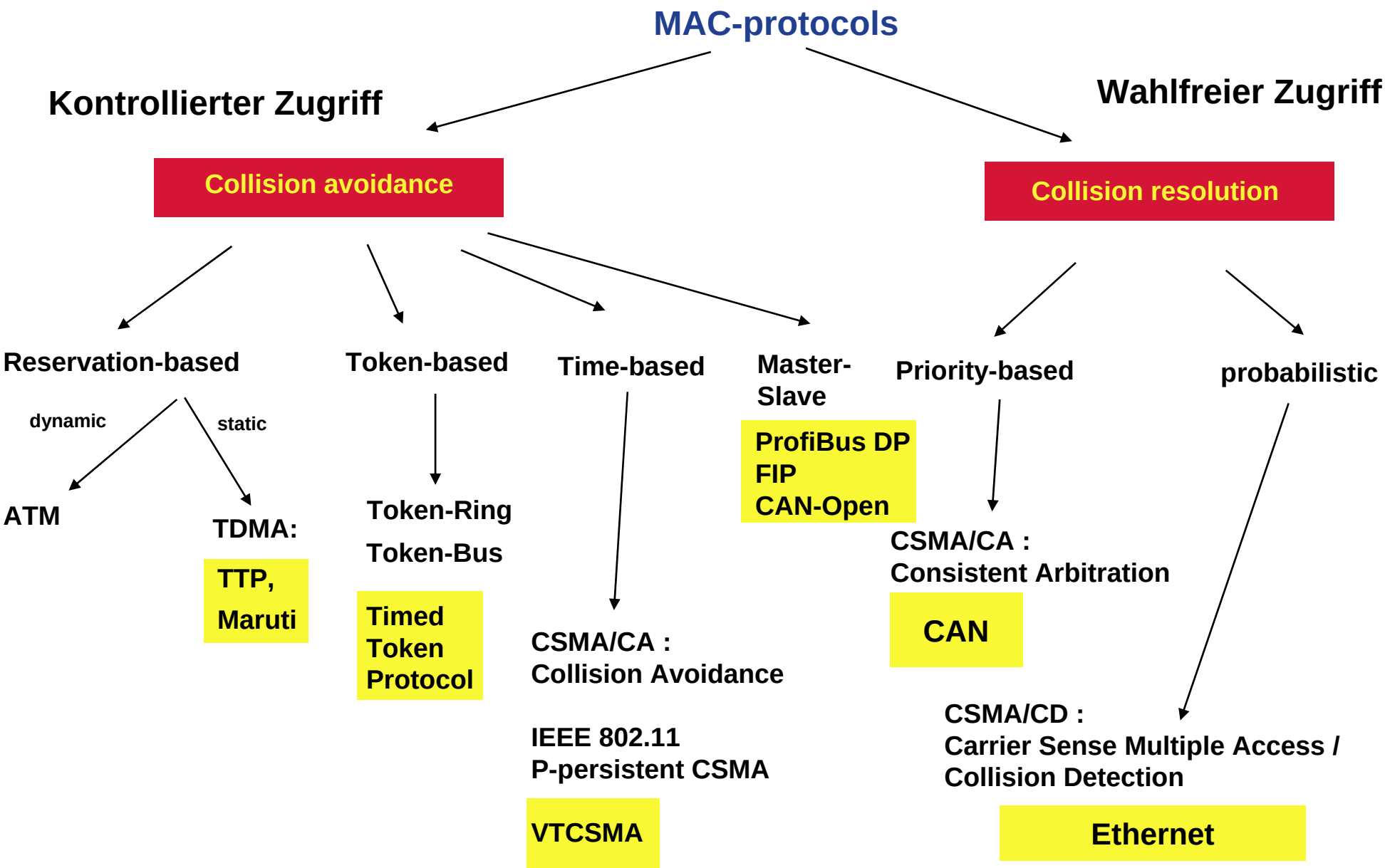
RT-Kommunikation am Beispiel

CAN-Bus

Grundlegende Konzepte des CAN-Bus:

- **Priorisierung der Nachrichten.**
- **Begrenzte, garantierte Nachrichtenverzögerung.**
- **Fehlererkennung in den Knoten und Signalisierung des Fehlerzustands.**
- **Automatische Wiederversendung der von Fehlern betroffenen Nachrichten sobald der Bus frei ist.**
- **Unterscheidung zwischen temporären und permanenten Fehlern und autonomes Abschalten defekter Rechnerknoten.**
- **Multimaster - Konfiguration.**
- **Garantie systemweiter Datenkonsistenz (entweder wird eine Nachricht von allen Knoten akzeptiert oder von keinem Knoten).**
- **Multicast-Empfang mit Zeitsynchronisation**

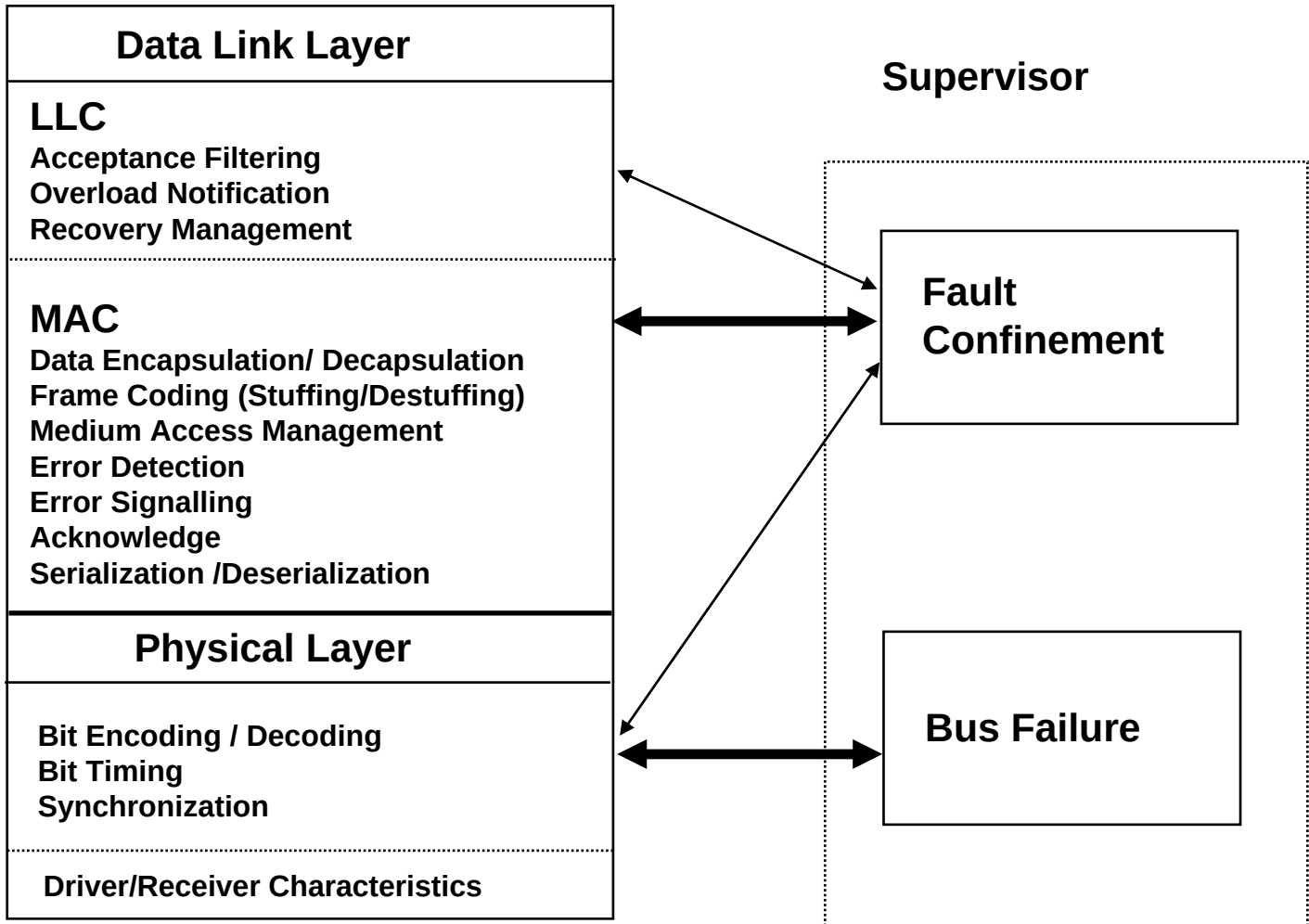

MAC-protocols
Kontrollierter Zugriff
Wahlfreier Zugriff
Collision avoidance
Collision resolution
Reservation-based
Token-based
Time-based
Master-Slave
Priority-based
probabilistic
dynamic
static
ATM
TDMA:
TTP,
Maruti
Token-Ring
Token-Bus
Timed
Token
Protocol
CSMA/CA :
Collision Avoidance
IEEE 802.11
P-persistent CSMA
VTCSMA
ProfiBus DP
FIP
CAN-Open
CSMA/CA :
Consistent Arbitration
CAN
CSMA/CD :
Carrier Sense Multiple Access /
Collision Detection
Ethernet

Unterschiedliche CAN-Standards

CAN Specification 1.2
CAN Specification 2.0

Die Spezifikationen unterscheiden sich hauptsächlich:

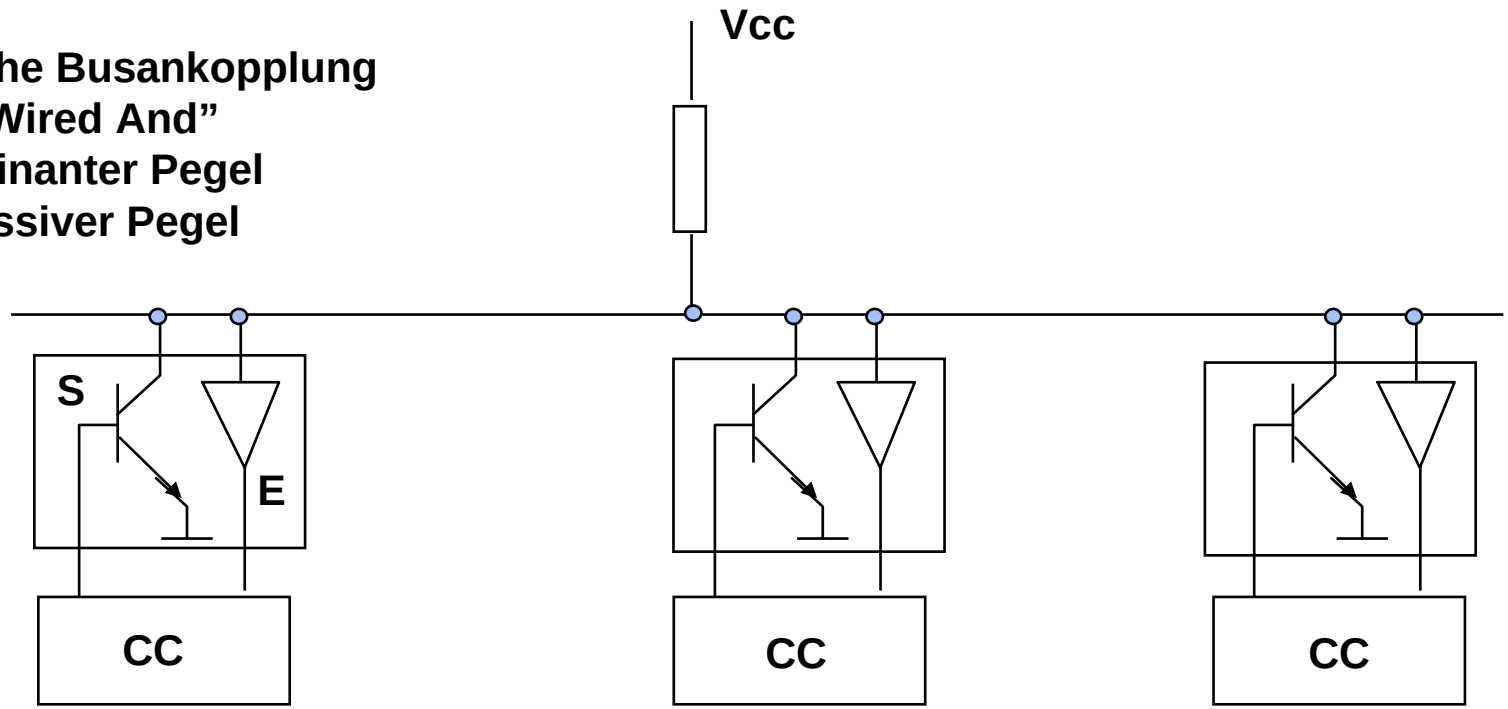
- **in der möglichen Länge der Nachrichten-Identifier**
- **in der Möglichkeit gruppenorientierter Kommunikation**
  Nachrichtenfilter = Masken für die längeren Identifier



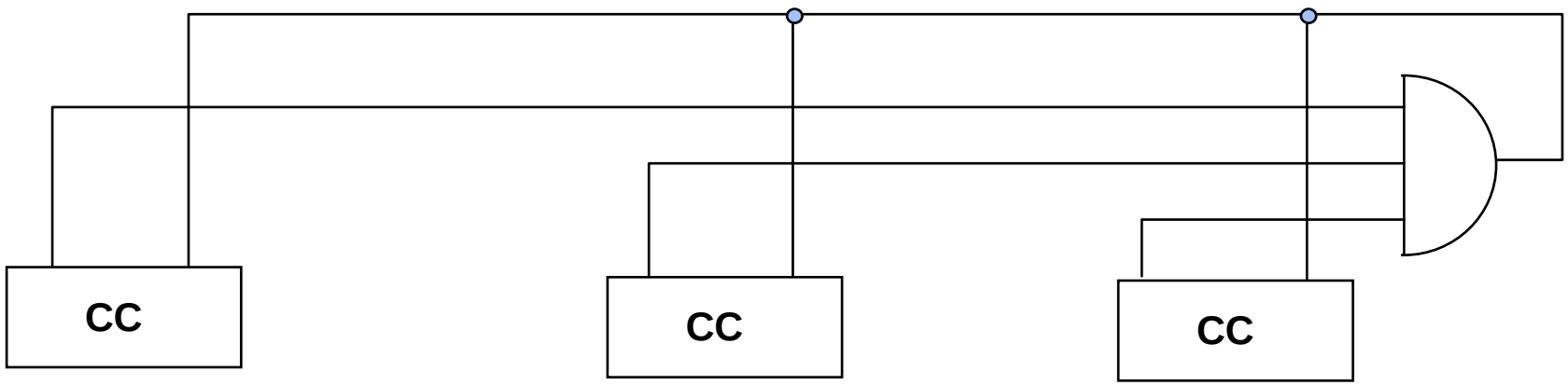
LLC = Logical Link Control
MAC = Medium Access Control

Die CAN Übertragungsschicht (physical layer)

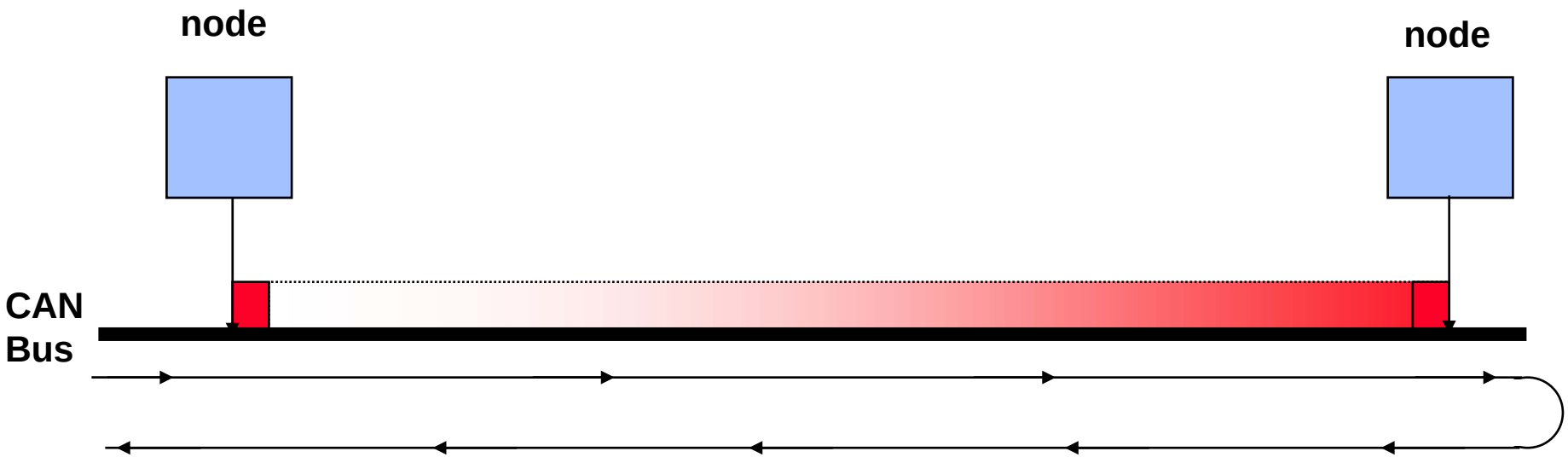
Physische Busankopplung
durch “Wired And”
0 = dominanter Pegel
1 = rezessiver Pegel



Äquivalentes logische Schaltbild



CAN Bit Synchronisation

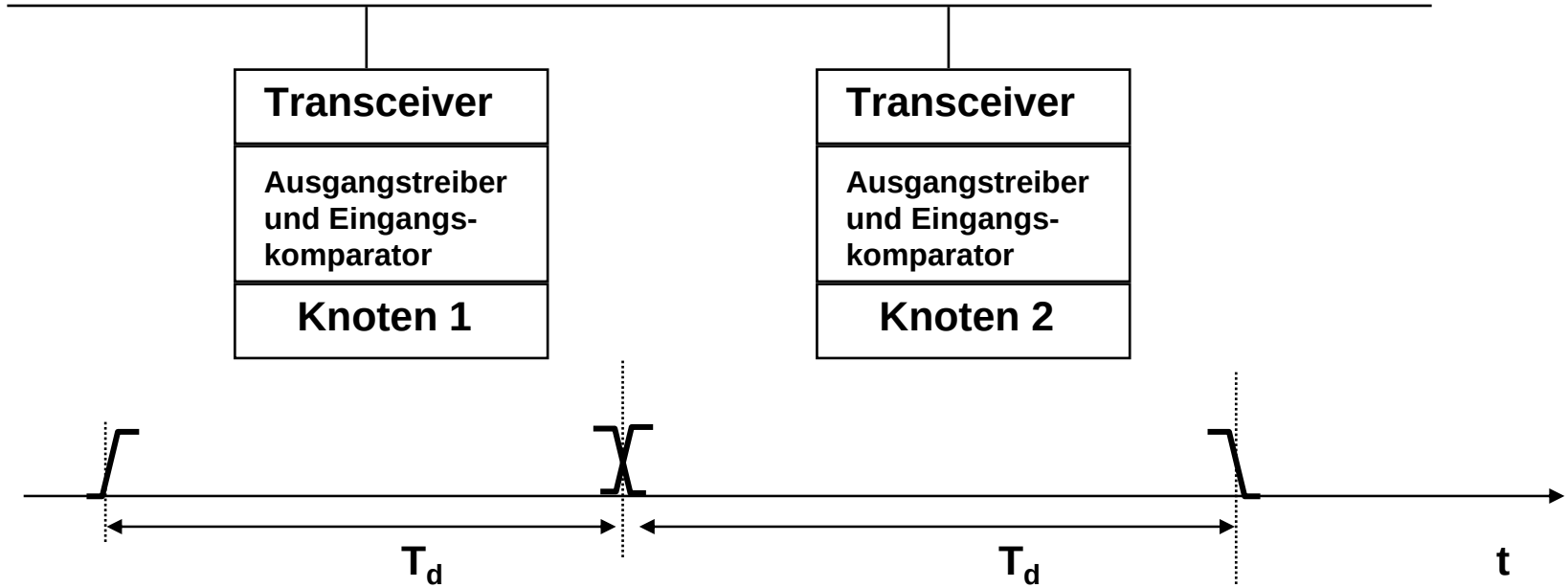


Bit rate dependend on the length of the bus

Dominant and Recessive Bit Values

Bit Monitoring

Ermittlung der Länge des Signalausbreitungs-Segments



Knoten 1 schaltet rezessives Bit auf den Bus

Signal von Knoten 1 erreicht Knoten 2. Knoten 2 schaltet dominantes Bit auf den Bus

Signal von Knoten2 erreicht Knoten 1

T_d : Max. Signallaufzeit zwischen Netzknoten + Schaltzeiten der elektr. Komponenten

Maximale Übertragungsraten auf dem CAN Bus in Abhängigkeit von der Buslänge

$$T_d = T_{TT\text{-}verzögerung} + T_{Leitungsverzögerung}$$

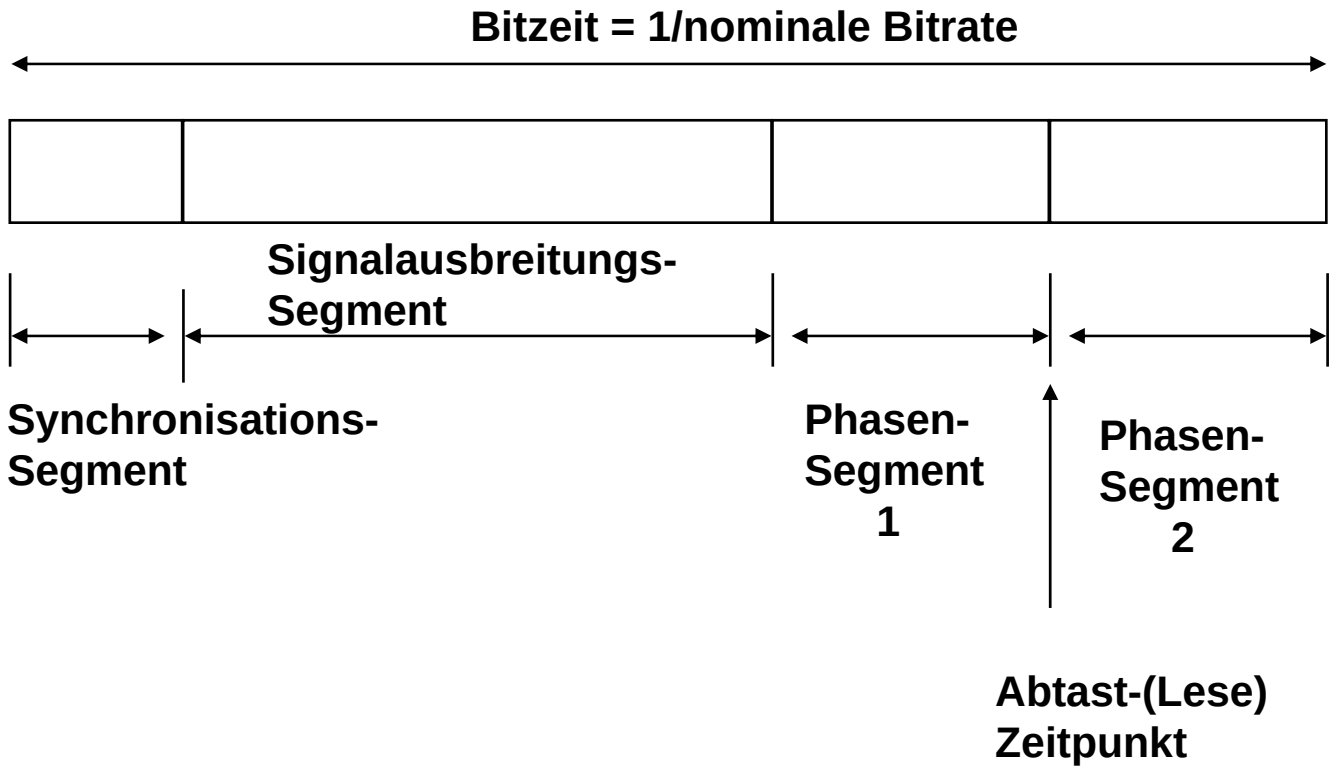
$T_{TT\text{-}verzögerung}$ ~ 100 ns

(Treiber, Eingangskomparator und Transceiver)

$T_{Leitungsverzögerung}$ ~ 0,2 m / ns bei verdrillten Zweidrahtleitungen

Bitrate (kBits/s)	maximale Netz ausdehnung (m)
1000	40
500	112
300	200
200	310
100	640
50	1300

Bit-Timing und Bitsynchronisation



Länge der Zeitsegmente werden in Vielfachen einer aus der Oszillatorperiode abgeleiteten Zeiteinheit (time quantum) spezifiziert:

Synch.-Segment	**1**	**Zeiteinheit**
Sig. Ausbr. Seg.	**1...8**	**Zeiteinheiten**
Phasensegment 1	**1...8**	**Zeiteinheiten**
Phasensegment 2	**1...8**	**Zeiteinheiten**

Die CAN Sicherungsschicht

Nachrichtenformate:

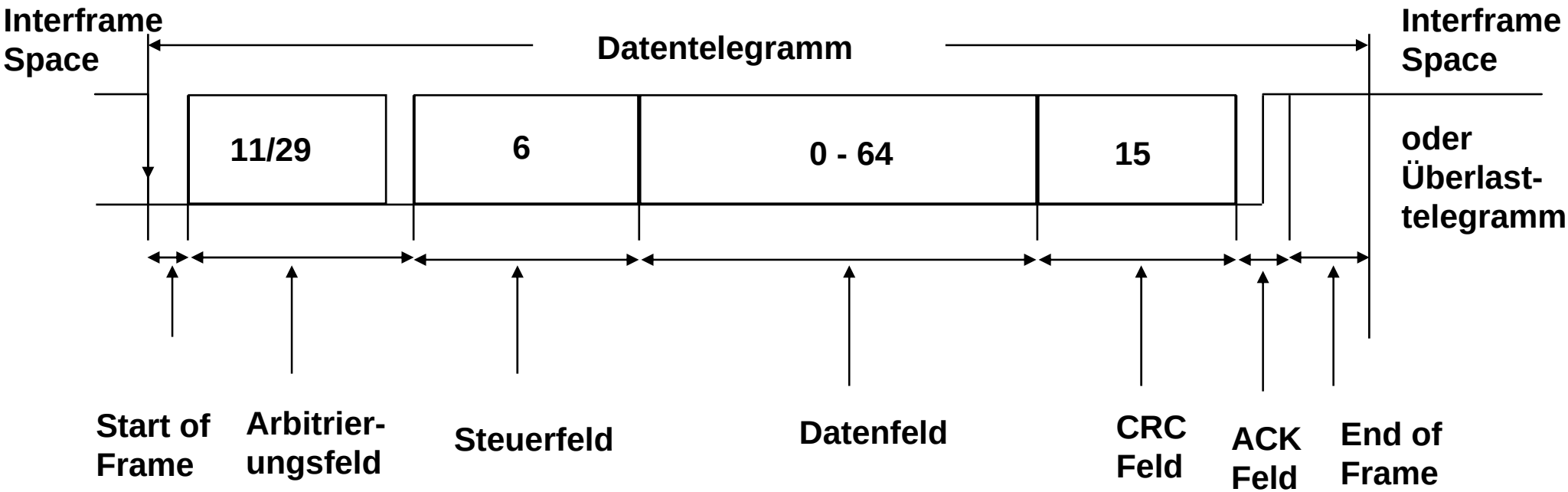
- **Datentelegramm (Data Frame)**
  Normale Datenübertragung auf Initiative des Senders.

- **Datenanforderungstelegramm (Remote Frame)**
  Auf Anforderung eines Busteilnehmers wird ein bestimmtes Datentelegramm mit demselben Identifier von einer anderen Datenquelle gesendet.

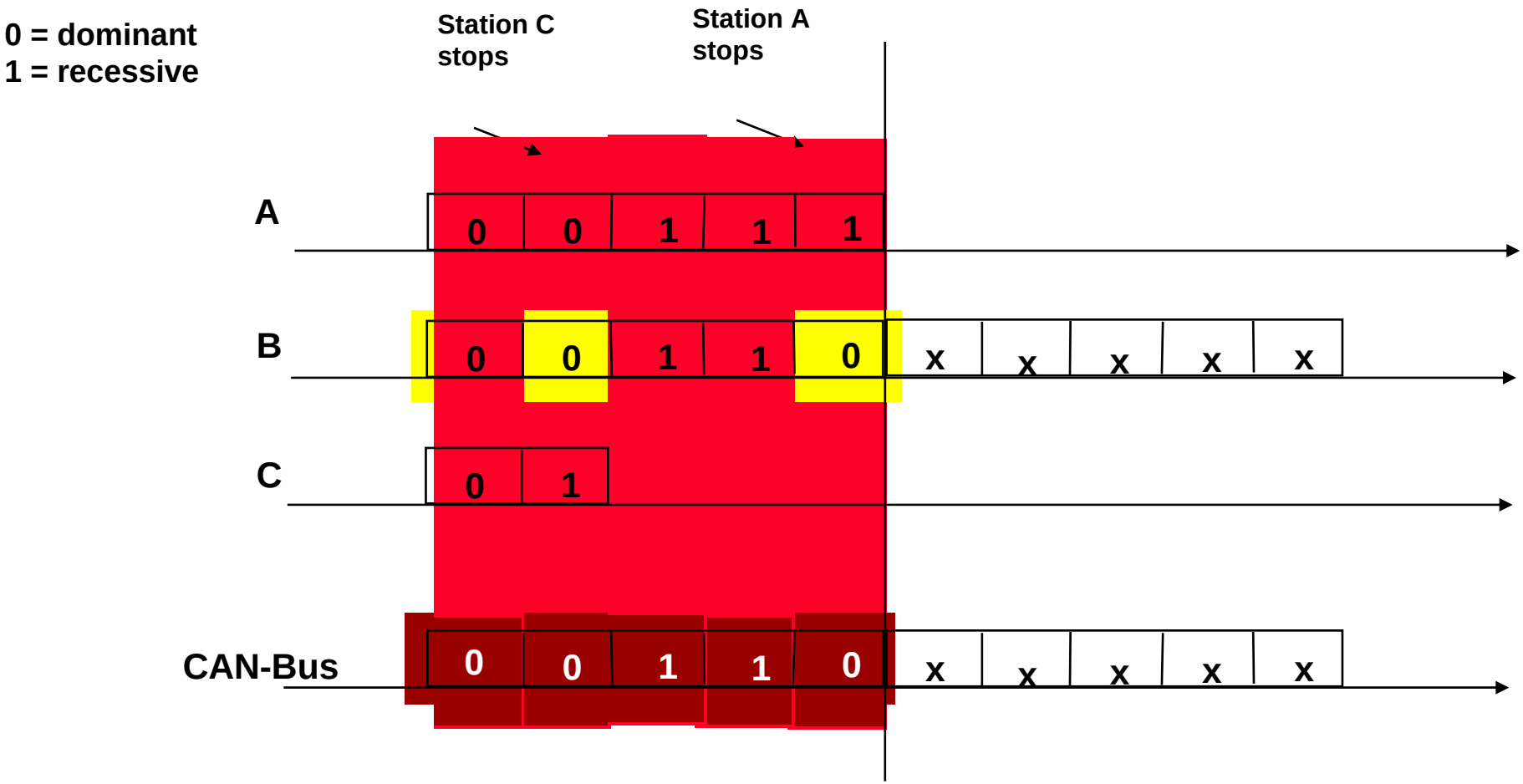
- **Fehlertelegramm (Error Frame)**
  Ein Busteilnehmer (Sender oder Empfänger) signalisiert einen von ihm erkannten Fehler.

- **Überlasttelegramm (Overload Frame)**
  Dient der Flußkontrolle. Es kann eine Verzögerung zwischen einem vorausgegangenen und einem nachfolgenden Daten- oder Datenanforderungstelegramm realisiert werden.

CAN Datentelegramm (Data Frame)

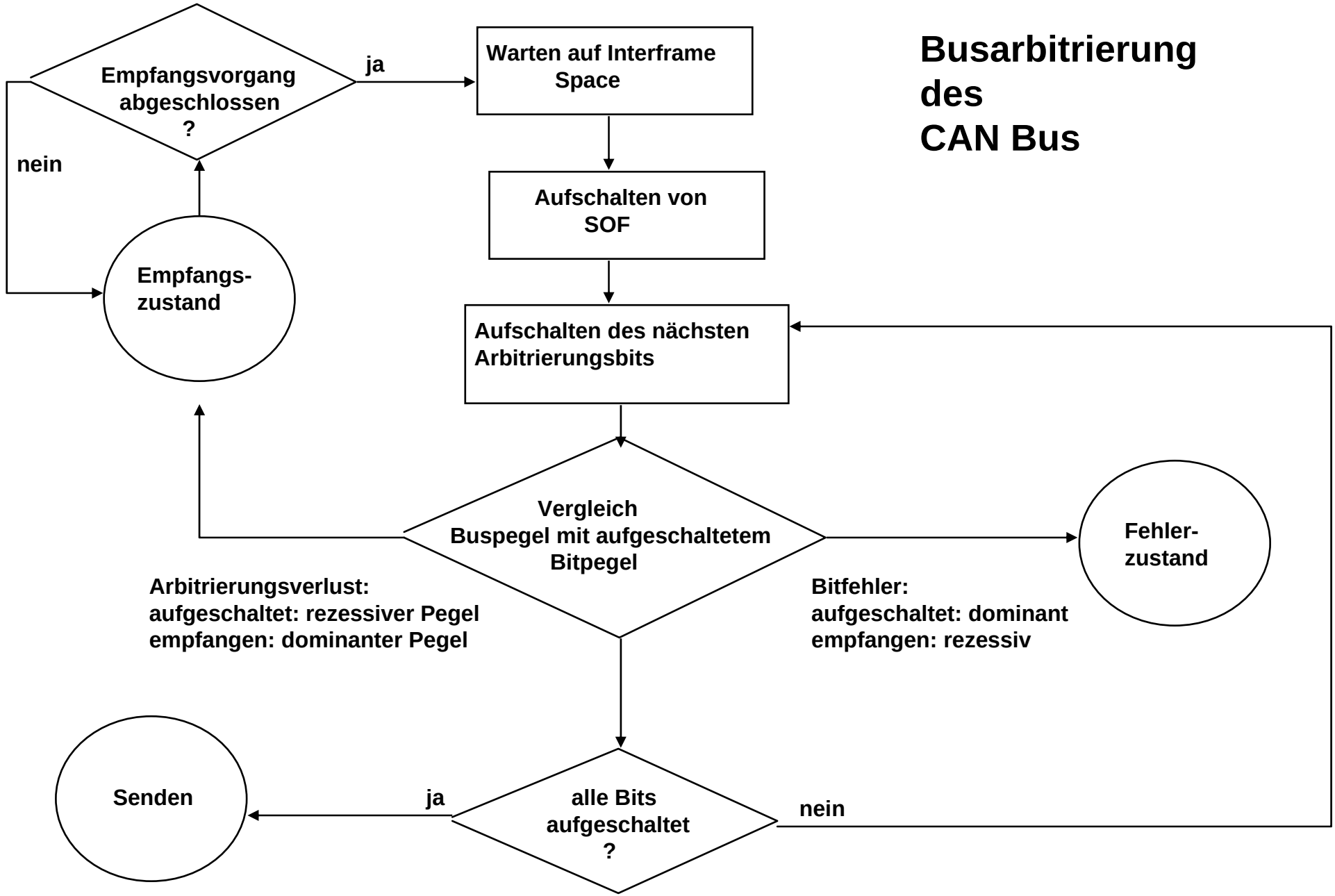


Arbitration on a CAN-Bus



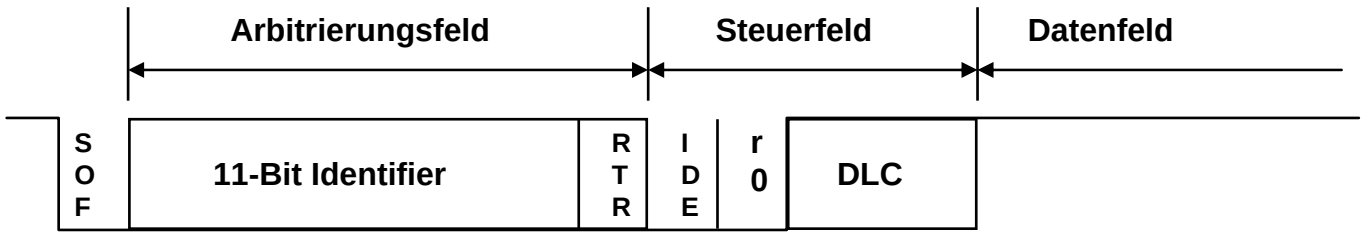
CAN enforces a global priority-based message scheduling

Busarbitrierung des CAN Bus

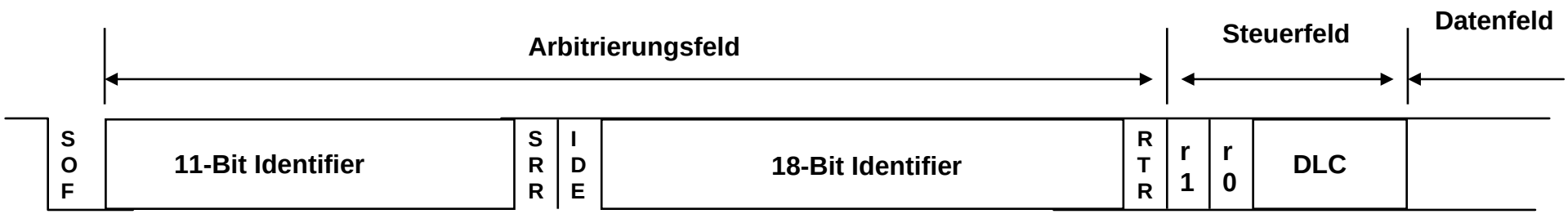


CAN Datentelegrammformate (Data Frame)

Standard Format SF (kompatibel zu CAN Spezifikation 1.2)

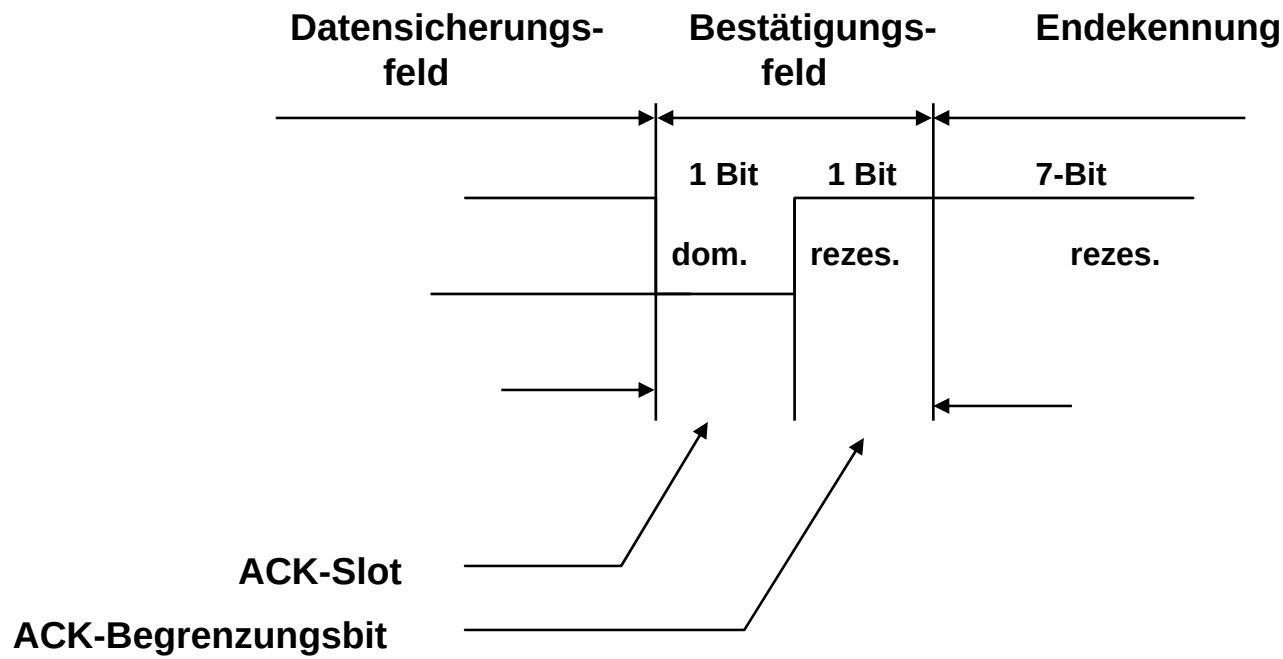


Erweitertes Format EF (CAN Spezifikation 2.0)



RTR: Remote Transmissin Request. In Data Frame: RTR = dominant. In Remote Frame: RTR = rezessiv.
Das SSR-Bit übernimmt in EF die Funktion des RTR-Bit
IDE: Identifier Extension. Im SF Bestandteil des Steuerfeldes, dominant, keine Bedeutung.
Im EF Bestandteil des Arbitrierungsfeldes, rezessiv, Telegramm wird als EF erkannt.
SRR: Subsitute Remote Request. Rezessiv, es steht in EF an der Stelle von RTR und sorgt für Kompatibilität.
DLC: Data Length Control. 0-8 Byte sind erlaubte Werte.
r0 und r1: reserved

Bestätigung einer Nachricht:

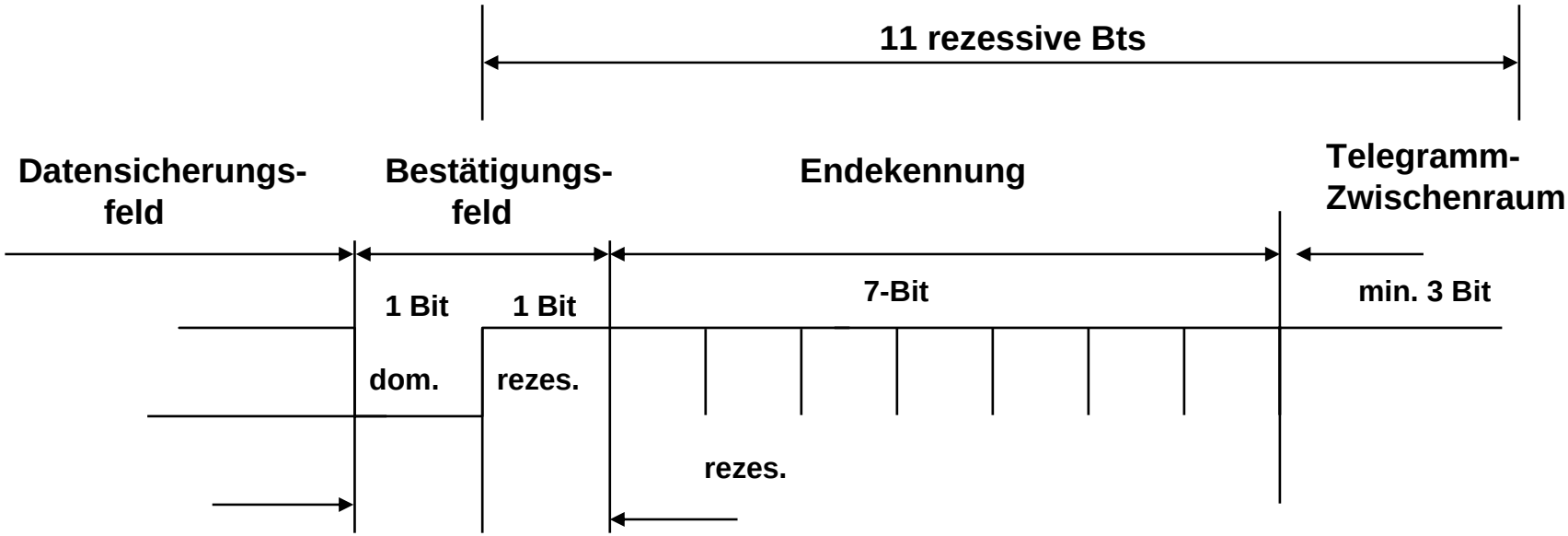


Stationsneutrale positive Empfangsbestätigung (Broadcast !)

Empfänger, die die Nachricht korrekt erhalten haben, legen während des Ack-Slots einen dominanten Pegel auf den Bus. Der Empfänger selbst schaltet auf einen rezessiven Pegel.

- **Nachricht wir durch einen einzigen fehlerfreien Empfänger als korrekt bestätigt.**
- **Systemweite Datenkonsistenz erfordert zusätzlich Signalisierung lokaler Fehler.**

Abschluß einer Nachricht

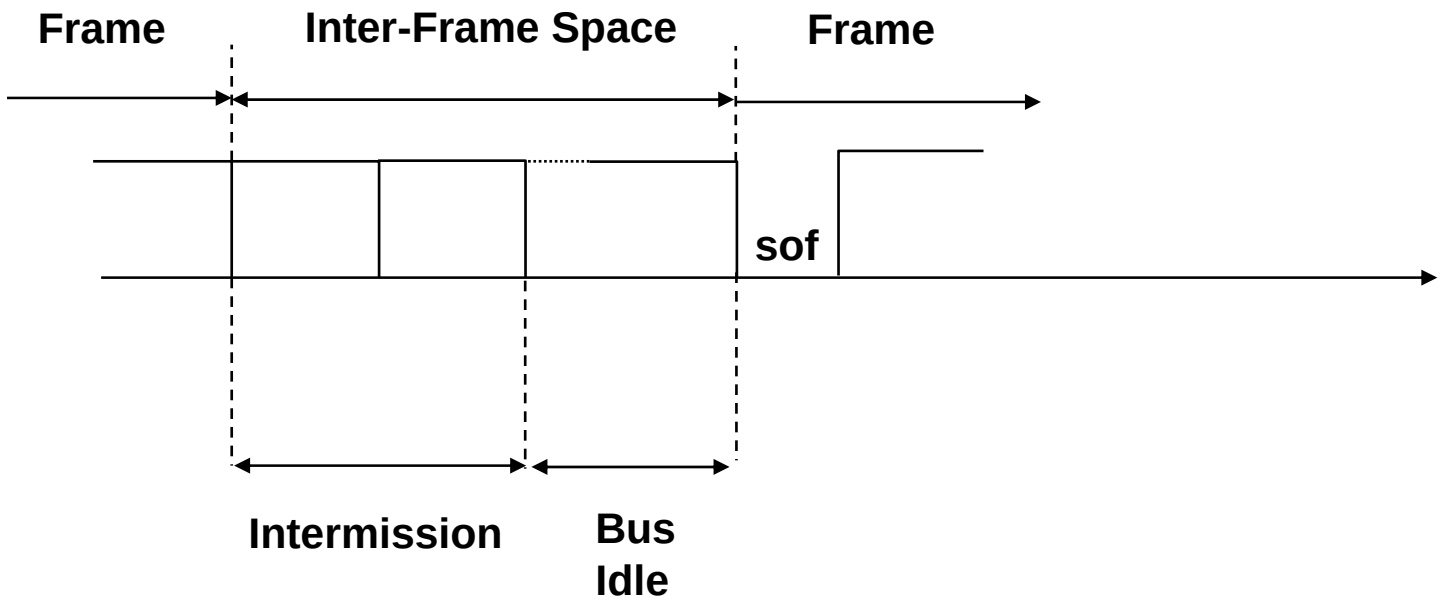


Ziel: Lokalisierung der Nachricht, in der ein Fehler auftritt.
Fehler soll innnerhalb der Nachricht, die gerade gesendet wird, erkannt und an alle Teilnehmer signalisiert werden.

Vorgehen: Endekennung besteht aus 7 rezessiven Bits.

1. **Ein Fehler kann durch Aufschalten von dominanten Bits signalisiert werden.**
2. **Ein völlig aus der Synchronisation gefallener Knoten, der die Endekennung als Daten interpretiert, wird durch die Verletzung der Bit-Stuffing Regel auf einen Fehler aufmerksam. Nach 5 rezessiven Bits erwartet er ein dominantens Bit. Durch die Länge der Endekennung von 7 Bit hat er dann noch die Möglichkeit, den Fehler zu signalisieren.**

Interframe Space

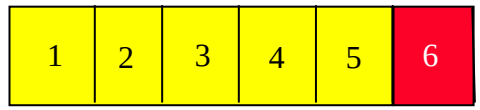


Intermission: kein Data- oder Remote Frame darf gestartet werden.

Intermission 1: Overload Frame darf aktiv gestartet werden
Intermission 2: Overload Frame reaktiv (nach Erkennung eines dominanten Pegel in I1)

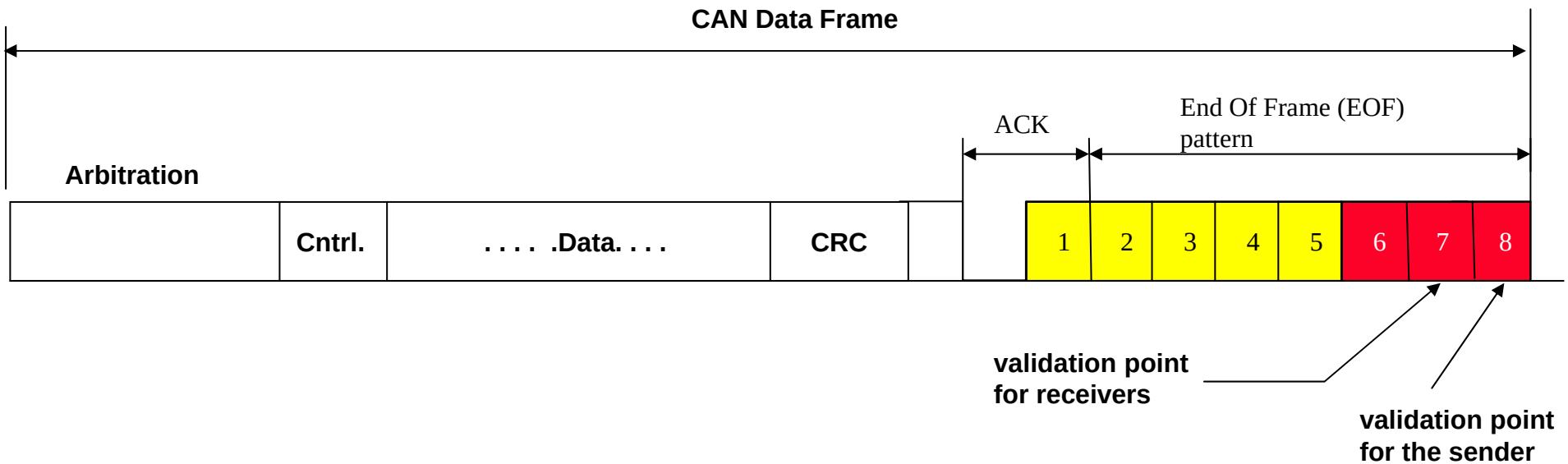
Error Detection and Error Signalling in CAN

Violation of the Bit-Stuffing Rule:
Used for Error Detection and Signalling



Bit-Stuffing enforces the following rule:

A sequence of 5 identical bit levels is followed by a complementary bit level



Fehlererkennungsmaßnahmen:

1.) Monitoring: Der Sender vergleicht das gesendete Bit lokal mit dem empfangenen Bit.
Fehlerart: lokale Fehler im Sender
Fehlererkennung: im Sender

2.) Cyclic Redundancy Check
Fehlerart: 5 beliebig verteilte Fehler im Codewort,
Burstfehler der Länge 15.
Fehlererkennung: im Empfänger

3.) Bitstuffing:
Fehlerart: transiente Störungen, Stuck-at-Fehler im Sender
Fehlererkennung: im Empfänger

4.) Formatüberwachung:
Fehlerart: Die im Format festgelegten Bitfolgen werden nicht eingehalten.
Fehlererkennung: im Empfänger

5.) Acknowledgment:
Fehlerart: kein Acknowledge
Fehlererkennung: im Sender, der Sender vermutet den Fehler bei sich.

“Restrisiko”

Die Restfehlerwahrscheinlichkeit gibt an, mit welcher Wahrscheinlichkeit auftretende Fehler nicht erkannt werden.

Ein Fehler bleibt unerkannt, wenn folgende Bedingungen erfüllt sind:

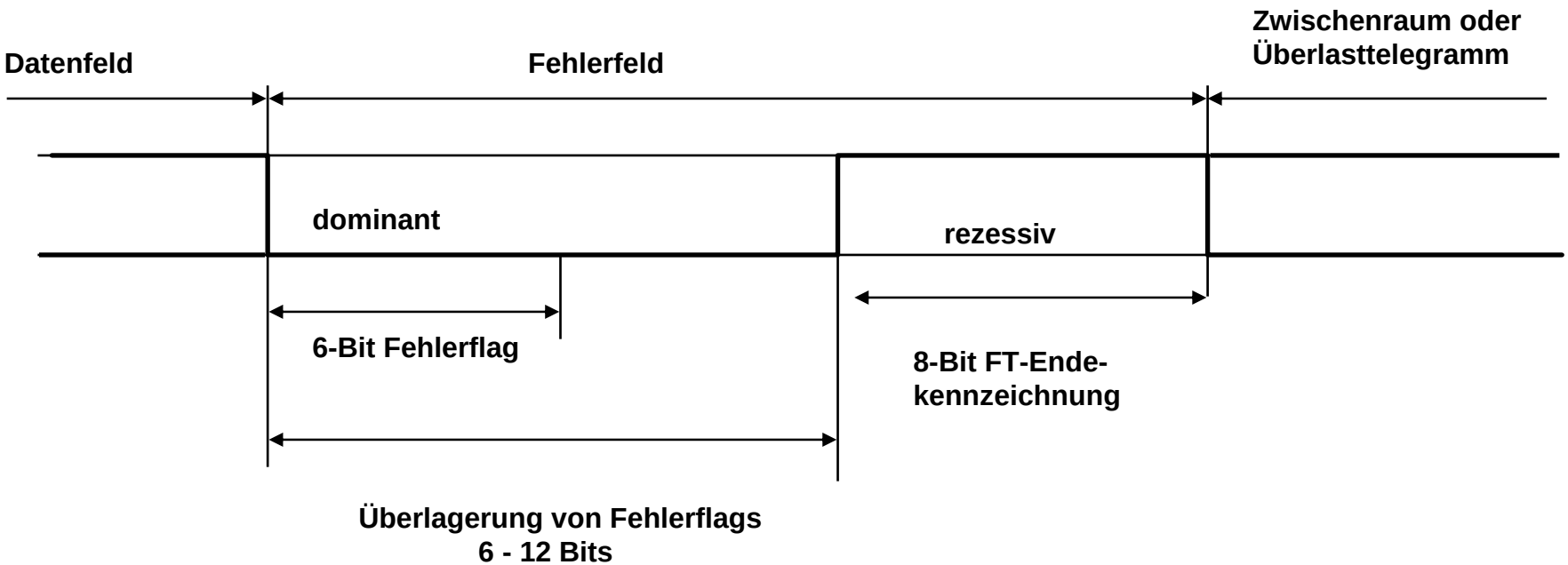
- **der Sender ist nicht gestört, d.h. er empfängt lokal seine gesendeten Daten korrekt (Monitoring)**
- **alle anderen Empfänger empfangen daselbe Bitmuster, das sich von dem vom Sender empfangenen unterscheidet und einen Fehler enthält, der nicht erkannt wird.**

Bitstuffing: Doppelfehler innerhalb von 6 Bits werden nicht erkannt.
CRC: Differenz zwischen gesendeter und empfangener Nachricht beträgt ein Vielfaches des Generatorpolynoms.
Frame-Fehler: Verlängern oder verkürzen von Nachrichten und gleichzeitiger Vortäuschung eines EOF (End-of-Frame) Feldes.

Anschätzung nach Unruh, Mathony und Kaiser:”Error Detection Analysis of Automotive Communication Protocols”, SAE International Congress, Nr. 900699, Detroit, USA, 1990

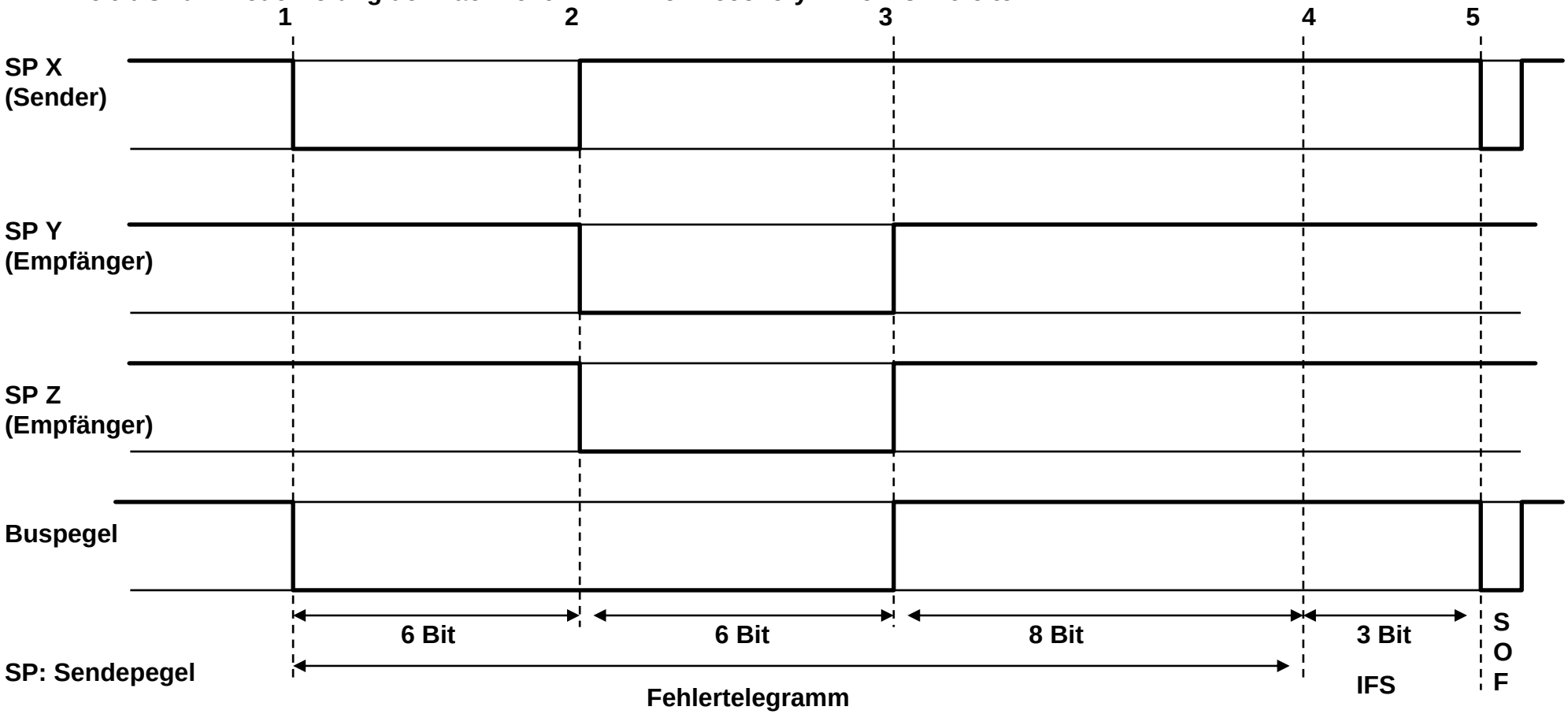
Szenario: Knoten: 10, Bitfehlerrate: $2 \bullet 10^{-2}$, Nachrichtenfehlerrate: 10^{-3}
Restfehlerwahrscheinlichkeit: $4{,}7 \bullet 10^{-14}$
Bei zunehmender Knotenzahl nimmt die Restfehlerwahrscheinlichkeit ab.

Fehlerbehandlung: 1. Fehlertelegramm FT



Fehlerbehandlung: Entstehung eines Fehlertelegramms durch Sendefehler

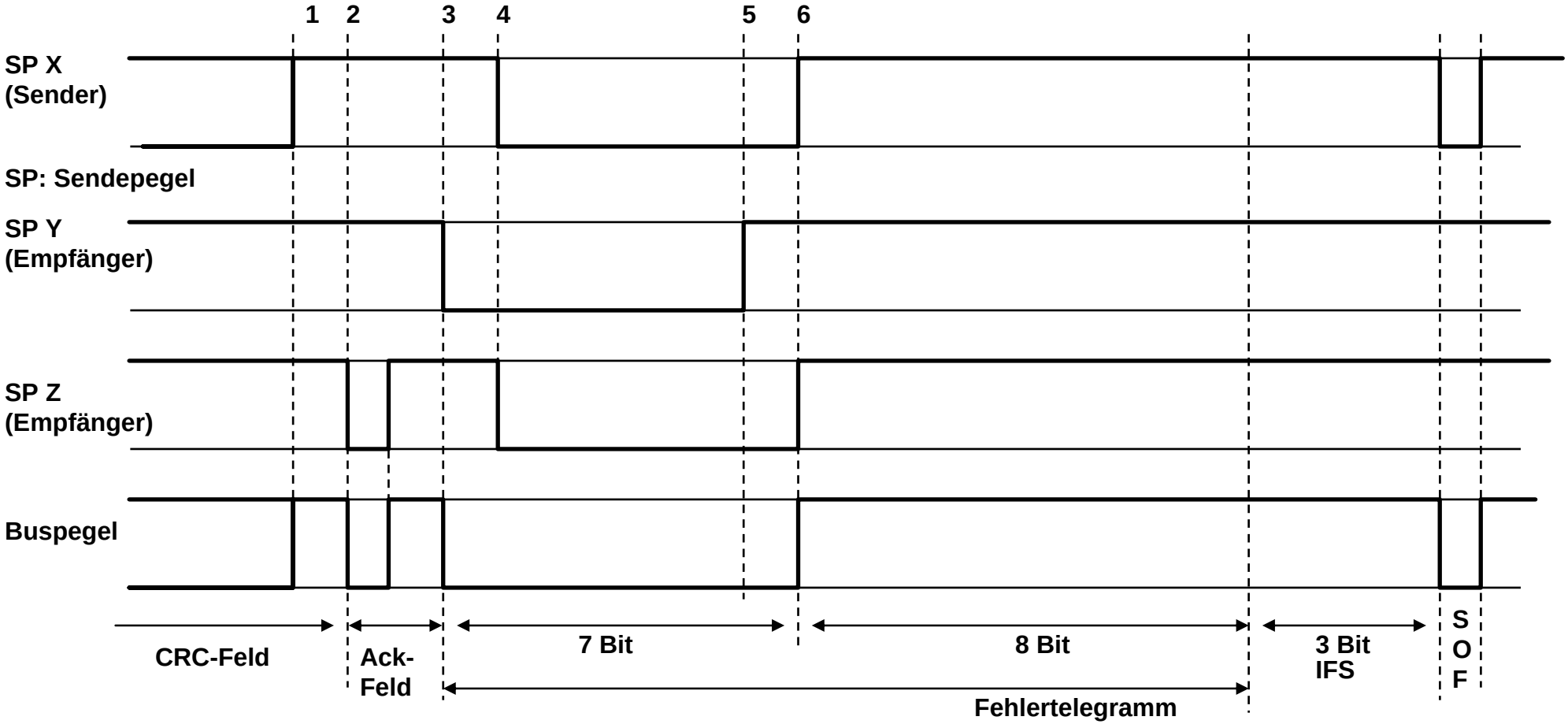
Zeit bis zur Wiederholung der Nachricht: min. Error Recovery Time: 23 Bitzeiten.



1. Konten X erkennt Sendefehler und schaltet sofort auf einen dominanten Pegel.
2. Knoten Y und Z erkennen duch die Verletzung der Bit-Stuffing Regel einen Fehler und schalten ihrereseits einen dominanten Pegel auf den Bus. Knoten X, der einen rezessiven Pegel sendet, erkennt, daß er den Fehler zuerst erkannt hat.
3. Es erfolgt die Kennung des Endes der Nachricht.
4. Der Nachrichten zwischenraum (IFS: Inter Frame Space) wird gesendet.
5. Knoten X versucht erneut zu senden. Konkurriert normal um die Belegung des Busses.

Fehlerbehandlung: Entstehung eines Fehlertelegramms durch Empfangsfehler

Zeit bis zur Wiederholung der Nachricht: min. Error Recovery Time: 20 Bitzeiten.

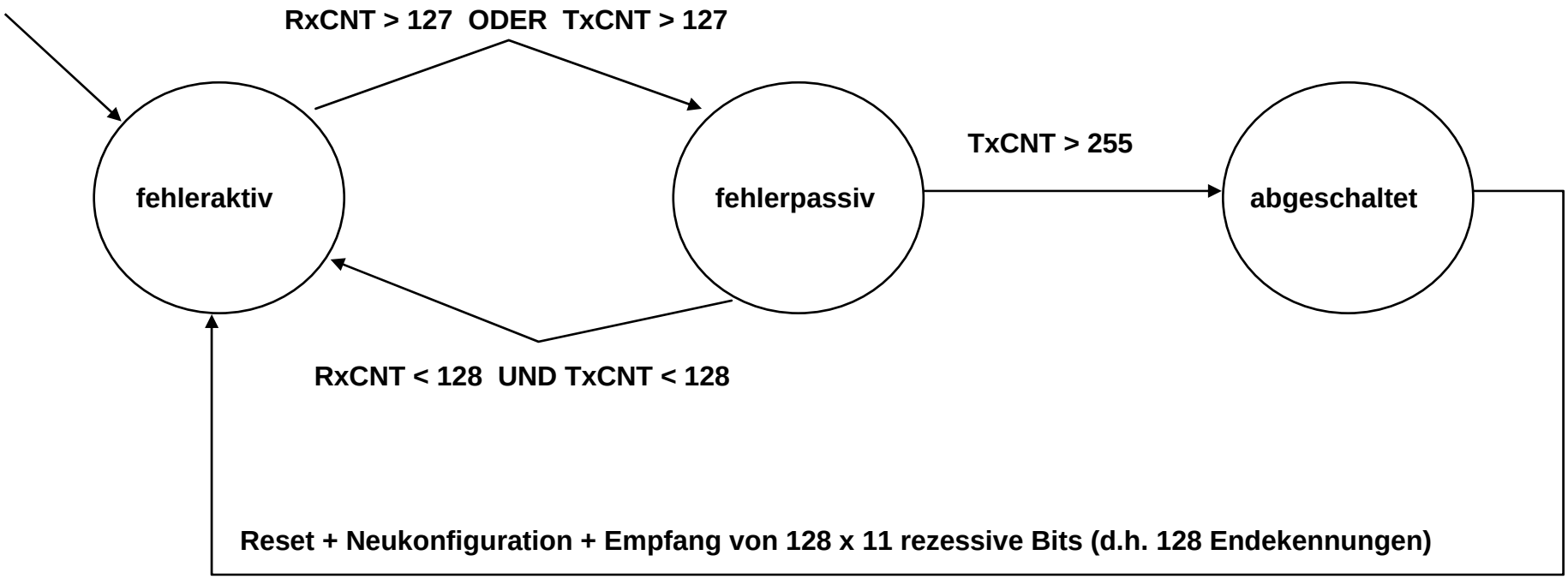


1. Knoten Y erkennt nach Abschluß der CRC-Prüfung einen Fehler.
2. Der Bestätigungsmechanismus, d.h. die Betätigung, daß mindestens 1 Knoten die Nachricht fehlerfrei empfangen hat, bleibt erhalten. D.h. es wird ein Ack-Zyklus durchgeführt.
3. Knoten Y signalisiert einen Fehler durch Aktivierung des Fehlertelegramms.
4. Knoten X und Z, die eine rezessive Endekennung erwarten, erkennen den Fehler und aktivieren ihrerseits ein Fehlertelegramm.
5. Knoten Y schaltet rezessiven Pegel auf den Bus und erkennt am noch anliegenden dominanten Pegel, daß er den Fehler erkannt hat.
6. Nach Endekennung und IFS kann die fehlerhafte Nachricht erneut gesendet werden, wenn X den Bus gewinnt.

Durchsetzung eines “Fail Silent” Verhaltens (Fehlereingrenzung)

Zustände eines CAN-Knotens im Hinblick auf Fehlereingrenzung:
- **fehleraktiv**
- **fehlerpassiv**
- **abgeschaltet**

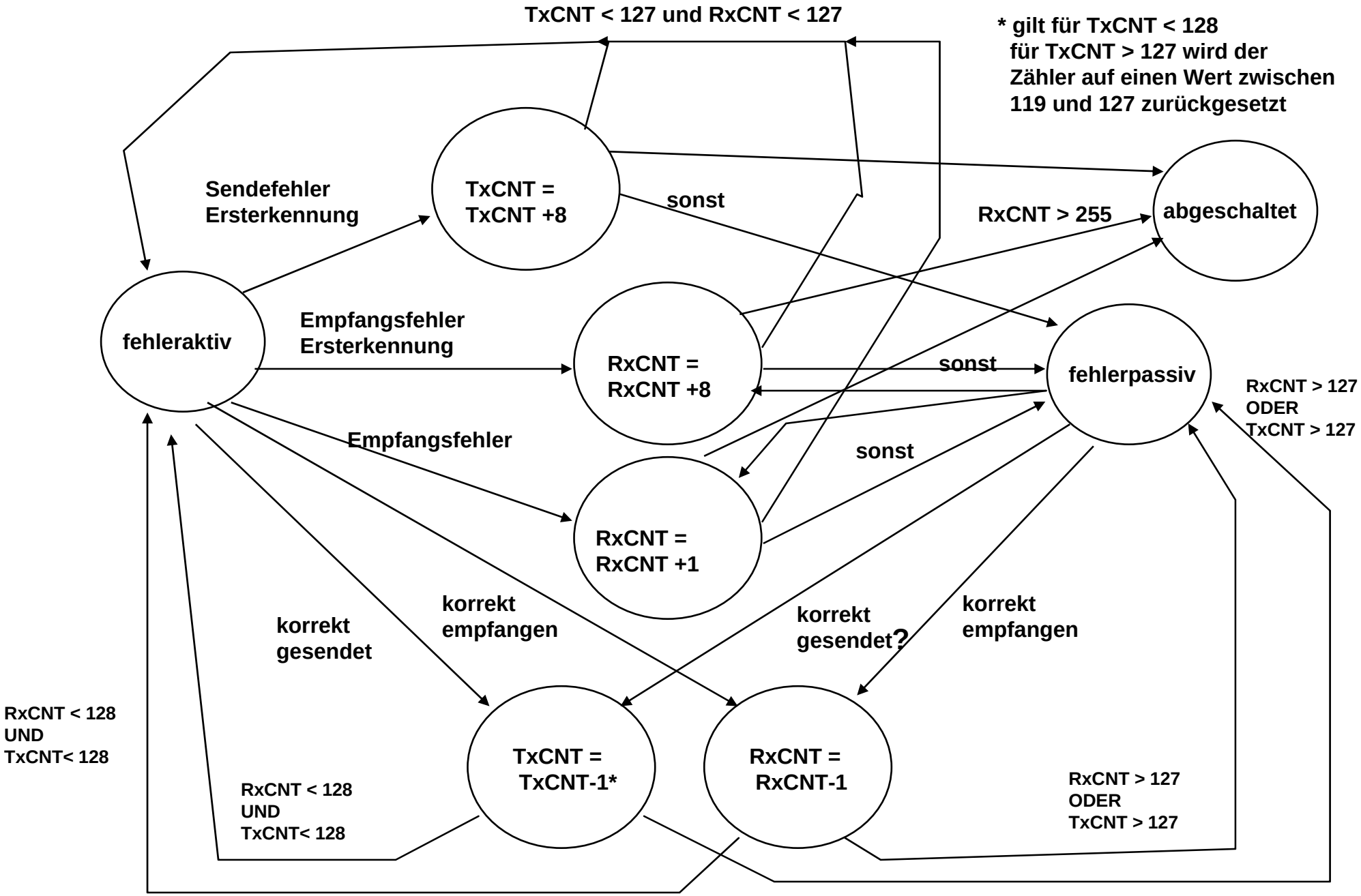
RxCNT: Wert des Empfangsfehlerzählers
TxCNT: Wert des Sendefehlerzählers



Fehlereingrenzung und Durchsetzung eines “Fail Silent” Verhaltens

Problem: Defekter Busteilnehmer kann den gesamten Nachrichtenverkehr blockieren.
Annahme: 1. Ein defekter Knoten erkennt Fehler zuerst.
2. Häufiges Ersterkennen eines Fehlers deutet auf einen Fehler im Knoten hin.

Lösungsansatz: Fehlerzähler für Sende- und Empfangsfehler. Knoten, die den Fehler zuerst erkannt haben, werden um den 8 bis neunfachen Wert inkrementiert.


TxCNT < 127 und RxCNT < 127
* gilt für TxCNT < 128
für TxCNT > 127 wird der
Zähler auf einen Wert zwischen
119 und 127 zurückgesetzt
Sendefehler
Ersterkennung
TxCNT =
TxCNT +8
sonst
RxCNT > 255
abgeschaltet
fehleraktiv
Empfangsfehler
Ersterkennung
RxCNT =
RxCNT +8
sonst
fehlerpassiv
RxCNT > 127
ODER
TxCNT > 127
Empfangsfehler
sonst
RxCNT =
RxCNT +1
korrekt
gesendet
korrekt
empfangen
korrekt
gesendet?
korrekt
empfangen
RxCNT < 128
UND
TxCNT< 128
TxCNT =
TxCNT-1*
RxCNT =
RxCNT-1
RxCNT < 128
UND
TxCNT< 128
RxCNT > 127
ODER
TxCNT > 127

Analyse der Totzeiten auf dem CAN-Bus
(Jose Rufino, Paulo Verissimo, Juli 1995)

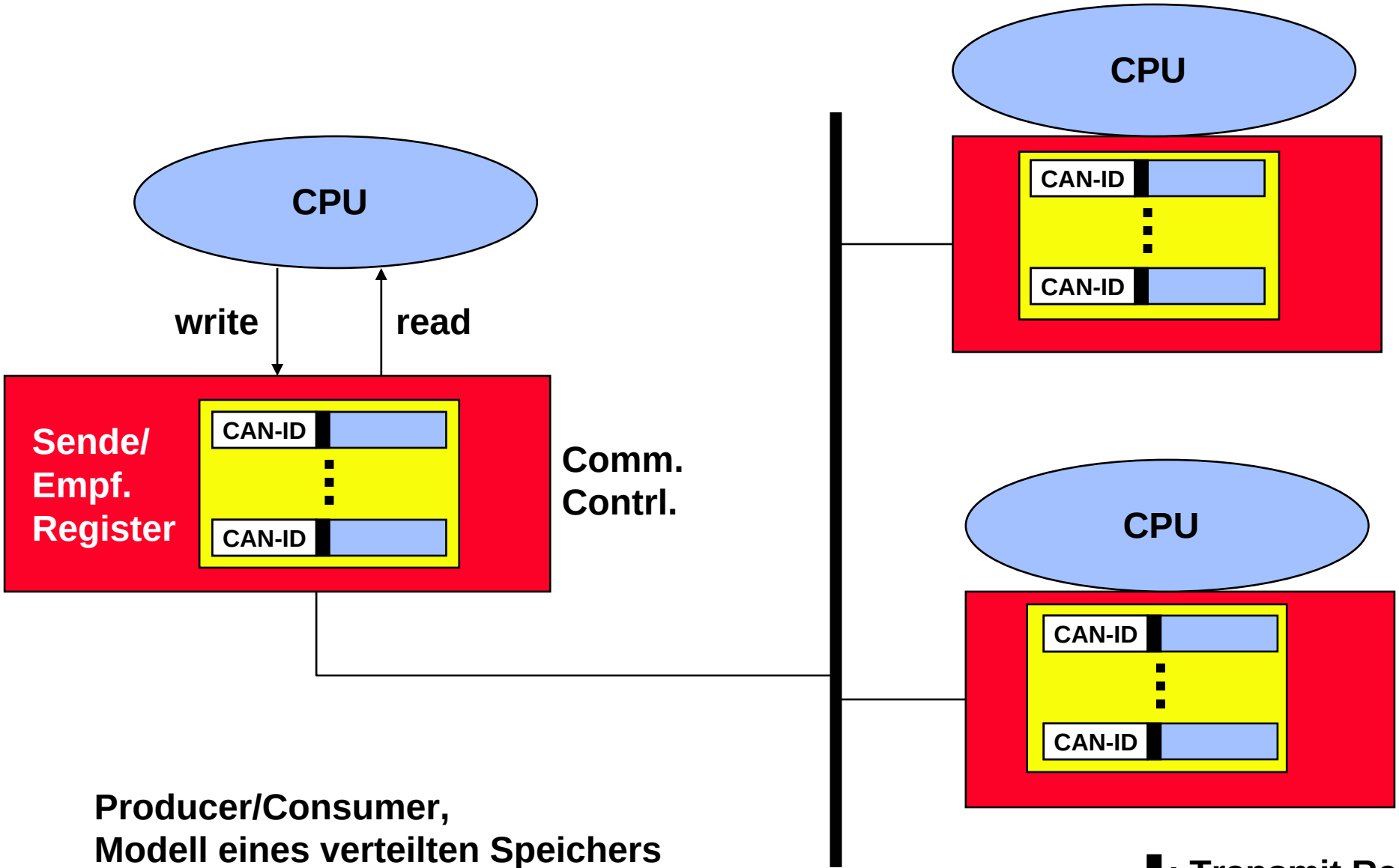
CAN-Bus, Datenrate 1Mbps

Szenario	Totzeit (µs)	
	min	max
Bitfehler	18.0	150.0
Stuff-Fehler	23.0	140.0
CRC-Fehler	54.0	143.0
Form-Fehler	52.0	150.0
Acknowledge-Fehler	53.0	142.0
Overload-Fehler	14.0	46.0
Overload Form Fehler	15.0	66.0
Medium Fehler	19.0	450.0
Senderfehler (bis Passivierung)		2400.0
Empfängerfehler (bis Passivierung)		2250.0

CAN-Bus Eigenschaften

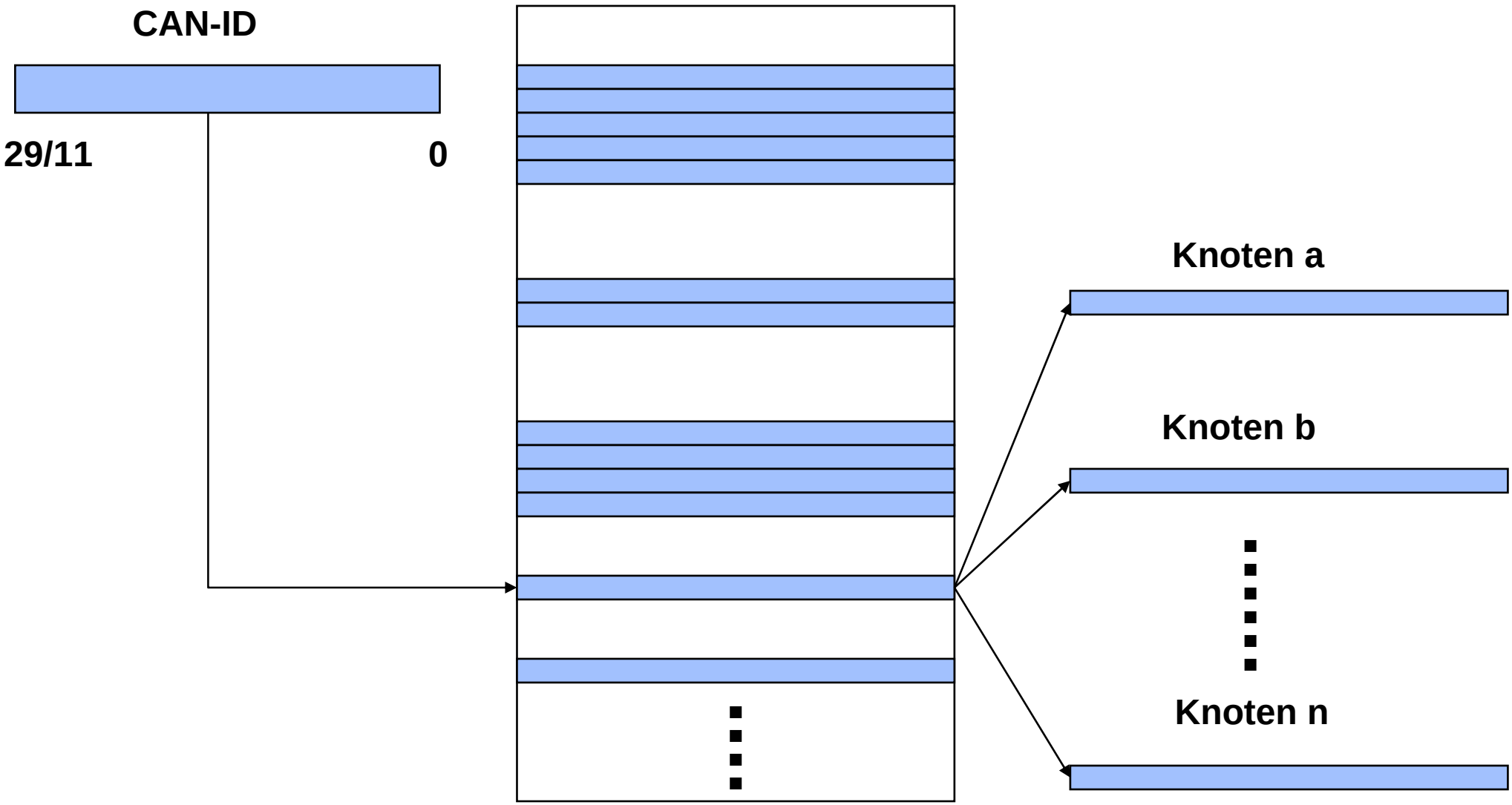
- **Ereignis-basierte Kommunikation für geringe Latenz**
- **Prioritäts-basierte verlustfreie Arbitrierung für garantierten Durchsatz**
- **Fehlerbehandlung:**
  - **Stations-neutrales positives Ack.**
  - **negatives Ack. im Fehlerfall (systemweite Fehlernachricht)**
  - **Identifikation defekter Stationen (ohne lokale Adressen !)**
  - **Sofortige Resynchronisation noch innerhalb des fehlerhaften Telegramms.**
- **Inhalts-basierte Adressierung für eine hohe Flexibilität (system elasticity)**

Das CAN-Kommunikationsmodell



Producer/Consumer,
Modell eines verteilten Speichers

▌: Transmit Request


CAN-ID
29/11
0
Knoten a
Knoten b
Knoten n

RT-Kommunikation am Beispiel

CAN-Bus

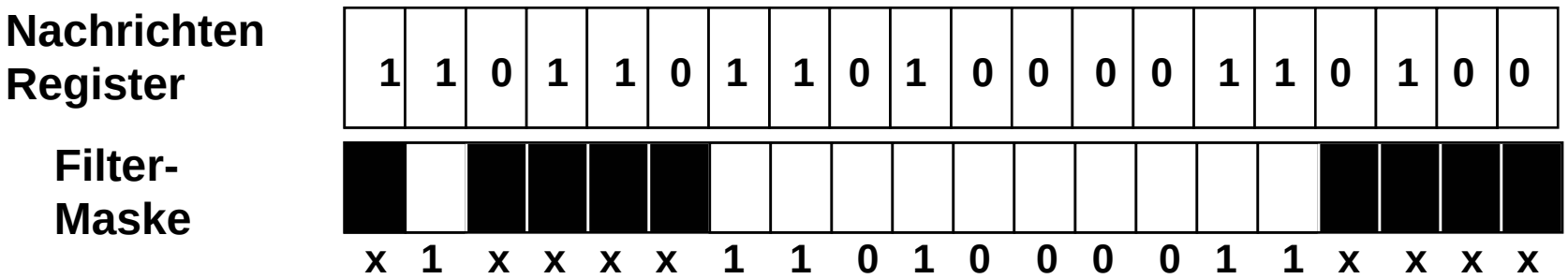
Das Routing Problem: **Wie kommt eine Nachricht vom Sender zum Empfänger ?**

CAN: Genereller Broadcast

Das Filter Problem: **Wie kann der Empfänger selektiv nur die Nachrichten empfangen die ihn wirklich interessieren?**

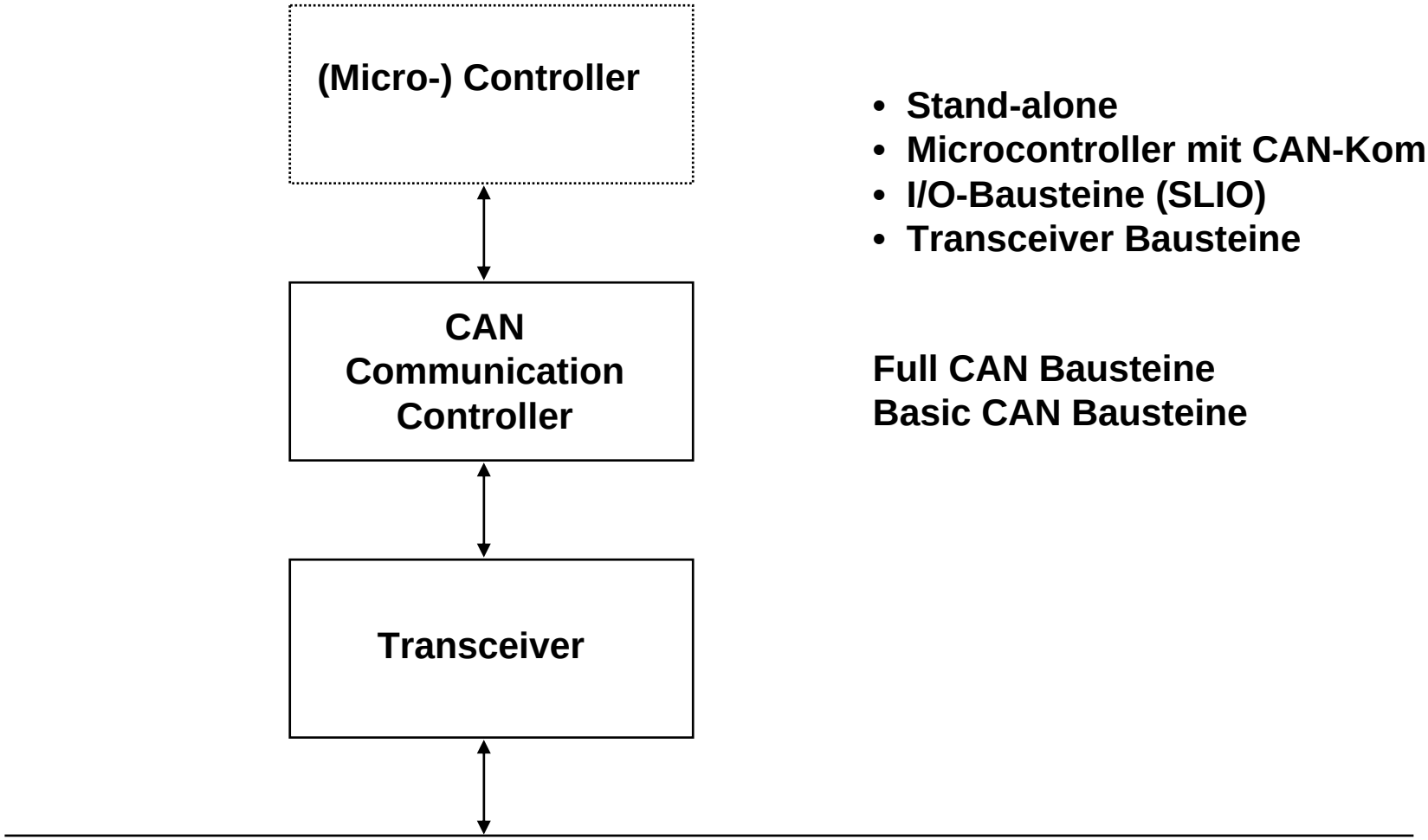
CAN: Nachrichtenfilter

Nachrichtenfilterung



Anzahl der Nachrichtenregister, Konfigurationsmöglichkeiten und Möglichkeiten der Nachrichtenfilterung sind abhängig vom verwendeten Kommunikationskontroller.

CAN Bausteine



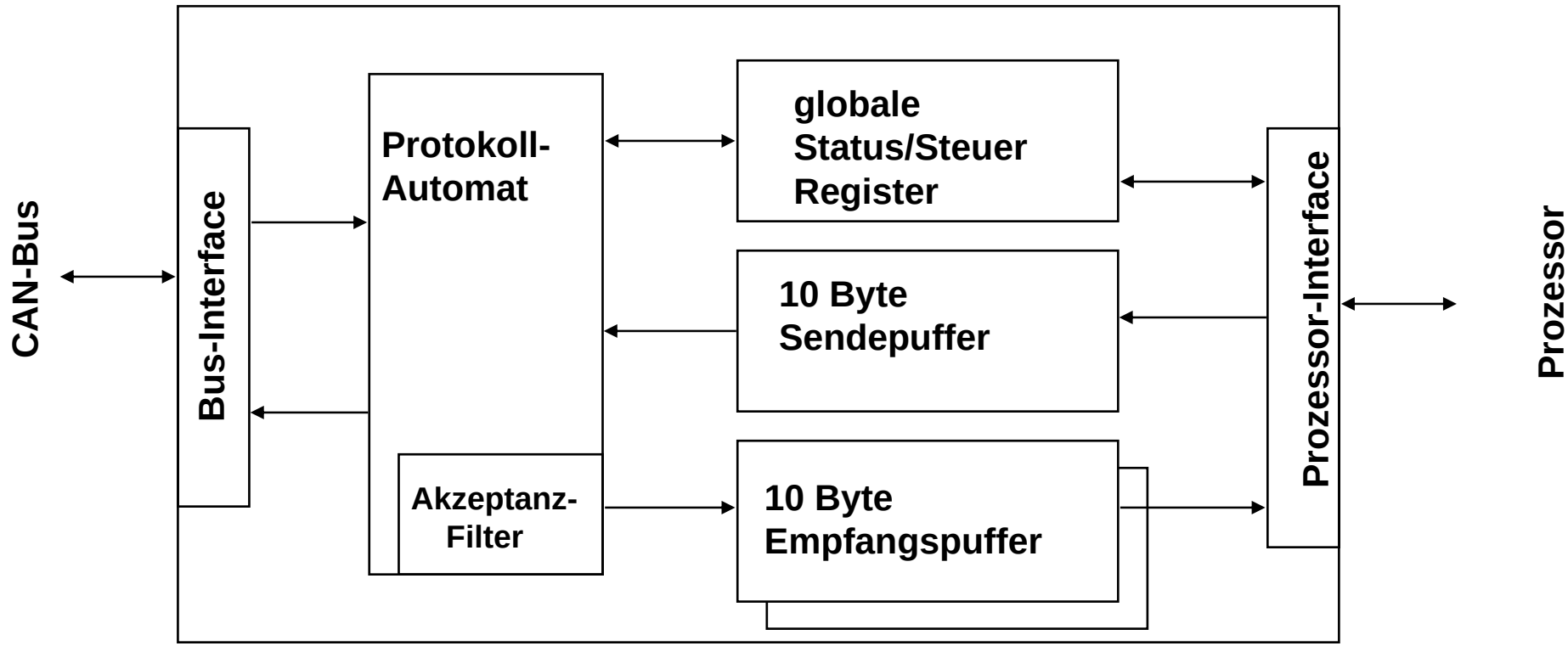
- **Stand-alone**
- **Microcontroller mit CAN-Komponente**
- **I/O-Bausteine (SLIO)**
- **Transceiver Bausteine**

Full CAN Bausteine
Basic CAN Bausteine

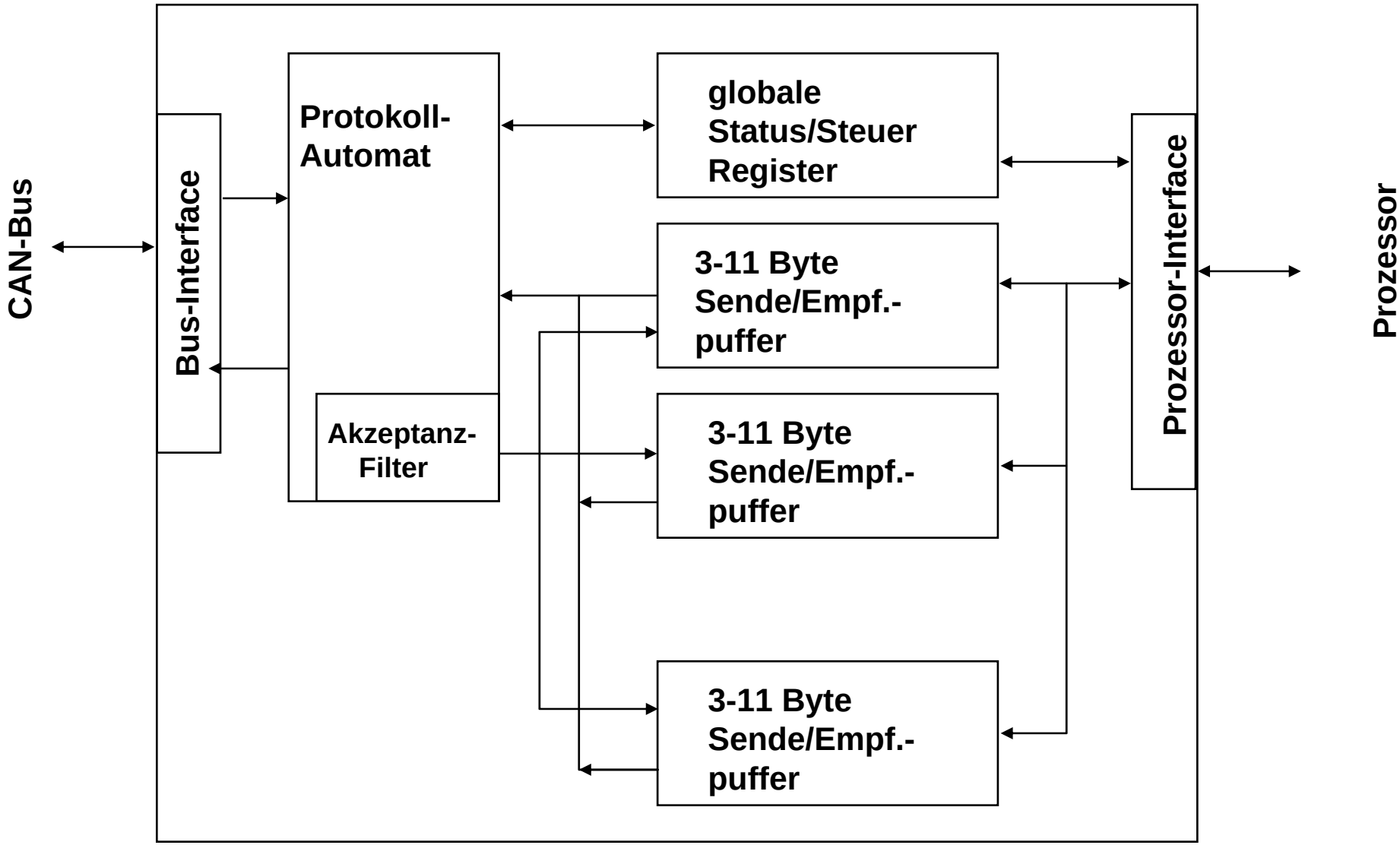
Aufgaben des CAN Controllers

- **Busarbitrierung**
- **Serialisierung der zu sendenden Telegramme**
- **Assemblierung der empfangenen Telegramme**
- **Berechnung bzw. Überprüfung der Checksumme**
- **Fehlererkennung und Fehlersignalisierung**
- **Bildung der CAN-Nachrichtenformate**
- **Einfügen bzw. Entfernen der zusätzlichen Bits beim Bit-Stuffing**
- **Erzeugen bzw. Überprüfen des Acknowledge-Bits**
- **Synchronisation des empfangenen Bitstroms**

Basic CAN


CAN-Bus
Bus-Interface
Protokoll-
Automat
Akzeptanz-
Filter
globale
Status/Steuer
Register
10 Byte
Sendepuffer
10 Byte
Empfangspuffer
Prozessor-Interface
Prozessor

Full CAN

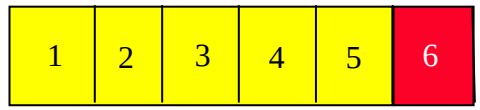

CAN-Bus
Bus-Interface
Protokoll-
Automat
Akzeptanz-
Filter
globale
Status/Steuer
Register
3-11 Byte
Sende/Empf.-
puffer
3-11 Byte
Sende/Empf.-
puffer
3-11 Byte
Sende/Empf.-
puffer
Prozessor-Interface
Prozessor

What CAN can’t

- **All-or-nothing property under all single (crash/omission) fault conditions**
- **Temporal guarantees for message transmissions**
- **Consistent order of messages**

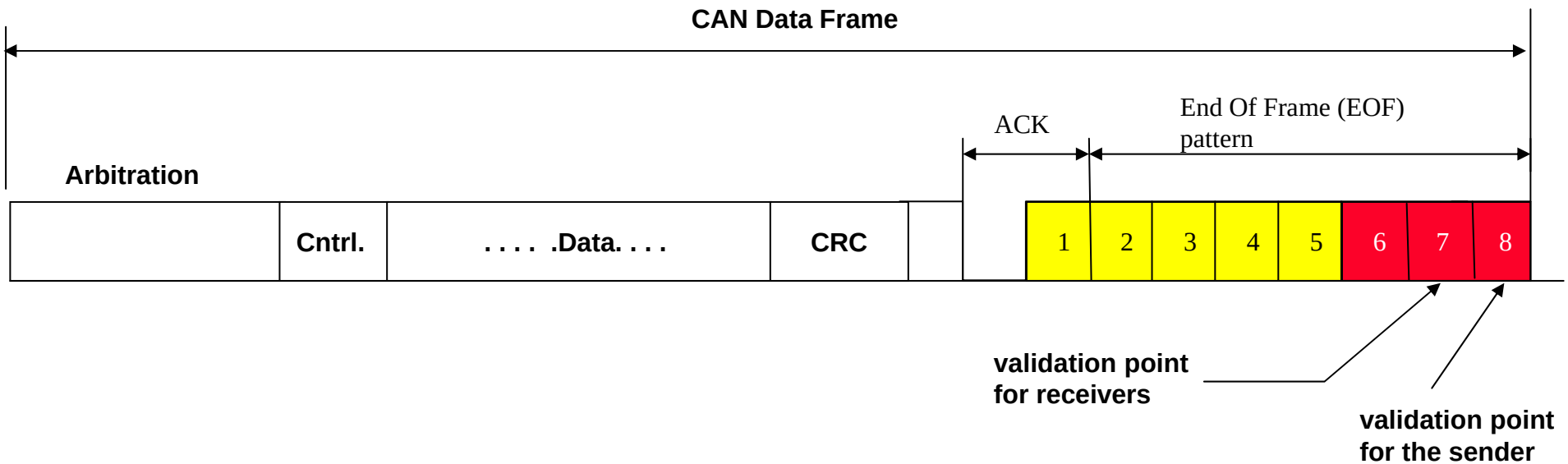
Error Detection and Error Signalling in CAN
The Case for Inconsistencies

Violation of the Bit-Stuffing Rule:
Used for Error Detection and Signalling



Bit-Stuffing enforces the following rule:

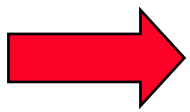
A sequence of 5 identical bit levels is followed by a complementary bit level



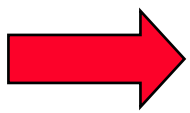
Konsequenzen aus dem Validierungsprotokoll

J. Rufino, P. Veríssimo, C. Almeida , L. Rodrigues: „Fault-Tolerant Broadcasts in CAN“,
Proc. FTCS-28, Munich, Germany, June 1998.

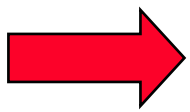
Jörg Kaiser, Mohammad Ali Livani: “Achieving Fault-Tolerant Ordered Broadcasts in CAN”
Proc. of the 3rd European Dependable Computing Conference, (EDCC-3), Prague, Sept. 1999



Inkonsistente Nachrichten-Duplikate

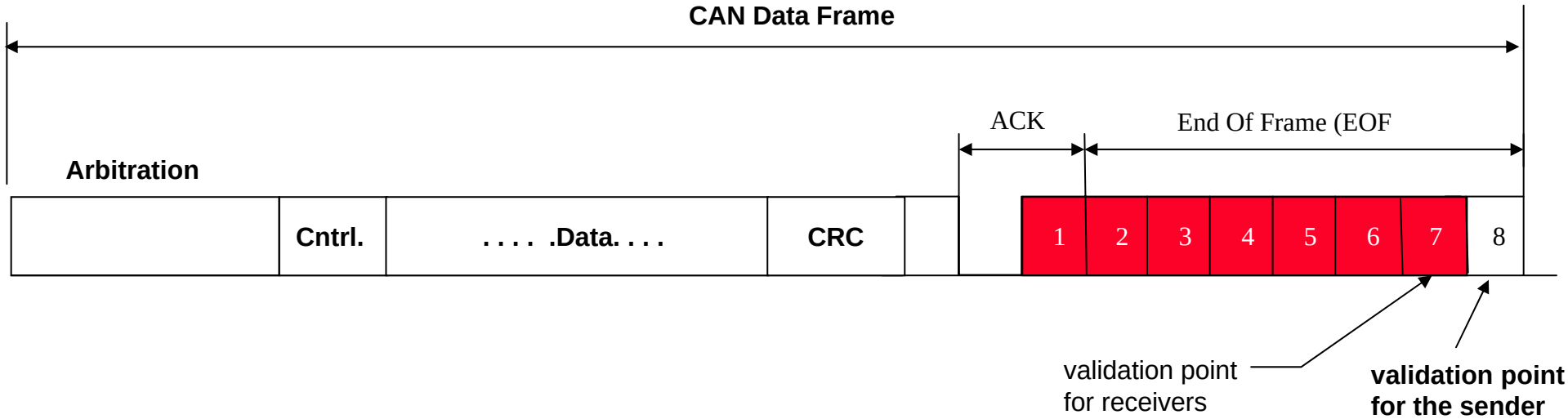


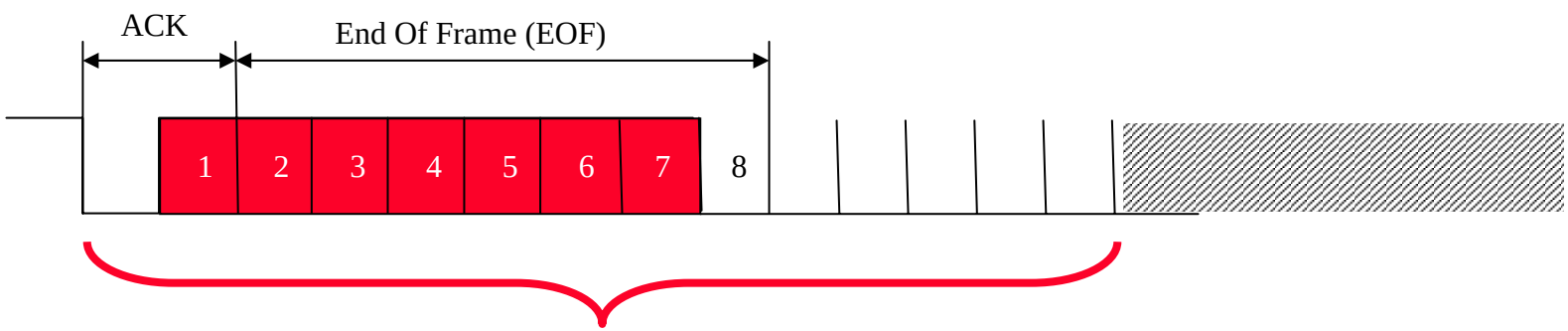
Inkonsistente Omission-Fehler



(Potentiell) Unbeschränkte zeitliche Verzögerungen

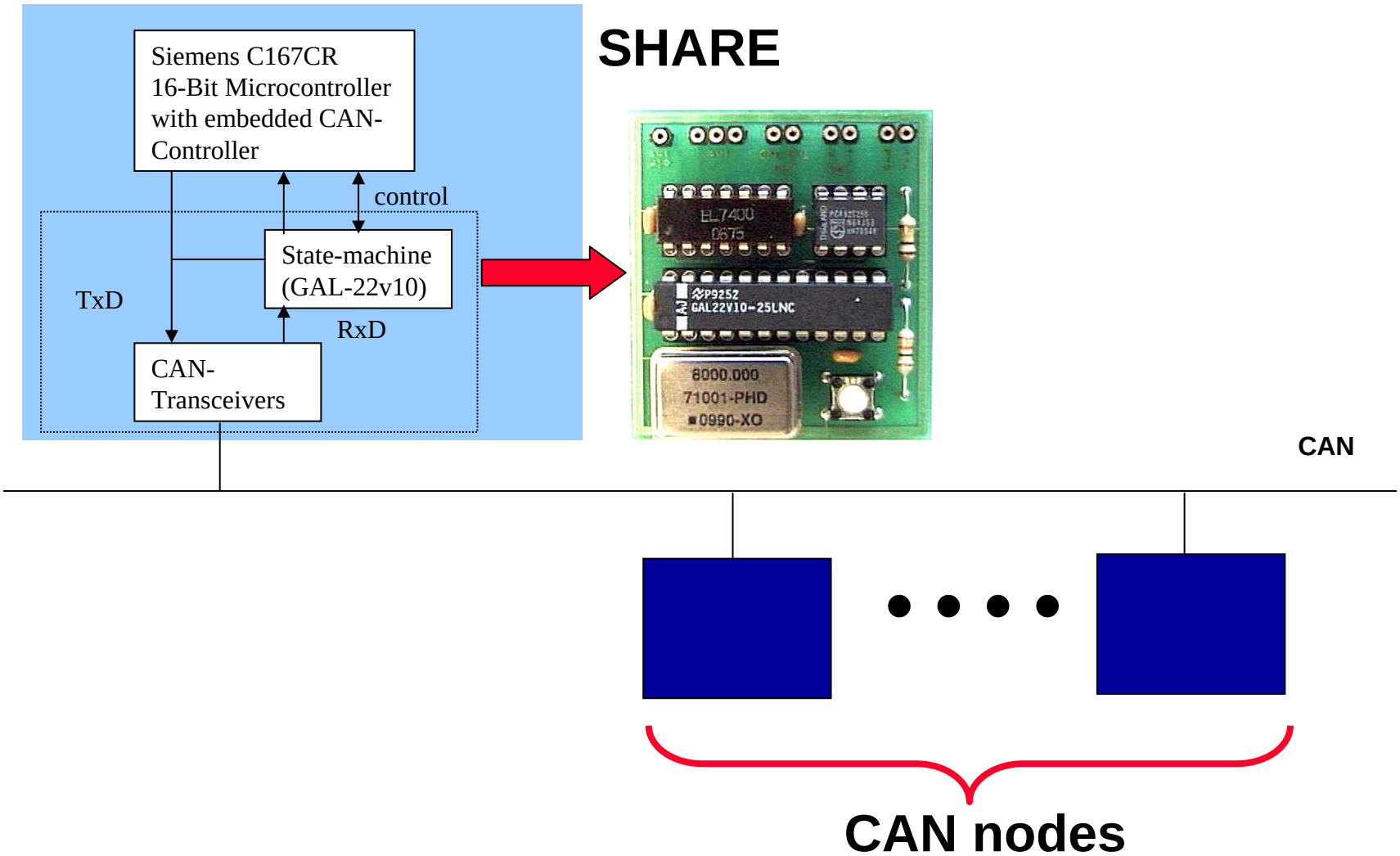
The Case for SHARE: Inconsistent Omissions





unique pattern : 1 dominant, 7 recessive, 6 dominant !

The Architecture of SHARE



ET vs. TT (Kopetz, RT Communication, 1996)

Szenario: 10 Rechnerknoten, verbunden über einen 100 kb Kommmunikationskanal.
Jeder Knoten soll 40 verschiedene Alarmsensoren überwachen, deren (binären) Zustand er innerhalb 100 ms an einen Alarm-Monitor übermitteln soll.

	ET-System	**TT-System**
Beispiel	**CAN-Bus**	**TTP**
Kontrolle	**Alarm-Ereignis**	**Polling alle 100msec/Knoten**
Alarm Kodierung	**1/Nachricht**	**40/Nachricht**
Nachrichtenlänge	**8+44 = 52**	**48 + 20 = 68**
Max. # der Nachr.	**ca. 2000**	**ca. 1500**
benötigte Nachr.	**0...4000**	**100 konstant**
Overhead/Nachricht	**44/62 Bit**	**20**
Max. Delay	**einige µsec**	**50-100 msec**
Problem	**max. # Nachrichten**	**Verzögerungszeit**
Entwurfsfragen	**kann man Alarmnachrichten zusammenfassen ?** **Wie lange sollte man “samplen” ?**	**was macht man, wenn der Alarm nach 20 msec übermittelt werden soll?** **Wie skaliert die Anzahl der Knoten?**

Koexistenz von zeitgesteuerten und ereignisgesteuerten Mechanismen auf dem CAN-Bus

???

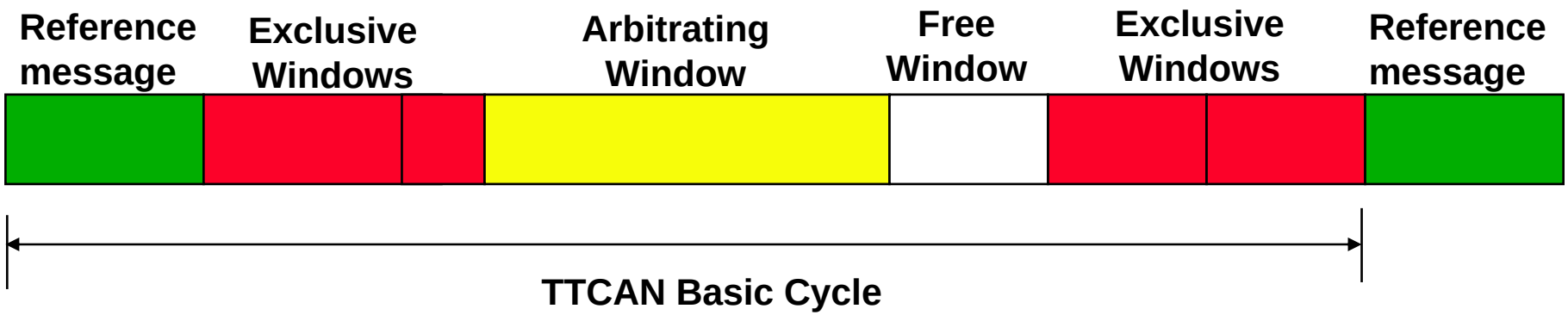
Geht es und was ist der Preis?

1. **Time Triggered CAN: TTCAN (Führer, Müller, Dieterle, Hartwich, Hugel, Walther,(Bosch))**
2. **Dynamische Prioritäten (Kaiser, Livani)**

Time Triggered CAN
TTCAN

Time Triggered CAN: TTCAN (Führer, Müller, Dieterle, Hartwich, Hugel, Walther,(Bosch))

Basic Cycle and Time Windows



reference message: indicates the start of a cycle,
exclusive window : used for critical periodic state messages,
arbitrating window: used for spontaneous state and event messages,
free window : window for further extensions and gap to the next exclusive window.

RETRANSMISSIONS ARE GENERALLY NOT ALLOWED IN TTCAN !!

Scheduling a Basic cycle on a node

Node n

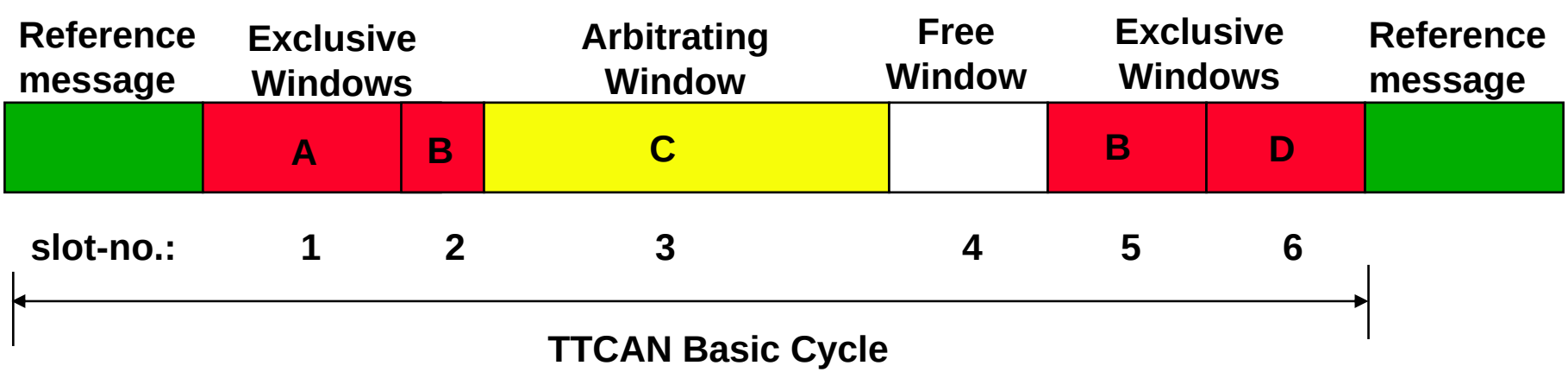
Send msg B in slot 2 and 5

Send msg F in slot 3

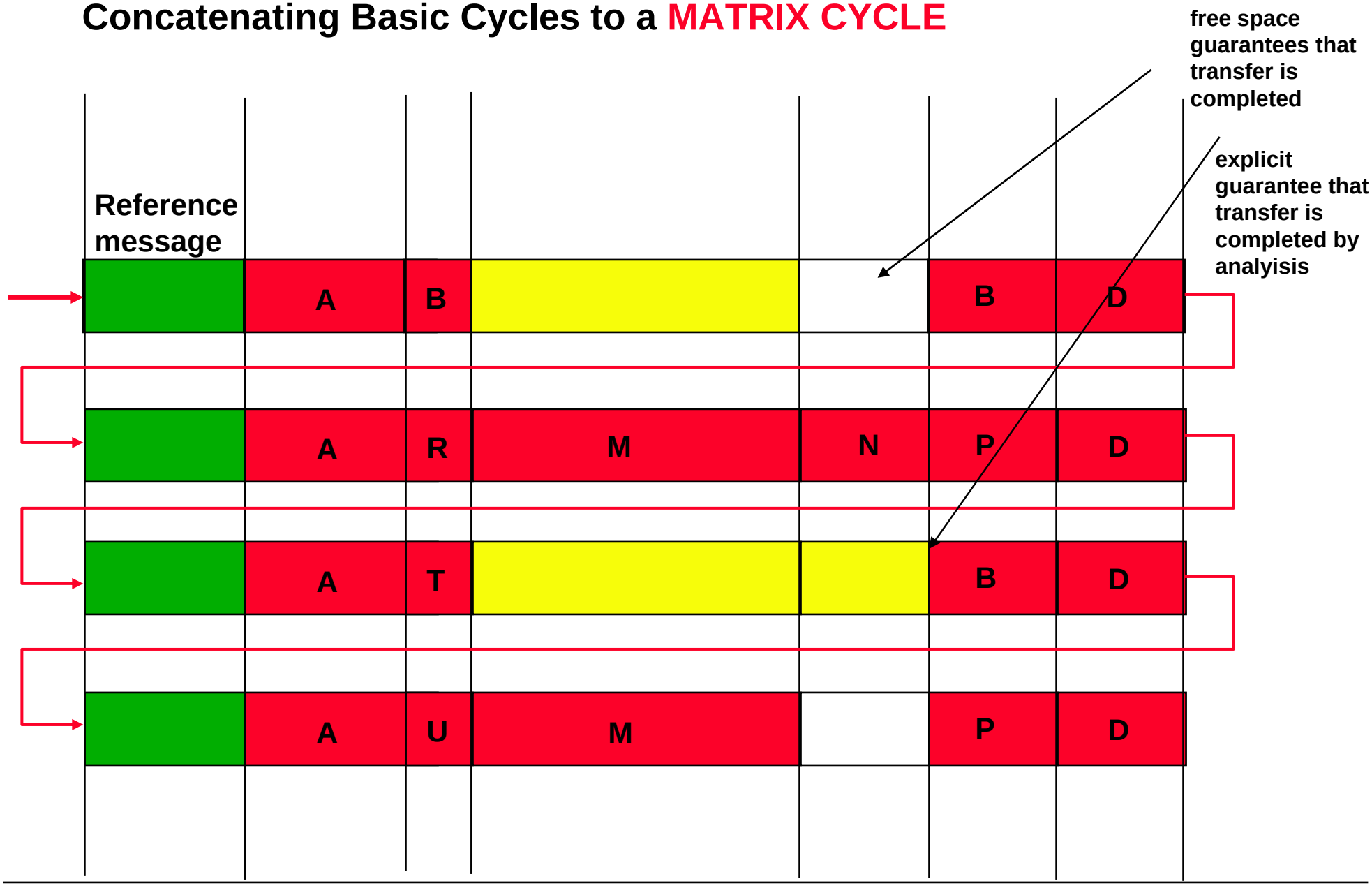
Receive msg D in slot 6

Constraint:

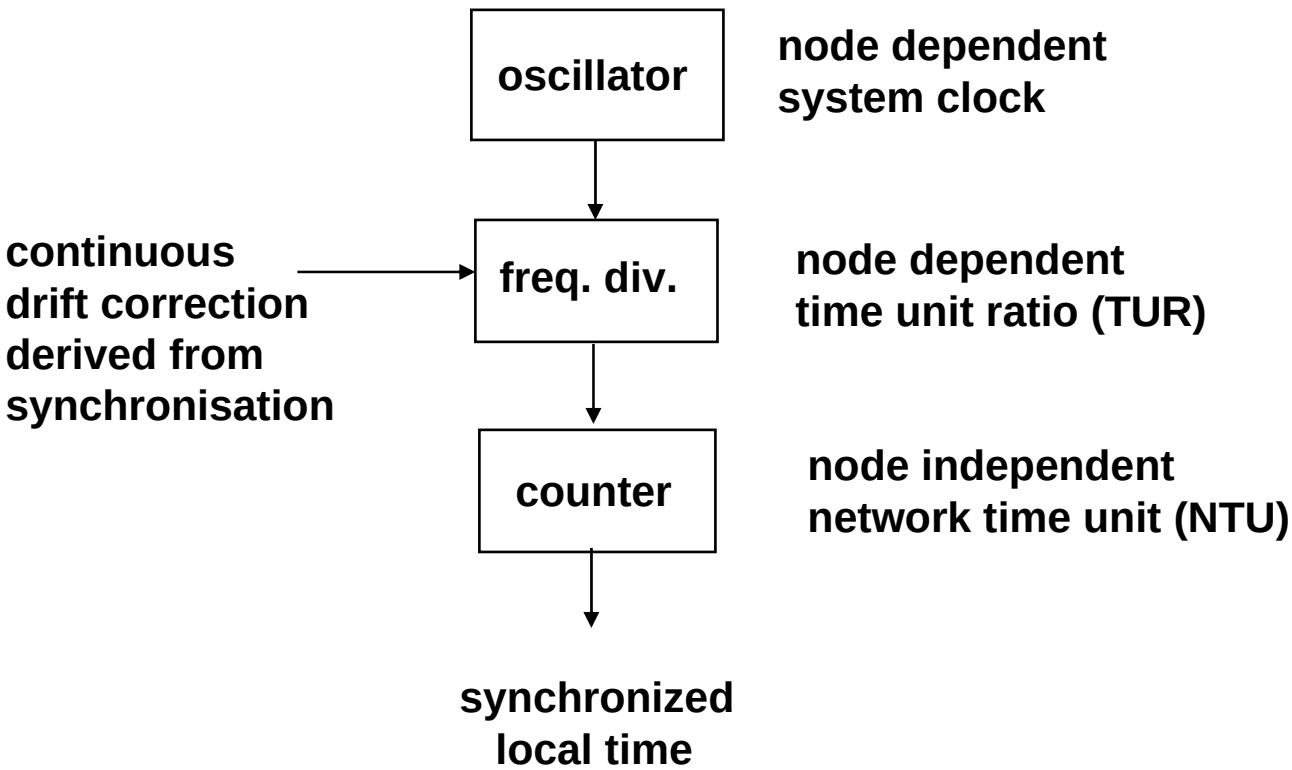
A message transfer in an arbitrating window must be successfully completed before the start of an exclusive window.



Concatenating Basic Cycles to a MATRIX CYCLE



Time and synchronization in TTCAN



Synchronization based on the existence of a Time Master.

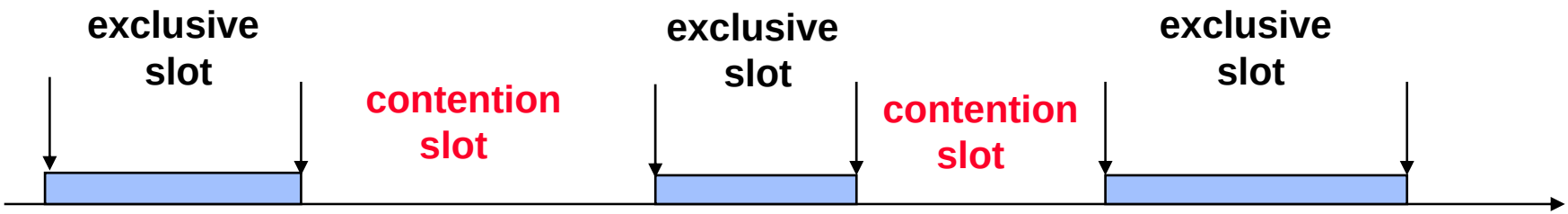
All nodes take a snapshot of their local time at the SoF (Start of Frame) bit of the reference message.

Because of dependability reasons, TTCAN supports redundant Time Masters.

Arbitration among Time Masters is based on the priority scheme of CAN.

Integration von TT- und ET- Kommunikation durch dynamische Prioritäten

Basic Idea: Reserve slots for hard real-time traffic and schedule soft real-time traffic in the remaining slots

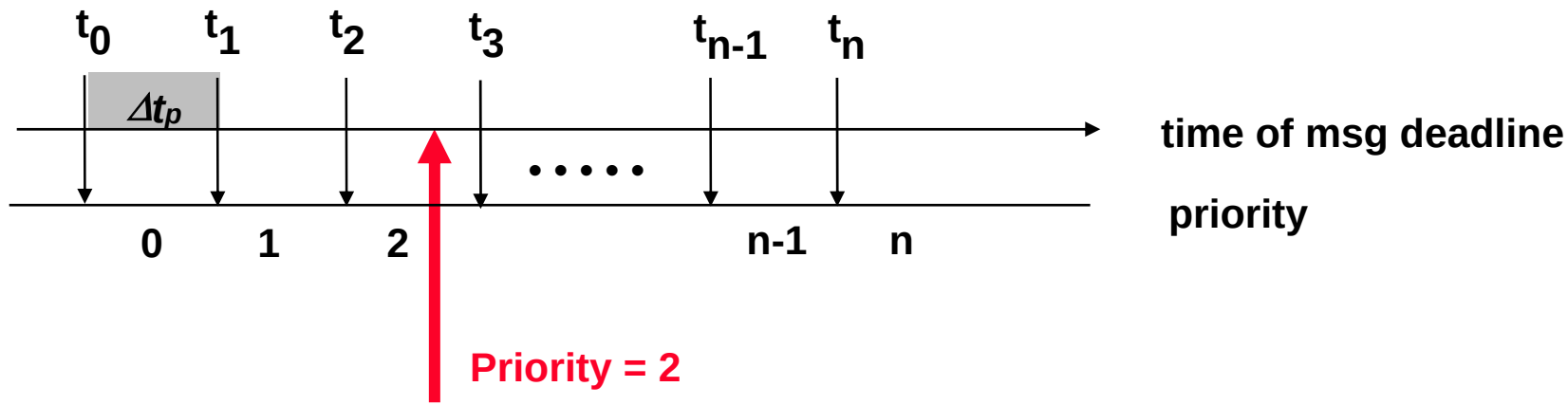


The priority scheme is used to enforce high priority message transmission in the exclusive slots.

What is the advantage over TDMA?

Mapping Deadlines to Priorities

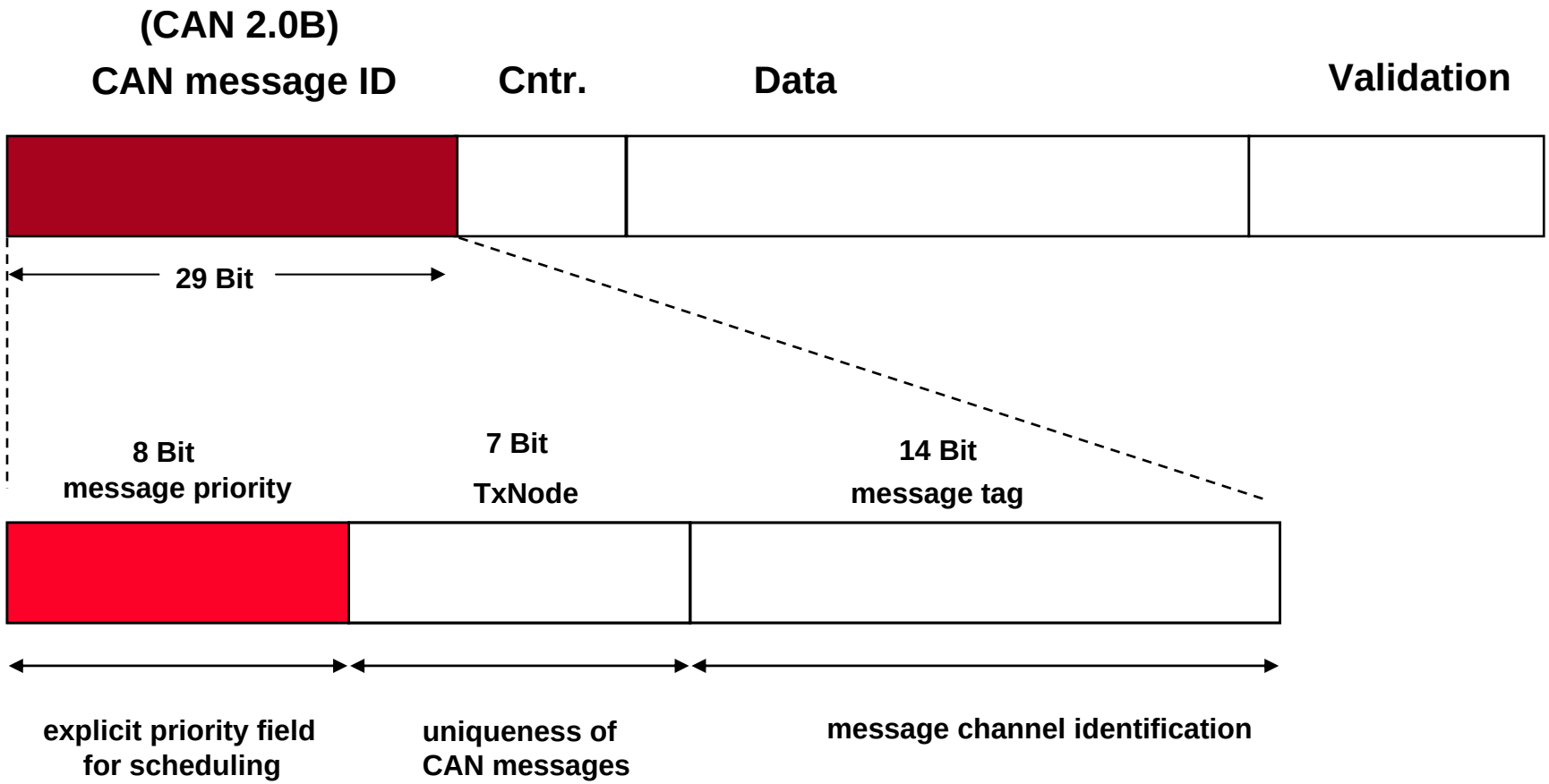
- Messages have deadlines
- Deadlines can be transformed into priorities



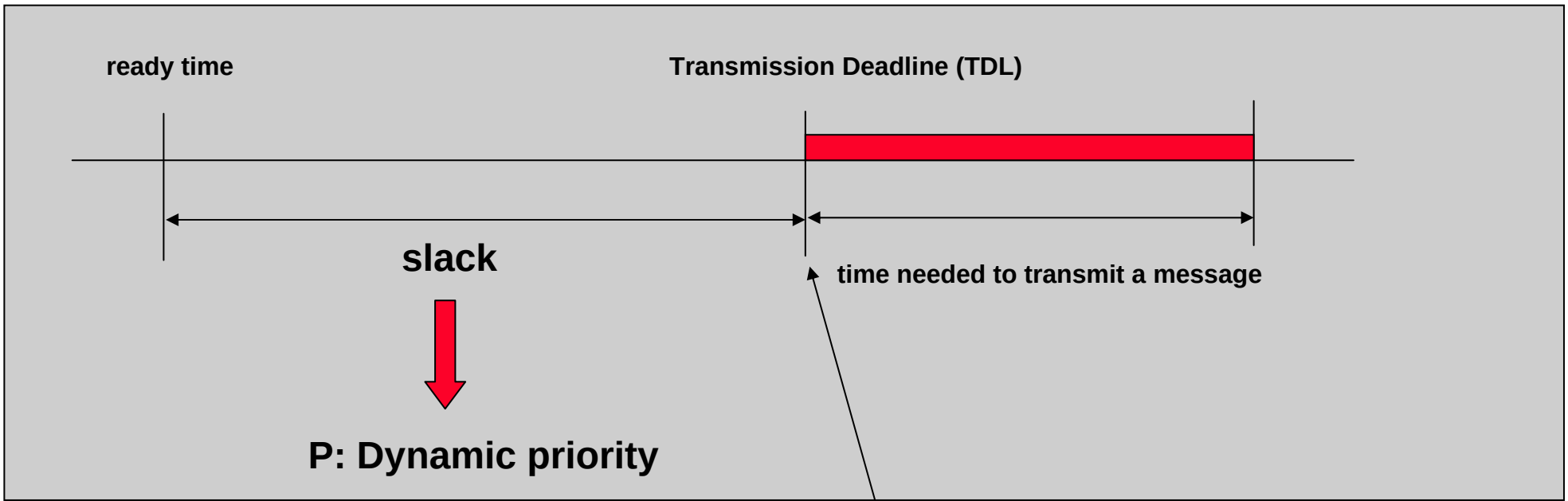
Wanted

a global priority-based message dispatcher

Structuring the CAN-ID



Scheduling messages with guarantees



At TDL: $P_{HRTM} > P_{SRTM} > P_{NRTM}$

How many HRT-slots can be guaranteed ?

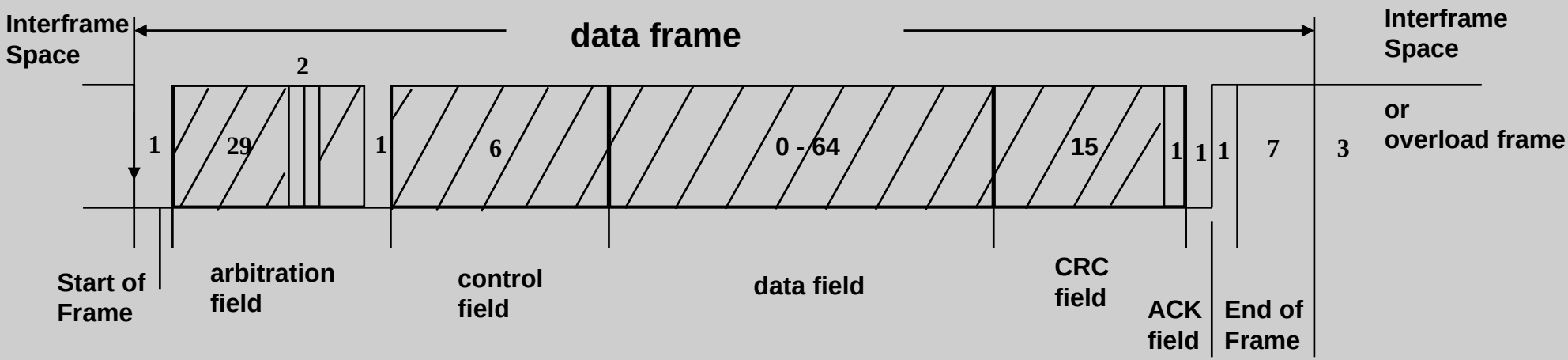
ΔC_{max} + δ_{clock}

ΔC_{max} **max. time interval (possibly under failure assumptions), which is neccessary to safely transmit a message to the destination**

ΔC_{max} is a worst case assumption under all anticipated load and failure conditions

δ_{clock} **max. offset, i.e. the difference between any two local clocks**

CAN Data Frame



longest possible message:

Format-Overhead:	67 bit times
Data:	64 bit times
Bitstuffing (max):	23 bit times
total:	154 bit times

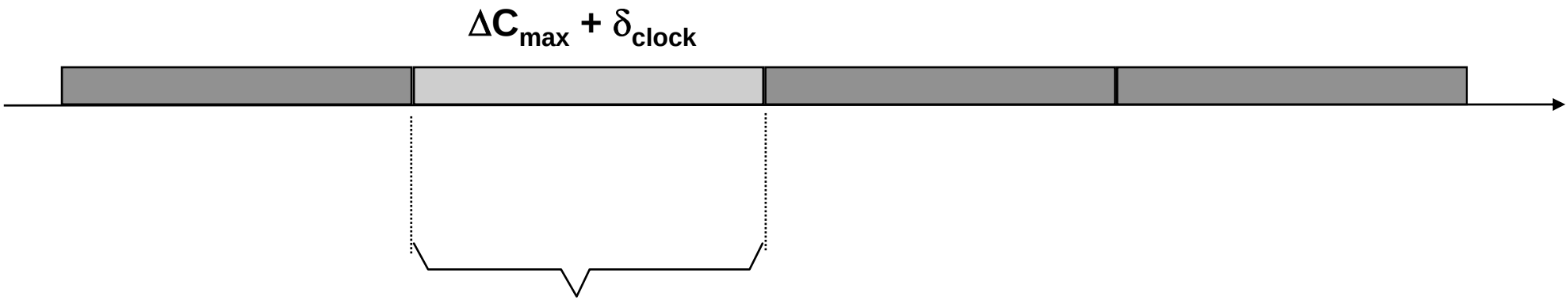
CAN Inaccessibility Times*

Data Rate 1 Mbps , Standard Format

Scenario	t_{inacc} (µs)		
Bit Errors	155.0	(174)	← worst case single
Bit Stuffing Errors	145.0		
CRC Errors	148.0		
Form Errors	154.0		
Ack. Errors	147.0		
Overload Errors	40.0		
Reactive Overload Errors	23.0		
Overload Form Errors	60.0		
Multiple Consecutive Errors (n=3)	195.0		
Multiple Successive Errors (n=3)	465.0		
Transmitter Failure	2480.0	(2784)	
Receiver Failure	2325.0		← worst case multiple

P. Verissimo, J. Ruffino, L. Ming:" How hard is hard real-time communication on field-busses?"

Utilization of CAN for HRT-messages

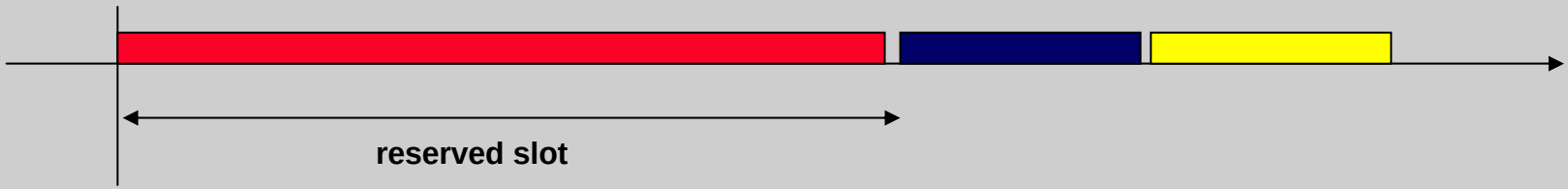


fault assumption n	m	ΔC_{max} + 50 µs δ_{clock} (µs)	HRT messages / sec. #
0	0	358	2793
0	1	532	1880
0	3	880	1136
1	0	2988	335
1	3	3664	273

Benefits of the approach

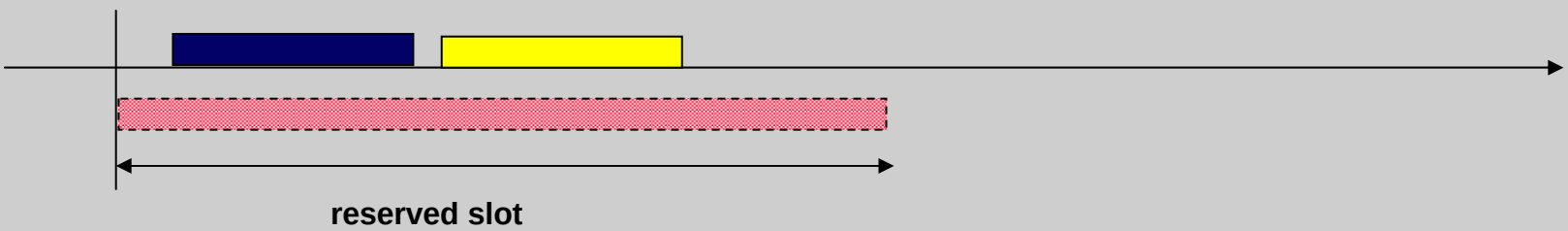
Media access controlled by global time only (TDMA)

All nodes need global time
Unused slots remain unused



Media access in a system controlled by our priority scheme

Only nodes with HRT-msg need global time
Unused slots can be used by msg which are ready to be transmitted



Fazit:

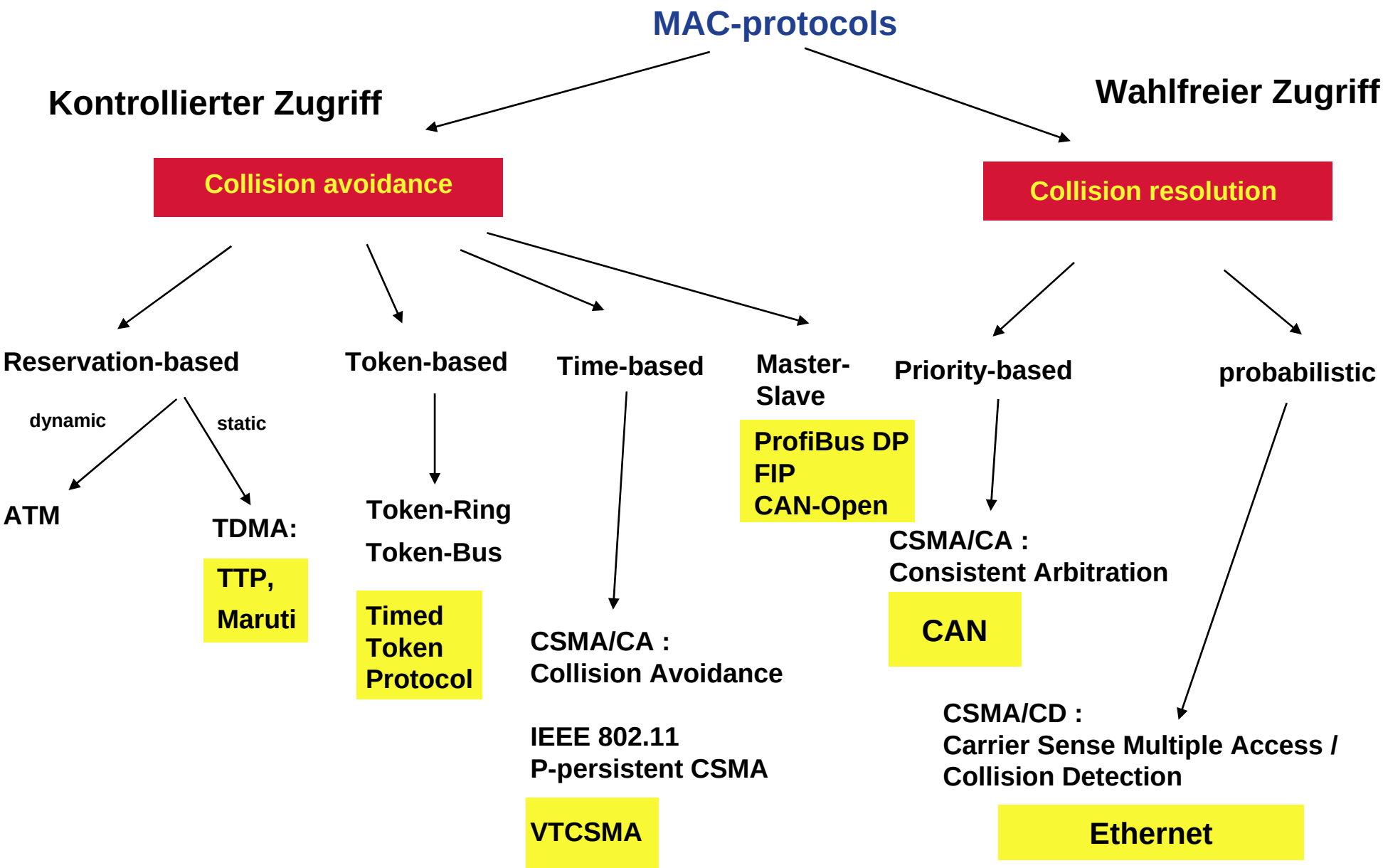
CAN bietet sehr gute Eigenschaften zur Realisierung eines ereignisgesteuerten Kommunikationsmodells.

Die konsistente Arbitrierung erlaubt die Planung und Analyse der Kommunikation nach bekannten Schedulingverfahren.

Durch Erweiterungen lassen sich pathologische Fehlersituationen weiter verringern.

Eine Koexistenz von zeitgesteuerten und ereignisgesteuerten Verfahren ist möglich.

Durch Ausnutzung der prioritätsbasierten Arbitrierung lassen sich Nachteile der zeitgesteuerten Verfahren vermeiden, allerdings ist die Protokolleffizienz von CAN geringer.


MAC-protocols
Kontrollierter Zugriff
Wahlfreier Zugriff
Collision avoidance
Collision resolution
Reservation-based
Token-based
Time-based
Master-Slave
Priority-based
probabilistic
dynamic
static
ATM
TDMA:
TTP,
Maruti
Token-Ring
Token-Bus
Timed Token Protocol
ProfiBus DP
FIP
CAN-Open
CSMA/CA :
Consistent Arbitration
CAN
CSMA/CA :
Collision Avoidance
IEEE 802.11
P-persistent CSMA
VTCSMA
CSMA/CD :
Carrier Sense Multiple Access /
Collision Detection
Ethernet