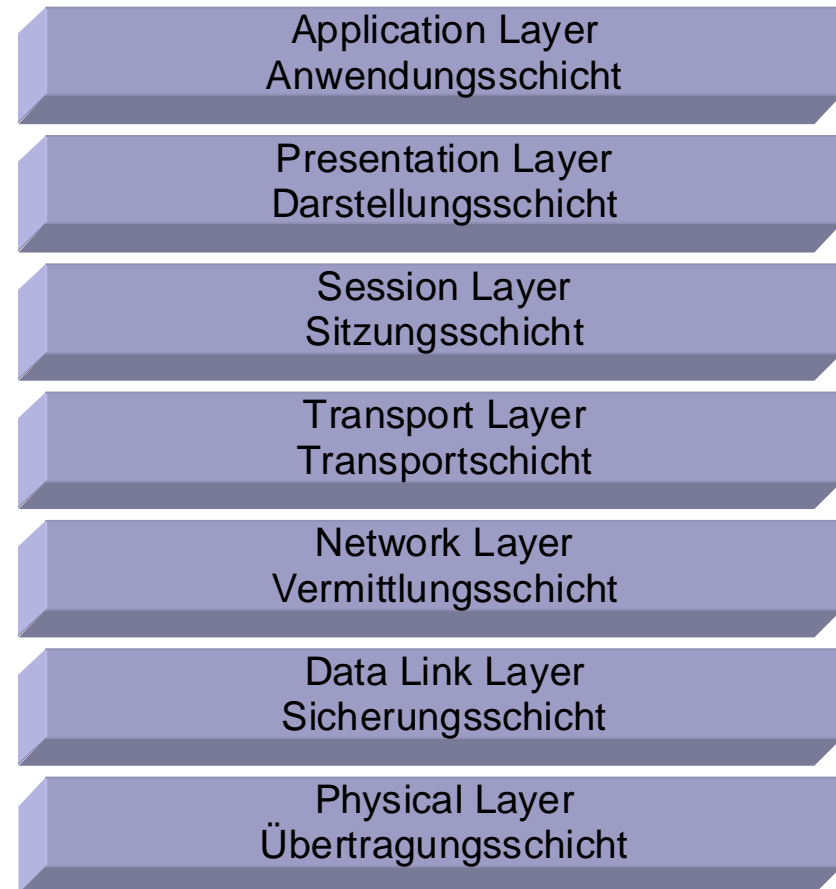
→? Frage nach Determinismus der Verfahren.

## Aufgaben der **Sicherungsschicht**:

- Aufteilung der Nachricht in **Datenpakete**
- Regelung des **Medienzugriffs**
- **Flusskontrolle**
- **Fehlererkennung**
  - Prüfsummen
  - Paritätsbits

## Aufgaben der **Vermittlungsschicht**:

- Aufbau von **Verbindungen**
- **Weiterleiten** von Datenpaketen
  - Routingtabellen
  - Netzwerkadressen



# Echtzeitkommunikation

## Aufgaben der **Transportschicht**:

- **Segmentierung** von Datenpaketen
- **Transport** zwischen Sender und Empfänger
- **Staukontrolle**

## Aufgaben der **Sitzungssicht**:

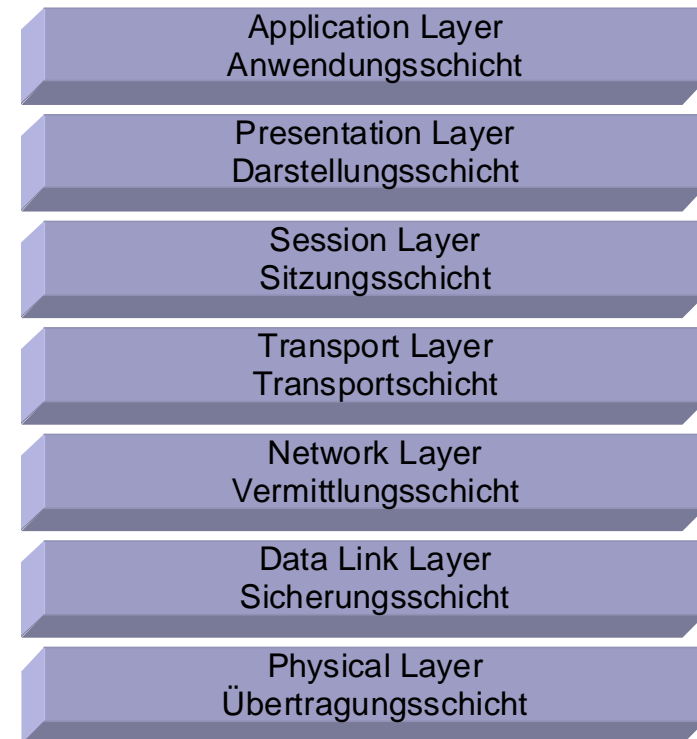
- Auf- und Abbau, Aufrechterhalten von **Verbindungen** auf Anwendungsebene
- Dienste zur **Synchronisation** der Kommunik.
- **Sicherheitsmechanismen** (Passwörter, etc.)

## Aufgaben der **Darstellungsschicht**:

- **Datenkonvertierung** ("Marshalling")
- **Datenkompression**
- **Verschlüsselung**

## Aufgaben der **Anwendungsschicht**:

- Anwendungsspezifische **Kommunikationsdienste**
  - Email, remote login, VoIP, ...



# Echtzeitkommunikation

Die **gesamte Übertragungszeit** einer Nachricht setzt sich damit zusammen aus:

- Zeit für die **Umsetzung der Protokolle** auf den einzelnen Schichten beim **Sender** (Schicht 7 bis 1)
- Wartezeit für **Medienzugriff**
- **Signallaufzeit** auf dem Medium
- Zeit für die **Umsetzung der Protokolle** auf den einzelnen Schichten beim **Empfänger** (Schicht 1 bis 7)

D.h. mit **jeder Schicht** vergrößert sich die **Übertragungszeit**. Zusätzlich erhöht jede Schicht auch den **benötigten Speicher**, da eine Nachricht typischerweise auf jeder Schicht **konvertiert** und **zwischengespeichert** werden muss.

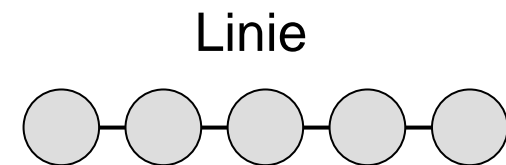
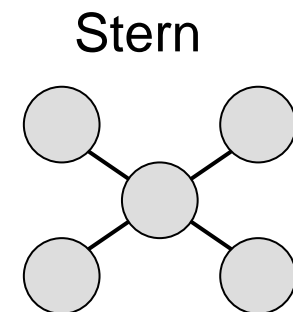
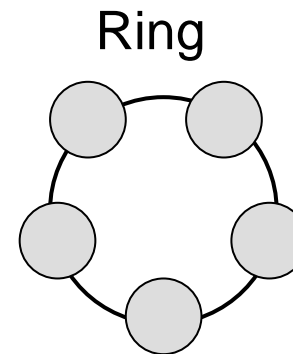
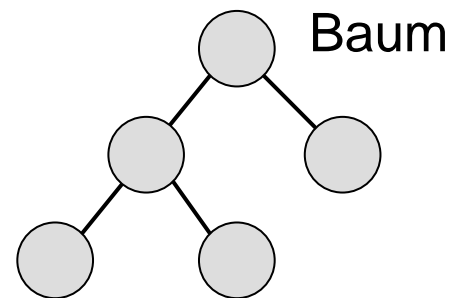
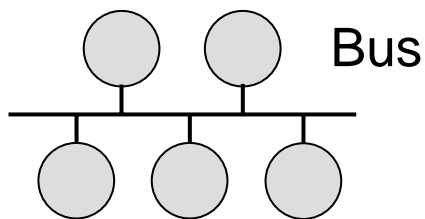
Daher **reduziert** man die **Anzahl der Schichten** in einem Echtzeitsystem meist auf:

- Anwendungsschicht
- Sicherungsschicht
- Übertragungsschicht

Die **Topologie** eines Netzwerks, d.h. die Struktur der Verbindungen zwischen den Knoten, hat insbesondere Einfluss auf seine

- **Ausfallsicherheit,**
- **Performance,**
- **Kosten,**
- **Skalierbarkeit.**

Mögliche Topologien:

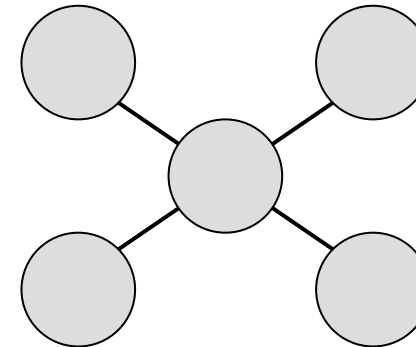


- Charakteristische **Eigenschaften** einer Topologie:
  - **Durchmesser**
    - maximale direkte **Entfernung** zwischen 2 Knoten **in Hops**. Der Durchmesser ist damit ein **Maß** für die maximale **Übertragungszeit**.
  - **Grad**
    - Anzahl **Verbindungen je Knoten**. Kann für alle Knoten gleich oder unterschiedlich sein. Je höher der Grad, desto mehr Verbindungen und damit höherer Verkabelungsaufwand (Kosten, Platz, Gewicht).
  - **Skalierbarkeit**
    - **Aufwand** (Anzahl Links + Knoten) **für die Erweiterung** des Netzwerks unter Beibehaltung seiner Topologie
  - **Konnektivität**
    - minimale Anzahl von Links, die getrennt werden müssen, um das Netz zu zerstören. Die Konnektivität ist ein Maß für die **Ausfallsicherheit** des Netzes.

## Stern-Topologie

### Vorteile:

- Ausfall eines Endgeräts hat keine Auswirkungen auf den Rest
- leicht skalierbar
- kein Routing erforderlich



### Nachteil:

- Ausfall des Verteilers bewirkt Ausfall des gesamten Netzes

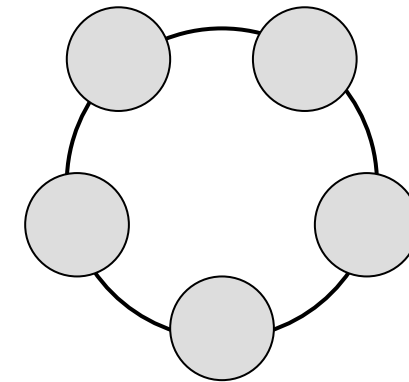
### Beispiel:

- Telefonnetz

## Ring-Topologie

### Vorteile:

- einfaches Routing
- leicht skalierbar, Grad bleibt konstant
- Jeder Knoten fungiert als Verstärker



### Nachteile:

- niedrige Konnektivität; Ausfall eines Knotens unterbricht die Kommunikation (in einer Richtung).
- hoher Durchmesser; lange Übertragungszeit zu entfernten Knoten
- hoher Verkabelungsaufwand

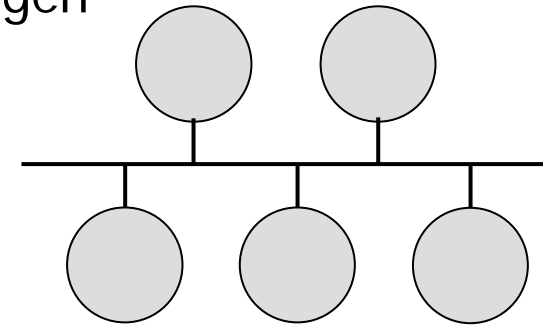
### Beispiel:

- Token Ring

## Bus-Topologie

### Vorteile:

- Ausfall eines Knotens hat keine Auswirkungen auf den Rest
- leicht skalierbar
- einfache Verkabelung



### Nachteile:

- geringe Konnektivität; nur ein Kabel
- Es kann immer nur eine Station senden; Datenstau
- Jeder Knoten muss jede Sendung mithören
- Stromverbrauch des Senders steigt mit der Anzahl der Knoten am Bus

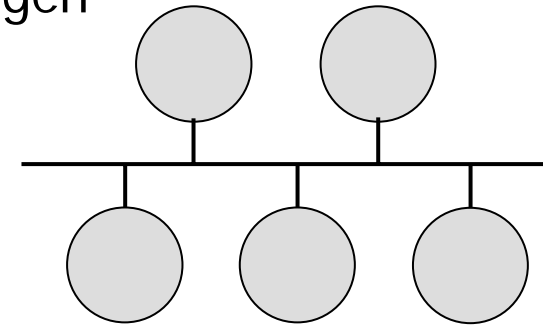
### Beispiel:

- CAN

## Bus-Topologie

### Vorteile:

- Ausfall eines Knotens hat keine Auswirkungen auf den Rest
- leicht skalierbar
- einfache Verkabelung



### Nachteile:

- geringe Konnektivität; nur ein Kabel
- Es kann immer nur eine Station senden; Datenstau
- Jeder Knoten muss jede Sendung mithören
- Stromverbrauch des Senders steigt mit der Anzahl der Knoten am Bus

### Beispiel:

- CAN

- **Bemerkung:**
- Die **physikalische Topologie** eines Netzes muss nicht zwangsläufig mit seiner **logischen Topologie** übereinstimmen.
- Z.B. kann sich ein Stern-Netzwerk auch wie ein Ring verhalten. Dabei leitet der Verteiler eine Nachricht immer an den jeweils nächsten Knoten weiter.

# Verteilte Systeme - Busszugriff

- **CSMA/CA** - Carrier Sense Multiple Access with Collision Avoidance - Protokollvariante von CSMA/CD, bei der Kollisionen durch Vorausmassnahmen vermieden werden
- **CSMA/CD** - Carrier Sense Multiple Access with Collision Detection - Protokollstrategie, bei der nach Feststellen einer offenen Leitung jeder Teilnehmer senden kann und bei einer waehrend des Sendens auftretenden Kollision fuer unterschiedl. (per Zufallsgenerator bestimmte) Zeit unterbricht (Konkurrenzbetrieb)
- **CSMA/CE** - Carrier Sense Multiple Access with Collision Elimination - Protokollvariante von CSMA/CD, bei der Kollisionen automatisch erkannt und behoben werden
- **CSMA/CP** - Carrier Sense Multiple Access with Collision Prevention - Protokollvariante von CSMA/CD, bei der Kollisionen durch Vorausmassnahmen verhindert werden.

## Beispiele:

- CSMA/CD: Ethernet
- CSMA/CA: RS232 mit RTS/CTS, CAN
- CSMA/CE und /CP: bei LAN-Protokollen

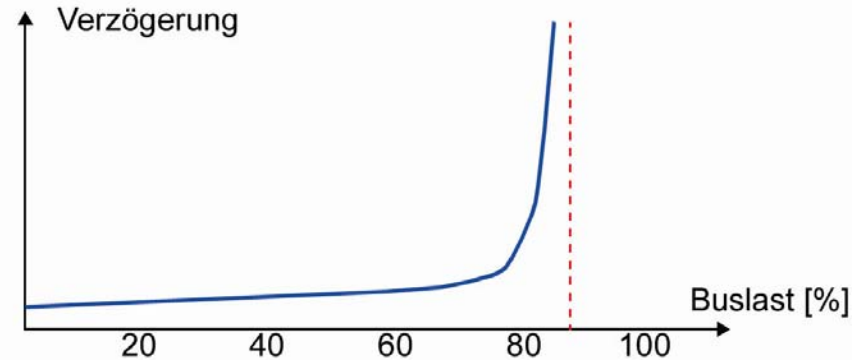
# Anforderungen an Bussysteme

Randbedingungen für echtzeitfähige Bussysteme:

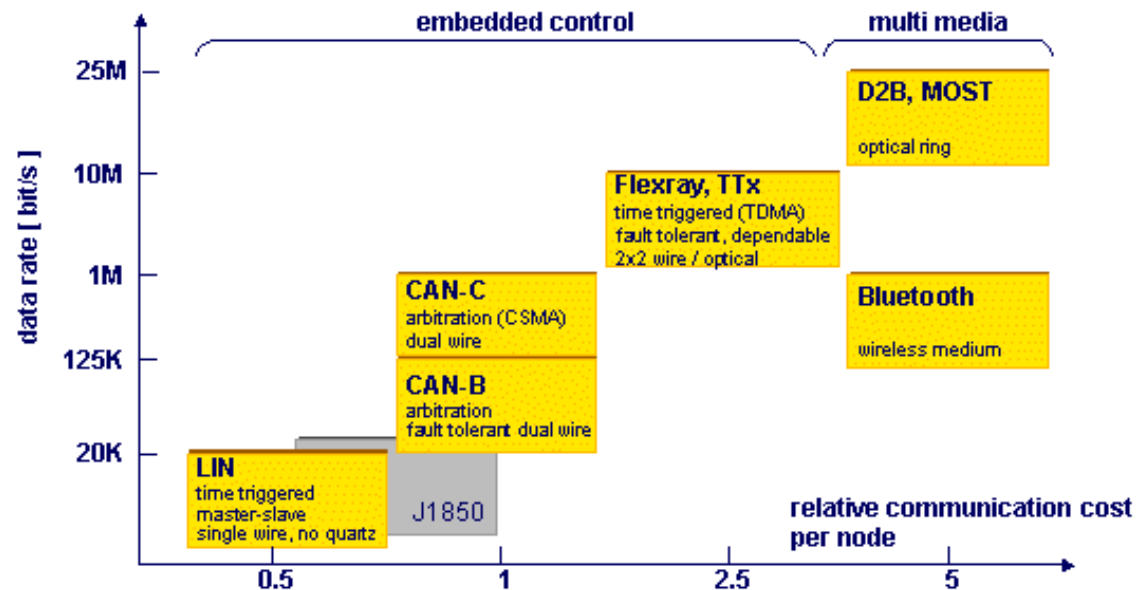
- Reaktionszeit
- deterministisch
- Bandbreite
- Sicherheit

Weiterhin:

- Kosten
- Störanfälligkeit
- Redundanz



*CSMA/CD*-Buszugriff von *Ethernet*: Übertragungszeiten nehmen mit steigender Buslast stark zu



# Eigenschaften von CAN

## Control Area Network (1981 von Intel und Bosch entwickelt)

- Ereignisgesteuert: Alle Knoten sind zu jedem Zeitpunkt gleichberechtigt auf das Kommunikationsmedium zuzugreifen.
- **C**arrier **S**ense, **M**ultiple **A**ccess / **C**ollision **D**etection + Methode zum Erkennen und Auflösen von Kollisionen
  - Senden können, sobald Bus frei ist (Carrier Sense), mehrerer Knoten gleichzeitig (Multiple Access)
  - Dabei werden mögliche Kollisionen erkannt (CD) und behoben.
- Bitweise Arbitrierung: Alle aktuellen Sender legen bitweise nacheinander die Bits ihrer Priorität auf den Bus
  - die höhere Priorität sendet weiter
  - die niedrigere(n) schalten sofort auf Empfang um
  - realisiert durch dominante („0“) / rezessive Bits („1“)
- Beachte Gegensatz zu Ethernet (CSMA/CD)
  - bei entdeckter Kollision nehmen **alle** Sender ihre Signale vom Bus
  - die sendewilligen Knoten warten jeweils eine zufällig gewählte Zeit und greifen dann erneut auf den Bus zu
  - Probleme bei hoher Netzlast:
    - unbestimmte Reaktionszeit
    - Bandbreite nimmt ab

# Steckbrief CAN

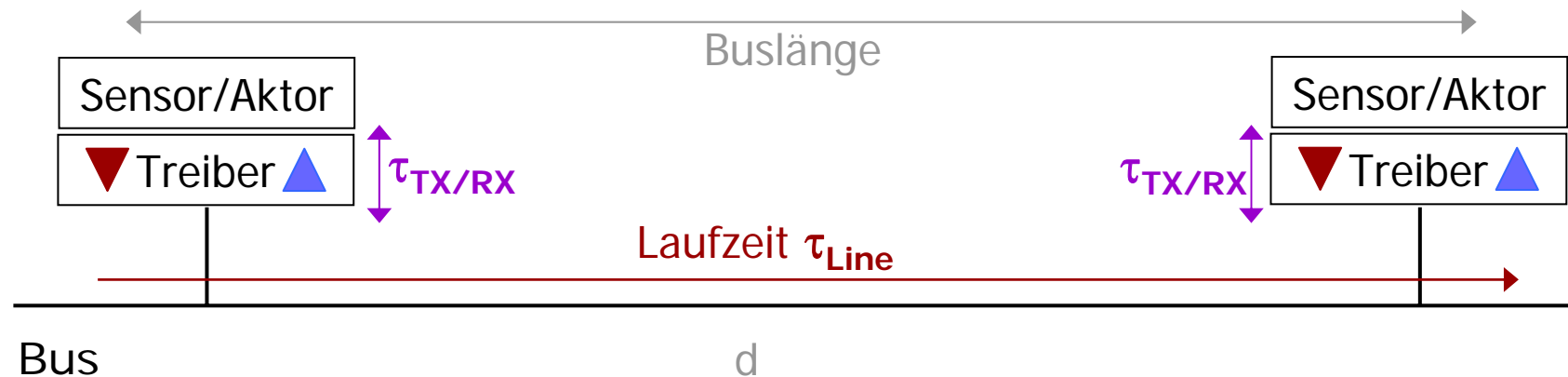
- Implementierung der **OSI**-Schichten **1, 2** und **7**
- Broadcast-Bus bis 1 MBit/s
- Nachrichten (message) 0-8 Byte
  - Daten Message
  - **R**emote **T**ransmission **R**equest Message
  - Jeder Knoten versendet aus seiner Queue zuerst die Nachricht mit der höchsten Priorität
- Eindeutiger Identifier (unique identifier)
  - kein expliziter Empfänger
  - Empfänger implementieren Filter für Nachrichten
  - **C**ollision **D**etection +
  - **C**ollision **R**esolution: höhere Priorität erhält Vorrang
- CAN message format:
  - 47 Kontroll-Bits
  - 0-64 Daten-Bits
  - 0-19 Stopf-Bits (stuffing bits)

Wenn jeder Teilnehmer des Busses ein Bit **erhalten und quittiert** haben muss, bevor das nächste Bit gesendet wird, wie hoch darf die **Bitrate** bei einer gegebenen **Buslänge** sein?

Buslänge d [m]	f Bits/s	CAN-Spezifikation
40 m	< 2.5 Mbit/s	1 Mbit/s
100 m	< 1 Mbit/s	500 kbit/s
500 m	< 200 kbit/s	125 kbit/s

$$\frac{2}{3} \cdot c = \frac{d}{t} \quad \Leftrightarrow \quad t = \frac{d}{c} \cdot \frac{3}{2} \quad \Leftrightarrow \quad f = \frac{1}{2t} = \frac{c}{d} \cdot \frac{1}{3} \quad (\text{Gl. 1-3})$$

# Bitraten/Buslänge-Beziehung (CAN)



Bei CAN wird die Reaktion anderer Busteilnehmer auf das Aussenden einer Bitfolge erwartet. Ein von einem **zweiten** Sender ausgesandtes Bit muss in ca. **2/3** der Bit-Zeit wieder beim **ersten** Sender eingetroffen sein, um dort eine Entscheidung zu ermöglichen. Bei **verlorener Arbitrierung** muss der erste Knoten innerhalb von **1/3** der Bitzeit **auf Empfang** umstellen.

Weiterhin ist die zu beachten:

- die Laufzeit auf der Busleitung
- Die Laufzeit durch die Sende/Empfangs-Treiber

Daraus ergibt sich die Abschätzung:

$$\tau_{Line} = \frac{d}{c} \cdot \frac{3}{2} \quad (\text{Gl. 1}) \quad \frac{2}{3} \cdot t_{Bit} \geq 4 \cdot \tau_{TX/RX} + 2 \cdot \tau_{Line} \quad (\text{Gl. 4})$$

# Bitraten/Buslänge-Beziehung (CAN) (2)

Buslänge [m]	max. Bitrate
40	1 MBit/s
100	500 kBit/s
500	125 kBit/s

Tab. 1: CAN Spezifikation

Geht man von  $\tau_{\text{Line}}$  aus  $c_{\text{Line}} = 17 \text{ cm/ns}$  und  $\tau_{\text{TXRX}} = 25 \text{ ns}$  aus, erhält man folgende „Worst-Case-Designregel“:

$$\text{Buslänge} \times \text{Bitrate} < 40 \text{ m} \times 1 \text{ Mbit/s}$$

(Gl. 2)

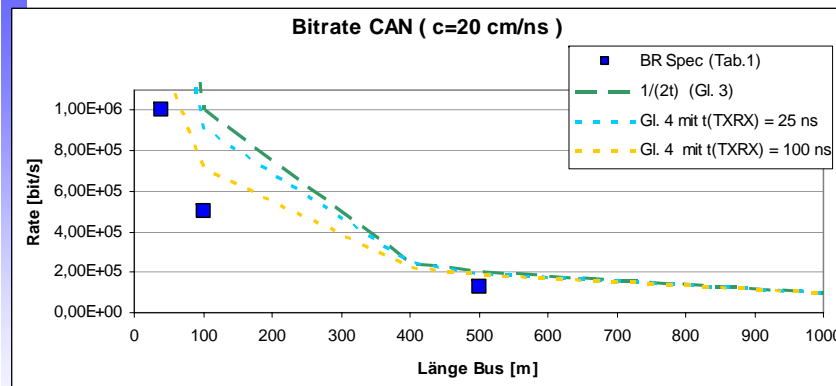
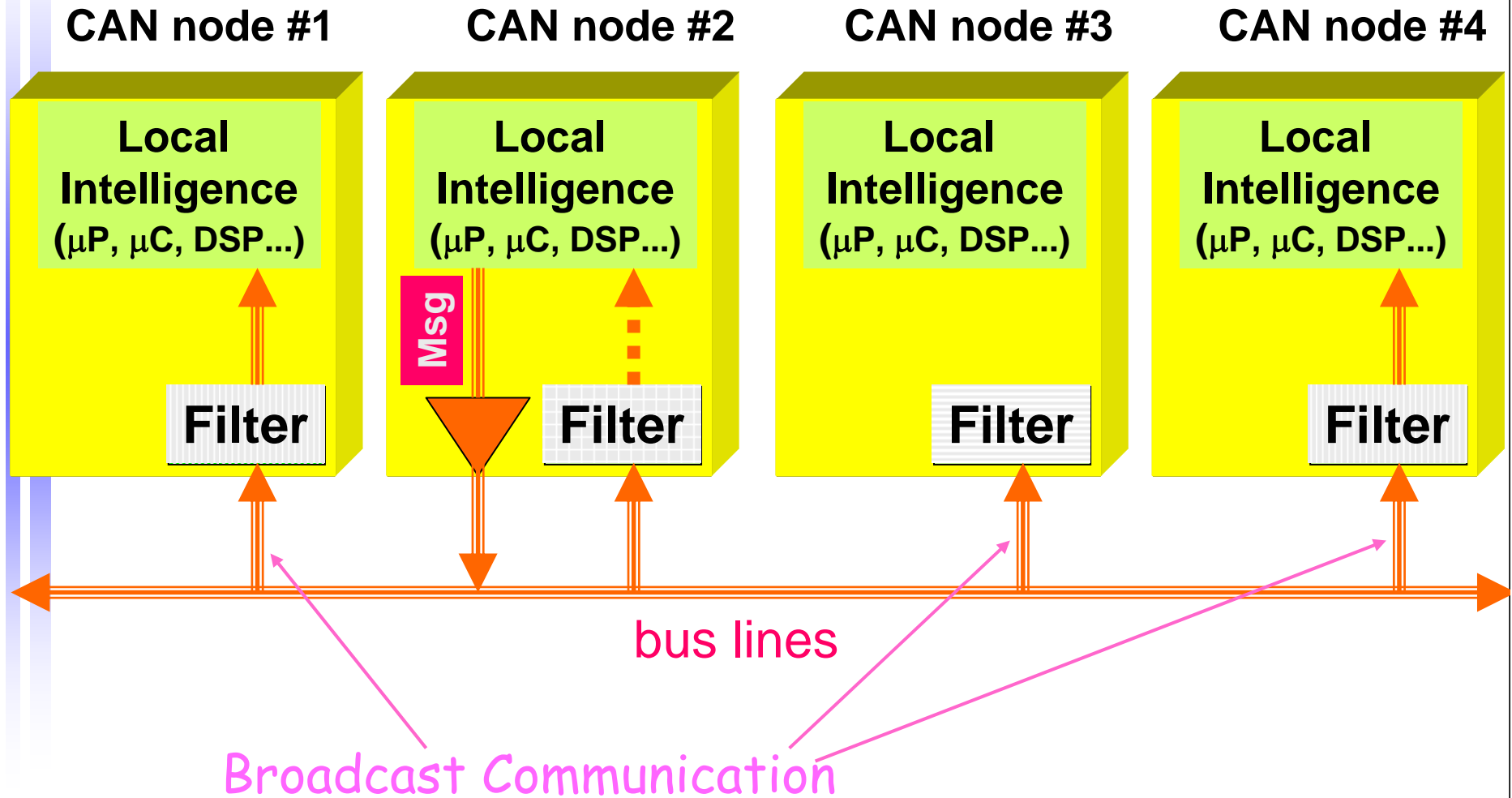


Bild: Maximal möglichen Bitrate in Abhängigkeit von der Kabellänge für CAN

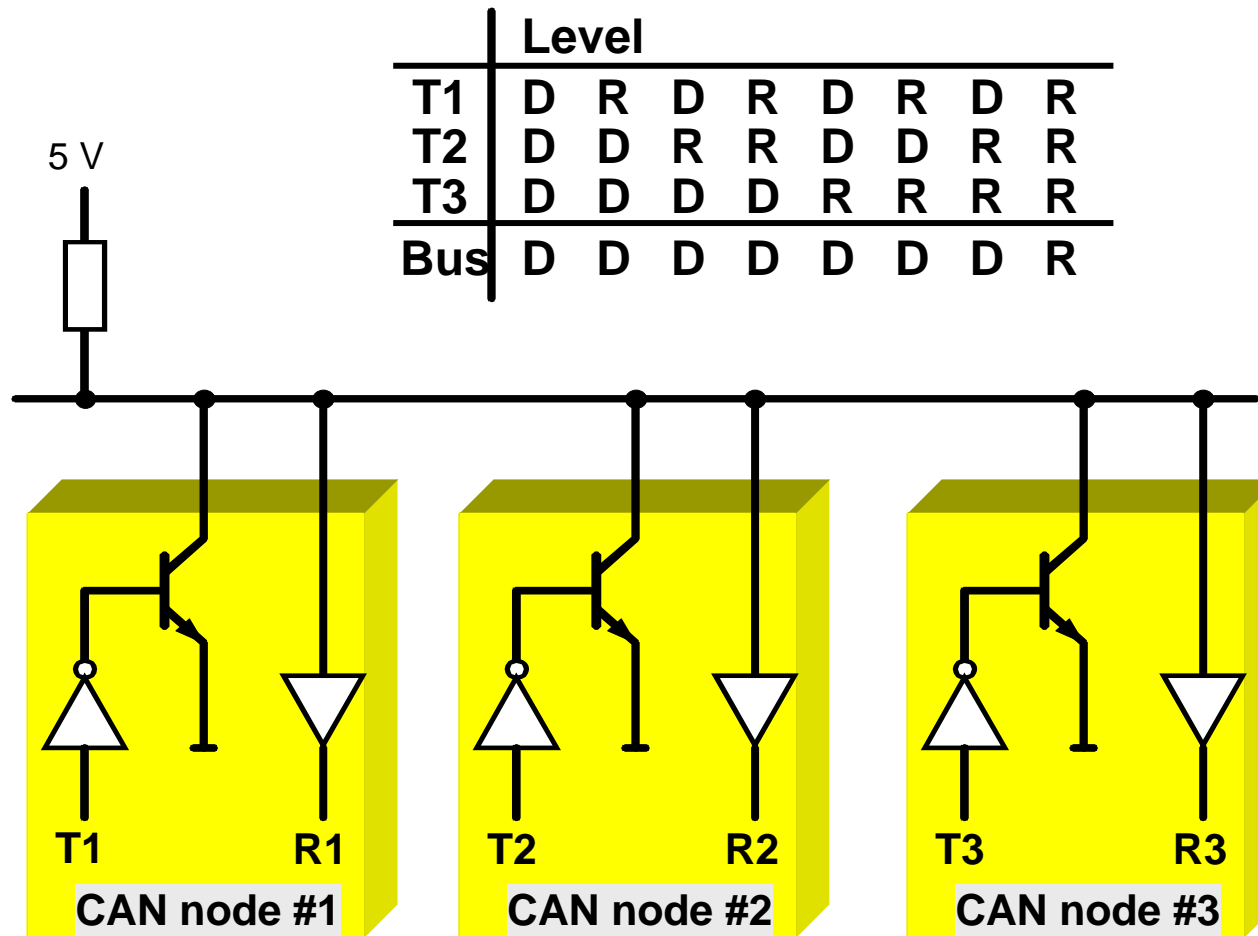
Länge [m]	Max Rate ( $\tau_{\text{TXRX}}=25\text{ns}$ )	Max Rate ( $\tau_{\text{TXRX}}=100\text{ns}$ )
10	< 5 Mbit/s	< 2 Mbit/s
40	< 2 Mbit/s	< 1,25 Mbit/s
100	< 900 kbit/s	< 710 kbit/s
500	< 195 kbit/s	< 185 kbit/s

Praktisches Limit für 1 MBit/s:  
Buslänge ca. 9 m

# CAN: Broadcast-Prinzip, Filter

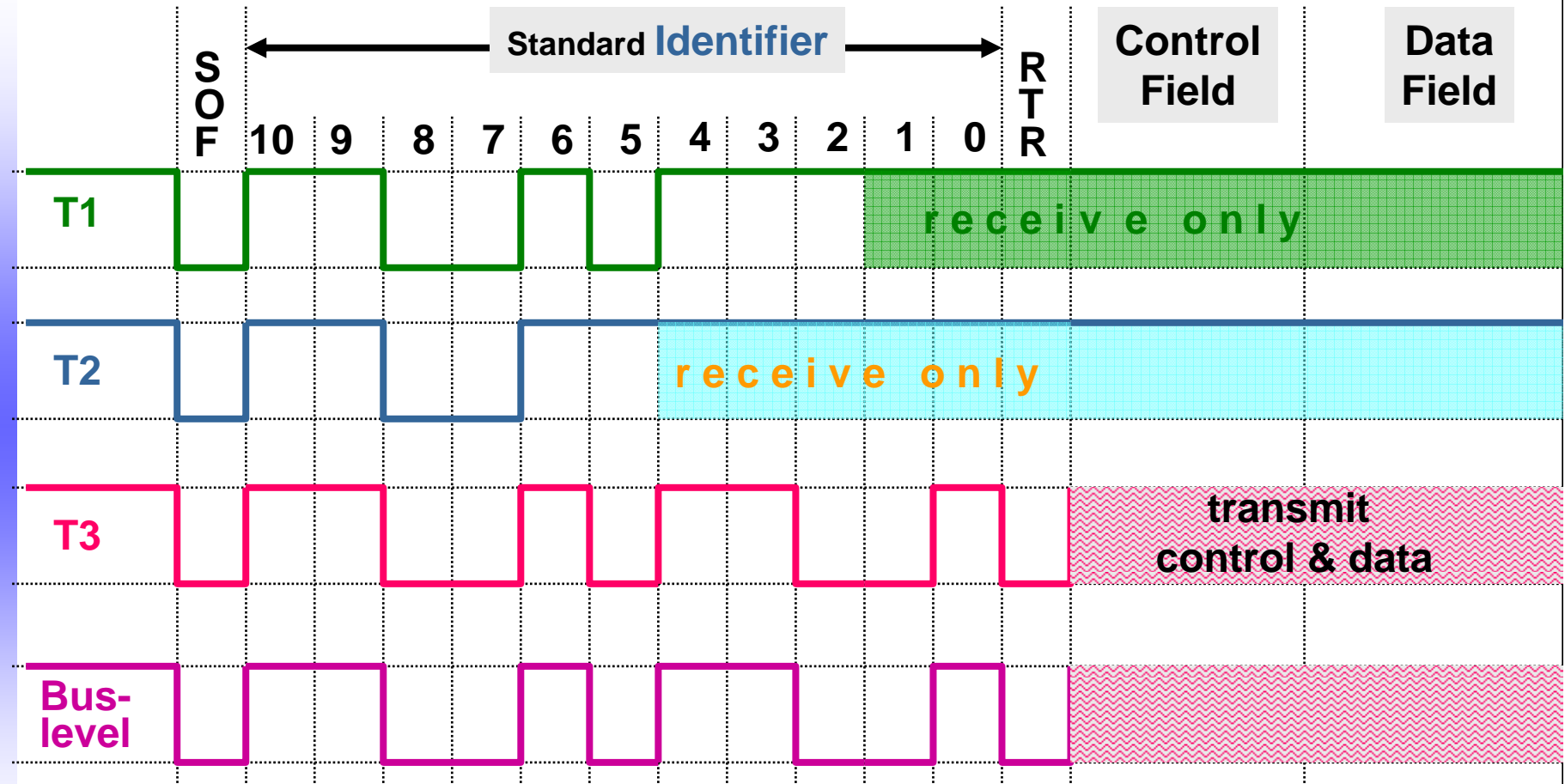


# CAN: Realisierung bitweise Arbitrierung



Verschaltung der Open-Collector-Ausgänge wirkt wie ein „wired“ AND.  
 Dominantes Bit „0“ setzt sich gegenüber rezessivem Bit „1“ durch.  
 → CSMA/CD + **A**rbitration on **M**essage **P**riority

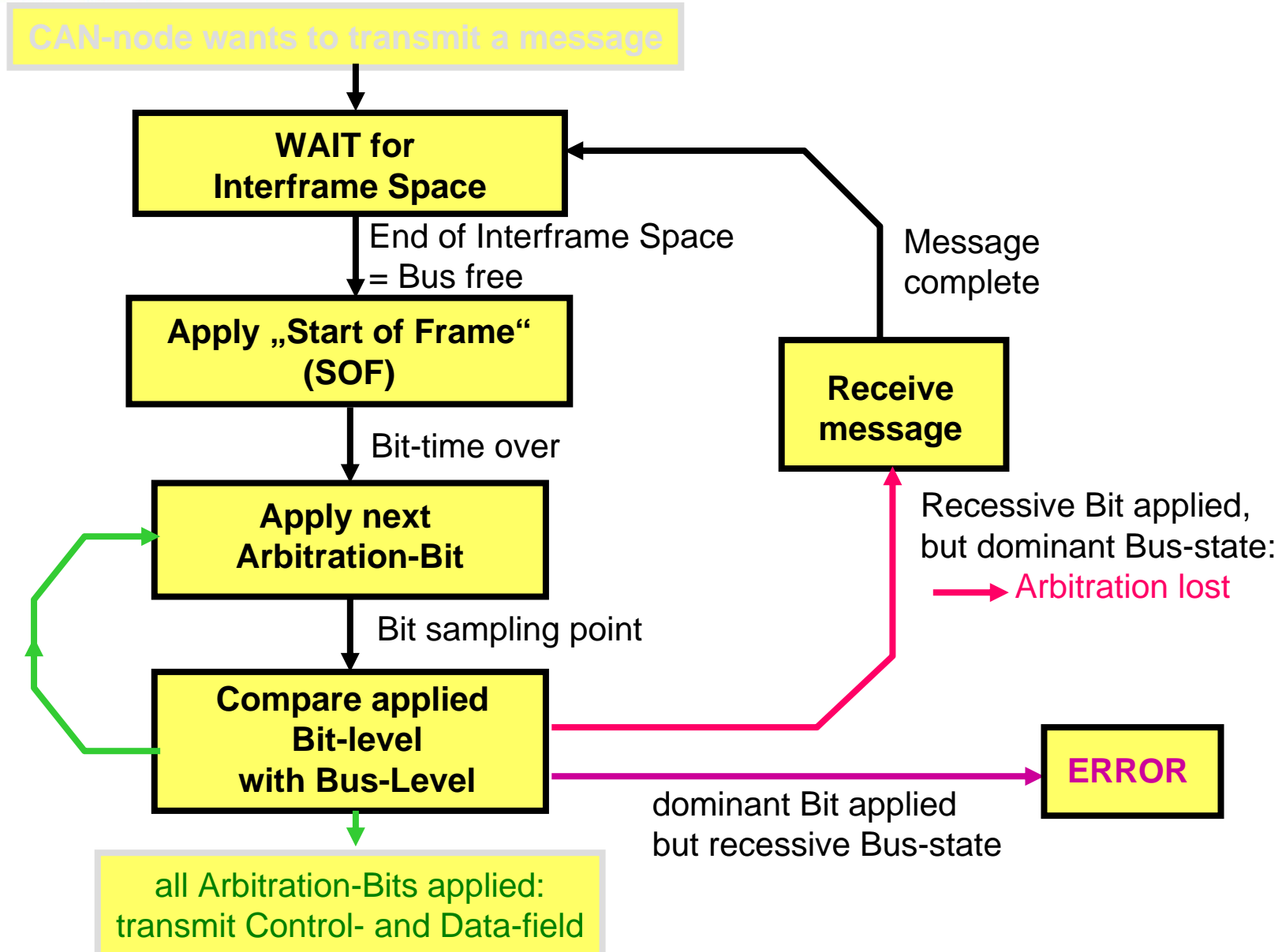
# CAN: Realisierung bitweise Arbitrierung (2)



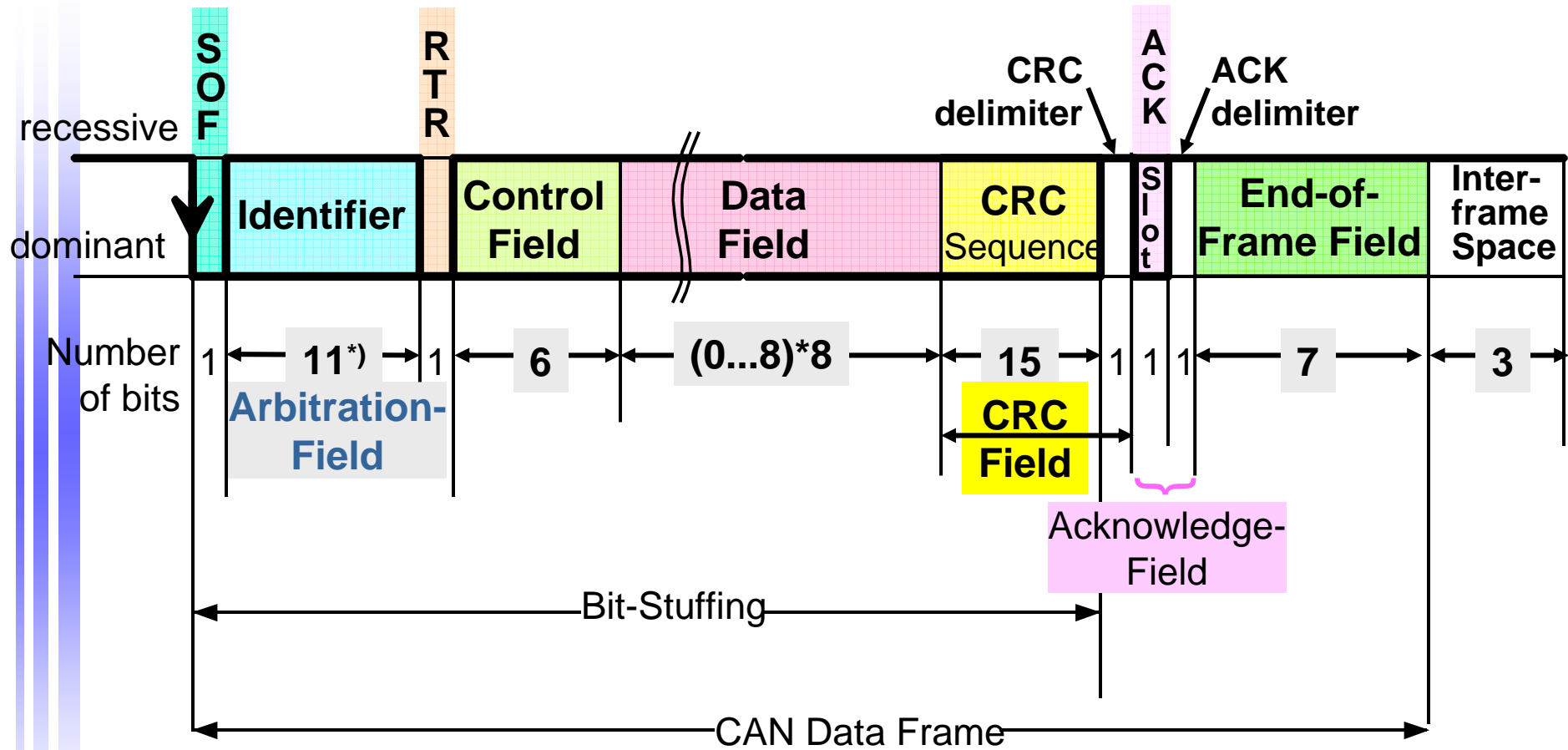
SOF: Start-of-Frame  
 Identifier. 11 Bit (CAN 2.0A)  
 29 Bit (CAN 2.0B)

RTR: Remote Transmission Request  
 enthält Daten oder  
 fordert zum Senden entspr. Daten auf

# CAN: Ablauf Arbitrierung



# CAN: Frame Format

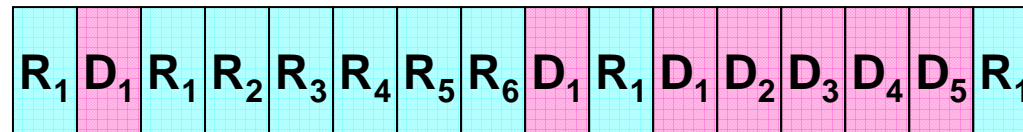


\*) Standard-Format: 11 Bit  
 Extended: 29 Bit

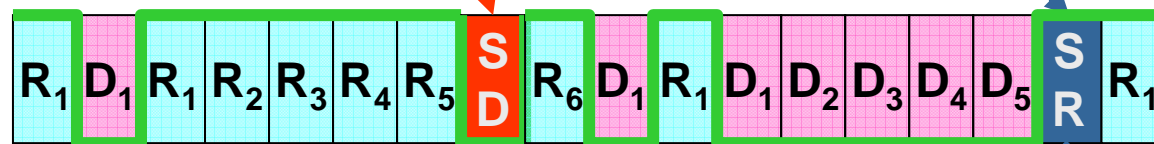
## Synchronisation auf dem Bus (bit stuffing)

Um eine zeitliche Synchronisation zu gewährleisten, sind hin und wieder Flanken (Übergänge  $0 \rightarrow 1$  oder  $1 \rightarrow 0$ ) notwendig. Bei Folgen von 5 und mehr gleichen Bits werden vom Sender ein inverses Bit eingefügt (*stuffing bit*) welches beim Empfänger wieder entfernt wird.

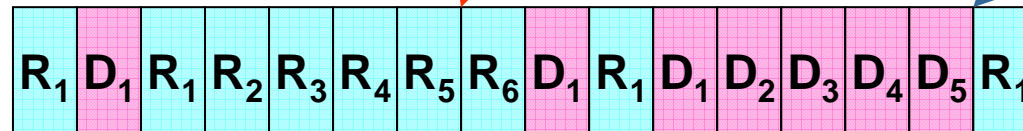
Bits to be transmitted:



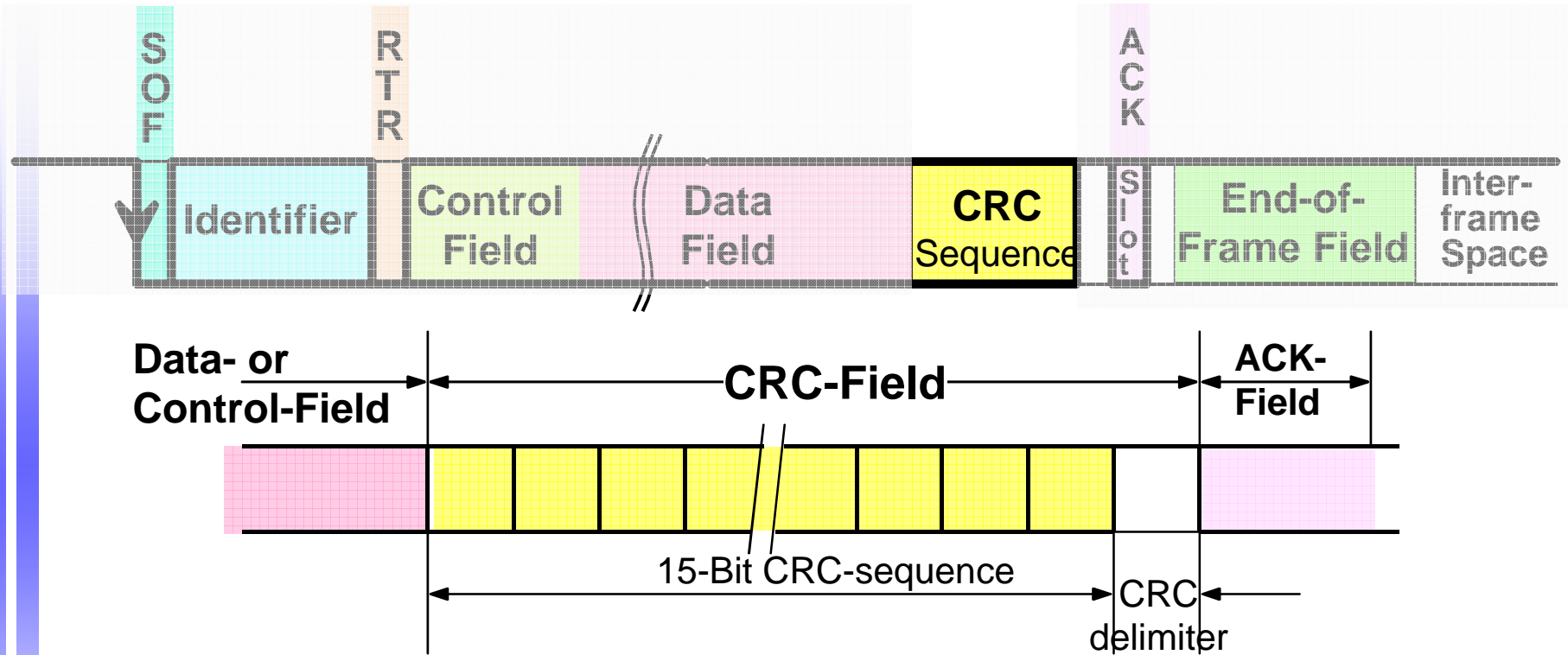
Bits at the Bus:



Received bits (Stuff-bits deleted):



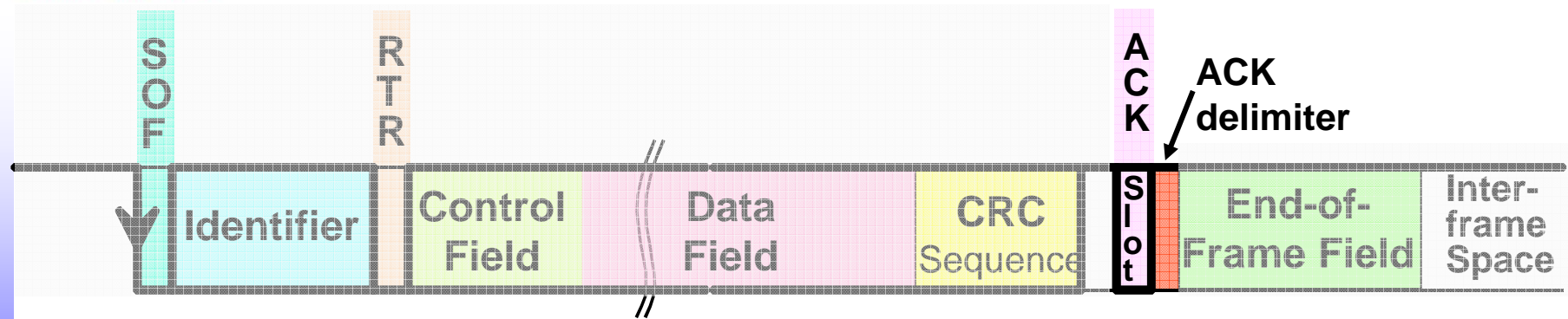
# CAN: Cyclic Redundancy Check



15 Bit CRC → Hamming-Distanz 6

→ Bis zu 5 Bit-Fehler können erkannt *und* korrigiert werden

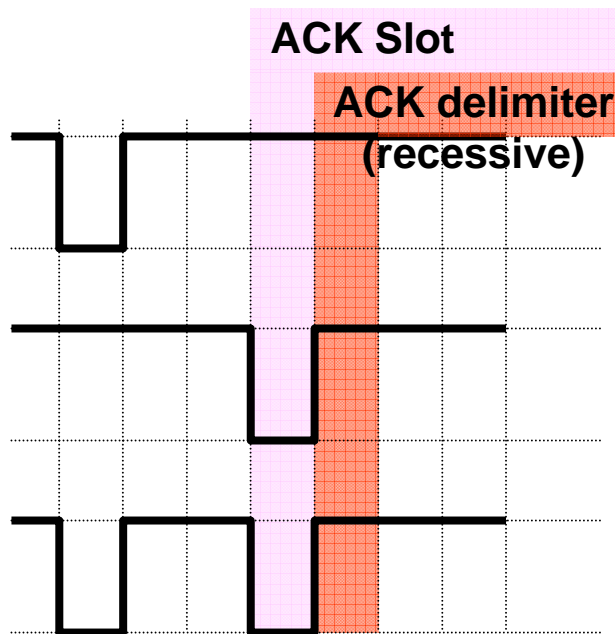
# CAN: Acknowledge



**Node A**  
 ○ Receive  
 ● Transmit

**Node B**  
 ● Receive  
 ○ Transmit

**CAN Bus**



Alle Knoten checken CRC, Stuffing-Bits etc.

Alle empfangenden Knoten quittieren – ungeachtet Korrektheit – mit einem dominanten Bit

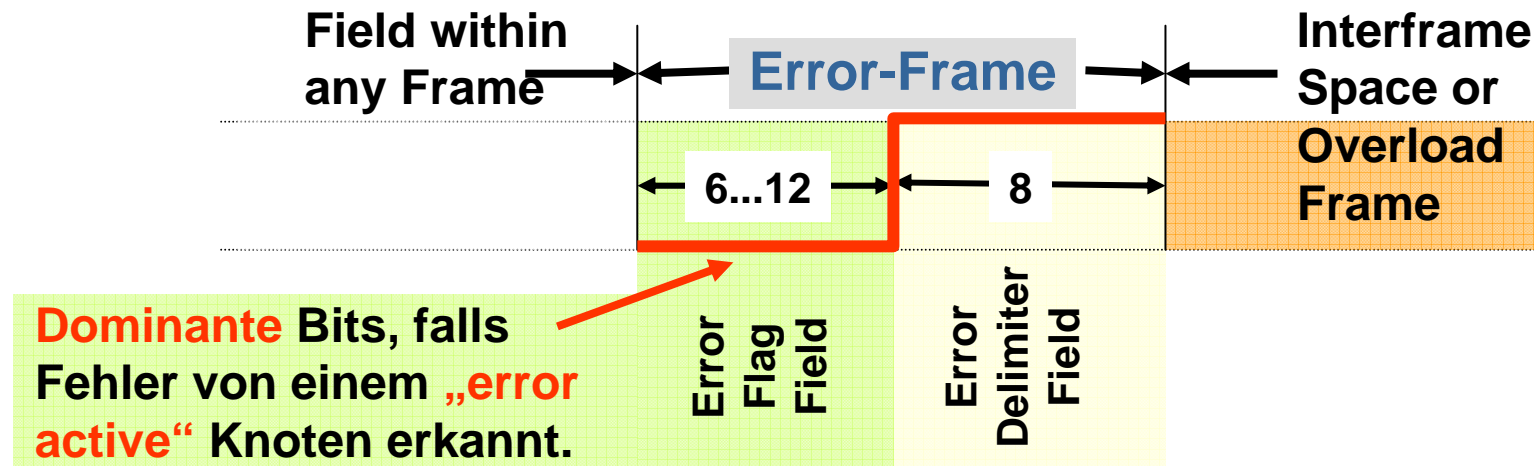
D.h. mindestens 1 Knoten hat korrekt empfangen

Falls nicht, wird Sendung wiederholt  
Kein Error-Frame!

End-of-Frame: 7 rezessive Bits

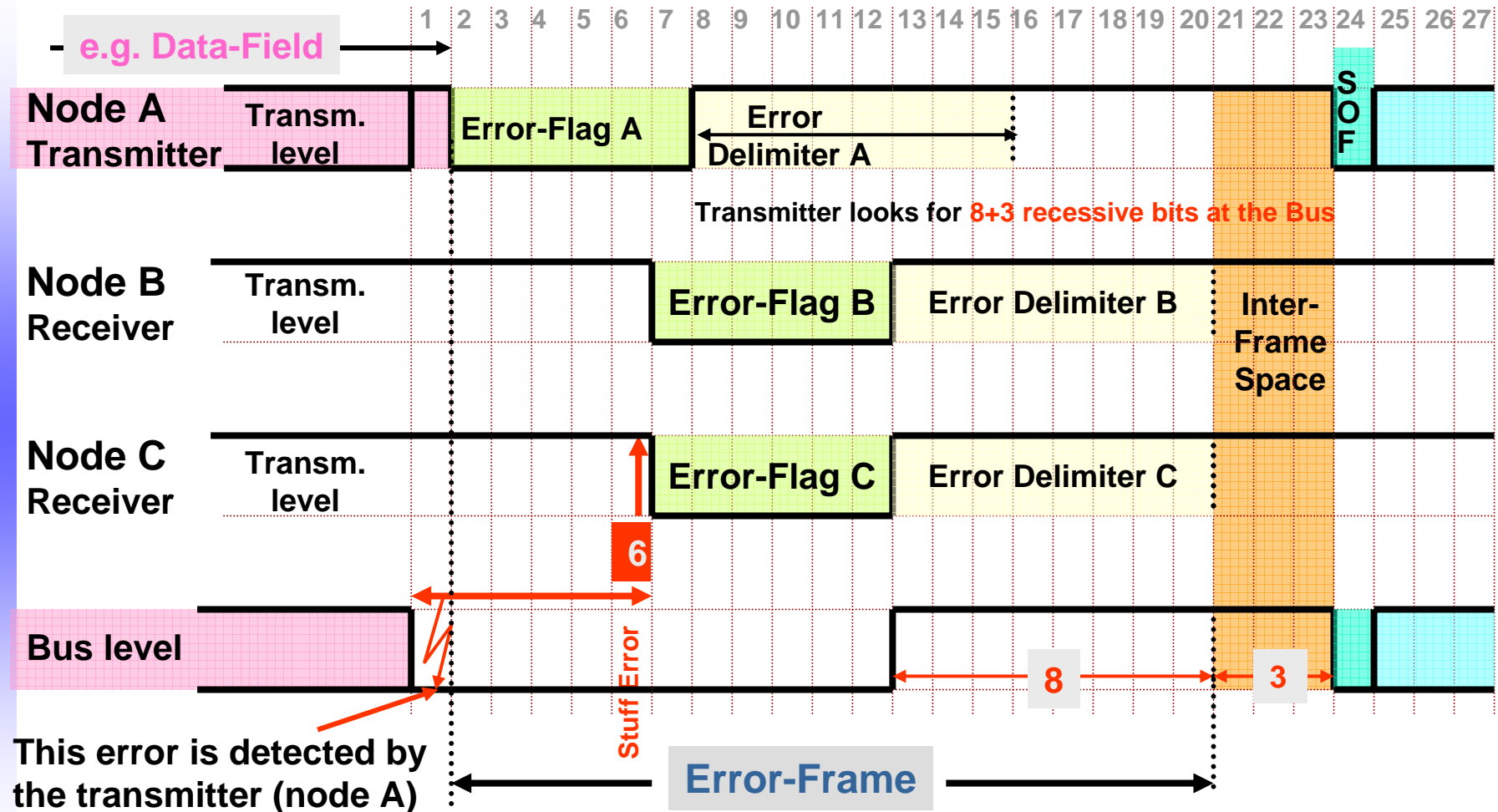
Entdeckt ein Knoten einen Fehler, sendet er einen Error-Frame aus

# CAN: Fehlerbehandlung



- „error active“ Knoten unterbricht bei Erkennung eines Fehler die Übertragung durch das Senden eines Error-Flags.
- „active error flag“ besteht aus 6 aufeinanderfolgenden dominanten Bits
- Alle Knoten erkennen eine Verletzung der Stopf-Bit (stuffing) Regel und senden ihrerseits einen Fehler-Rahmen
- → 6-12 dominante aufeinanderfolgende Bits auf dem Bus
- Nach Senden des „error delimitierte“-Feldes → normaler Busbetrieb

# CAN: Fehlerbehandlung (2)



# CAN: Fehlererkennung

- Die Wahrscheinlichkeit  $\rho$ , dass ein fehlerhafter CAN-Standard-Frame nicht erkannt wird, beträgt:

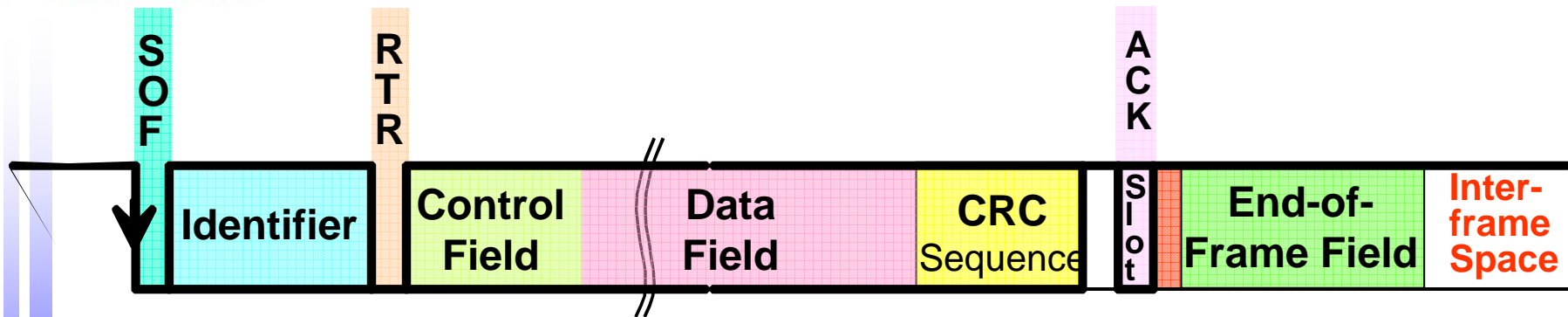
$$\rho = 4.7 \times 10^{-11} \times \text{Fehlerrate}$$

- Beispiel:  
1 Bit-Fehler pro 0.7 s,  
Übertragungsrate: 500 kBit/s,  
Betrieb: 8 h / d, 365 d / a (d:Tag, a:Jahr)

→ statistisch: **1** unerkannter Fehler in **1000** Jahren

- Extended CAN etwas schlechter als Standard CAN

# CAN: Rahmen Format



- 1 Startbit
- + 11 Identifierbits
- + 1 RTR Bit
- + 6 Controlbits
- + 64 Datenbits
- + 15 CRC Bits
- + 1 CRC Delimiter
- + 19 (maximale Anzahl) Stuffbits
- + 1 ACK Slot
- + 1 ACK Delimiter
- + 7 EOF Bits
- + 3 IFS (Inter Frame Space) Bits
- = 130 Bit

CAN 2.0A Identifier = 11 Bit  
 → Summe ≤ 130 Bit

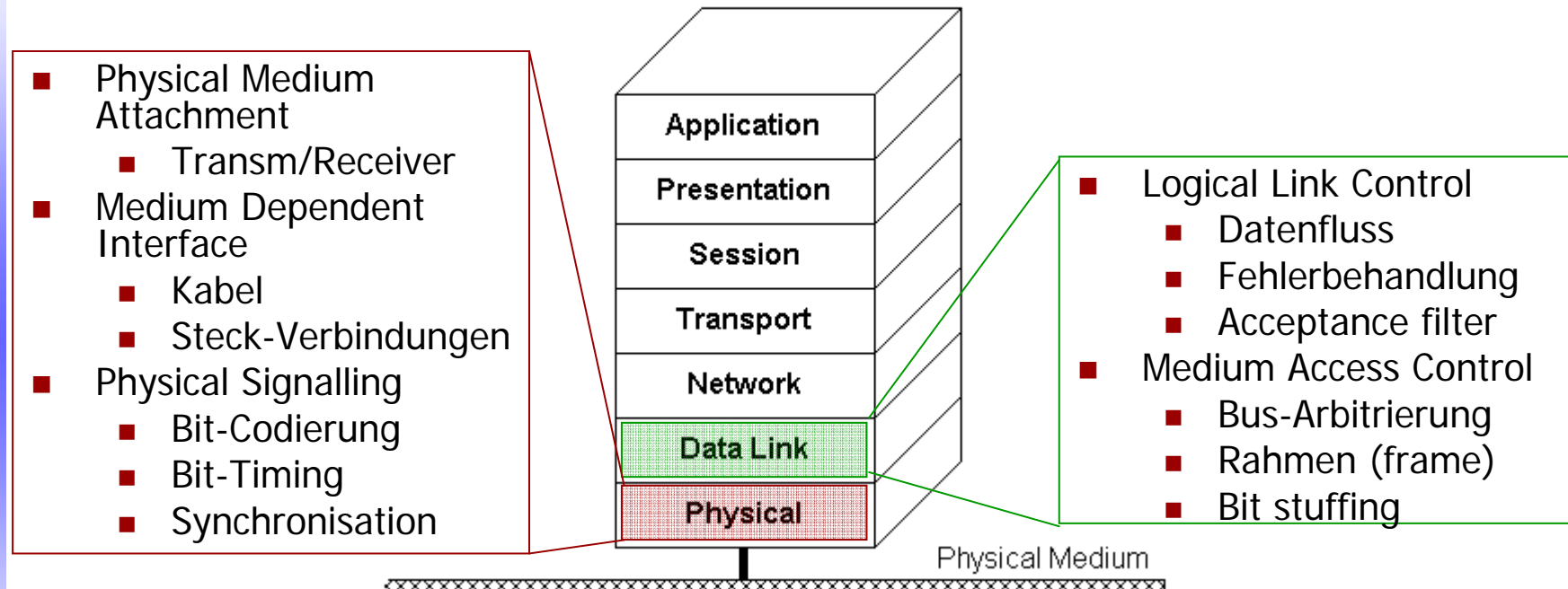
CAN 2.0B Identifier = 29 Bit  
 → Summe ≤ 154 Bit

Bei 1MBit pro Sekunde (1 bit/μs):  
 Übertragungsdauer bei

- Standard-Frame: ≤ 130 μs
- Extended Frame: ≤ 154 μs

# CAN: die physikalische Schicht

## The OSI Reference Model



# CAN: Bit-Timing und Synchronisation

Prinzipiell müssen folgende Fälle bei der Kommunikation zwischen Knoten berücksichtigt werden:

- Oszillator des Empfängers läuft etwas schneller/langsamer
- Laufzeit des Signals auf dem Bus

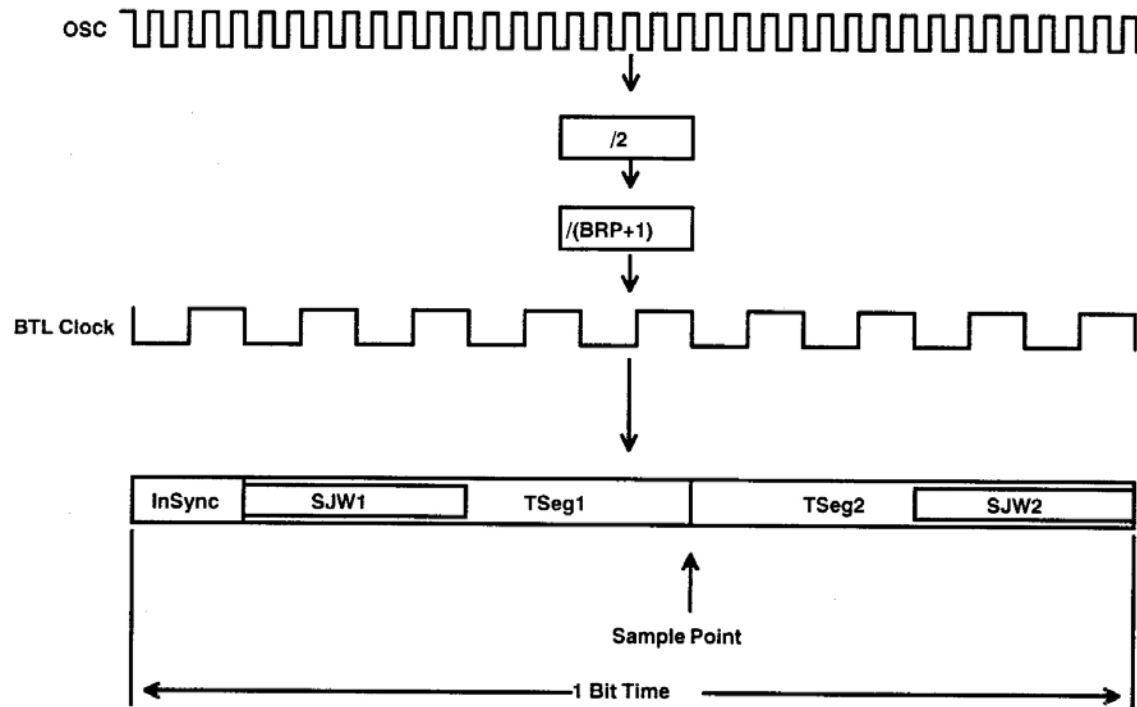
Man unterscheidet zwei Arten der **Synchronisation**:

- **harte Synchronisation**: wird an der fallenden Flanke des Startbits durchgeführt (oder an weiteren fallenden Flanken). Legt den absoluten Beginn eines Bits fest.
- **weiche Synchronisation**: wird innerhalb des CAN-Telegramms an Flanken durchgeführt, um die zeitliche Länge von Bits anzupassen.

*Frage: Warum ist fallende Flanke dafür günstiger als ansteigende?*

(siehe S. 24)

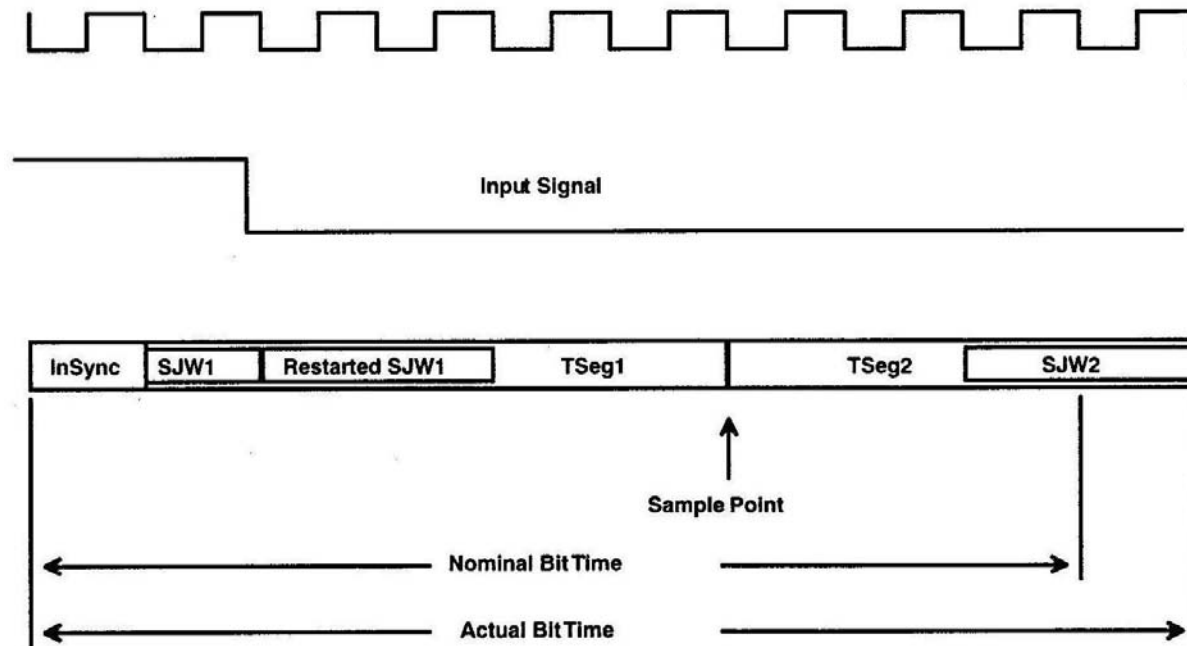
# Prinzip: Bit-Timing und Synchronisation



BTL: Bit Timing Logic  
 BRP: Baud Rate Prescaler  
 SJW: Synchronisation  
 Jump Width

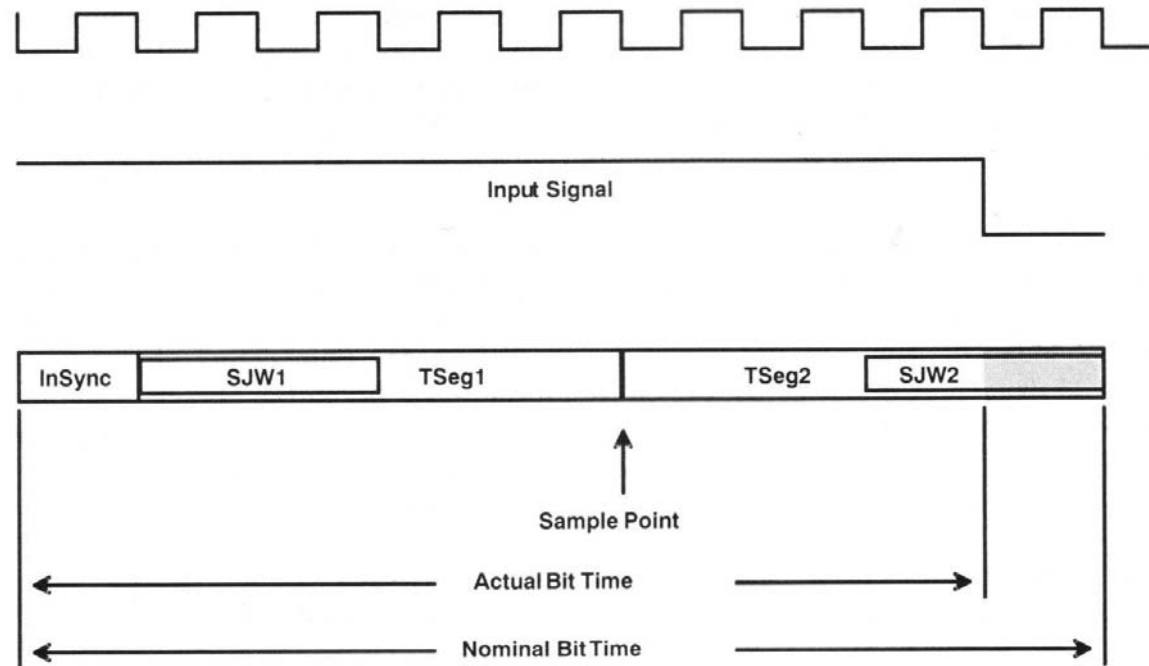
- Bitzeit ist in drei Segmente unterteilt: InSynchron, TSEG1 und TSEG2
- Der Abtastzeitpunkt (Sample) ist immer genau zwischen TSEG1 und TSEG2
- Die programmierbaren Teile jedes Zeitsegments – das Synchronisationssegment SJW – werden zur Synchronisation benutzt.
- Die Flanke wird innerhalb des InSynchron-Segments erwartet. In diesem Fall startet Empfänger TSEG1 und tastet an dessen Ende ab. Danach wird TSEG2 gestartet. Nach dessen Ablauf wird das nächste Bit erwartet.

# Synchronisation langsamerer Sender



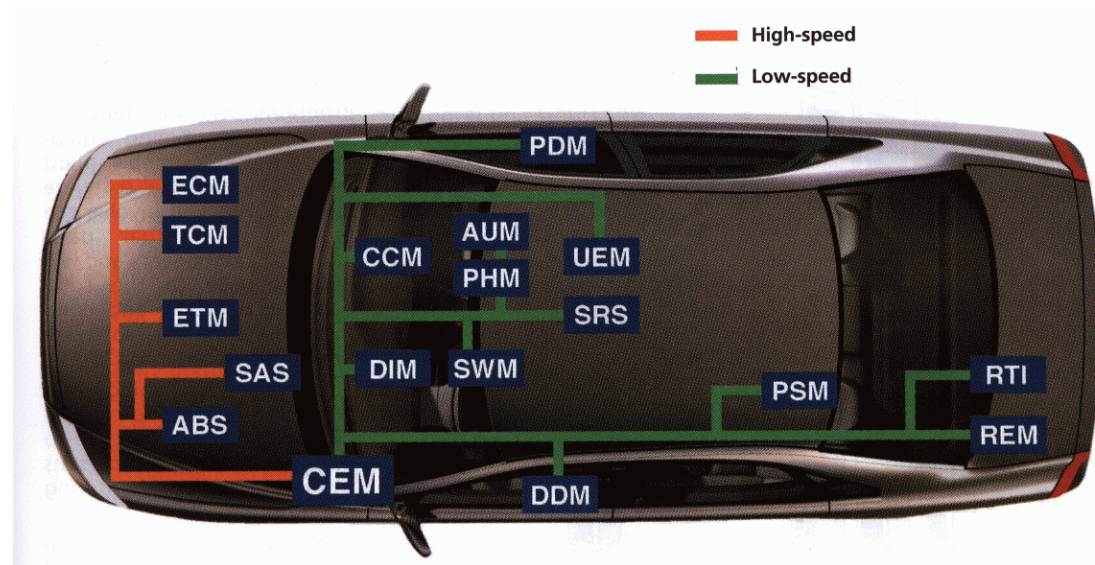
- Bitzeit des Senders länger als die des Empfängers (Sender hat niedrigere Übertragungsrate)
- Flanke wird erst im SJW1-Segment detektiert
- SJW1 wird erneut gestartet.
  
- Resynchronisation nur bei Flankenwechsel, sonst läuft normales Bit-Timing ab

# Synchronisation schnellerer Sender



- Bitzeit des Senders kürzer als die des Empfängers (Sender hat höhere Übertragungsrate)
- Flanke am Ende des Bits wird schon im SJW2-Segment detektiert
- SJW2 wird sofort gestoppt und
- Empfänger startet Bitzeit für das nächste Bit gestartet.
  
- Erneute Synchronisation erfolgt erst wieder mit einer Flanke

# Verteilte Systeme



## Warum verteilte Systeme?

- Komponenten sind physikalisch verteilt
- Verteilte Systeme sind leistungsfähiger/preiswerter als eine einzige zentrale CPU
- Sicherheit: Ausfall betrifft einzelne Komponente nicht das ganze System.
- Redundanz einfacher möglich
- System skaliert: einzelne Komponenten können hinzugefügt werden.

# Response Time Analysis bei CAN

Modell:

- Die Prozesse  $P_j \in \mathcal{P}$  versenden Nachrichten
- Prozess  $P_j$  lässt sich beschreiben zu  $P_j = \langle T_j, C_j, J_j \rangle$ , dabei ist
  - $T_j$  die Periode (minimale Zeit zwischen zwei Sendungen)
  - $C_j$  minimale Übertragungsdauer für einen Rahmen (Frame)
  - $J_j$  der Jitter (Schwankung in der Periodizität  $T_j$ )
  - $R_j$  ist die Response Zeit des Prozesses  $P_j$

Für die Berechnung der Response-Zeit auf dem CAN-Bus kann gut das Modell des **Fixed Priority Scheduling** auf einem Prozessor herangezogen werden.

Sei Priorität  $\pi_i < \pi_j$ , ist die Anzahl  $n_i$  der Interferenzen, die  $P_i$  durch den Prozess  $P_j$  mit höherer Priorität erleiden kann

$$n_i = \left\lceil \frac{R_i}{T_j} \right\rceil$$

(die Klammer bedeutet die **Ceiling**-Funktion: kleinste Ganzzahl größer oder gleich der gebrochenen Zahl in der Klammer)

## Response Time Analysis bei CAN (2)

Die maximale Interferenz  $I_i$  beträgt also:

$$I_i = \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

In einem realen System kann aber nicht davon ausgegangen werden, dass alle Sendeprozesse zur gleichen Zeit und mit konstanter Periode gestartet werden → **Jitter**.

Betrachten wir den Fall, dass  $P_j$  um  $J_j$  früher startet: Dann hat  $P_i$

- 2 Interferenzen, wenn  $R_i > T_j - J_j$ ,
- 3 Interferenzen, wenn  $R_i > 2T_j - J_j$ , ... ,
- $n$  Interferenzen, wenn  $R_i > (n-1) T_j - J_j$

Daraus folgt:  $\frac{R_i + J_j}{T_j} > (n-1)$  oder  $(n-1) = \left\lceil \frac{R_i + J_j}{T_j} \right\rceil - 1$

Die maximale Interferenz  $I_i$  mit **Jitter**  $J_j$  beträgt also:

$$I_i = \left\lceil \frac{R_i + J_j}{T_j} \right\rceil C_j$$

# RTA bei CAN

Scheduling bei CAN ist Fixed Priority und nicht-preemptiv:

- Die Prozesse  $P_j \in \mathcal{P}$  versenden Nachrichten
- Prozess  $P_j$  lässt sich beschreiben zu  $P_j = \langle T_j, C_j, J_j \rangle$ , dabei ist
  - $T_j$  die Periode (minimale Zeit zwischen zwei Sendungen)
  - $C_j$  minimale Übertragungsdauer für einen Rahmen (Frame)
  - $J_j$  der Jitter in Queuing oder Release
  - $B_j$  die Blockierung durch alle Prozesse kleiner Priorität
  - $w_j$  die Interferenz durch alle Prozesse höherer Priorität
  - $R_i = J_i + w_i + C_i$  : Response Zeit des Prozesses  $P_i$
  - $\tau_{bit}, \tau_{res}$  die Zeit(unschärfe) für 1 Bit. Bei 1MBit/s  $\rightarrow 1 \mu s$

$$R_m = J_m + w_m + C_m \quad (\text{Gl. 3}) \text{ worst case Release Time}$$

$$w_m = B_m + \sum_{\forall j \in hp(m)} \left\lceil \frac{w_m + J_j + \tau_{bit}}{T_j} \right\rceil C_j \quad (\text{Gl. 4}) \text{ wobei } hp(m) \text{ alle Nachrichten mit höherer Priorität als } m \text{ sind}$$

## RTA bei CAN (2)

In Gl. 4 kann die Blockierung im worst case Fall wie folgt angegeben werden:

$$B_m = \max_{\forall k \in lp(m)} (C_k) \quad (\text{Gl. 4}) \quad \text{wobei } lp(m) \text{ die Nachrichten mit kleinerer Priorität als } m \text{ sind}$$

(Gl. 2) Gl. 4 kann rekursiv beschrieben werden:

$$w_i^{n+1} = B_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i^n + J_j + \tau_{res}}{T_j} \right\rceil C_j$$

Dabei wird  $w_i^0 = 0$  gesetzt. Iteration ist beendet, wenn  $w_i^{n+1} = w_i^n$

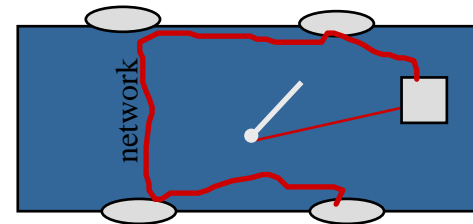
$C_m$  ist die maximale Zeit, die zur Übertragung der Nachricht  $m$  benötigt wird. CAN-Rahmen haben einen Overhead von 47 Bit, eine Stuffing-Breite von 5 Bit, jedoch werden nur maximal 34 von 47 Bits des Overhead gestopft.

$s_m$ : Anzahl Daten-Bytes (0-8)

$$C_m = \left( \left\lfloor \frac{34 + 8s_m}{5} \right\rfloor + 47 + 8s_m \right) \tau_{bit} \quad (\text{Gl. 6})$$

# Problem bei eventgesteuertem Bus

- nur Nachricht mit höchster Priorität ist determiniert.
- niedrige Prioritäten können verdrängt werden
- höhere Prioritäten können durch niedrigere zeitweilig blockiert werden.
- Reihenfolge von Nachrichten nicht determiniert
- Einfluss von Jitter
  - selbst kleine Schwankungen der Periodizität der einzelnen Nachrichten haben ungünstigen Einfluss auf Response-Zeit
  - nicht periodisch auftretende Nachrichten haben negativen Einfluss



→ Übergang zu zeitgesteuerten Protokollen bei kritischen Anwendungen (z.B. drive-by-wire, X-by-wire)

- [1] Real-Time Systems and Programming Languages; Alan Burns, Andy Wellings; Pearson-Addison Wesley; 3rd Ed 2001
- [2] Hard Real-Time Computing Systems; Giorgio C. Buttazzi; Springer; 2005
- [3] Analysing Real-Time Communications: Controller Area Network (CAN); K. W. Tindell, H. Hansson A. J. Wellings; 1994
- [4] Informations- und Kommunikationstechnik im Kraftfahrzeug, Michael Bolle
- [5] Echtzeitkommunikation, Bernd Schürmann
- [6] Friedhelm Stappert, RTS-Vorlesung SS09
- [7] Echtzeitanwendungen und Kommunikation mit Mikrocontrollern, P. Mutschler