
Accessing the shared communication medium

Media
Access
Conntrol



Network-topologies

- **Bus**
- **Ring**
- **Star**
- **Tree**
- **grid, mesh**
- **fully connected**

assessment criteria:

Overhead, latency, tolerance of transmission errors and network partitions



What are the impairments of predictability ?

Load

Failures



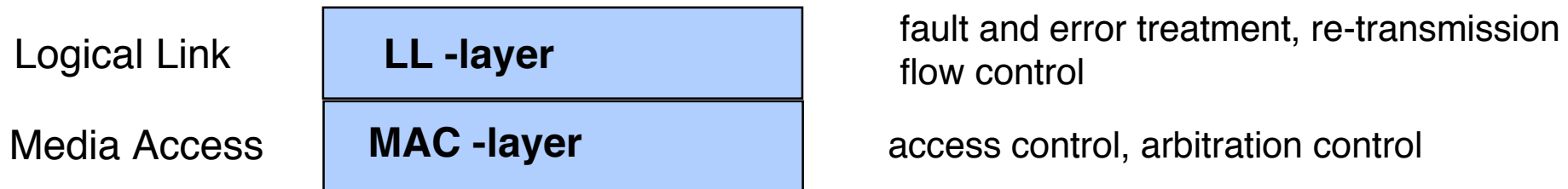
**Network contention
Arbitration conflicts**

**transmission errors,
lost messages**



Media Access Control & Logical Link Layer

transfer of data blocks
flow control
fault and error handling
message re-transmissions



MAC-protocols

controlled access

random access

Collision avoidance

Collision resolution

Reservation-based

dynamic

static

ATM

TDMA:

**TTP,
Maruti**

Token-based

**Token-Ring
Token-Bus**

**Timed
Token
Protocol**

Time-based

**CSMA/CA :
Collision Avoidance**

**IEEE 802.11
P-persistent CSMA**

LON, VTCSMA

**Master-
Slave**

**ProfiBus DP
FIP
CAN-Open**

Priority-based

**CSMA/CA :
Consistent Arbitration**

CAN

**CSMA/CD :
Carrier Sense Multiple Access /
Collision Detection**

Ethernet

probabilistic



Predictability in random access networks:

probabilistic

very low overhead and latency in low load conditions
very flexible wrt. extensibility
thrashing in high load situations

Collision avoidance

balances the latency against the collision probability
maintains a good average throughput in medium load situations
may adapt to high load conditions

Consistent arbitration with Collision Resolution

needs support from the physical layer
maintains a constant throughput in all load conditions
supports sophisticated fault handling



Controlled Access by Collision Exclusion:

Master/Slave

all control information in one place
maximum of control
easy to change

Global Time

Easy temporal co-ordination
Minimal communication overhead

Token-based

Decentralized mechanism
Integration of critical and non-critical messages



CAN-Bus Controller Area Network



CAN Milestones

1983	Start of the Bosch internal project to develop an in-vehicle network
1986	Official introduction of CAN protocol
1987	First CAN controller chips from Intel and Philips Semiconductors
1991	Bosch's CAN specification 2.0 published
1991	CAN Kingdom CAN-based higher-layer protocol introduced by Kvaser
1992	CAN in Automation (CiA) international users and manufacturers group established
1992	CAN Application Layer (CAL) protocol published by CiA
1992	First cars from Mercedes-Benz used CAN network
1993	ISO 11898 standard published
1994	1st international CAN Conference (iCC) organized by CiA
1994	DeviceNet protocol introduction by Allen-Bradley
1995	ISO 11898 amendment (extended frame format) published
1995	CANopen protocol published by CiA
2000	Development of the time-triggered communication protocol for CAN (TTCAN)



The CAN Standard

Developed by BOSCH, <http://www.semiconductors.bosch.de/pdf/can2spec.pdf>

CAN Specification 1.2

CAN Specification 2.0

Difference between the specifications mainly is:

- **the different length of message identifiers (CAN-ID)**

Standard CAN: 11 Bit IDs (defined in CAN 2.0 A ← 1.2)

Extended CAN: 29 Bit IDs (defined in CAN 2.0 B)

CAN-Controller Implementations:

Basic CAN: 1 Transmit + 1 Receive (Shadow) Buffer

Extended CAN: 16 Configurable Transmit/Receive Buf.

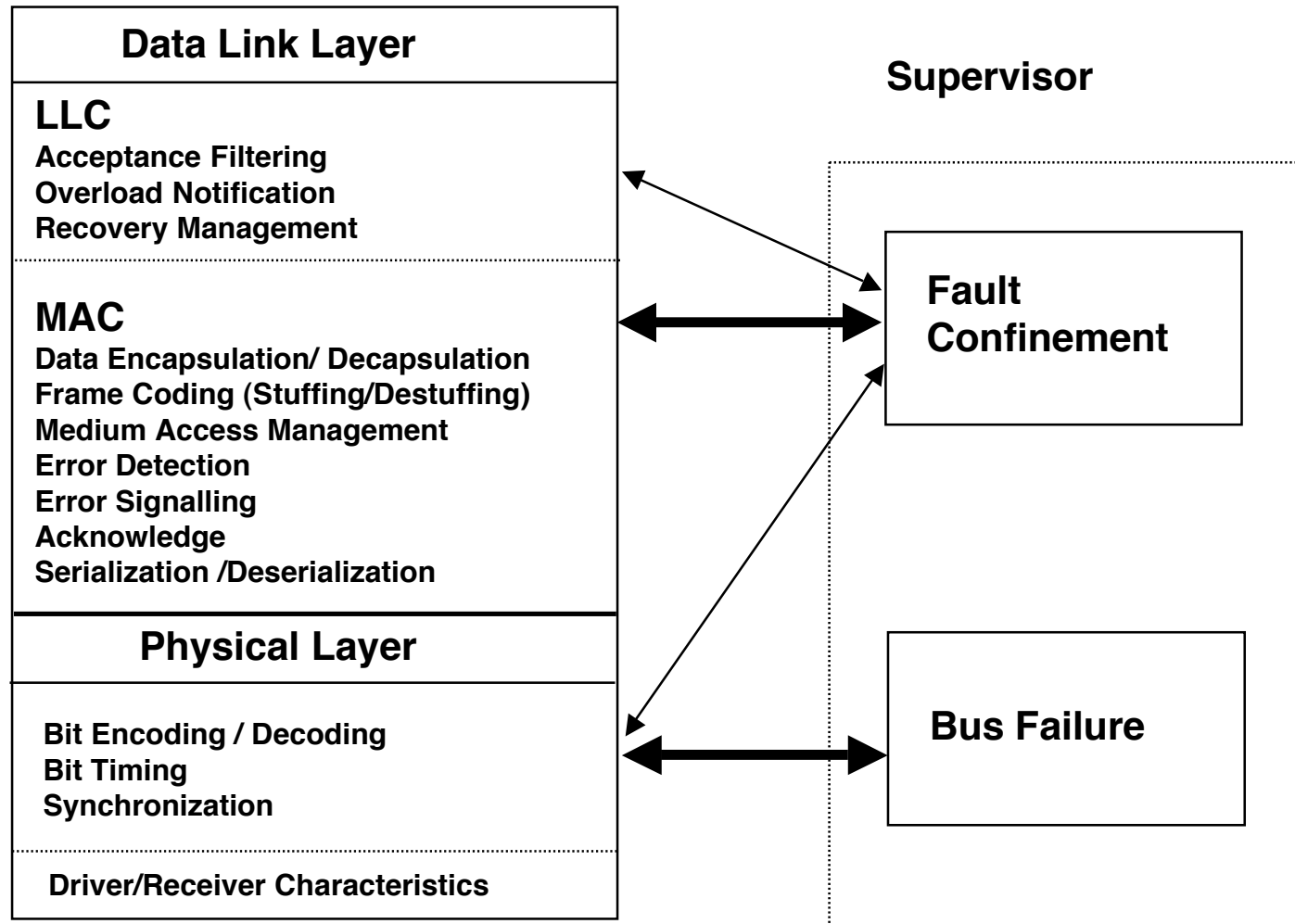


Basic CAN properties

- **Prioritised messages**
- **Bounded and guaranteed message delay for the highest priority message.**
- **Constant throughput in all load situations**
- **Error detection and signalling in the nodes.**
- **Automatic re-transmission.**
- **Fail silent behaviour of nodes.**
- **Consistent message delivery.**
- **Multicast with time synchronization.**



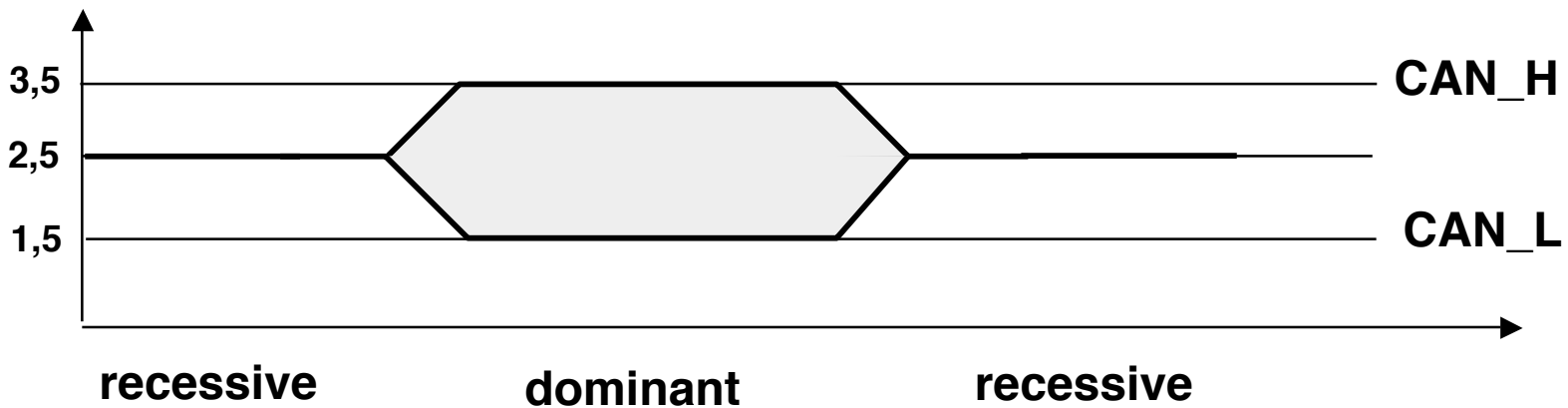
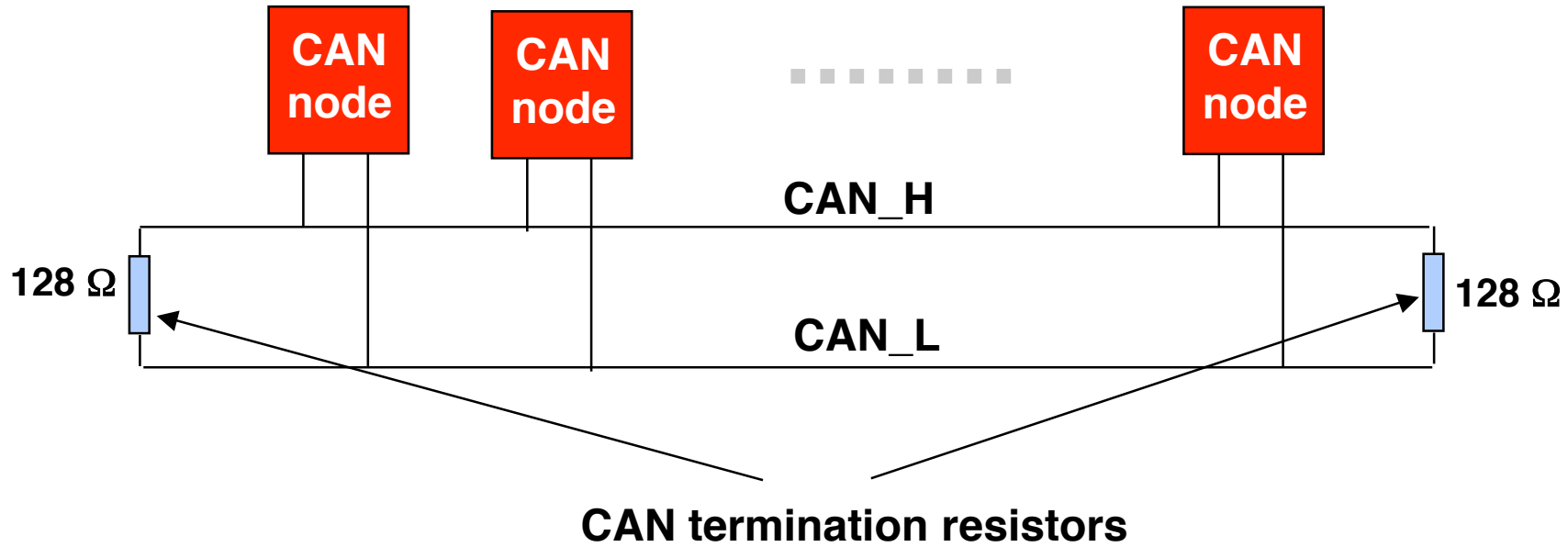
Layers defined by the CAN standard



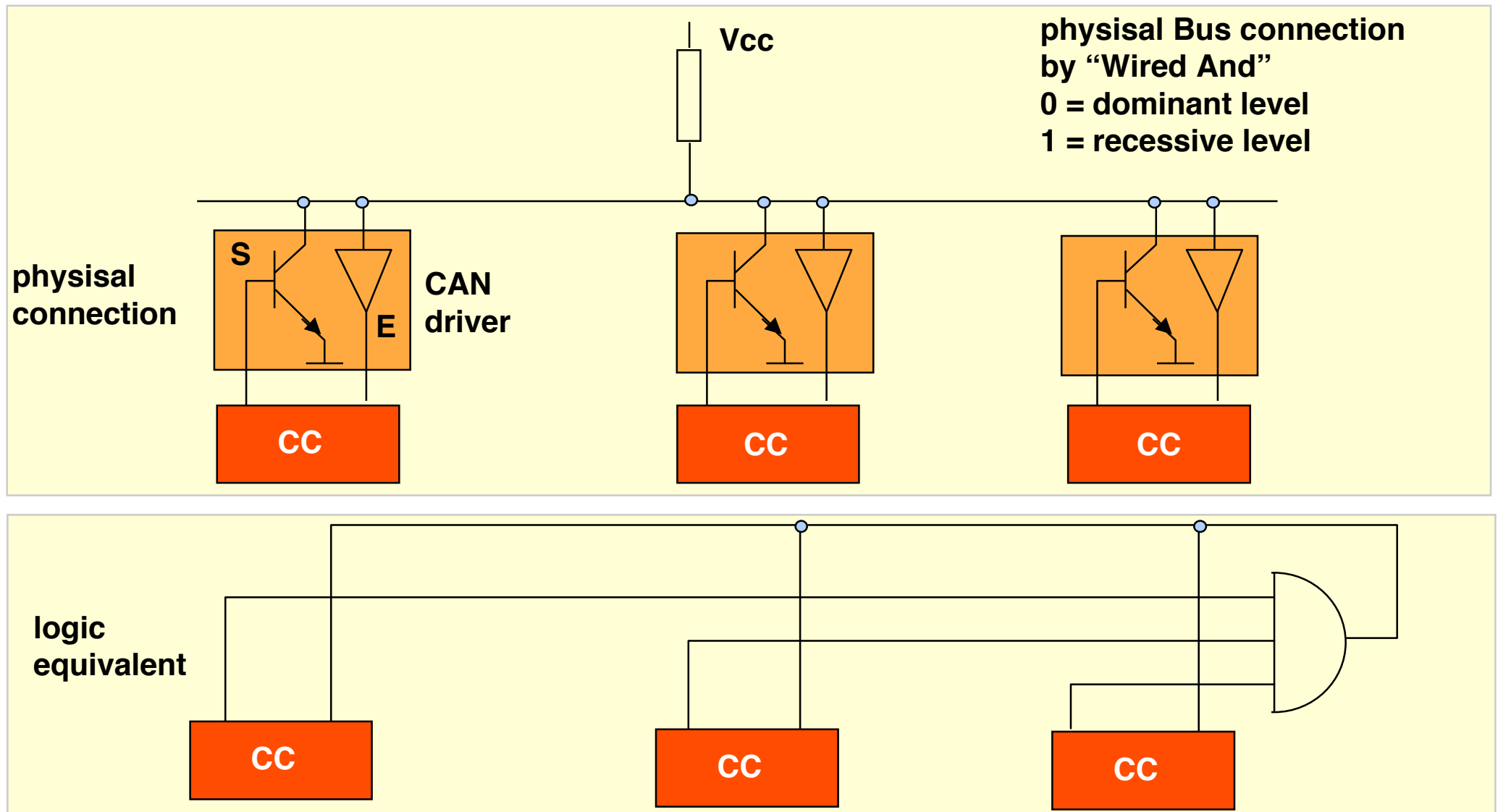
LLC = Logical Link Control
MAC = Medium Access Control



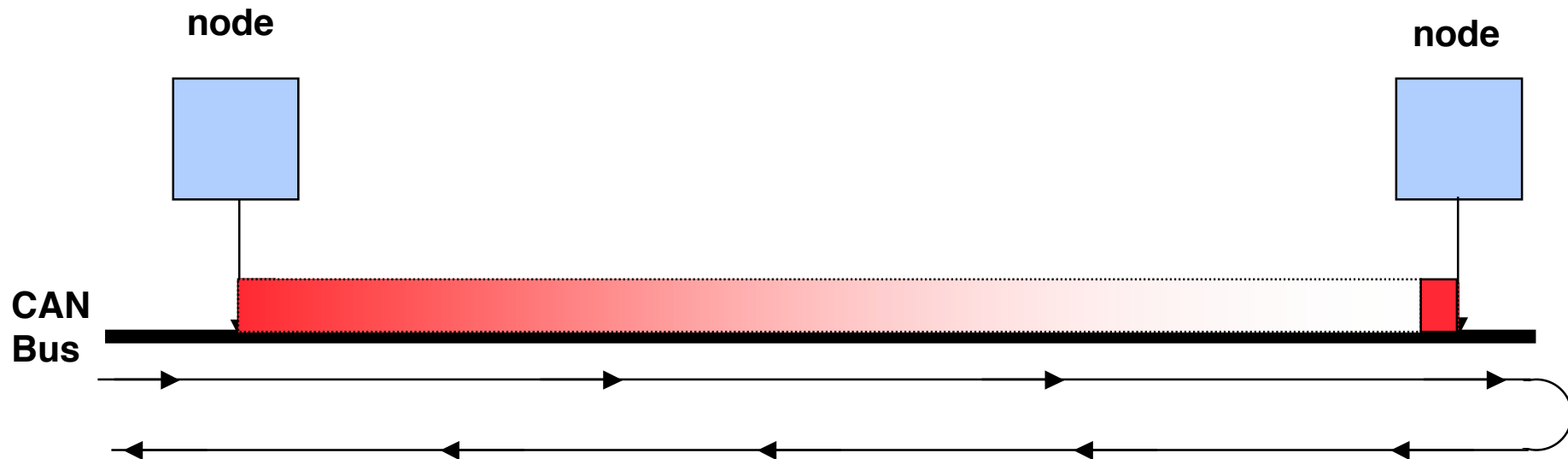
CAN differential transmission scheme



The CAN physical layer



CAN Bit Synchronisation



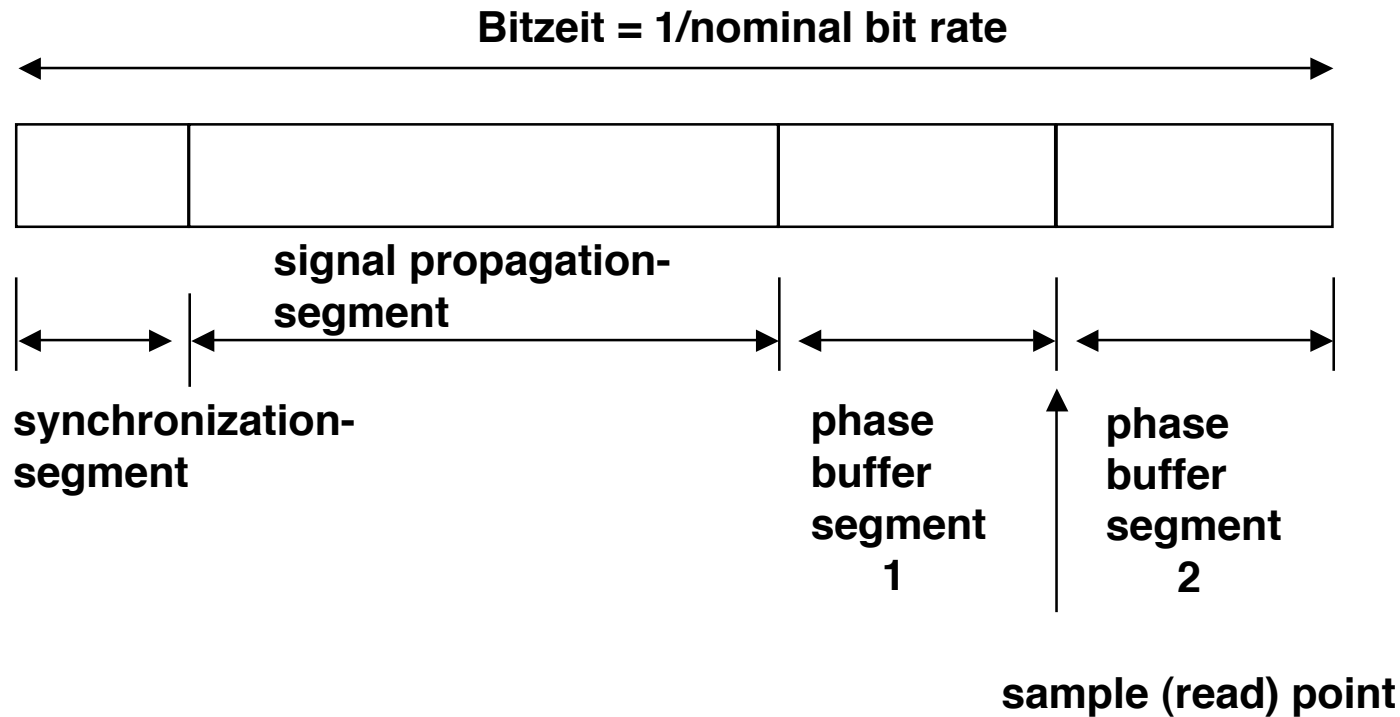
After a certain time, all nodes have seen the value of a bit

Bit rate dependend on the length of the bus

Bit Monitoring



Bit-timing and bit synchronization



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

synch.-segment	1	time quanta
sig. propag. seg.	1...8	time quantas
phase buffer seg. 1	1...8	time quantas
phase buffer seg. 2	1...8	time quantas



CAN transfer rates in relation to the bus length

$$T_d = T_{TT\text{-delay}} + T_{\text{line delay}}$$

$$T_{TT\text{-delay}} \sim 100 \text{ ns}$$

(driver, transceiver, comparator logic, etc.)

$$T_{\text{line delay}} \sim 0,2 \text{ m / ns twisted pair}$$

Bitrate (kBits/s)	max. network extension (m)
1000	40
500	112
300	200
200	310
100	640
50	1300



CAN payload

payload # of bytes	Std. frame kbits/sec	extended frame kbits/sec
0	--	--
1	71,1	61,1
2	144,1	122,1
3	216,2	183,2
4	288,3	244,3
5	360,4	305,3
6	432,4	366,4
7	504,5	427,5
8	576,6	488,5



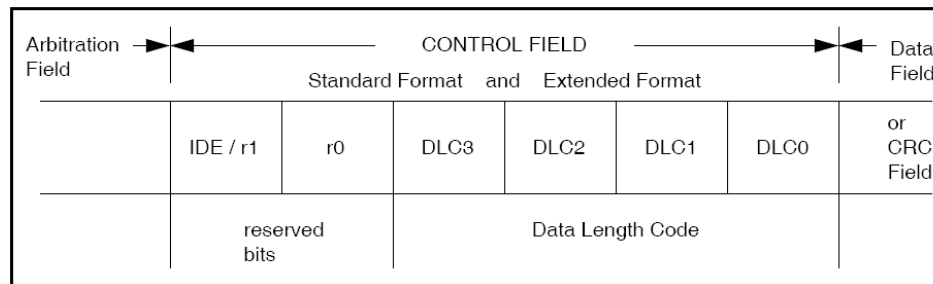
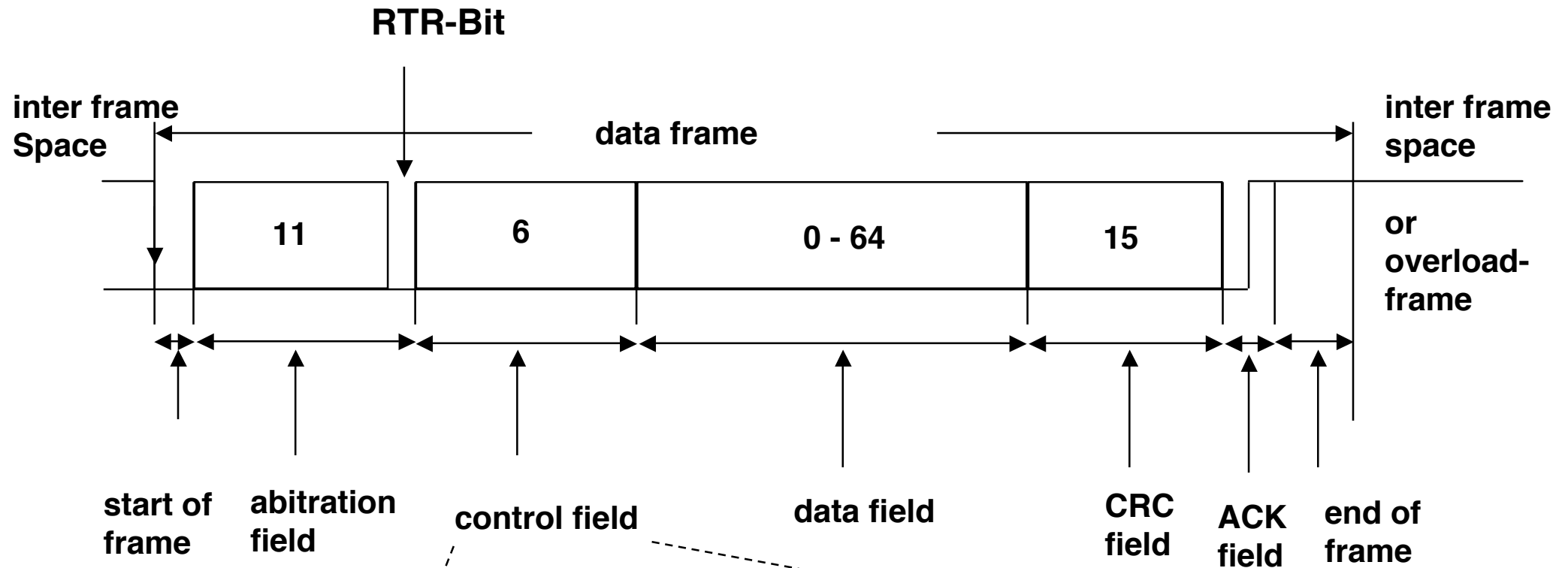
The CAN MAC and Logical Link Control (LLC) levels

Frame types and formats:

- **Data Frame** normal data transmission initiated by the sender
- **Remote Frame** participant requests frame which is sent with the identical frame ID from some other participant.
- **Error Frame** participant signals an error which it has detected
- **Overload Frame** used for flow control. Results in a delayed sending of the subsequent frame.

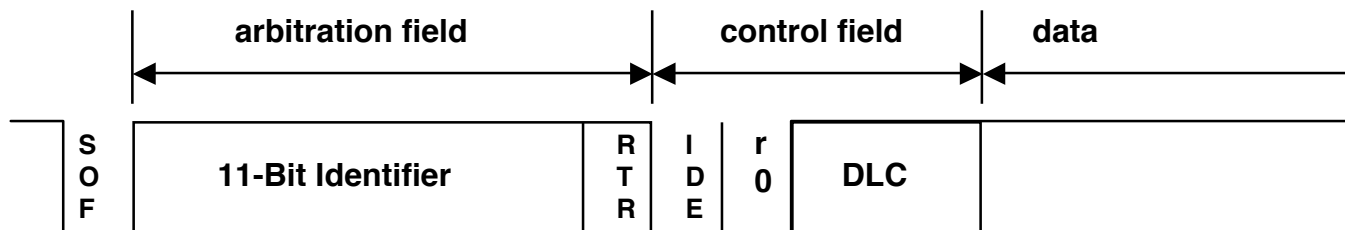


CAN Standard Data Frame

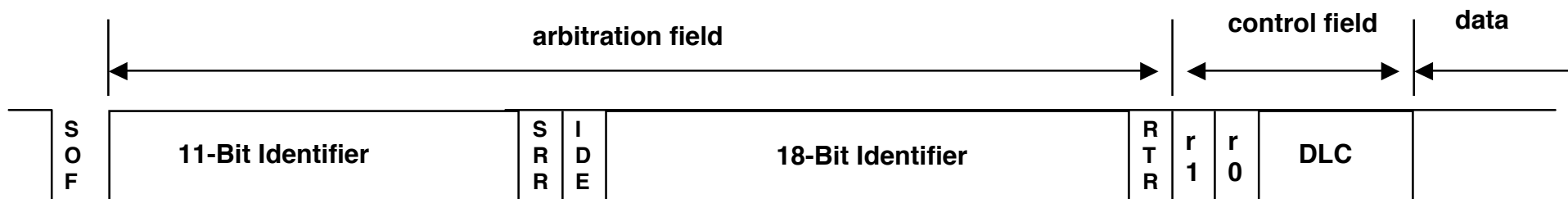


Compatibility between standard and extended frames

Standard Format SF (compatible to CAN Spezifikation 1.2)



Extended Format EF (CAN Spezifikation 2.0)

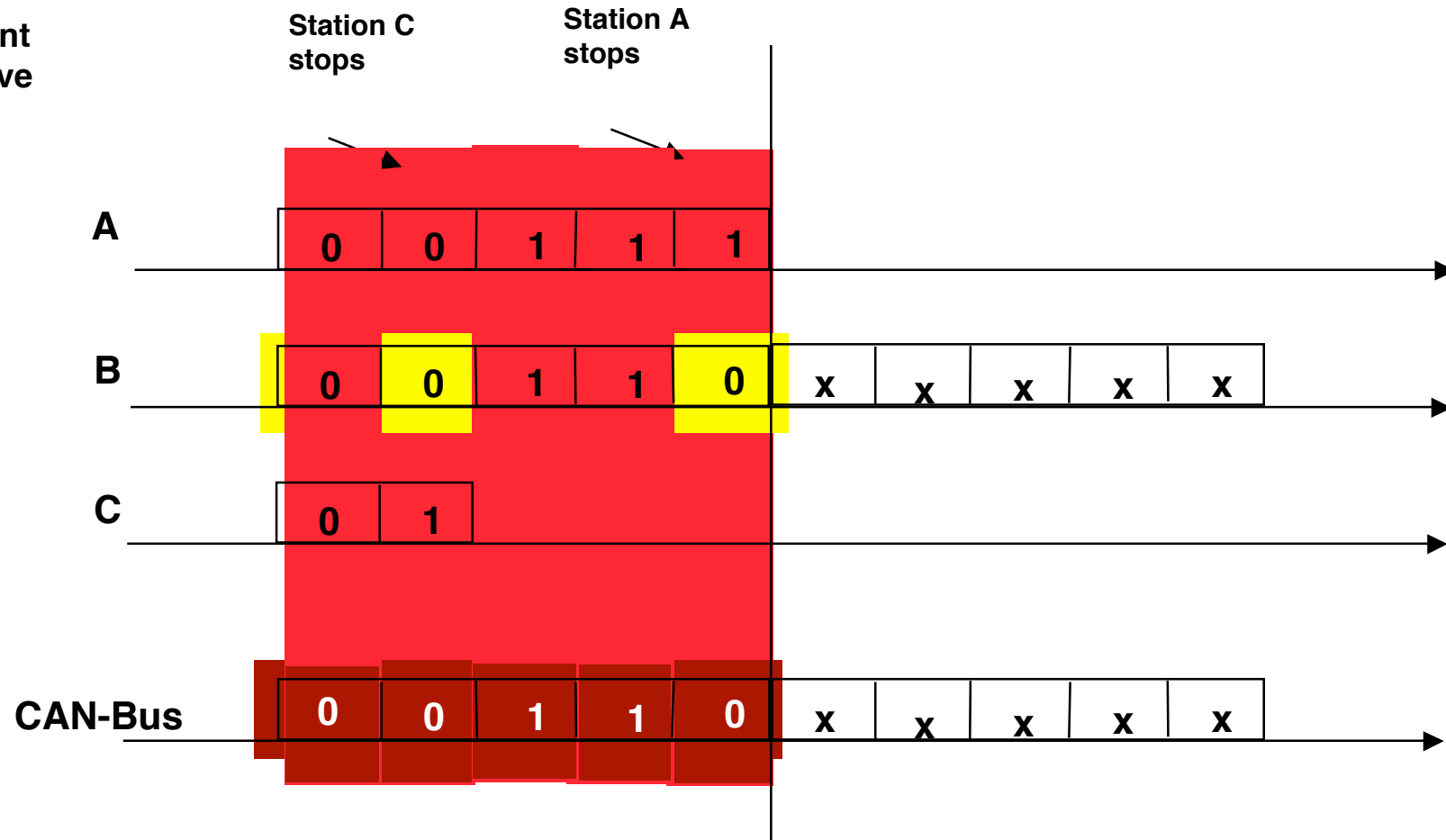


- RTR:** Remote Transmissin Request. In Data Frame: RTR = dominant. In Remote Frame: RTR = recessive. In the EF, the SSR-Bit has the funktion or the RTR-Bit
- IDE:** Identifier Extension. In the SF this is part of the control field, has a dominant value but is not interpreted. In the EF it is part of the addressing field, has a recessive value and causes the format to be recognized as EF.
- SRR:** Subsitute Remote Request. Recessive, replaces RTR in the EF for compatibility reasons.
- DLC:** Data Length Control. 0-8 Byte.
- r0, r1:** reserved



Arbitration on a CAN-Bus

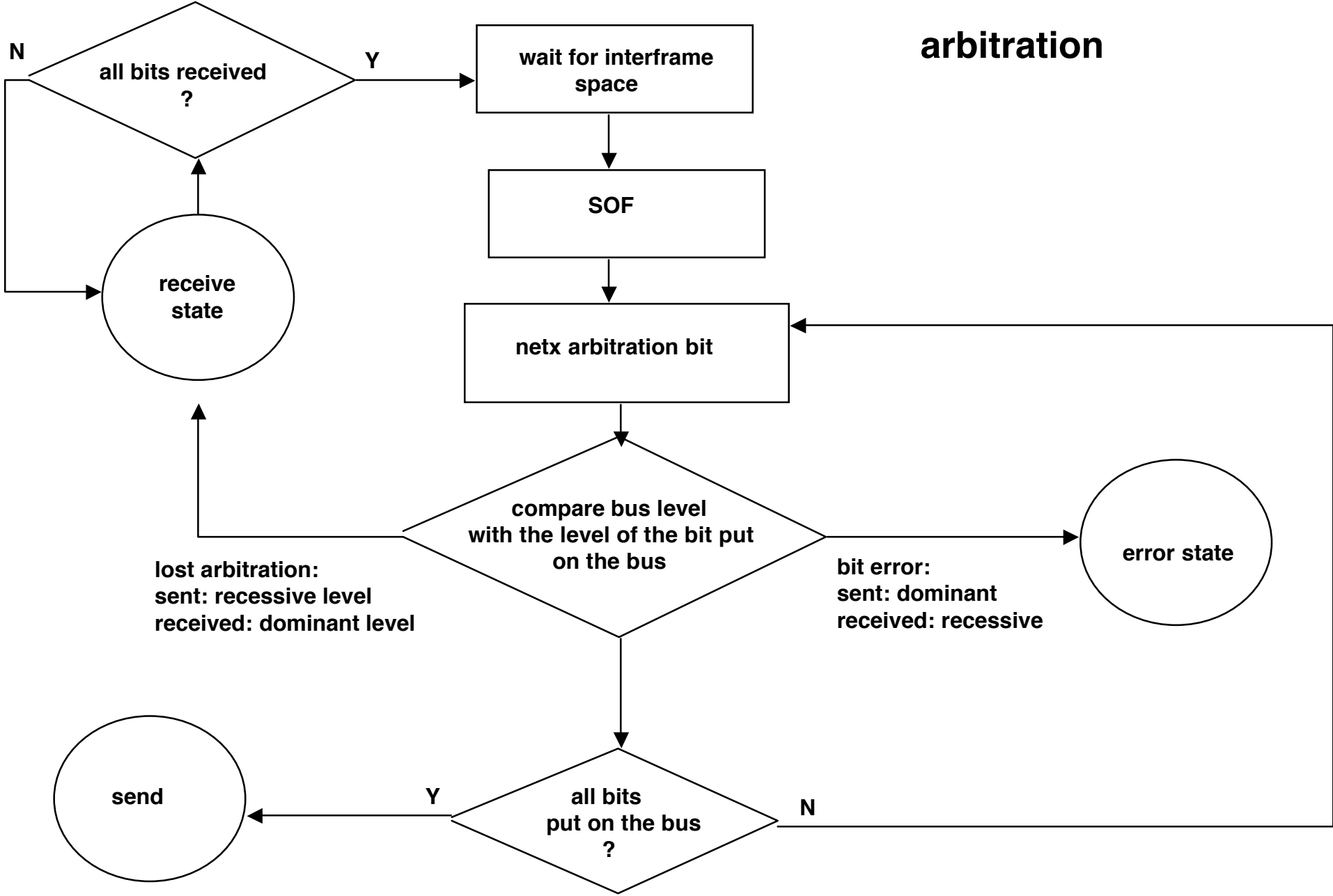
0 = dominant
1 = recessive



CAN enforces a global priority-based message scheduling

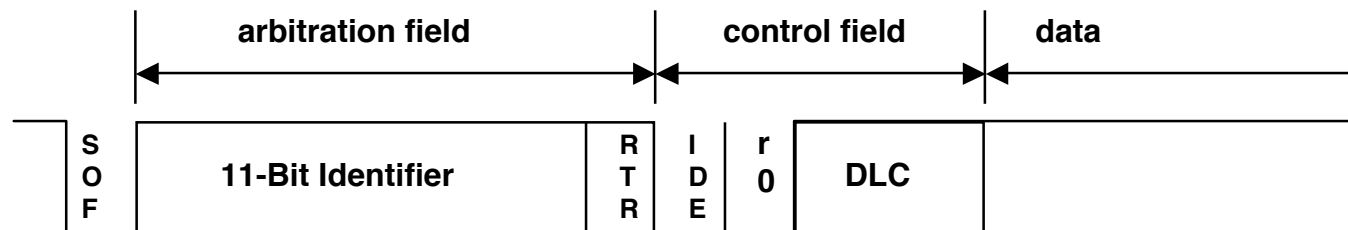


arbitration

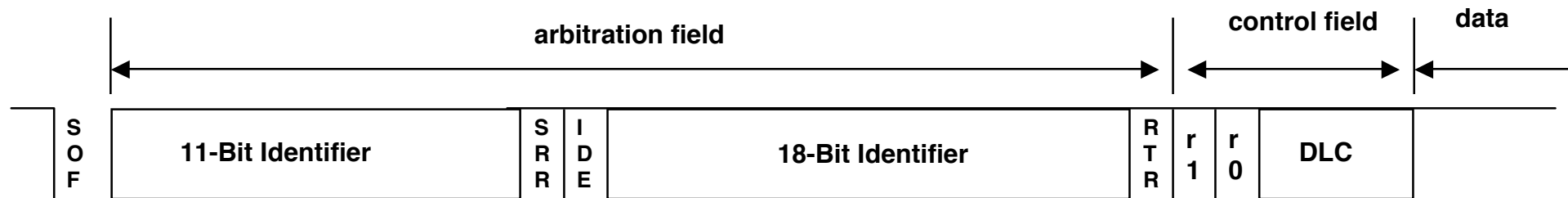


CAN Data Frame

Standard Format SF (compatible to CAN Spezifikation 1.2)



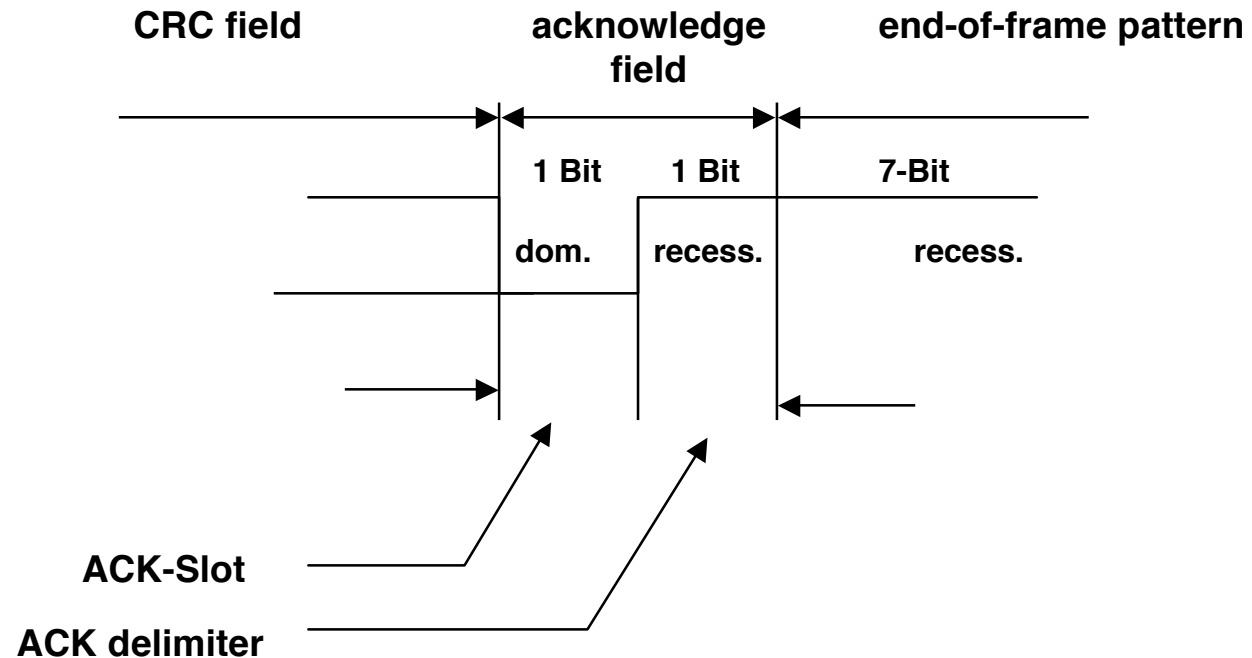
Extended Format EF (CAN Spezifikation 2.0)



- RTR:** Remote Transmissin Request. In Data Frame: RTR = dominant. In Remote Frame: RTR = recessive. In the EF, the SSR-Bit has the funktion or the RTR-Bit
- IDE:** Identifier Extension. In the SF this is part of the control field, has a dominant value but is not interpreted. In the EF it is part of the addressing field, ha s a recessive value and causes the format to be recognized as EF.
- SRR:** Subsitute Remote Request. Recessive, replaces RTR in the EF for compatibility reasons.
- DLC:** Data Length Control. 0-8 Byte.
- r0, r1:** reserved



Anonymous acknowledgement of a CAN message



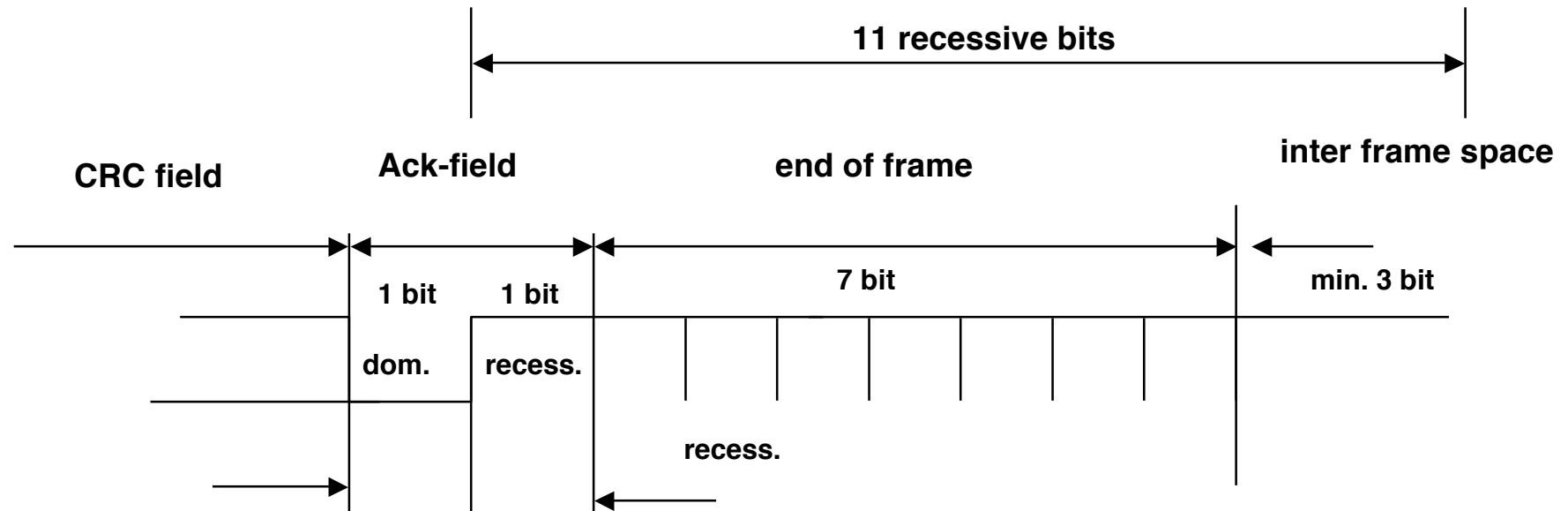
positive anonymous acknowledgement (Broadcast !)

receivers that correctly received a message(a matching CRC sequence) report this in the ack-slot by superscibing the recessive bit of the sender by a dominat bit. The sender switches to a recessive level.

- ➡ Message is acknowledged by a single correct reception on a correct node.
- ➡ Systemwide data consistency requires additional signalling of local faults.



Termination sequence of a frame



Goals:

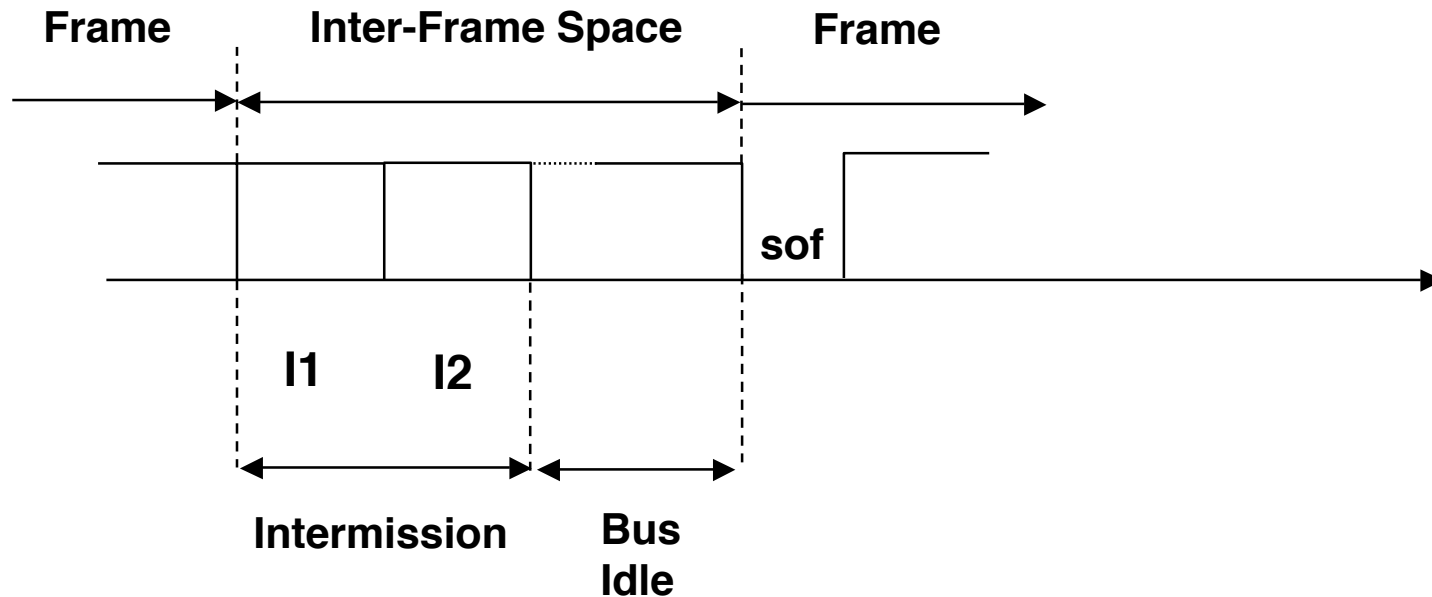
1. Detecting AND signalling the error within the actual frame in which it occurred
2. Identifying the node which may have caused the error.
3. Creating a systemwide view on the reception state of the message.

Approach: End of frame pattern consisting of 7 recessive bits.

1. Any error detection is signalled by putting a dominant bit on the bus.
2. An out-of-sync node, not being aware of the EOF sequence will signal an error at position "6".



Interframe Space



Intermission: no data- or remote Frame may be started

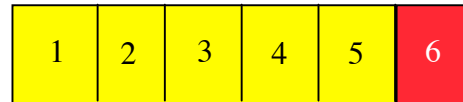
Intermission 1: active overload Frame may be started

Intermission 2: re-active overload frame (after detecting a dominant bit in I1)

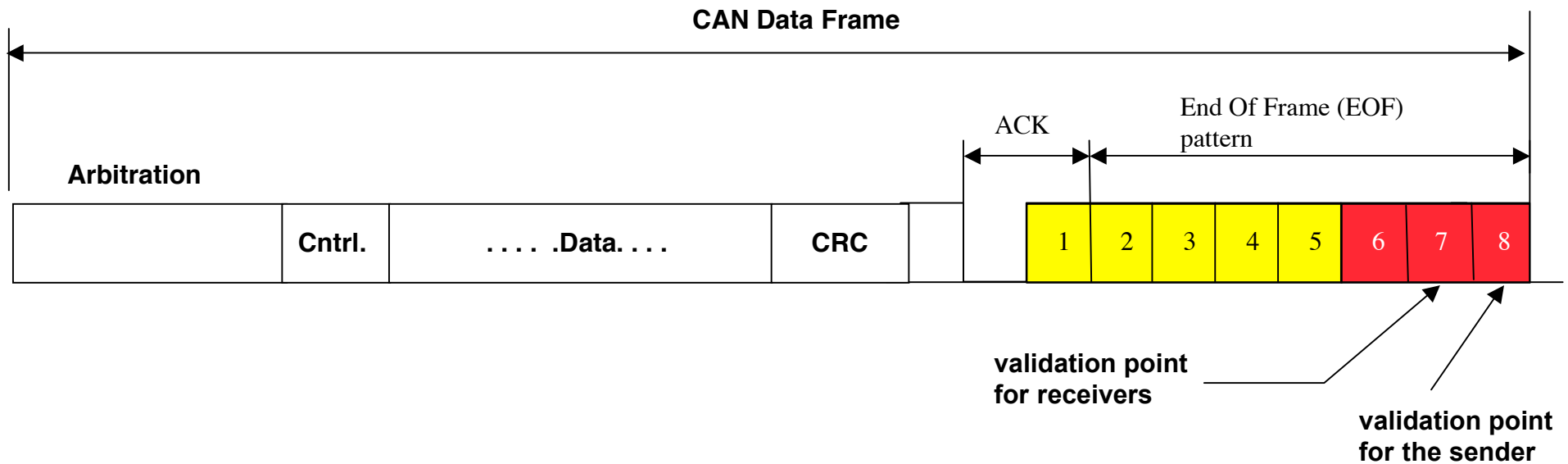


Error Detection and Error Signalling in CAN

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



Error detection

1.) **Monitoring: Sender compares the bit sent with the bit actually on the bus.**

Type of faults: local sender faults

Error detection: sender based

2.) **Cyclic Redundancy Check:**

Type of faults: 5 arbitrarily distributed faults in the code word,
burst error max. length 15.

Error detection: receiver based

3.) **Bitstuffing:**

Type of faults: transient faults, stuck-at-faults in the sender

Error detection: receiver based

4.) **Format control:**

Type of faults: the specified sequence of fields is violated.

Error detection: receiver based

5.) **Acknowledgment:**

Type of faults: no acknowledge

Error detection: sender based, sender assumes local fault.



Risk of undetected errors

Bit monitoring: An error will not be detected if

- the sender is correct and monitoring doesn't detect an error
- all other nodes receive the same bit pattern which is different from that of the sender and contains a non-detectable error.

Bit-stuffing: double errors within 6 bits will not be detected

CRC: difference between frame sent and received is a multiple of the generator polynome.

Frame errors: the frame is shortened or additional bits are added. At the same time a correct end-of-frame sequence is generated.

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

Scenario:

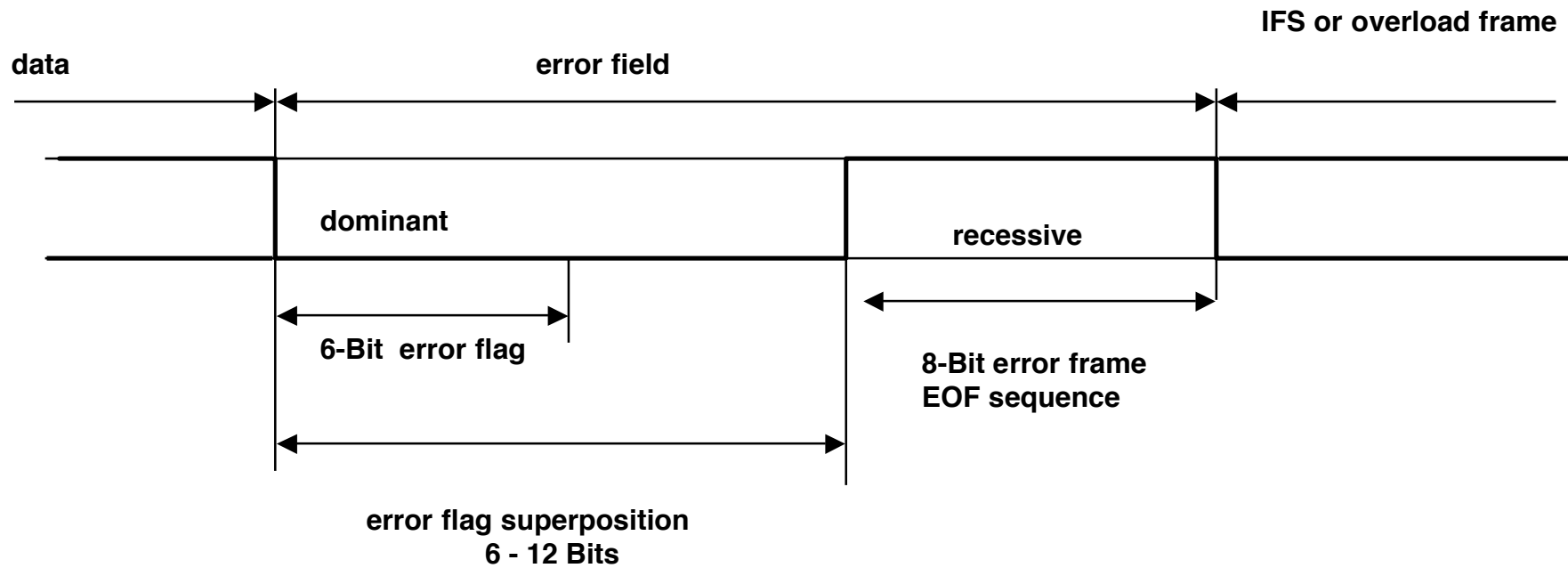
nodes: 10, Bit error rate: $2 \cdot 10^{-2}$, message error rate: 10^{-3}

risk of undetected errors: $4,7 \cdot 10^{-14}$

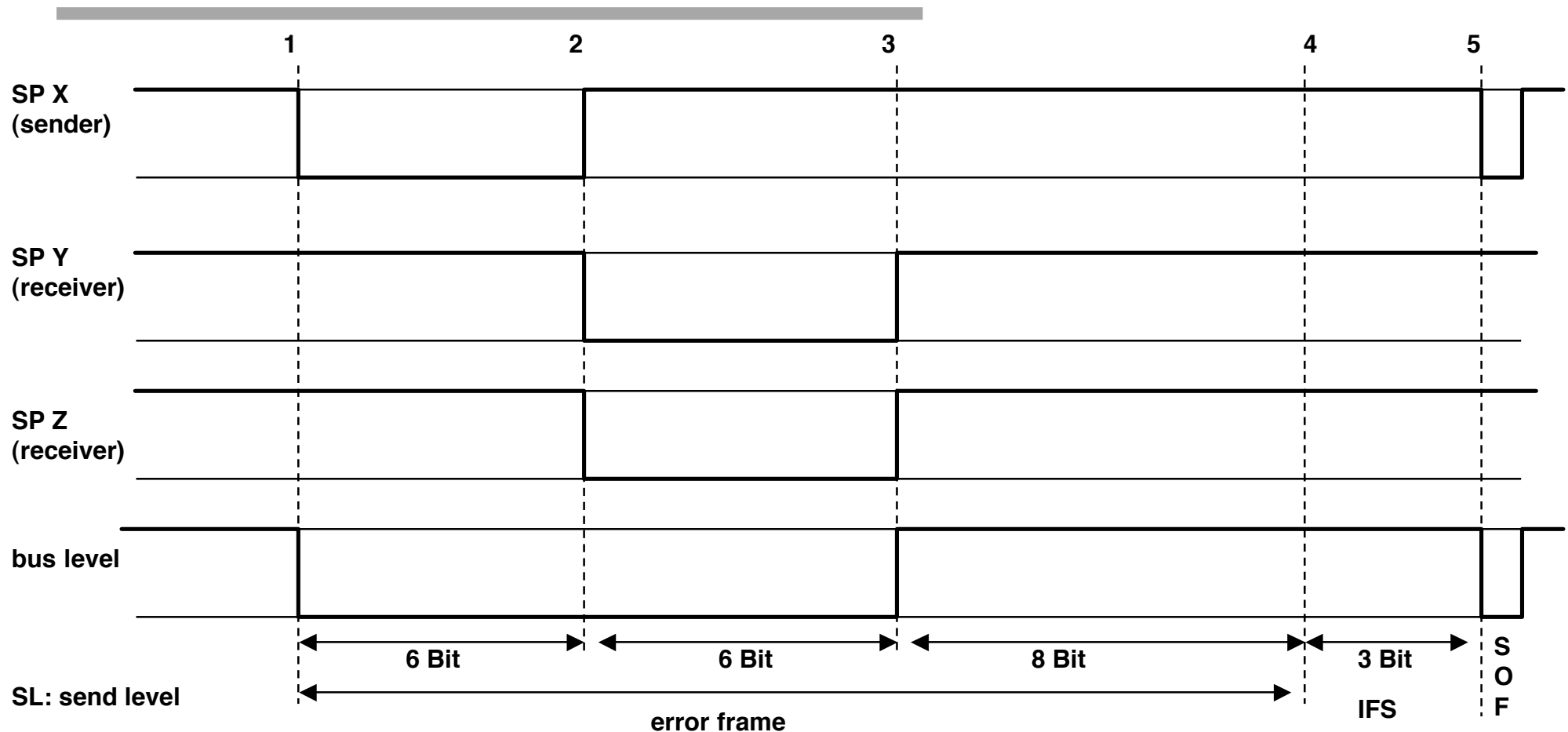
When the number of nodes increase, the probability of undetected errors decreases.



CAN error frame



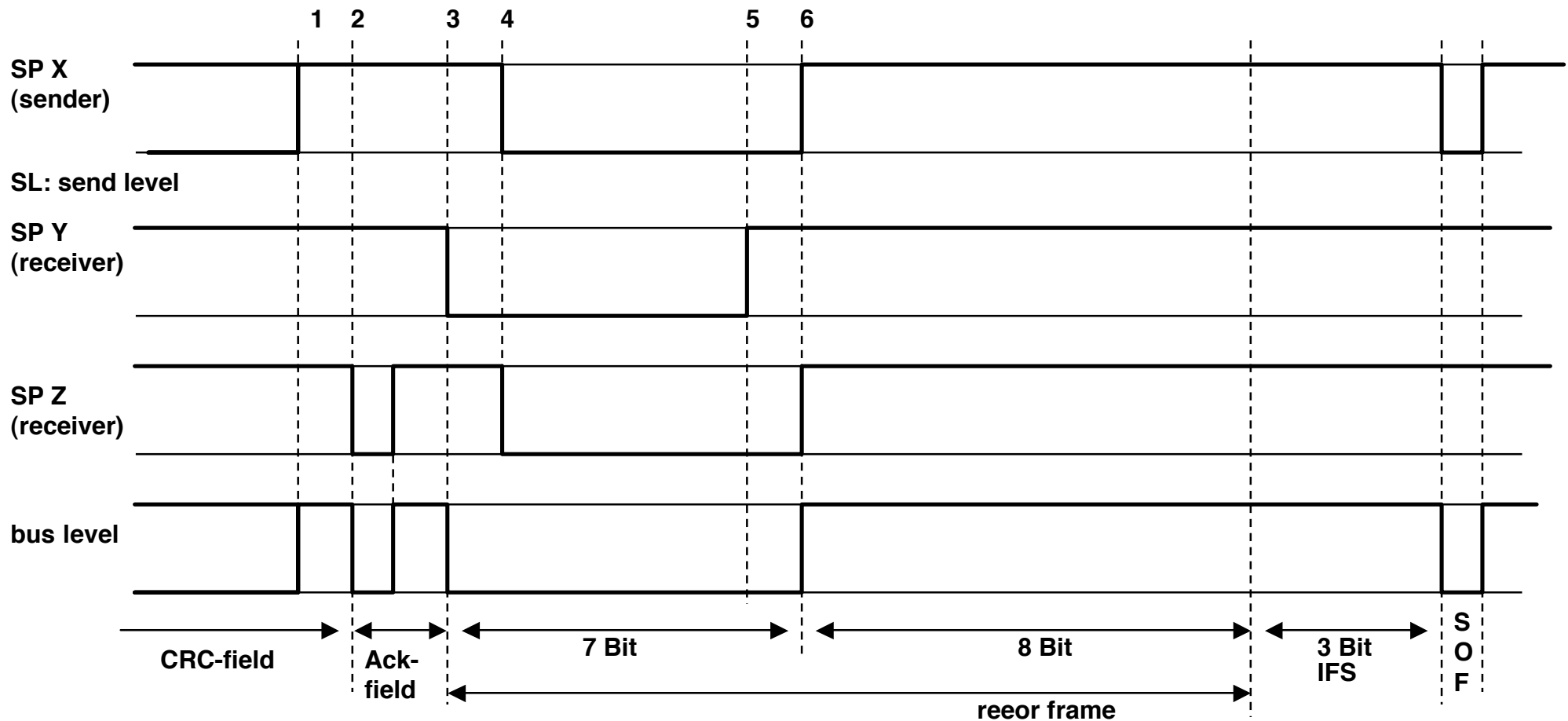
Error frame resulting from a sender fault



time to re-transmit a faulty message frame: min. error recovery time: 23 bit times



Error frame resulting from a receiver fault



time to re-transmit: min. error recovery time: 20 bit times



Enforcing fault confinement and a “Fail Silent” behaviour

Problem: Faulty component may block the entire message transfer on the CAN-Bus.

Assumption:

1. A faulty node detects the error first.
2. frequently being the first which detects an error --> local fault in the node

approach: error counter for receive and transmit errors. If error was first detected by the node, the counter is increased by 8-9.

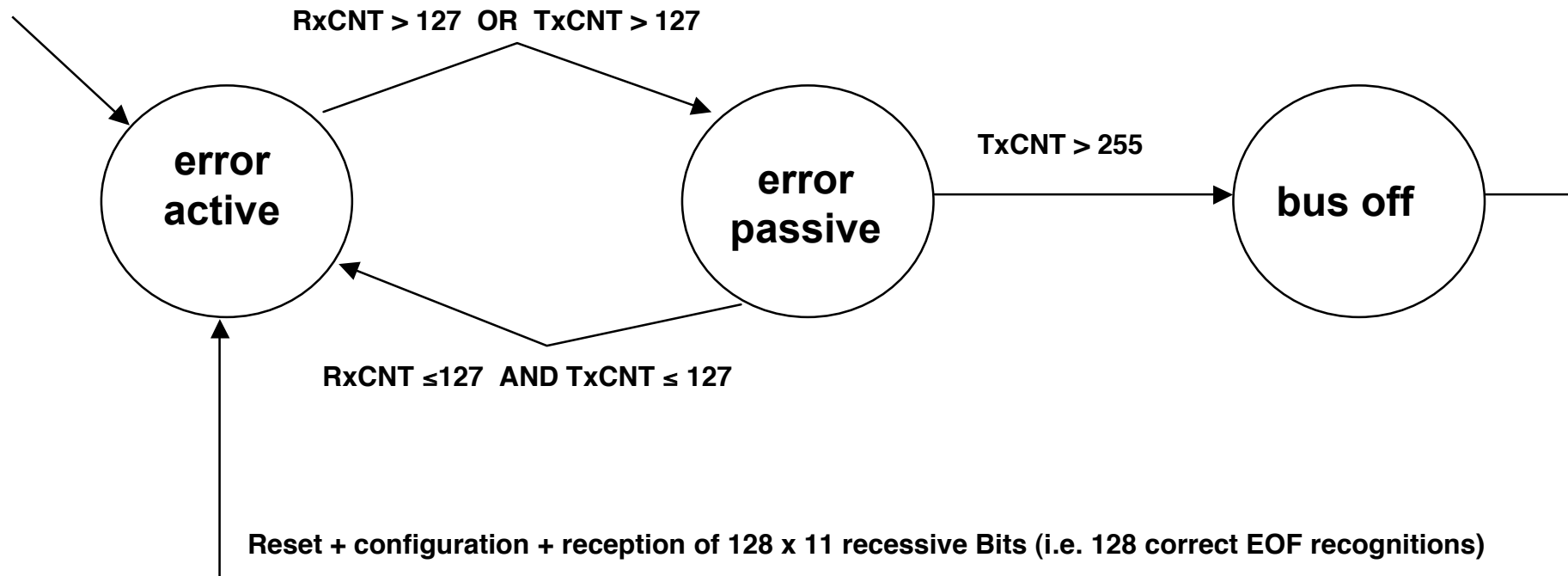


Enforcing fault confinement and a “Fail Silent” behaviour

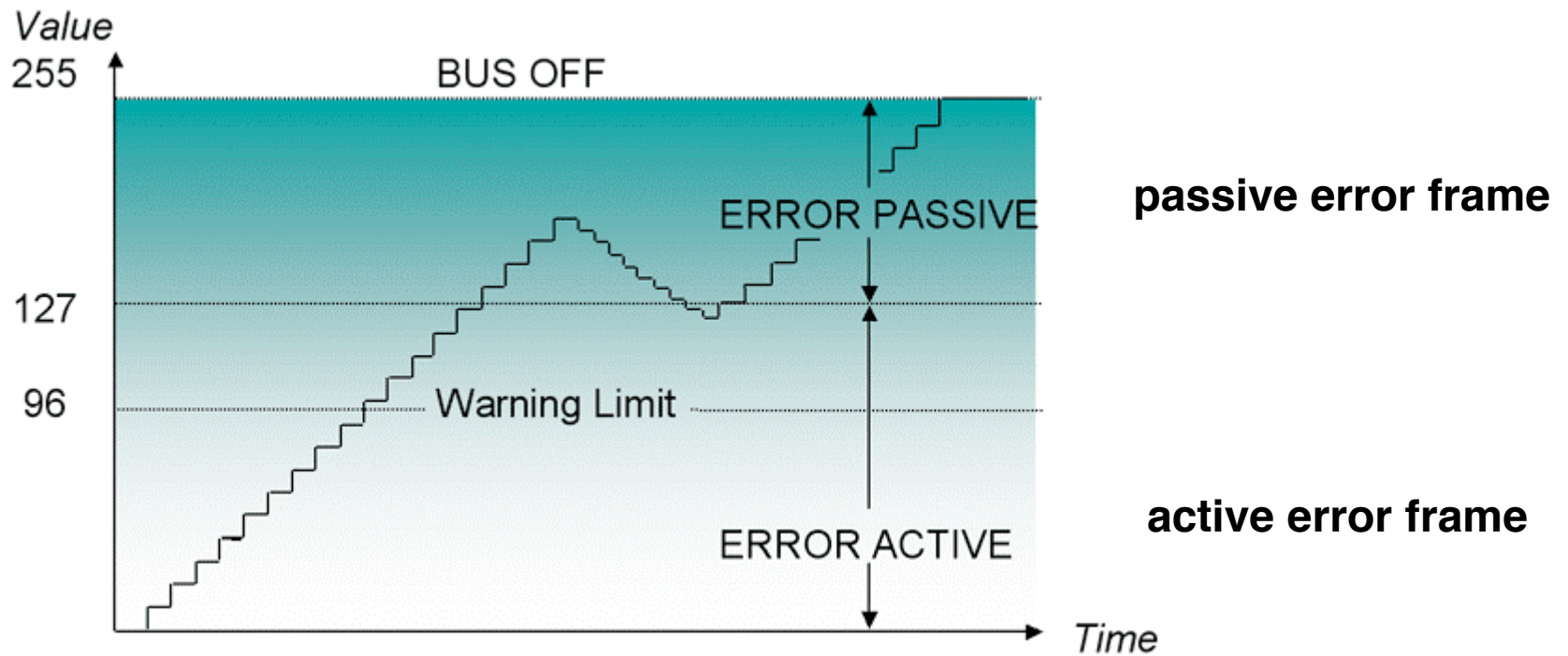
States of a CAN node:

- error active
- error passive
- bus off

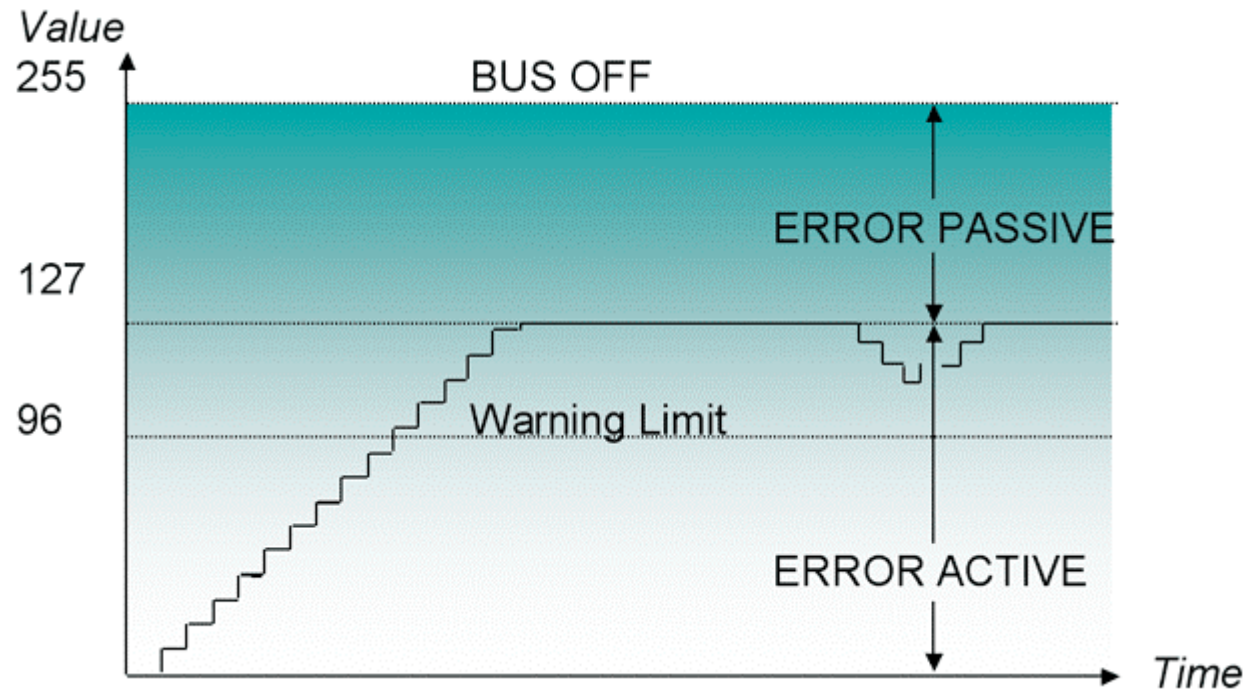
RxCNT: Value of the receive counter
TxCNT: Value of the transmit counter



CAN bus Error Handling - Transmit Error Counter

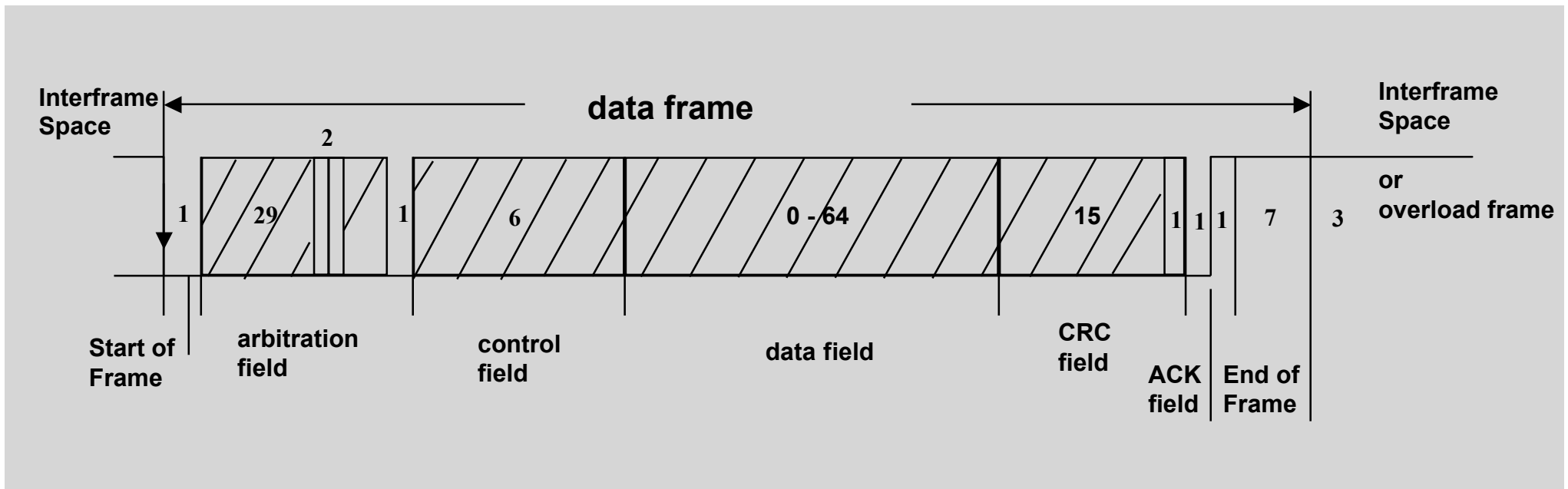


CAN bus Error Handling - Receive Error Counter



Analysis of CAN inaccessibility

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} (μ s)	
Bit Errors	155.0	← 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	
Transmitter Failure	2480.0	← worst case multiple
Receiver Failure	2325.0	

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



Predictability of various Networks*

Worst Case Times of Inaccessibility*		t_{inacc} (ms)	
ISO 8002/4 Token Bus (5 Mbps)		139.99	Token-based Protocols
ISO 8002/5 Token Ring (4 Mbps)		28278.30	
ISO 9314 FDDI (100 Mbps)		9457.33	
Profibus (500 kbps)		74.80	
CSMA/CD	unbounded		CSMA Protocols
CSMA/CA	stochastic		
CAN-Bus (1Mbps)		2.48	

The worst-case-delay of the Timed-Token-Protocol** is $2 \cdot TTRT$ (Target Token Rotating Time)

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

** J.N. Ulm: "A Timed Token Ring Local Area Network and its Performance Characteristics"

R.M. Grow: "A Timed Token Protocol for local Area Networks"



CAN-Bus Properties (summary)

- ➔ **Event-triggered communication with low latency**
- ➔ **Priority-based arbitration with collision resolution for guaranteed throughput**
- ➔ **error handling:**
 - anonymous positive acknowledge**
 - negative ack. in case of an error (systemwide messaging)**
 - identification of faulty nodes**
 - immediate synchronisation and retratnsmission**
- ➔ **content-based addressing with a high flexibilitx (system elasticity)**



High level issues

Routing: How does a message reach a receiver ?

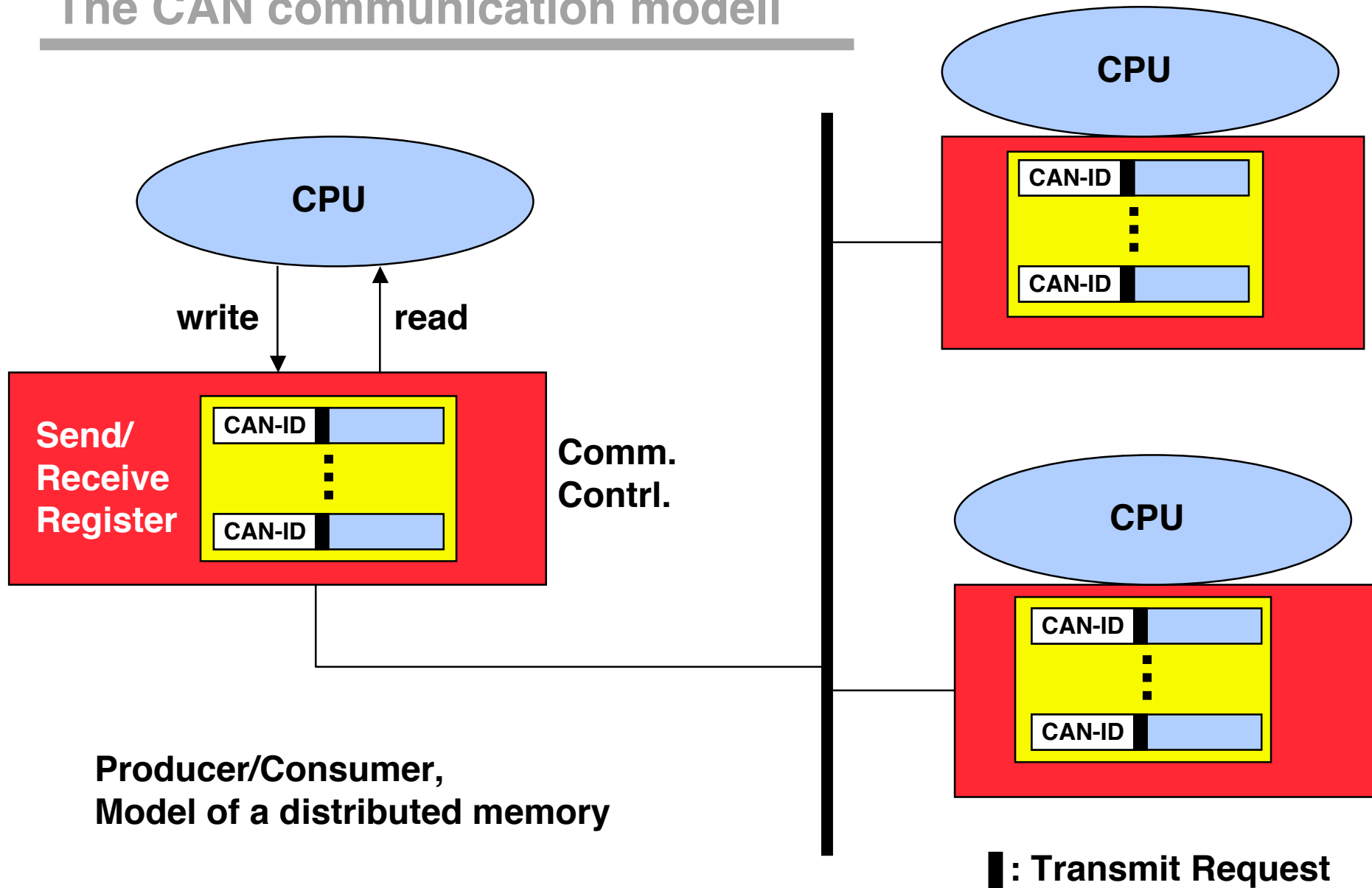
CAN: Broadcast, message subjects

Filtering: How can the receiver only receive those messages selectively in which it is interested in ?

CAN: message filters



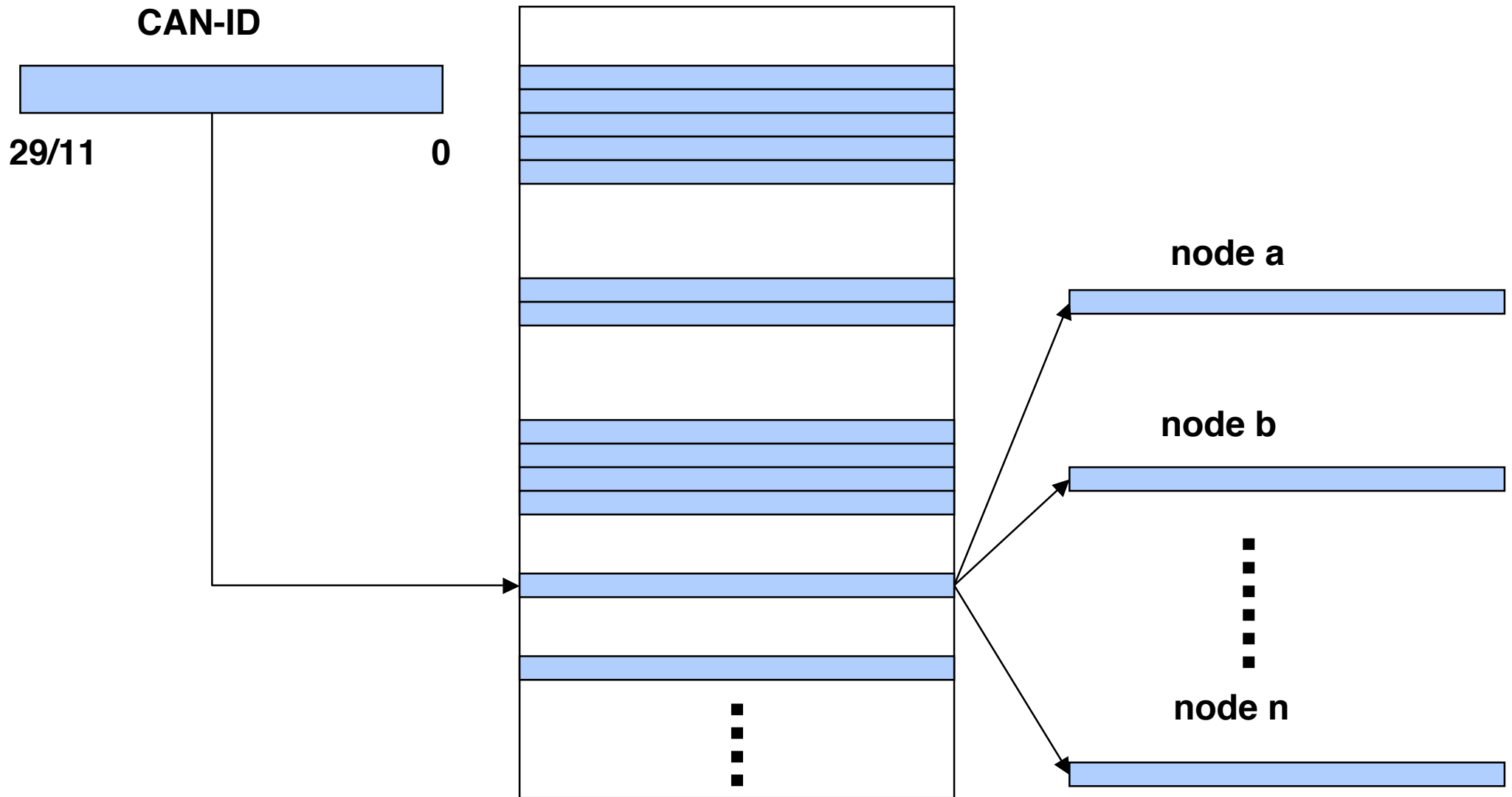
The CAN communication modell



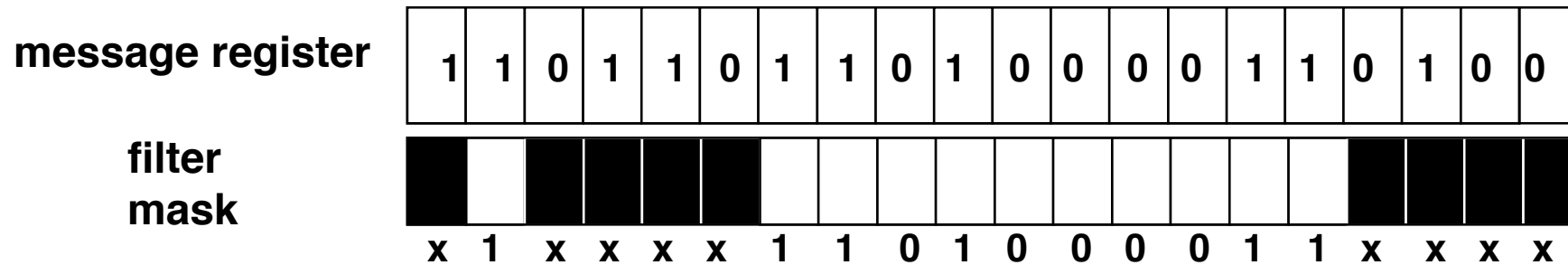
**Producer/Consumer,
Model of a distributed memory**



The CAN communication model



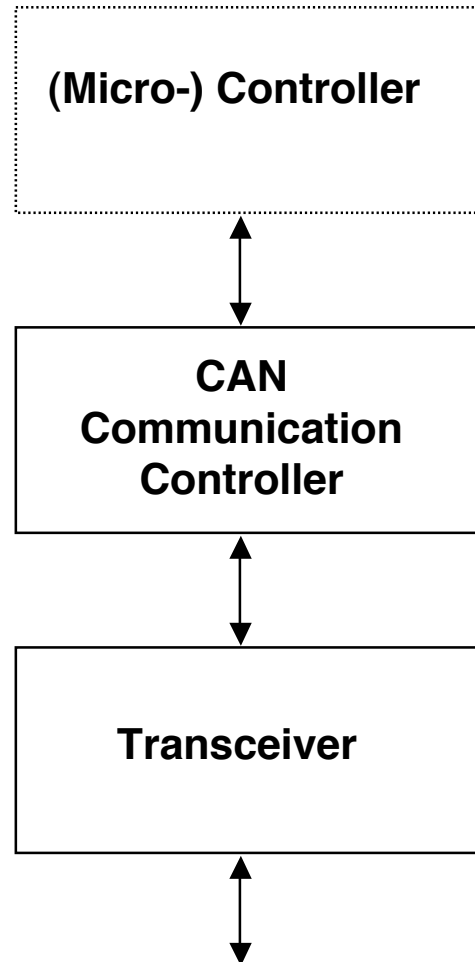
message filtering



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

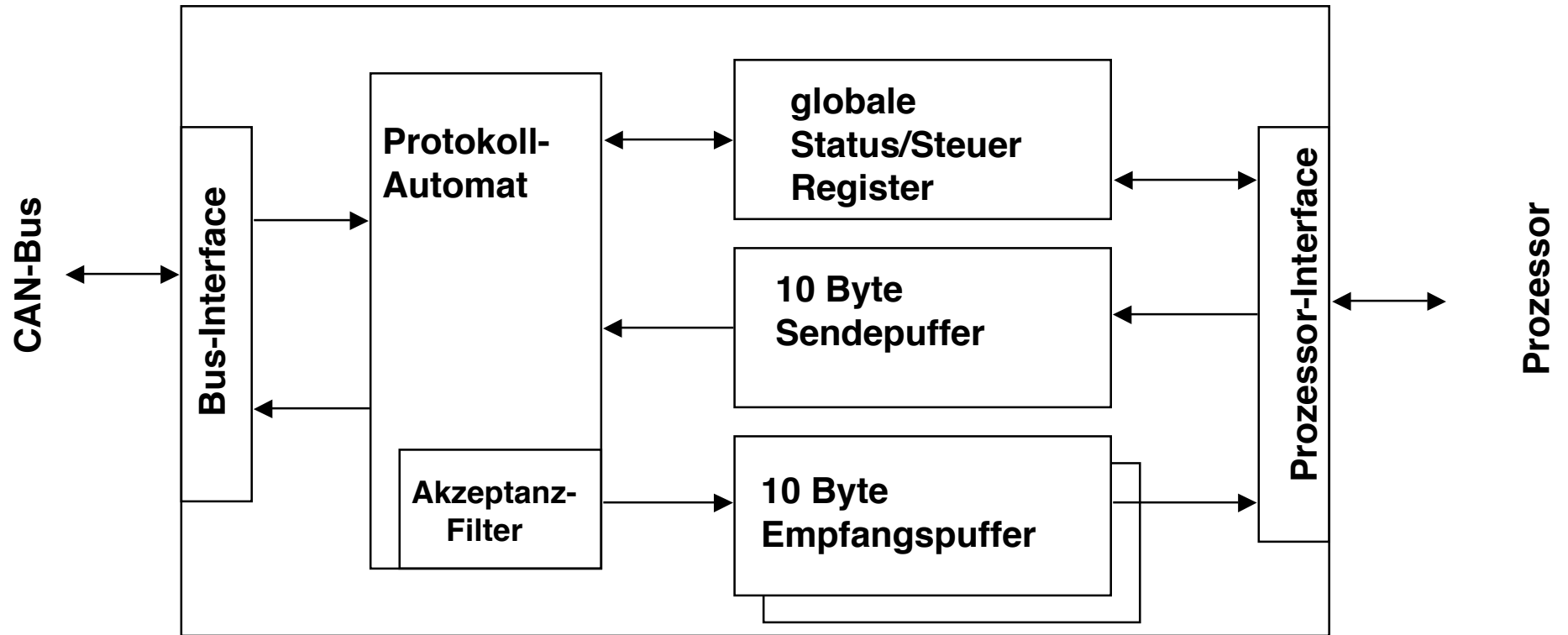


Aufgaben des CAN Controllers

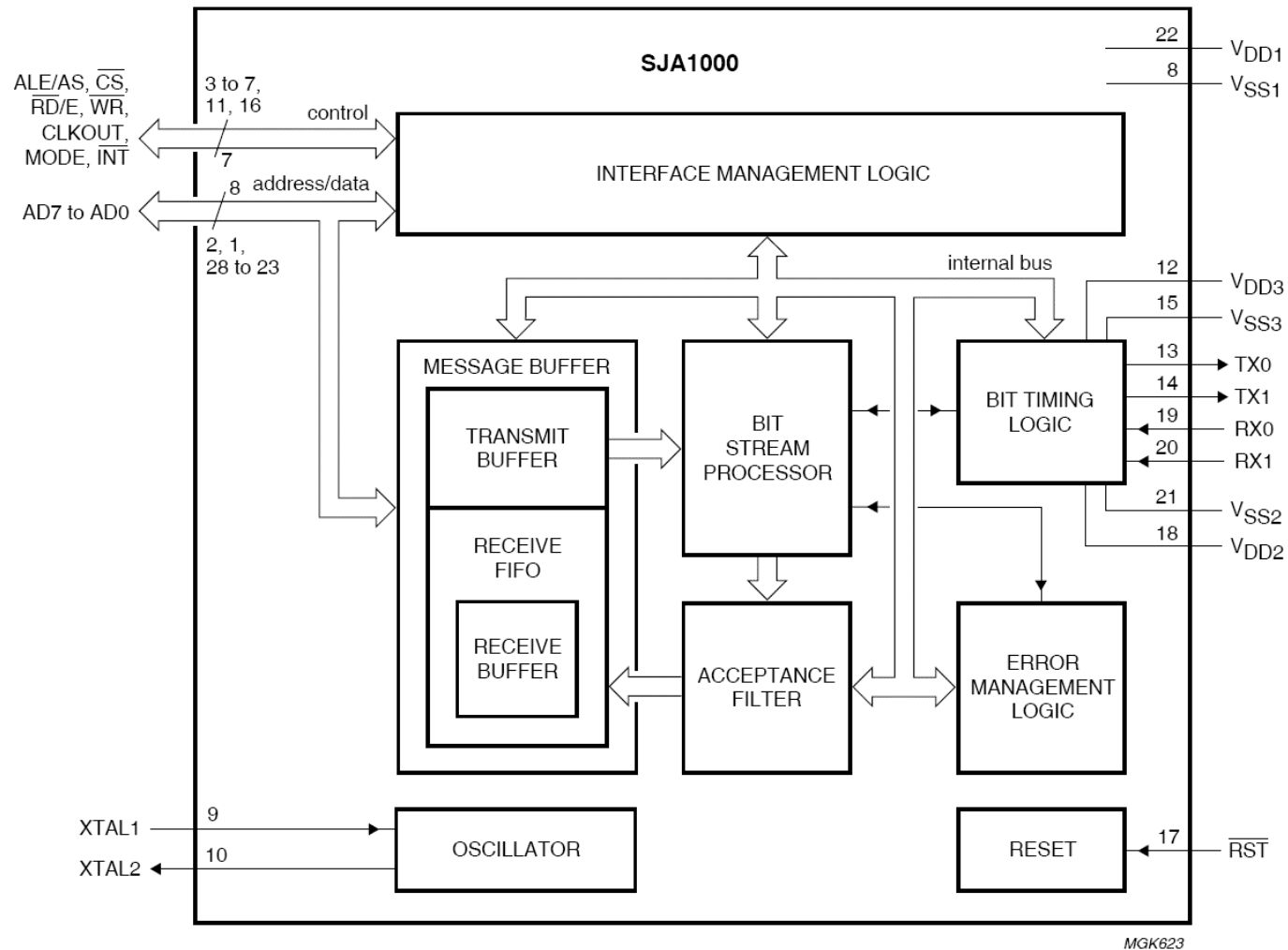
- **Busarbitrierung**
- **Serialisierung der zu sendenden Telegramme**
- **Assemblierung der empfangenen Telegramme**
- **Berechnung bzw. Überprüfung der Checksumme**
- **Fehlererkennung und Fehlersignalisierung**
- **Bildung der CAN-Nachrichtenformate**
- **Einfügen bzw. Entfernen der zusätzlichen Bits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



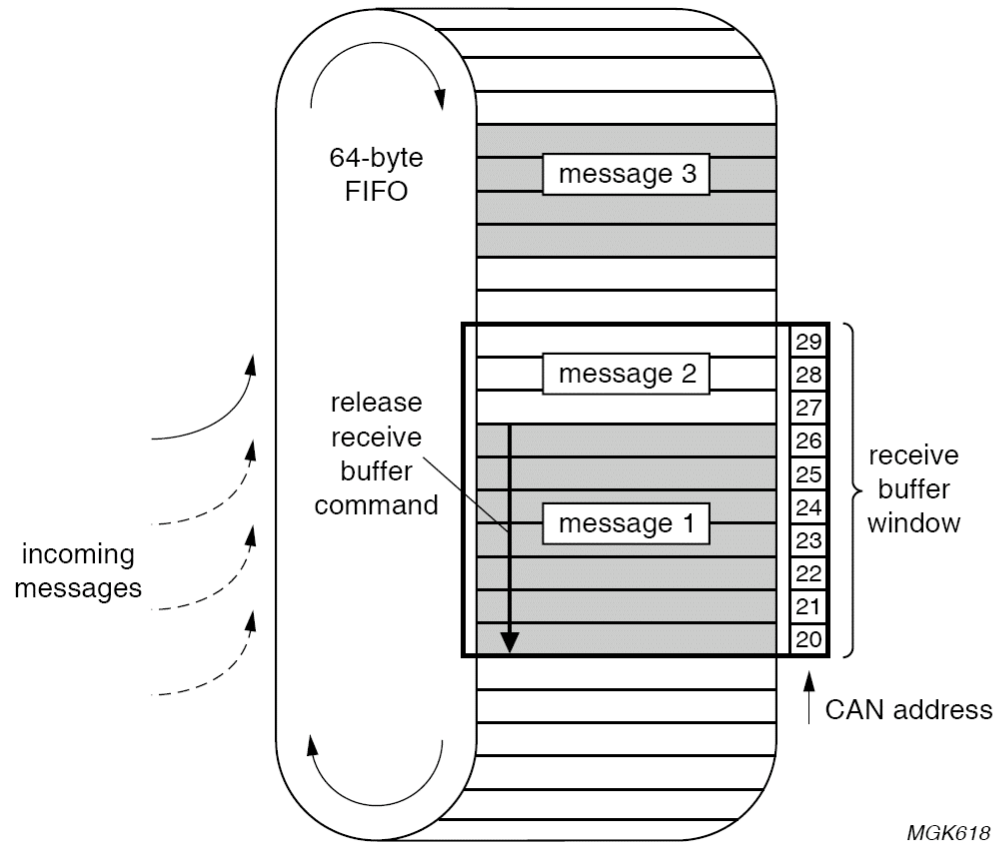
Basic CAN



SJA1000 (Philips)

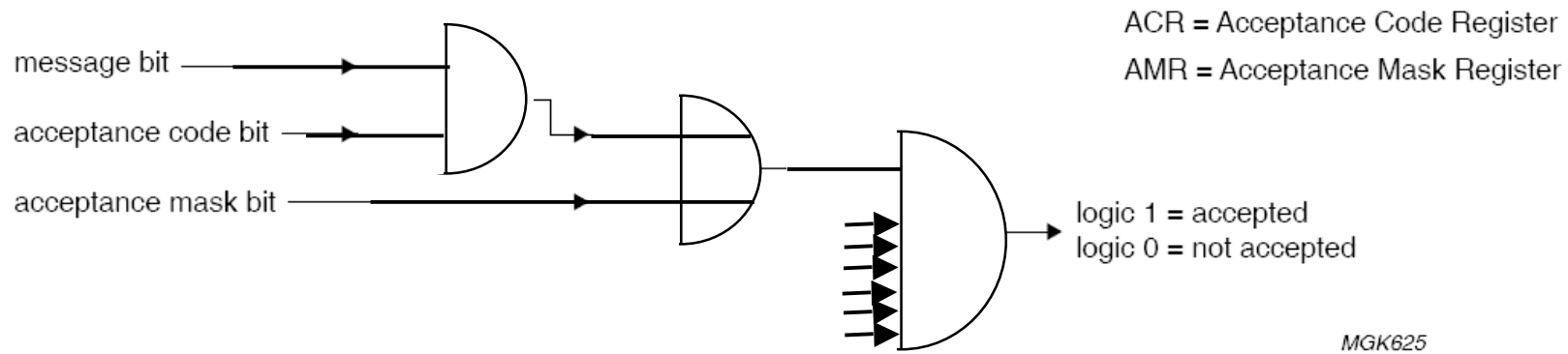
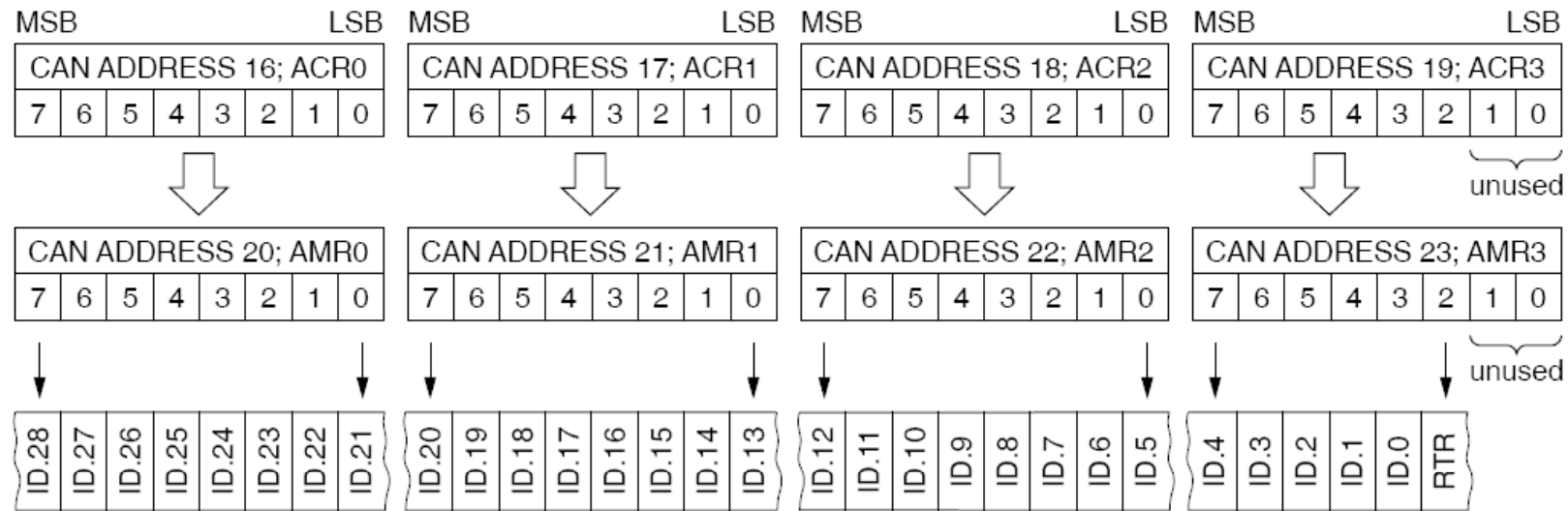


SJA1000 (Philips)



SJA1000 (Philips)

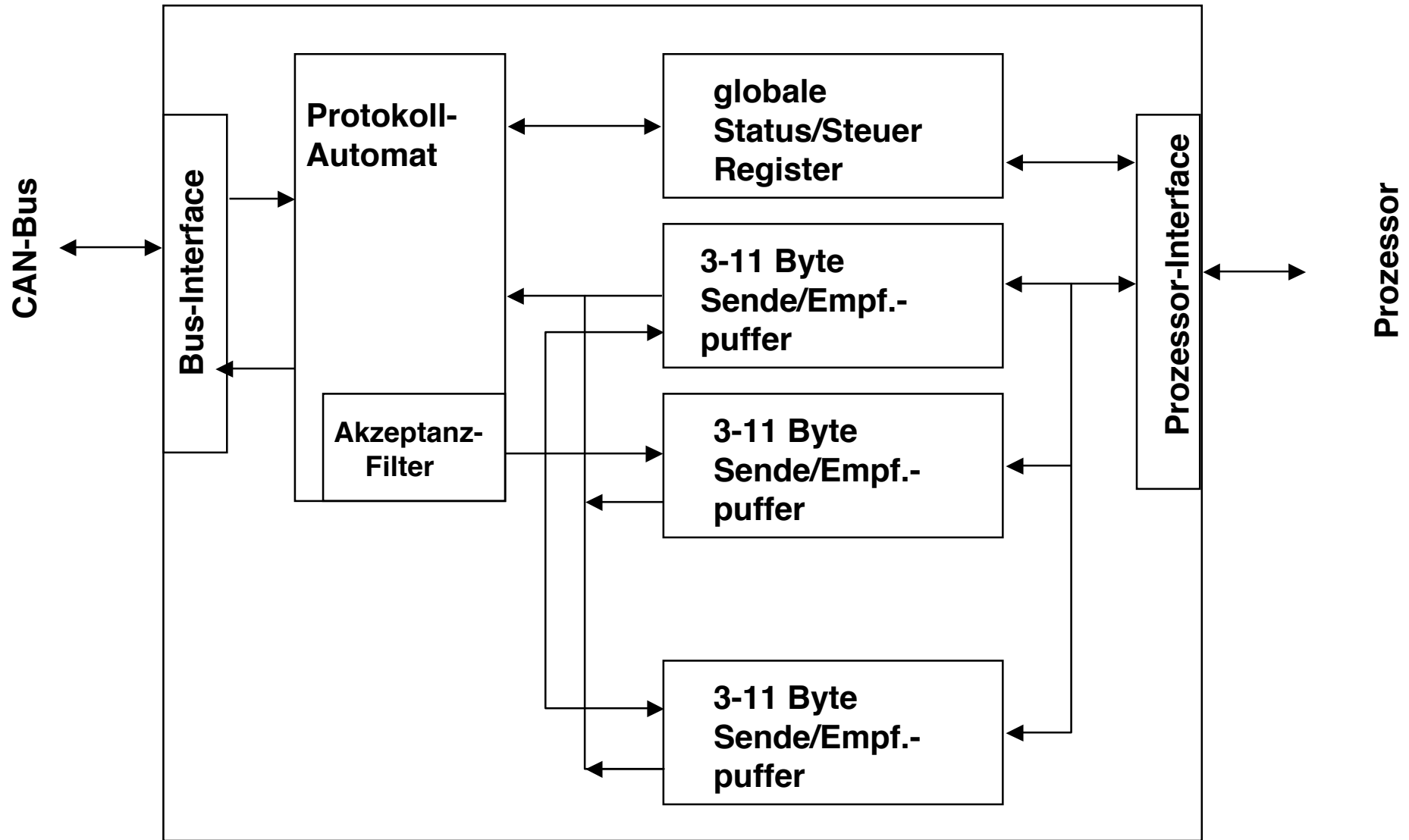
single mask option



ACR: defines the pattern of CAN message IDs which are accepted.
AMR: defines a mask of "don't care" positions.



Full CAN



What CAN can't

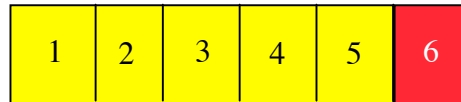
- **All-or-nothing property under all single (crash/omission) fault conditions**
- **Temporal guarantees for message transmissions**
- **Consistent order of messages**



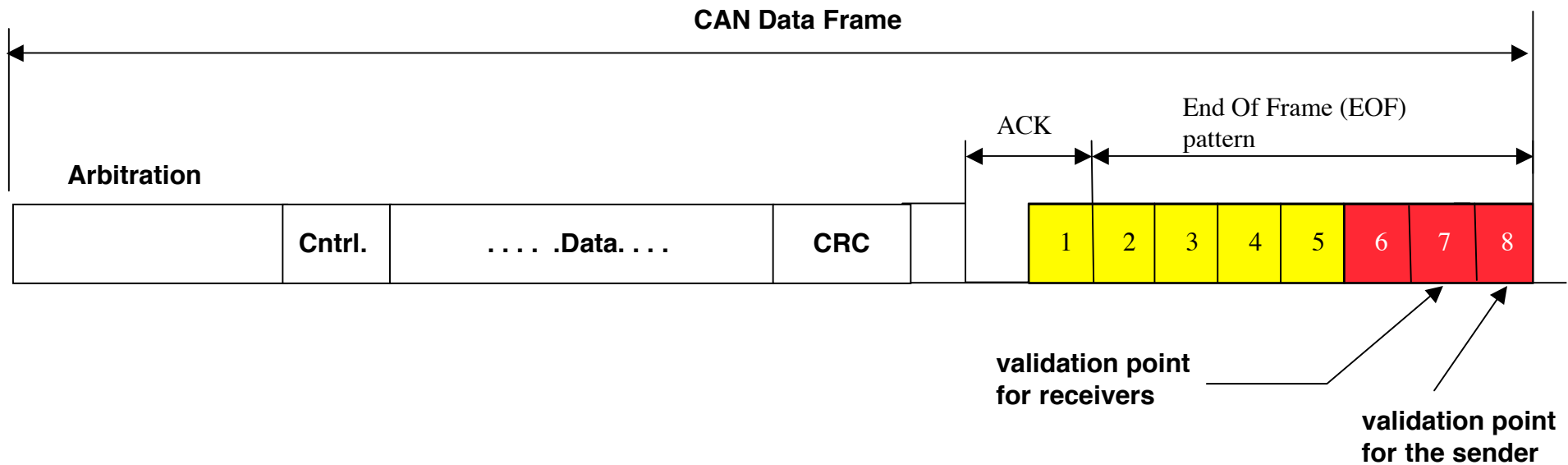
Error Detection and Error Signalling in CAN

The Case for Inconsistencies

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



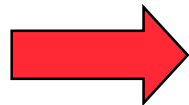
Bit-Stuffing enforces the following rule:
A sequence of 5 identical bit levels
is followed by a complementary bit level



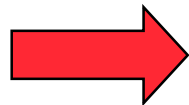
Konsequenzen aus dem Validierungsprotokoll

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

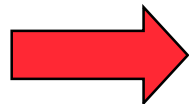
**J. Kaiser, Mohammad Ali Livani: “Achieving Fault-Tolerant Ordered Broadcasts in CAN”
*Proc. of the 3rd European Dependable Computing Conference, (EDCC-3), Prague, Sept. 1999***



Inkonsistente Nachrichten-Duplikate



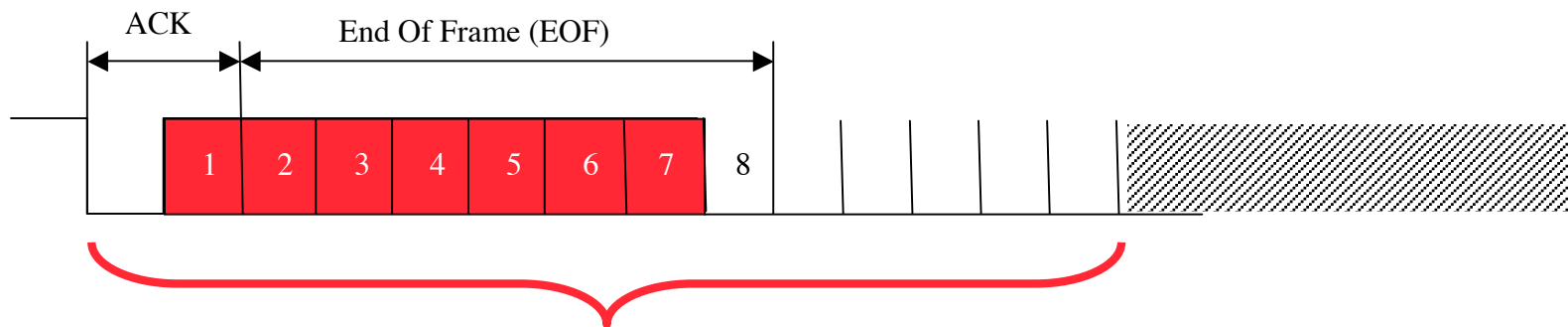
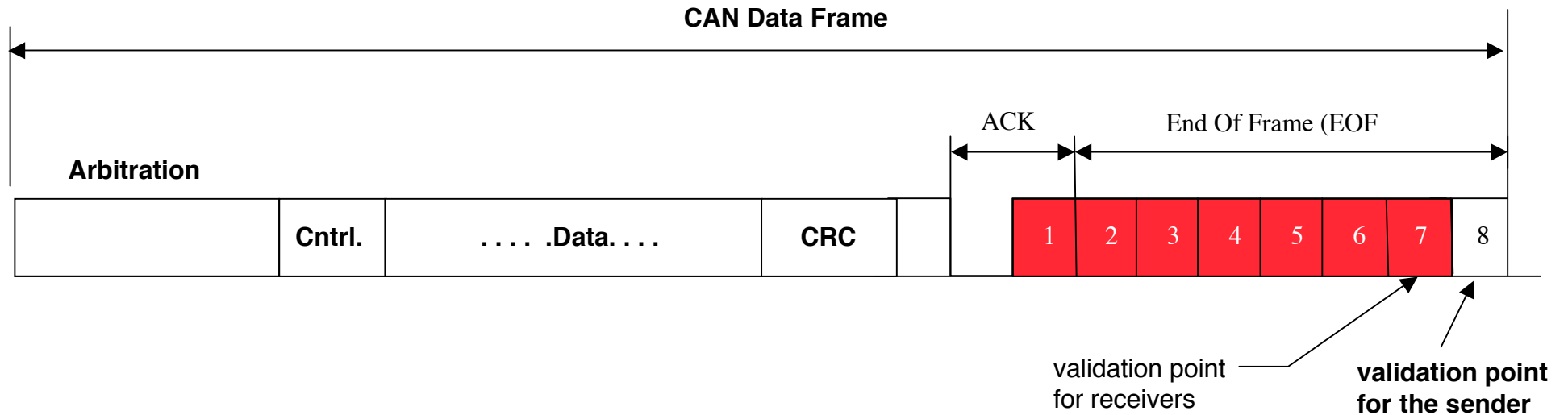
Inkonsistente Omission-Fehler



(Potentiell) Unbeschränkte zeitliche Verzögerungen



The Case for SHARE: Inconsistent Omissions



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



The Architecture of SHARE

