Karlsruher Institut für Technologie

Institut für Technik der Informationsverarbeitung ITIV

Prof. Dr.-Ing. Dr. h.c. Jürgen Becker *(becker@kit.edu)*

Dr.-Ing. Jens Becker *(jens.becker@kit.edu)*

Institut für Technik der Informationsverarbeitung (ITIV)

Communication Systems and Protocols

Session 7: Data transmission

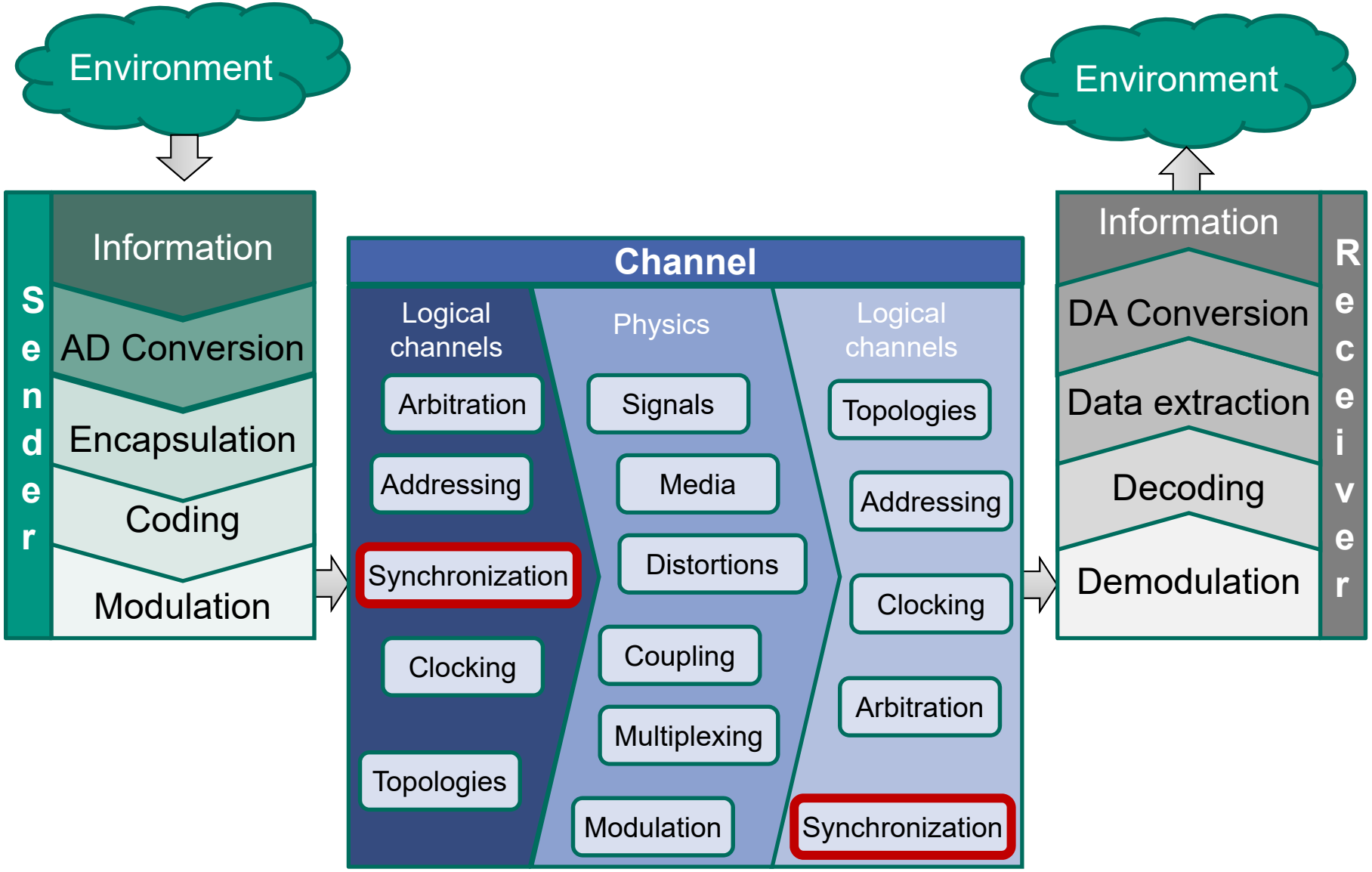
Clicker Session: Recapitulation

- https://arsnova.eu/mobile/#id/33969518

Recapitulation

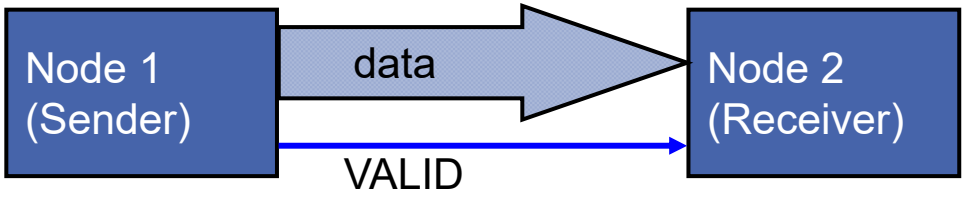
- Arbitration
- Specifications
  - Technical parameters
  - Protocol

Synchronization

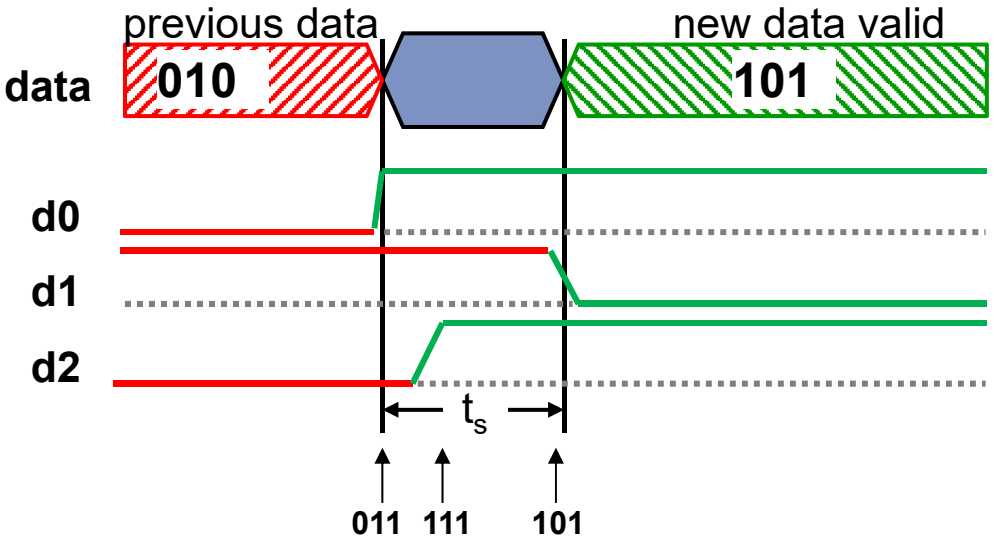


Data-Skew

- Sender outputs new information to data line
- Due to varying signal delays on different (parallel) signal paths the data on the bus may be invalid for a period of time t_s



- Example:
  Signal change 010 → 101



- Sender marks valid new data with a VALID signal

Different types of Synchronisation

- Low-level Synchronization (clocking)
  - Communication partners have to be able to separate individual bits
  - Common time base using
    - Clock line
    - Suitable line code
    - synchronized local clocks
    - …

- High-Level Synchronization (e.g. hand shaking)
  - Logical Synchronization of communication process
    - When does a data frame start/end?
    - Is the receiver ready for reception?
    - …

Synchronous vs. Asynchronous Transmission

- Asynchronous:
  - Transmission can occur any time
  - Mark beginning and end of transmission
  - There can be periods where „nothing“ is being transmitted

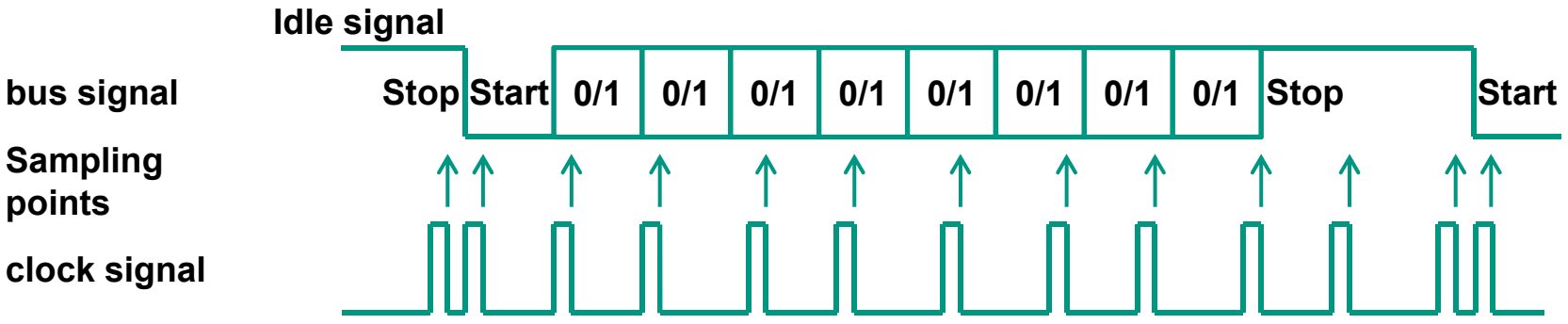
- Synchronous:
  - Transmission at dedicated time points
  - Synchronization even without payload data being send
  - Even if there is no data transmission required, empty packets will be send

Synchronisation Schemes

	Synchronous Transmission	Asynchronous Transmission
Parallel Transmission	Shared clock line *(see Session 4)*	Handshake-mode
Serial Transmission	Suitable line code or scrambler (shared clock line) *(see Session 4)*	Start-Stop-mode

Start-Stop mode

- When no transmission occurs, bus is in idle state → no signal changes occur on the bus
- Begin of a transmission is signaled using a well defined edge on the bus signal → start bit
- This edge is used to synchronize sender and receiver clock
- Every bit has a well defined step size and thus can be separated using the internal clocks → baud rate
- At the end, it has to be ensured that the bus goes to idle level in order to be able to detect a new transmission → stop bit

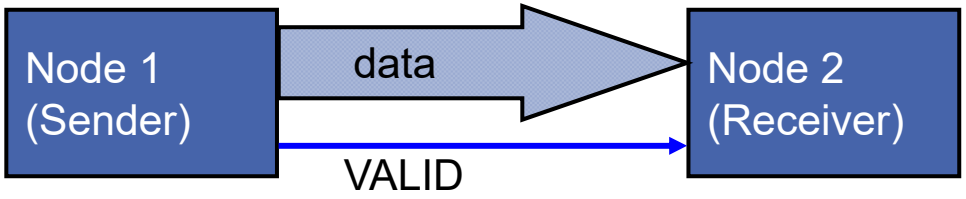


Start-Stop mode: Format definition

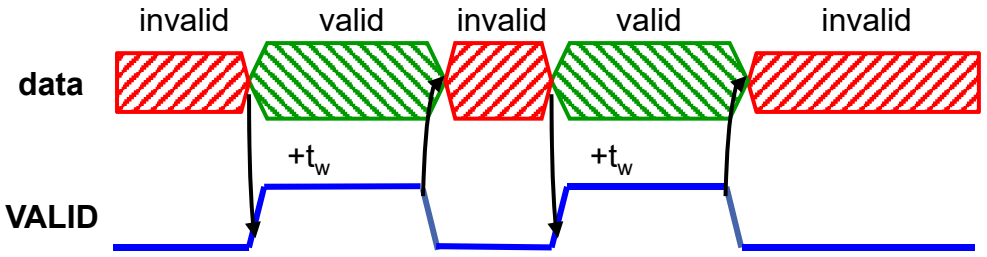
- Format definition for serial transmission using start-stop mode
  - **X-Y-Z**
- With:
  - X: number of data bits (normaly 5-8)
  - Y: parity check (none, even, odd)
  - Z: number of stop bits (1, 2)
- Notes:
  - There is always on start bit (denotes begin of transmission)
  - Parity checks will be explained in session 7
  - The number of stop bits in the definition is the **minimum** number. There can be more „stop bits“ if no transmission is going on

Simplex Handshake

If data is transmitted in multiple subsequent cycles, it has to be regulated when data of one cycle is processed and when a new cycle may be started.



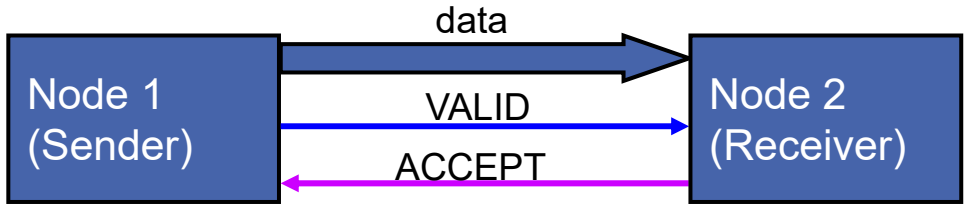
- One always waits for a fixed period of time t_w



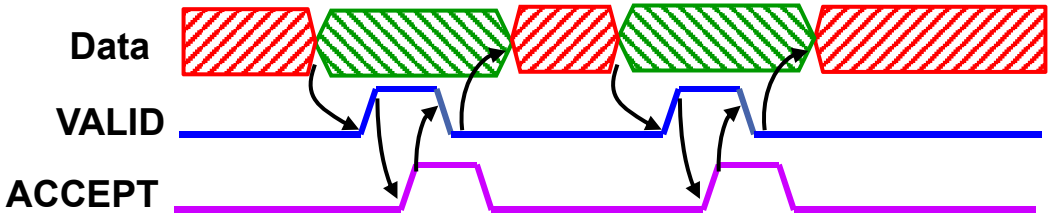
  - Disadvantage: Waiting time t_w is dependent on the slowest node on the bus and is always invariant in length (synchronous mode)

Half-Duplex Handshake

If data is transmitted in multiple subsequent cycles, it has to be regulated when data of one cycle is processed and when a new cycle may be started.



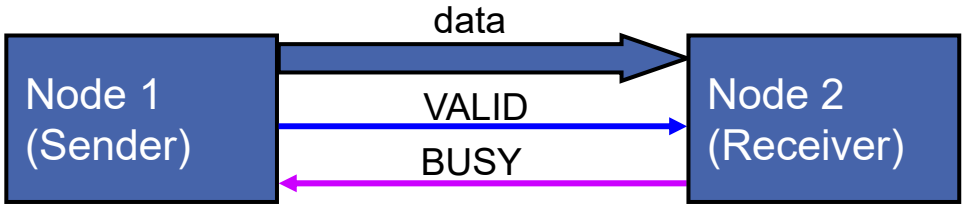
- Receiver asserts ACCEPT signal if data is not needed any longer.



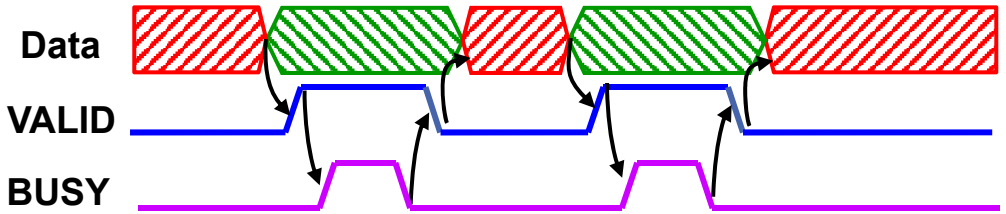
  - Disadvantage: Problems with node synchronization if ACCEPT signal is asserted for too long (sender will remove subsequent data from the bus before receiver has read it)

Half-Duplex Handshake II

If data is transmitted in multiple subsequent cycles, it has to be regulated when data of one cycle is processed and when a new cycle may be started.



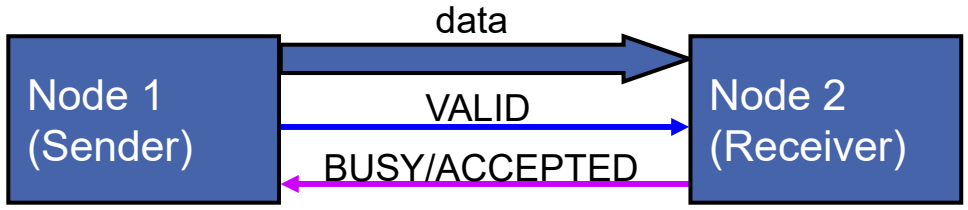
- Receiver asserts BUSY-Signal as long as data has to be available.

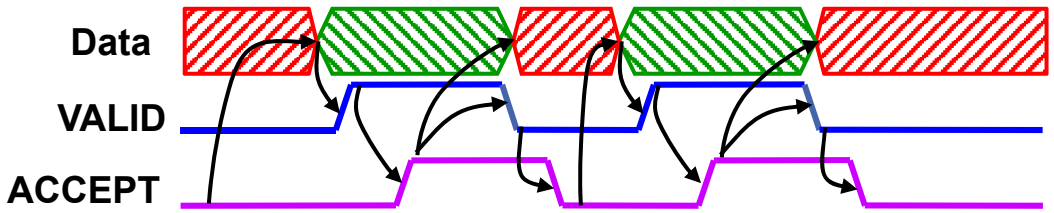


  - Disadvantage: Problems with node synchronization if BUSY signal is asserted too late (data will be removed from the bus too soon)

Full Duplex Handshake

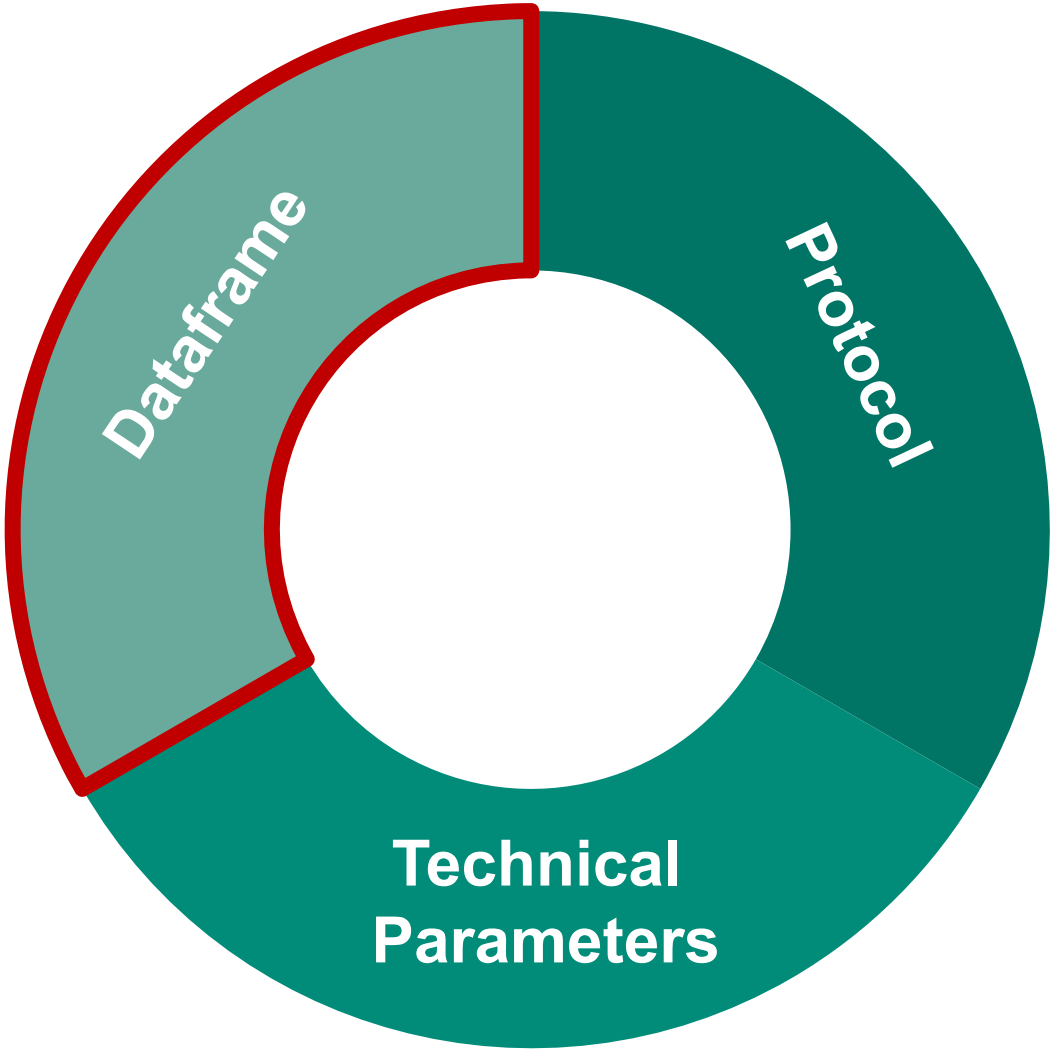
- Safe synchronization by regarding all edges of the control signals
  1. Sender is only allowed to put data on the bus if ACCEPT=0. Valid data on the bus is signaled with VALID=1.
  2. Via VALID=1 the receiver recognizes new valid data and processes it. As soon as the data has been consumed, the receiver asserts ACCEPT=1.
  3. Only if the receiver has acknowledged the data (ACCEPT=1), the sender is allowed to remove the data from the bus and sets VALID=0.
  4. When the receiver detects the de-asserted VALID, it sets the signal ACCEPT=0 as well. Only after this the sender is allowed to start a new cycle.





- Always safe independent of the sender's or receiver's speed.

Parts of a Specification



Dataframe - General Structure

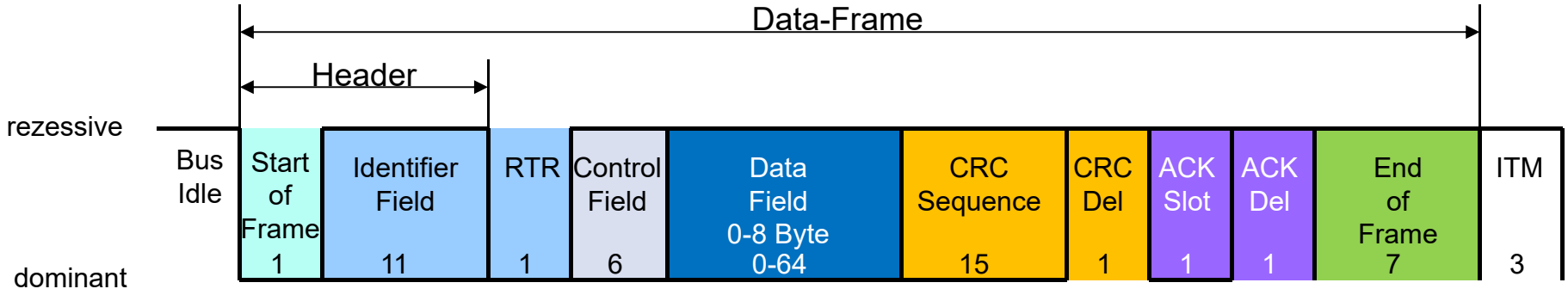
control information <header>	data field <body>	data validation <trailer>
Synchronization Addresses Instructions Arbitration	Fixed or arbitrary length Bits, Bytes, Symbols	Correction information Error detection End notification

Header of a dataframe

- Header is send as first part of a dataframe it contains
  - Information for synchronization (see session 4)
  - Address of target node (see session 6)
  - Information for Arbitration (see sessions 5 and 6)
  - Additional control information (e.g. priorities, change between commands and data, type of packet, etc.)

- Information fields are usually distinguished by their position (length) in the header

- Example: CAN Dataframe

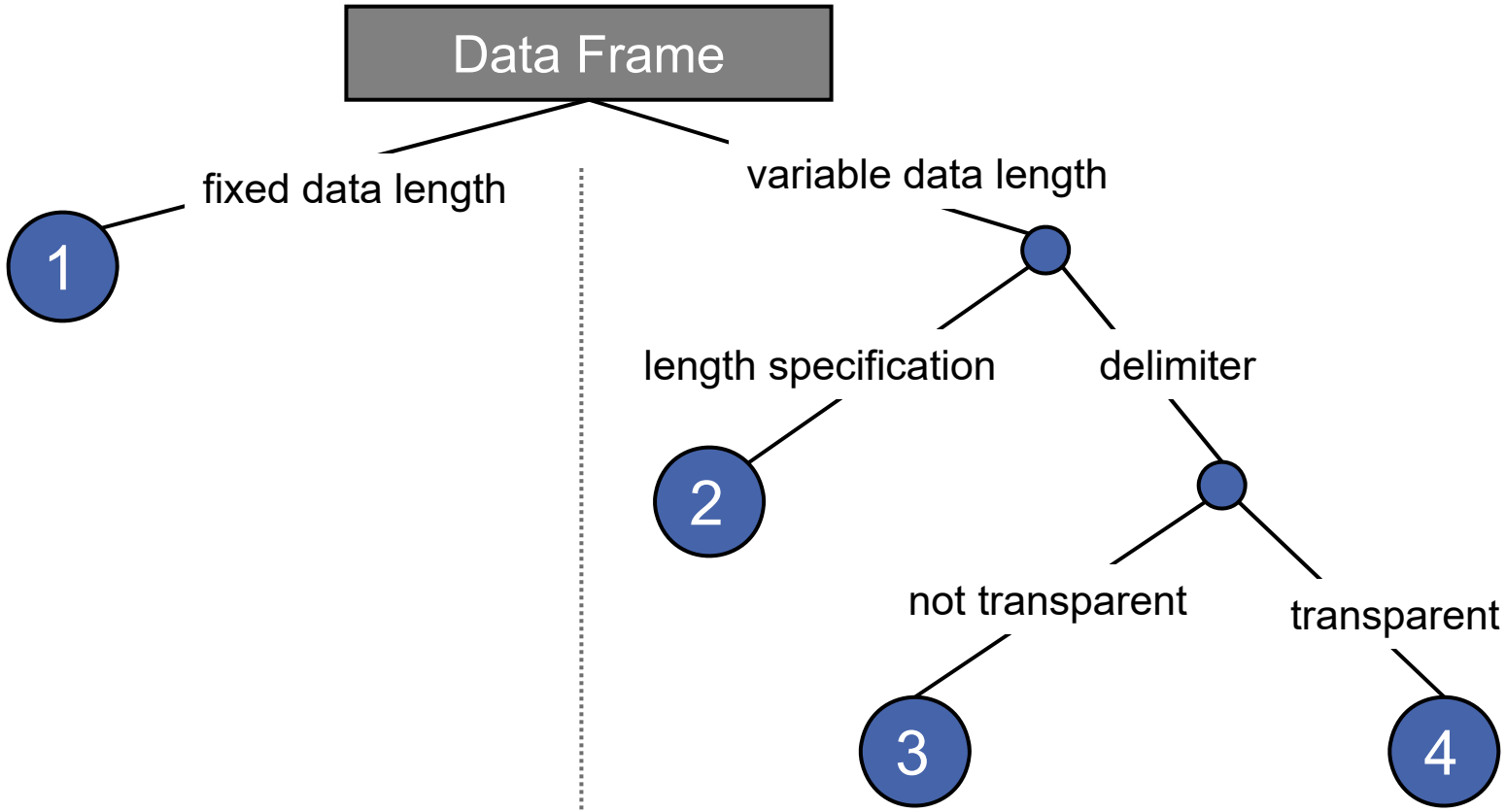


Dataframe – Length of information field

control information <header>	data field <body>	data validation <trailer>
Synchronization Addresses Instructions Arbitration	Fixed or arbitrary length Bits, Bytes, Symbols	Correction information Error detection End notification

- The meaning of a bit in a dataframe is usually determined by ist position
- What happens if data of variable length is to be transmitted?

Determination of data field length



: Explicit field boundaries
: Eventually „empties“

: Efficient utilization
: field boundaries

Examples for Dataframes

1. Fixed data length
   - Serial interface, RS-232

2. Field length in header
   - CAN-Bus Data packet: data field length 0-8 Bytes
   - Length denoted in control field

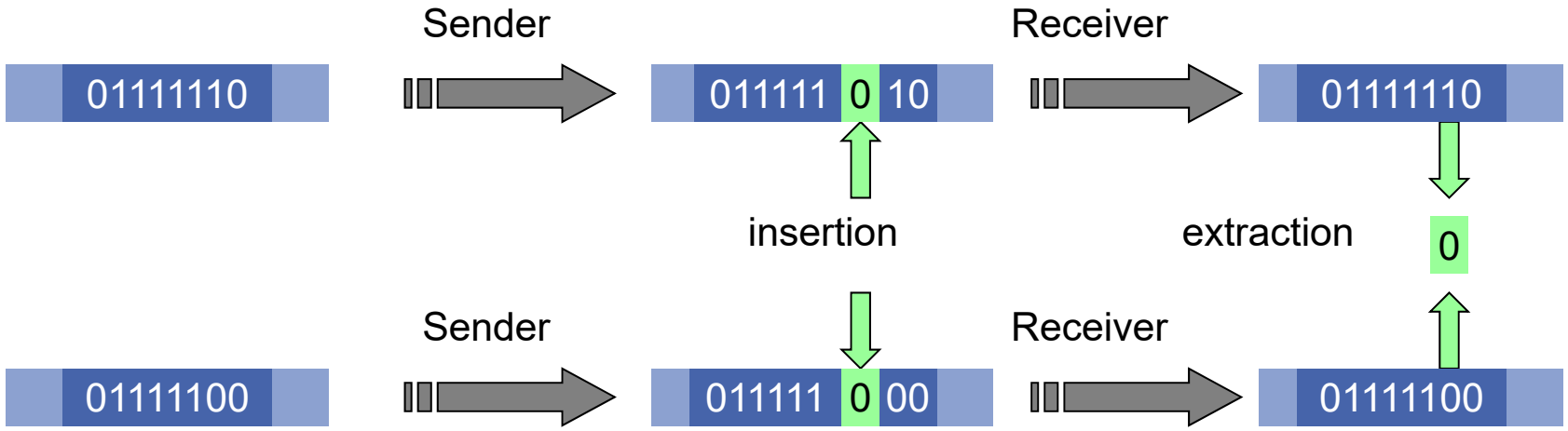
3. Reserved symbols, data field without such symbols
   - unusual
4. Arbitrary data, symbol stuffing necessary
   - Predefined symbols for start and end of transparency, e.g. DLE, STX, ETX (as used in ASCII)

Bit Stuffing (Transparency)

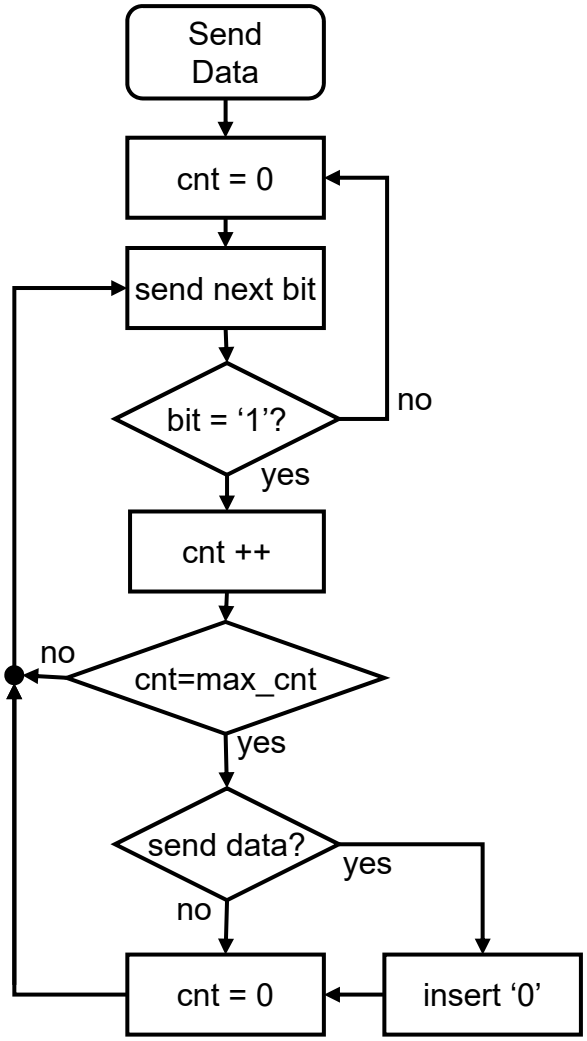
- Bit Stuffing
  - A sequence of *n* ‚0‘s or ‚1‘s is defined as control symbol for Data Start/End. This must not appear within the data stream.
  - Insertion of a ‚1‘ after *n-1* ‚0‘s and vice versa

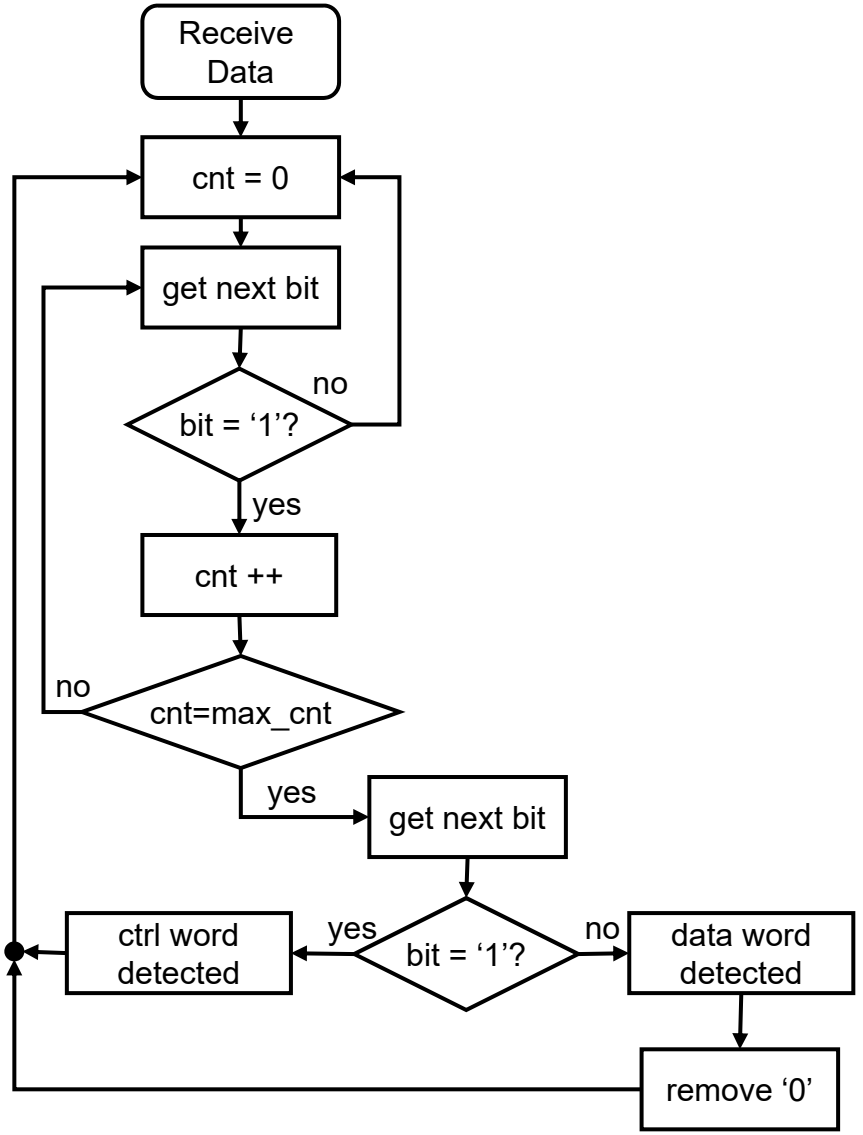
Example:

- after five ‚1‘s in a data word a ‚0‘ is inserted. The receiver will remove every ‚0‘ after a sequence of five ‚1‘s.



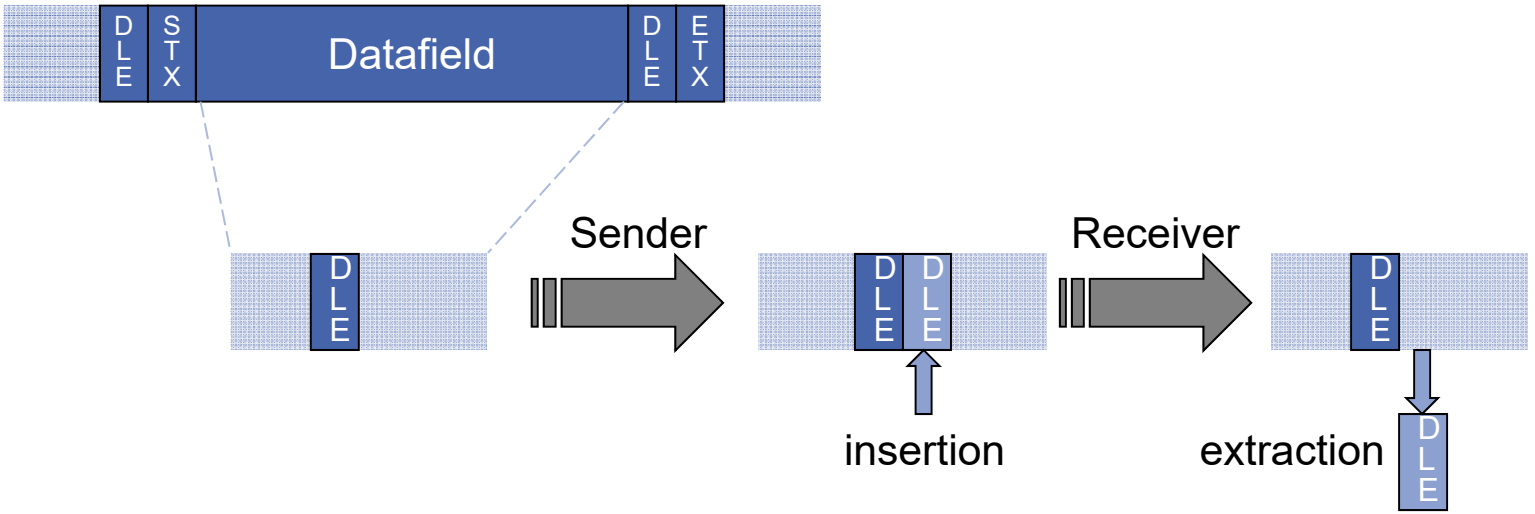
Bit Stuffing – Algorithm (Example)





Symbol Stuffing (Transparency)

- Symbol Stuffing
  - When using frame formats with variable data field lengths, a symbol is necessary that marks the start and end of a data field:
    Data Link Escape (DLE)
  - This symbol can also be a regular symbol to be transmitted
    → Masking necessary
  - If DLE is part of the data field, the symbol is doubled and receiver removes the second DLE symbol.

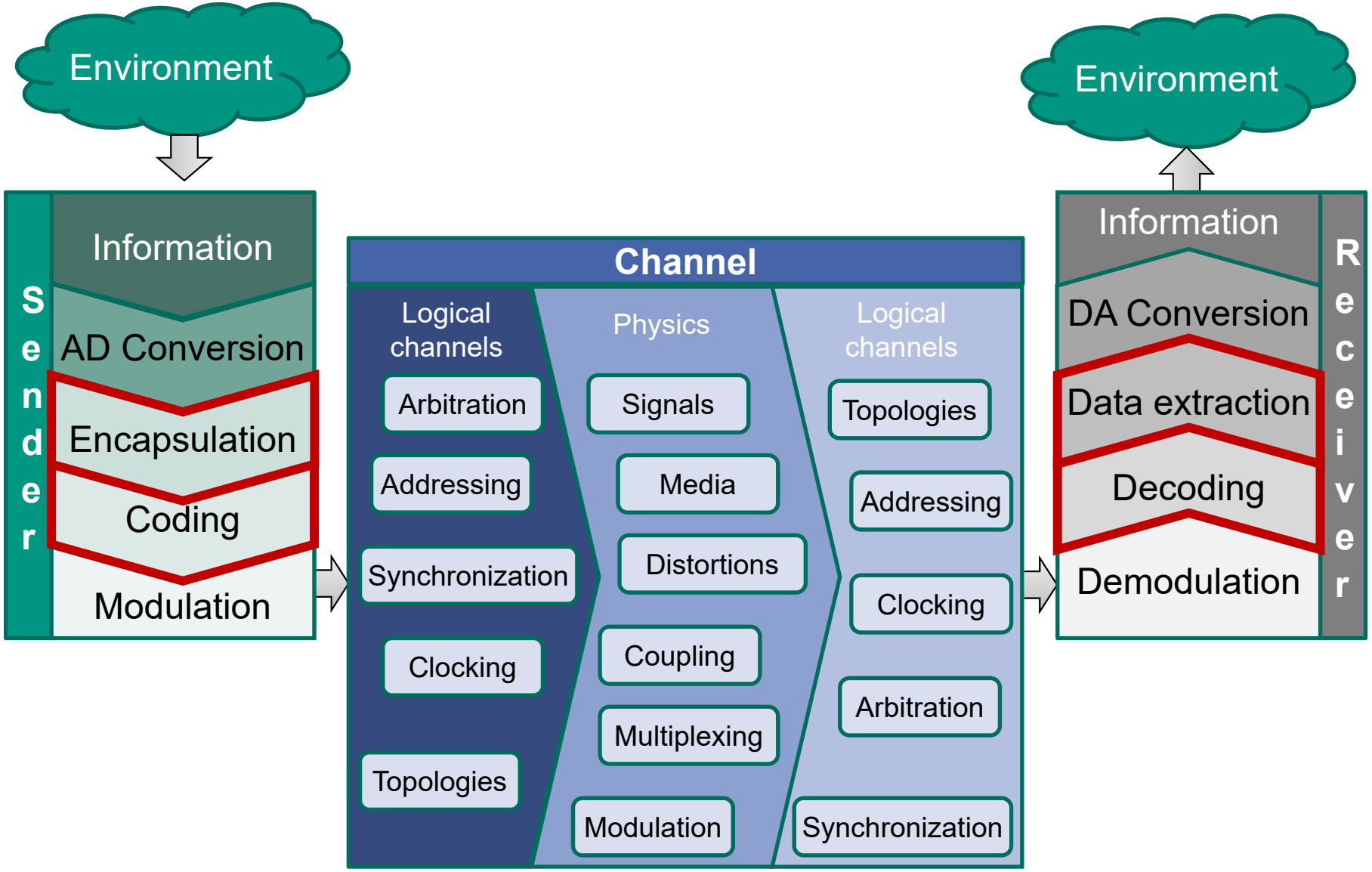


Dataframe – Data validation

control information <header>	data field <body>	data validation <trailer>
Synchronization Addresses Instructions Arbitration	Fixed or arbitrary length Bits, Bytes, Symbols	Correction information Error detection End notification

- Data might get corrupted during transmission
- How do we ensure that correct data has been received?

Error Detection



The difference between Safety and Security

Safety:
Protection of the environment from failures of the system.

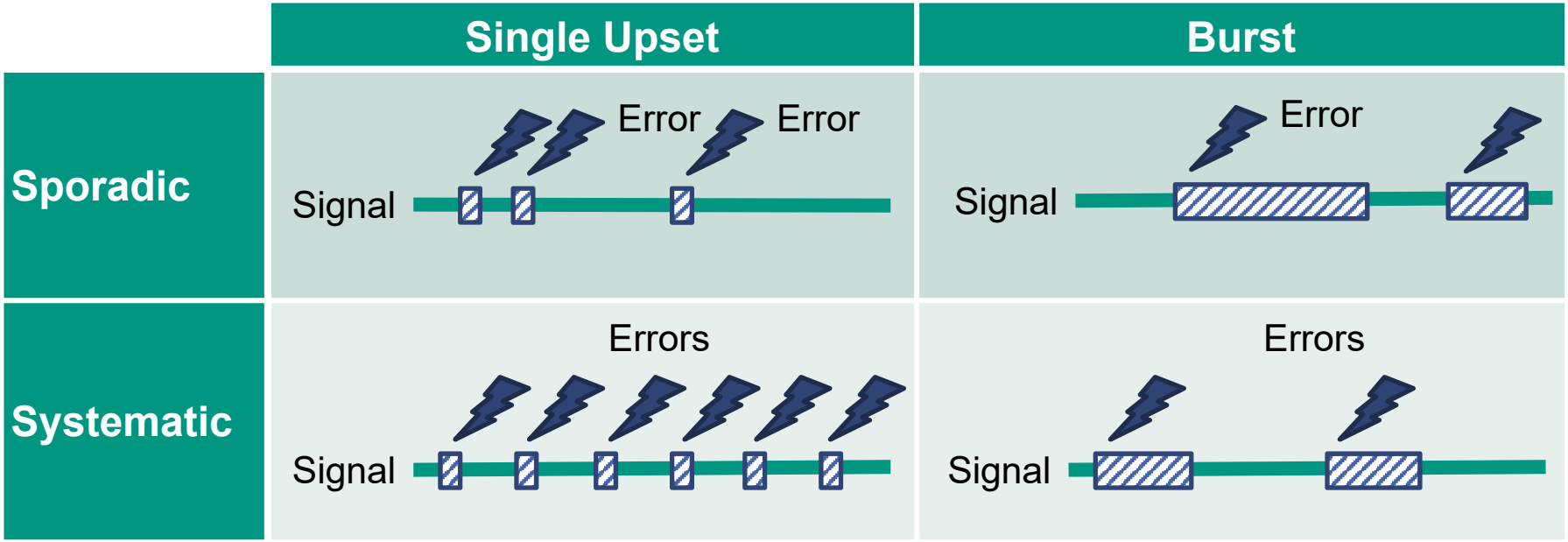
- In our case:
  - Changes in data during transmission have to be detected, so that wrong data can cause no harm to others.
  - Protection against unwittingly errors

Security:
Protection of the information against attacks from the environment.

- In our case:
  - Willful change of information by attackers

- Security is not handled in this lecture

Defect classes



- Single upset: Only one bit is changed
- Burst: Multiple bits in sequence are changed

- Sporadic: Changes happen randomly
- Systematic: Changes follow a certain pattern

How to Deal with Errors/Defect Classes

- Errors can only be processed if they are known
  - Analysis of possible error sources is part of the design process
  - Error avoidance, error detection and/or error correction are part of the specification of a communication system
- Error avoidance can be done on physical level, e.g.:
  - Proper media (shielding, twisted pairs)
  - Signaling (differential signaling, voltage levels)
  - System design (termination)
- Nevertheless, still errors can occur
  - Error detection/correction on data level is required
  - Adding redundancy to the data, to detect/correct errors
    - Hamming codes (see lecture “Digitaltechnik”)
    - Parity checking
    - CRC

Redundancy

Redundancy:

The number of bits used to transmit a message minus the number of bits of actual information in the message.

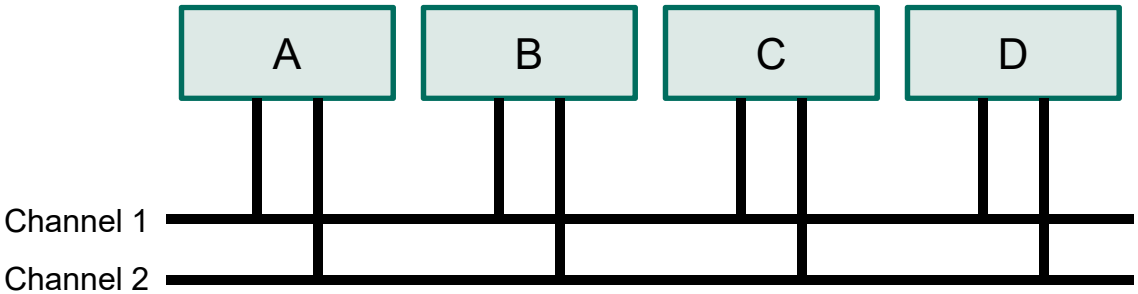
(Source: wikipedia.org)

- Redundancy can be external or in the information itself
- External (Hardware) redundancy
  - Implement the same functionality multiple times
  - E.g. transmission of the same information over two different lines
- Information redundancy
  - Natural redundancy: „wasted“ capacity of a channel that does not contain new information
  - Constructed redundancy: additional data, that has been added to protect the information

External Redundancy

- Add additional redundancy by implementing multiple instances of the communication system as a whole or in parts
  - See lecture „Systems and Software Engineering“

- Example: FlexRay
  - Two independent communication lines that can be used to transmit the same data twice, each on one line
  - An interference on one line should not have exact the same effects on the other line
  - If the received packets for these two lines differ, they are rejected and have to be transmitted again

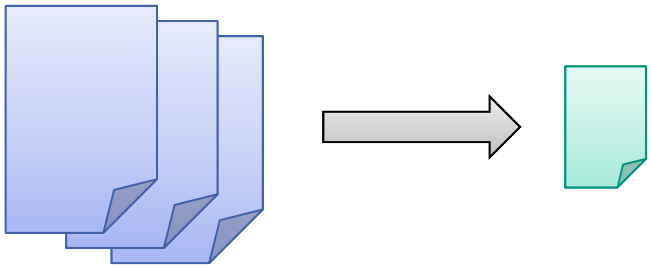


Constructed vs. “Natural” Redundancy

- In “natural” information inherent redundancy can exist
  - Speech, music, pictures, …
  - it includes unneeded data (e.g. voice can be transmitted with small bandwidths)
- “Natural” redundancy is…
  - … usually created randomly and often distributed uniformly
  - … has no dependable information value
- Replace “natural” redundancy with systematically constructed redundancy
- Processing of data for transmission:
  1. Sample natural information
  2. Reduce inherit redundancy to reduce amount of data (compression)
  3. Add artificial redundancy, e.g. to add error detection capabilities

Constructed Redundancy: Hash functions

- Hash function maps large data sets (keys) to smaller data sets of fixed length (hash values):



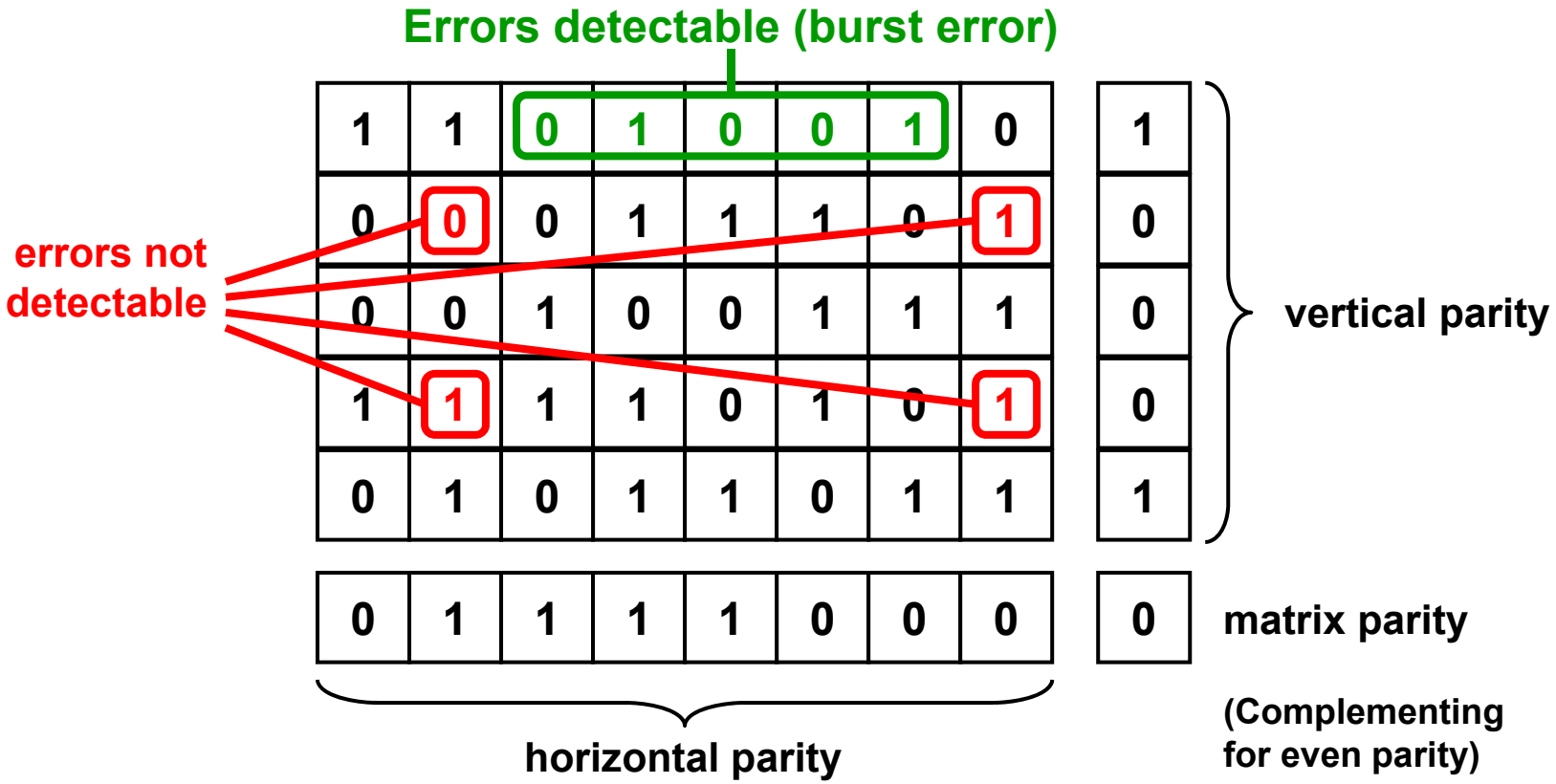
- Properties of good hash functions (for communication purposes):
  - Different inputs should lead to different hash values (reduce number of collisions)
  - Efficient calculation of hash
  - Small change in input values → large change in hash value

- Hash used as checksum to detect errors:
  - Cross sum
  - Parity
  - Cyclic Redundancy Check (CRC)

Methods for Error Detection based on Redundancy (recap)

- **Parity**
  - Complementing a data word with an extra (single) bit to make the number of bits within the data word carrying ‚1‘s <even> or <odd>
  - Can be used column-wise (vertical parity) or row-wise (horizontal parity)
  - Detection of an odd number of errors (1, 3,..).
- **Block Check**
  - A set of n data words is protected by vertical parity and horizontal parity
  - Detection of burst errors of length n is possible
- **XOR Block Check**
  - For every 2 data words their XOR concatenation is send as a third data word. These 3 data words allows to reconstruct all data from any two data words. It has to be known which data word is erroneous!
  - Used in RAID arrays.
- **Hamming Code**
  - Exploiting the hamming distance between valid code words, meaning that not all code words that can be generated with the available bits are valid code words

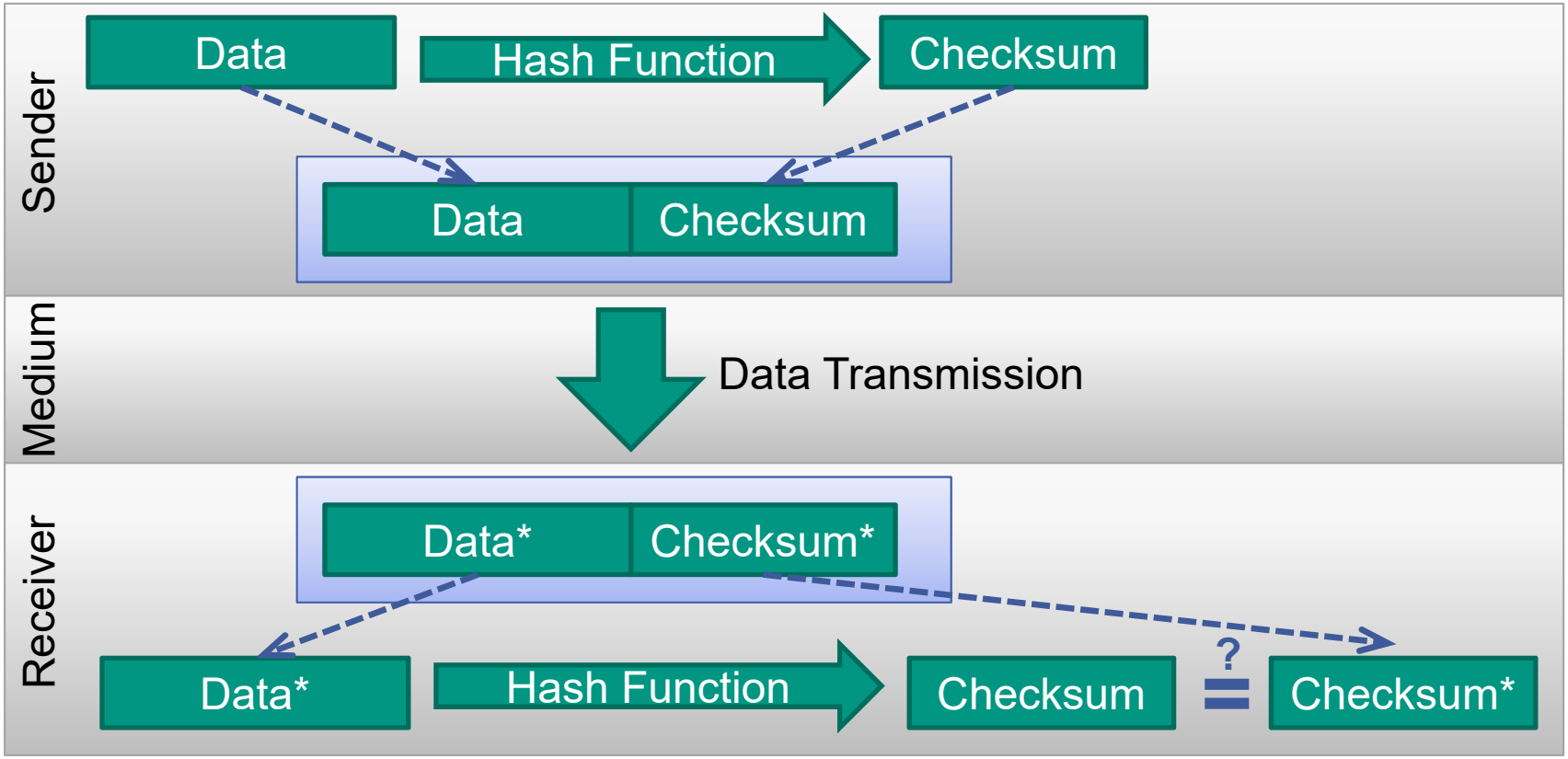
Example: Block Check (recap)



- Detectable bit errors:
  - All odd number of errors
  - All even number of errors with odd number of bit errors per row/column
  - All 2-Bit errors
  - Most 4-Bit errors (except “rectangular” bit errors)

Cyclic Redundancy Check (CRC) – Approach

- Generate a checksum (Hash)
- Send checksum together with raw data
- Receiver checks received data and received checksum match



CRC –mathematical description

- Generation of checksum uses a generator polynomial *G(x)* on sender and receiver side
  - most significant bit (MSB) and least significant bit (LSB) must be ‚1‘
- Calculation on sender side
  - Data vector *M(x)* of length *m* bit (exceeding length of *G(x)*)
  - Checksum equals the modulo of the division $(x^r M(x))/G(x)$
    - *r:* degree of generator polynomial
    - $x^r M(x)$ adds *r* zero bits to the end of the data vector
  - Checksum is appended to the data
    - Corresponds to the addition of the modulo: $x^r M(x) + R$
- Calculation on receiver side
  - Division of received data by *G(x)*
    - If the modulo of the division is not equaling zero an error has occurred
    - If the modulo of the division equals zero, no error has been detected

Example for CRC calculation

- See blackboard notes