Karlsruher Institut für Technologie

Institut für Technik der Informationsverarbeitung

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 5: Data transmission

Clicker Session: Recapitulation

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

Recapitulation

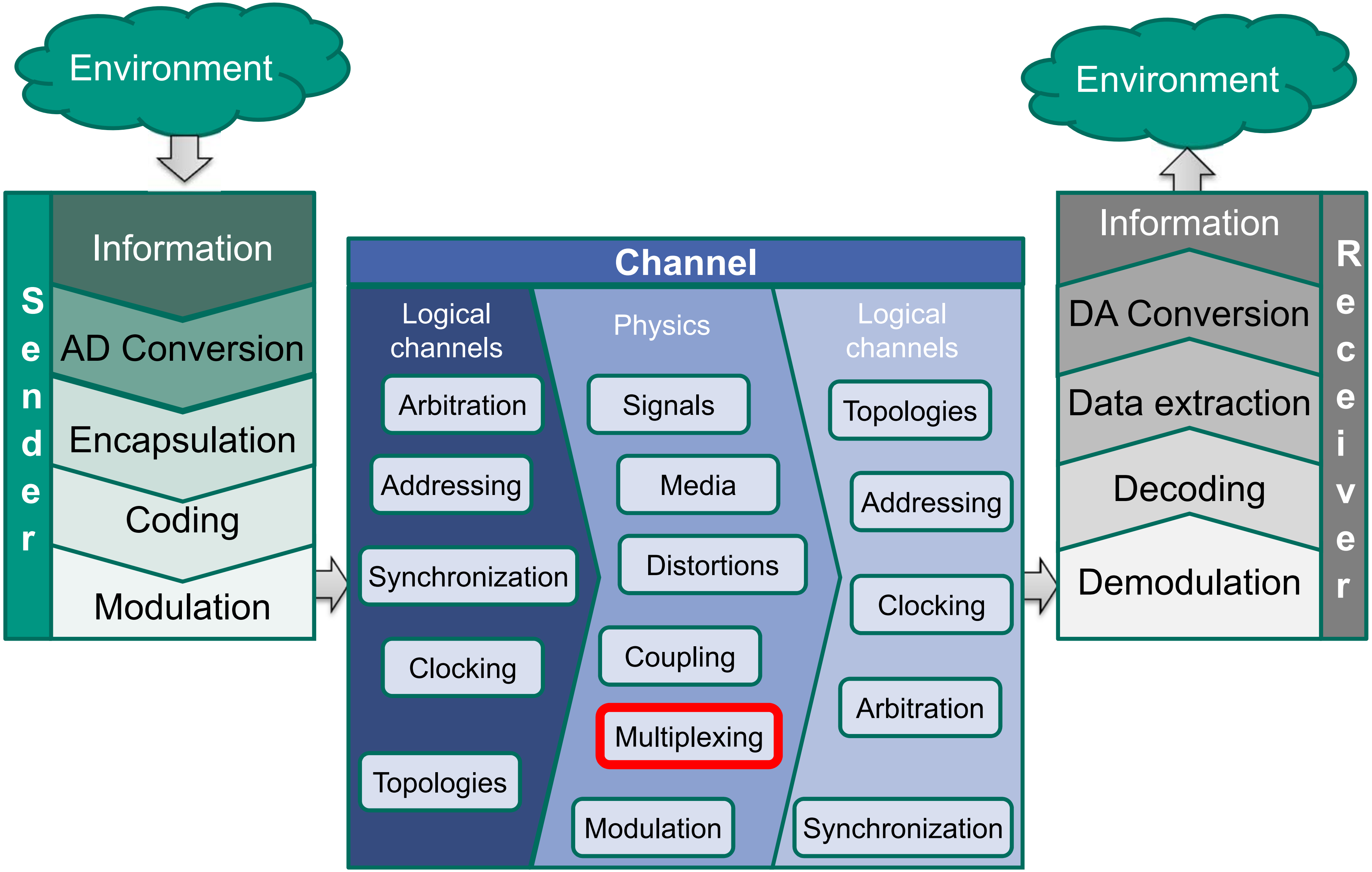
- Sampling
  - Sampling Methods
  - Technical restrictions
- Line Codes

Homework

- Classify the Line Codes according to the properties given in the last two slides (+, ++ , - , -- , O)

Line Code	DC Current	Required Bandwidth	Clock Recovery
NRZ	--	+	-
RZ	--	O / -	+ / -
NRZI	--	+	-
AMI	++	+	O
Manchester	++	O / -	++

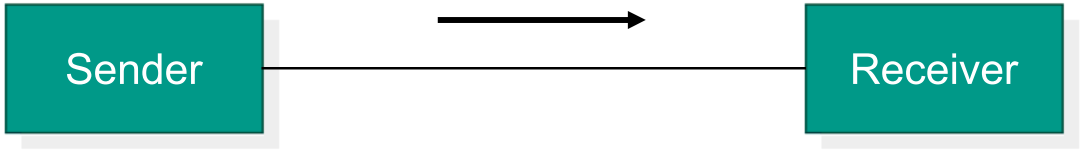
Transmission System – Multiplexing



Types of Transmissions

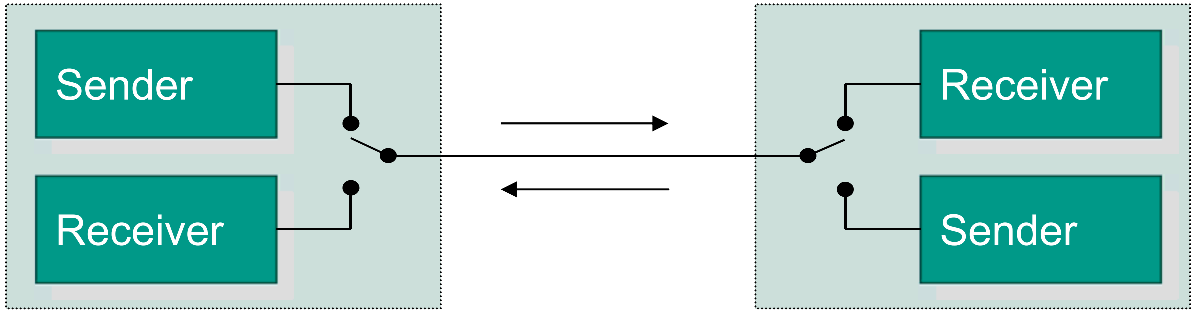
A transmission on a channel can be classified according to the direction of the data transmitted, as well as according to the degree of parallelism.

- Simplex transmission (respectively directional operation)
  - Only a single direction
  - Application: television, terrestrial radio, process data acquisition

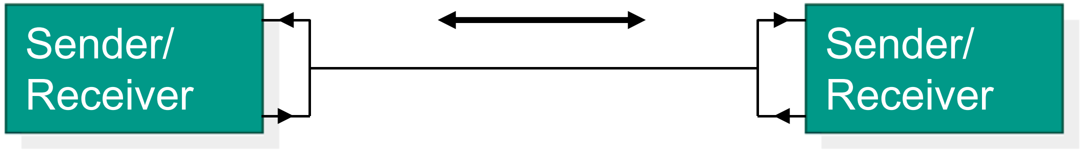


Types of Transmissions

- Half duplex transmission (also: alternating operation)
  - In both directions, but not simultaneously
  - Switching in terminal equipment
  - Application: radio communication, 10base-T Ethernet , ...

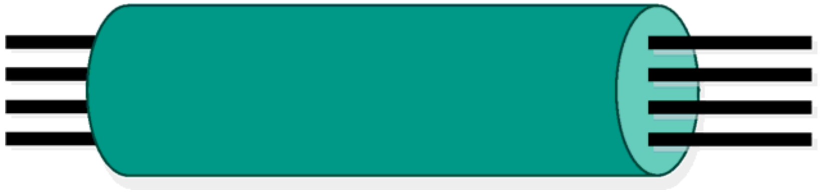


- Full Duplex transmission
  - In both directions simultaneously
  - Application: Telephone, RS 232, 100Mbit Ethernet,...



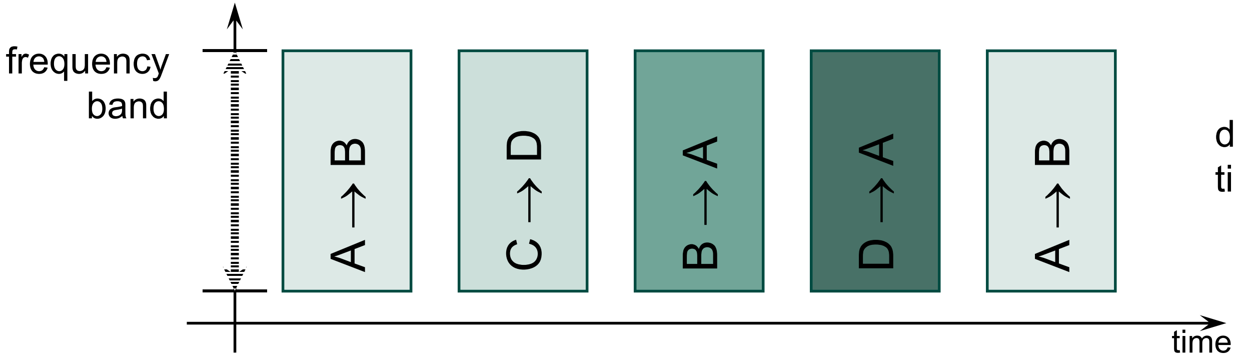
Multiple use of Media by multiplexing

- Multiple signals can be transmitted in multiplex mode
- Space division („copper multiplex“)
  - Representation of signals on multiple lines, baseband



bundle of
wires (pairs)

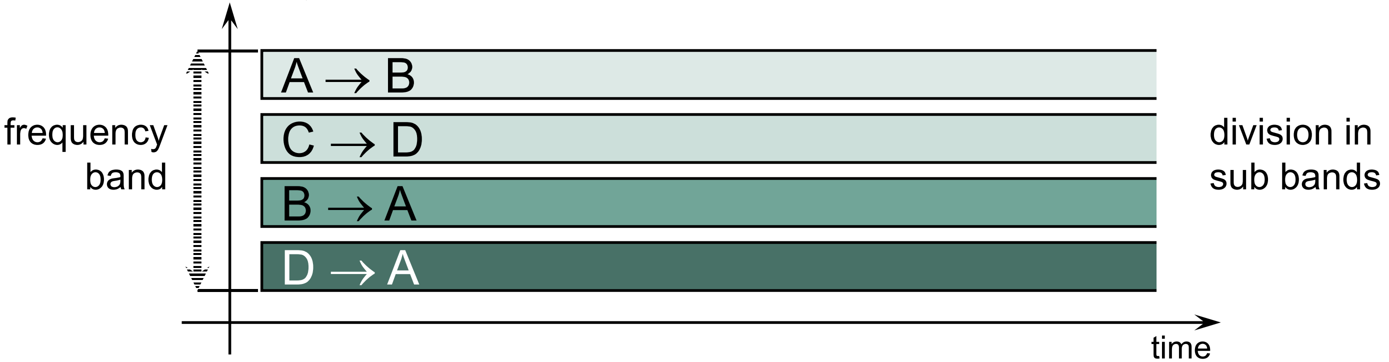
- Time division
  - Representation using temporal shifts (slots), base band



division into
time slots

Multiple Use of media

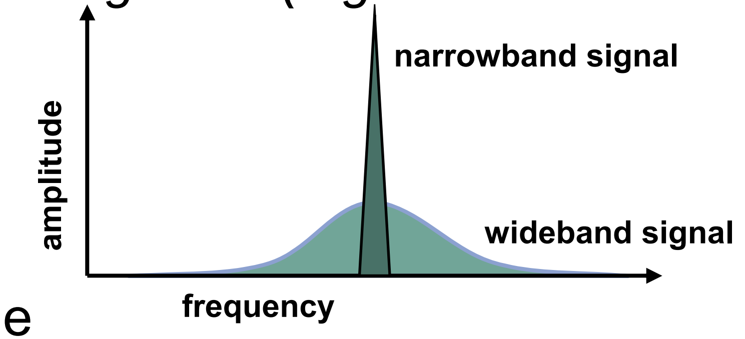
- Frequency division multiplex
  - Representation using multiple frequencies at the same location, at the same time, modulation



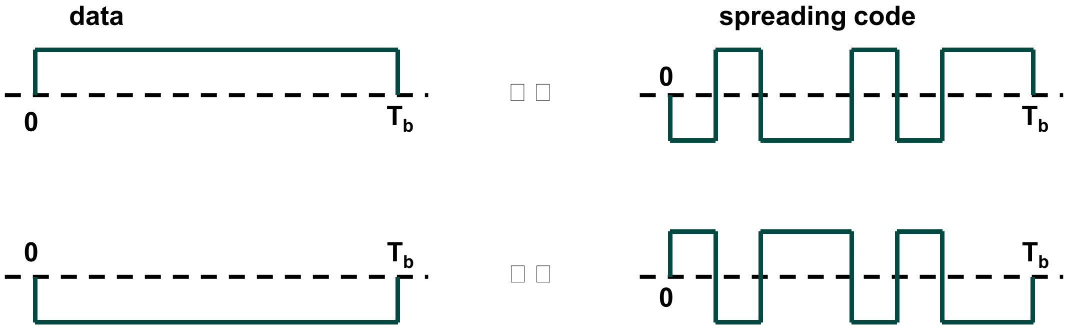
- Wavelength Division multiplex (special form of frequency division)
  - transmission of multiple wave lengths (colors) over a single optical waveguide.
  - currently up to 80 individual wavelengths possible
  - especially suitable for wide area networks (WAN)
  - within a single wavelength time multiplex possible – currently data rates of 80 GBits/s feasible (theoretically resulting in 6.4 Tbit/s per fiber)

Code Division Multiple Access (CDMA)

- frequency spread:
  - Transmitted signal is distributed over a wide spectrum using a so called *spreading code (e.g. Walsh function).*



- Spreading Code
  - One bit is encoded with multiple sub-bits (Chips)
  - □ Increase of bandwidth by spreading factor (SF)



Different ways for frequency spreading

1. Modification of the carrier frequency during a transmission using pseudo random sequences (PRS) with significantly high clock or chip rate r_{Chip}

   □ *Frequency Hopping Spread Spectrum, FHSS*

2. Logic interconnection of the data bits with a pseudo random sequence **before** modulation

   □ a bit is then represented using multiple chips

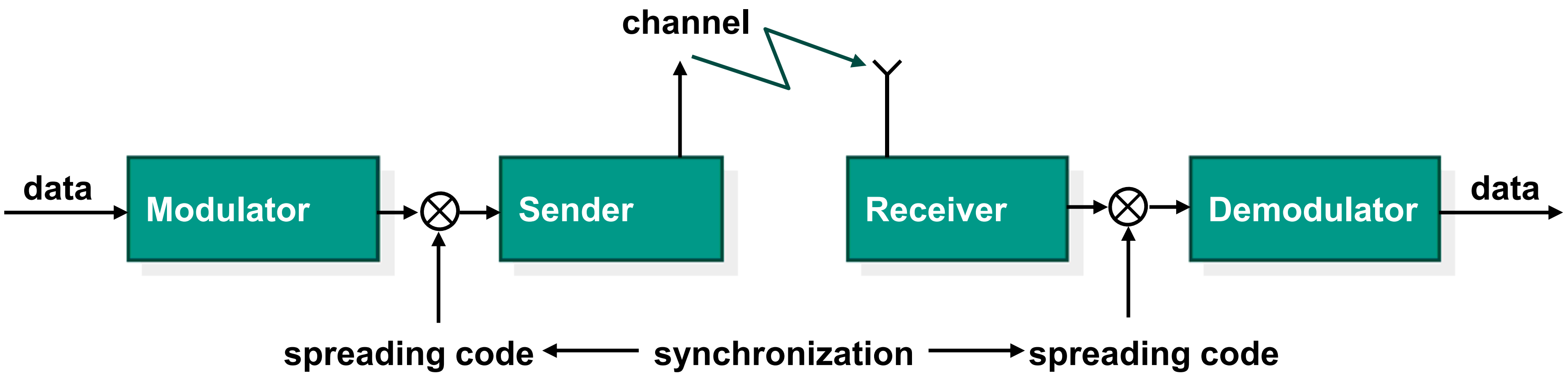
   □ Direct Sequence Spread Spectrum, DSSS

- Precondition for both schemes: The pseudo random sequence has to be know to the receiver in order to be able to decode the message

Advantages and Disadvantages

- Pro:
  - More robust against narrowband disturbances because signal energy is distributed on a broader spectrum
  - Lower sensitivity against interferences from other nodes
  - Better protection against eavesdropping
    - As the signal looks like white noise, it is impossible to detect when a transmission is going on
    - Eavesdropper has to know the used pseudo random sequence in order to be able to reconstruct the send data
- Contra:
  - More effort required an sender and receiver side
  - Requires channel with larger bandwidth

CDMA Transmission Path



- Data is transformed using a spreading sequence and decoded back into the original data by the receiver.
- Different nodes can be distinguished by their different spreading codes.
  - Logic combination with the correct spreading code will extract the original data
  - Transmissions of other nodes will appear similar to white noise and can be filtered out.

Spreading codes

- All nodes are sending at the same time and use the same frequency band
- Different transmissions are separated using individual spreading codes

- Requirements for spreading codes
  - The spectrum of the spread data function shall look like white noise
    - □ Pseudo random sequences
  - Spreading functions have to be orthogonal (orthogonal means that the inner product of two functions equals to 0)
    - □ e.g. Walsh functions

- As the Walsh functions do not generate a spectrum that is similar to white noise, they are combined with pseudo random sequences

Formation rule for Walsh functions

1. Start with a logic “0” as element a_{11} of a 2x2 matrix

2. The element a_{11} is repeated at position a_{12} and a_{21}
3. At position a_{22} the inverse of a_{11} is inserted, in this case a logic “1”

$$\begin{matrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{matrix}$$

$$\begin{matrix} 0 & 0 \\ 0 & 1 \end{matrix}$$

4. This matrix now becomes element a_{11} of a new 2x2 matrix, where the steps 2. and 3. are repeated
5. The steps 2. to 4. are repeated until the required length of the Walsh function is reached

Formation rule for Walsh functions

- Substitution in resulting matrix
  - Zeros with +1
  - Ones with -1

$$\begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{matrix} \quad \Rightarrow \quad \begin{matrix} +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 \end{matrix}$$

Walsh Function f_0

Walsh Function f_1

$\vdots$

Walsh Function f_n

Properties of Walsh functions

- All Walsh functions (except function 0) contain the same amount of logic ones and zeros
- The functions are orthogonal to each other, the cross correlation of two different functions equals to 0
  - Example: Walsh function 1 and 5

$$\begin{array}{ccccccccc} +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & \\ & & & & & & & & \text{times} \\ +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & \\ \hline +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & \text{Sum} = 0 \end{array}$$

- The autocorrelation of a function equals to eight (in this example)
  - Example: Walsh function 1

$$\begin{array}{ccccccccc} +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & \\ & & & & & & & & \text{times} \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & \\ \hline +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & \text{Sum} = 8 \end{array}$$

Data transmission with CDMA

- Multiple nodes want to send data at the same time using the same frequency band
  - Every node has a unique spreading code assigned to it
- Every bit to be send is encoded with the spreading code assigned to the corresponding sender
  - Transmission of „0“: Spreading sequence is send as is
  - Transmission of „1“: Spreading sequence is inverted
- Spreading sequences of all nodes are superimposed and form **one** signal on the media
  - Attention: The signal on the media is now multi-valued and not binary any more!

Reception with CDMA

- On receiver side the transmitted signal is correlated with the Walsh function that has been assigned to the node
- As the Walsh functions of the other nodes are orthogonal, only the signal that is meant for the receiver is filtered
  - If the result of the correlation is +8, a “0” has been send
  - If the result of the correlation is -8, a “1” has been send

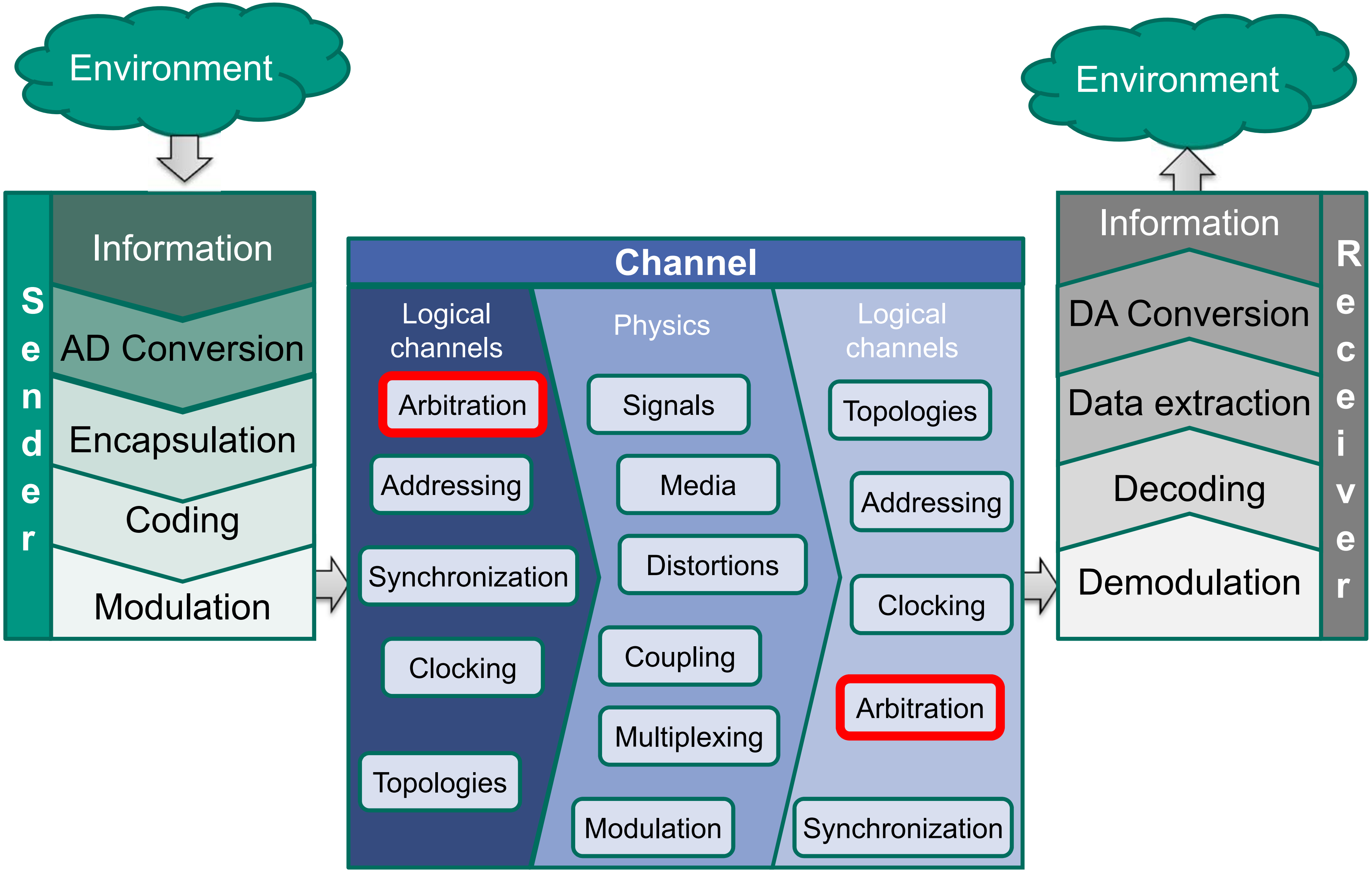
  - Example: Receiver using the Walsh function #6

$$\begin{array}{ccccccccc} -2 & +6 & +2 & +2 & -2 & -2 & +2 & +2 & \\ &&&&&&&& \text{times} \\ +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & \\ \hline -2 & +6 & -2 & -2 & +2 & +2 & +2 & +2 & \text{Sum} = +8 \end{array}$$

    - Sum= +8 □ a “0” has been send

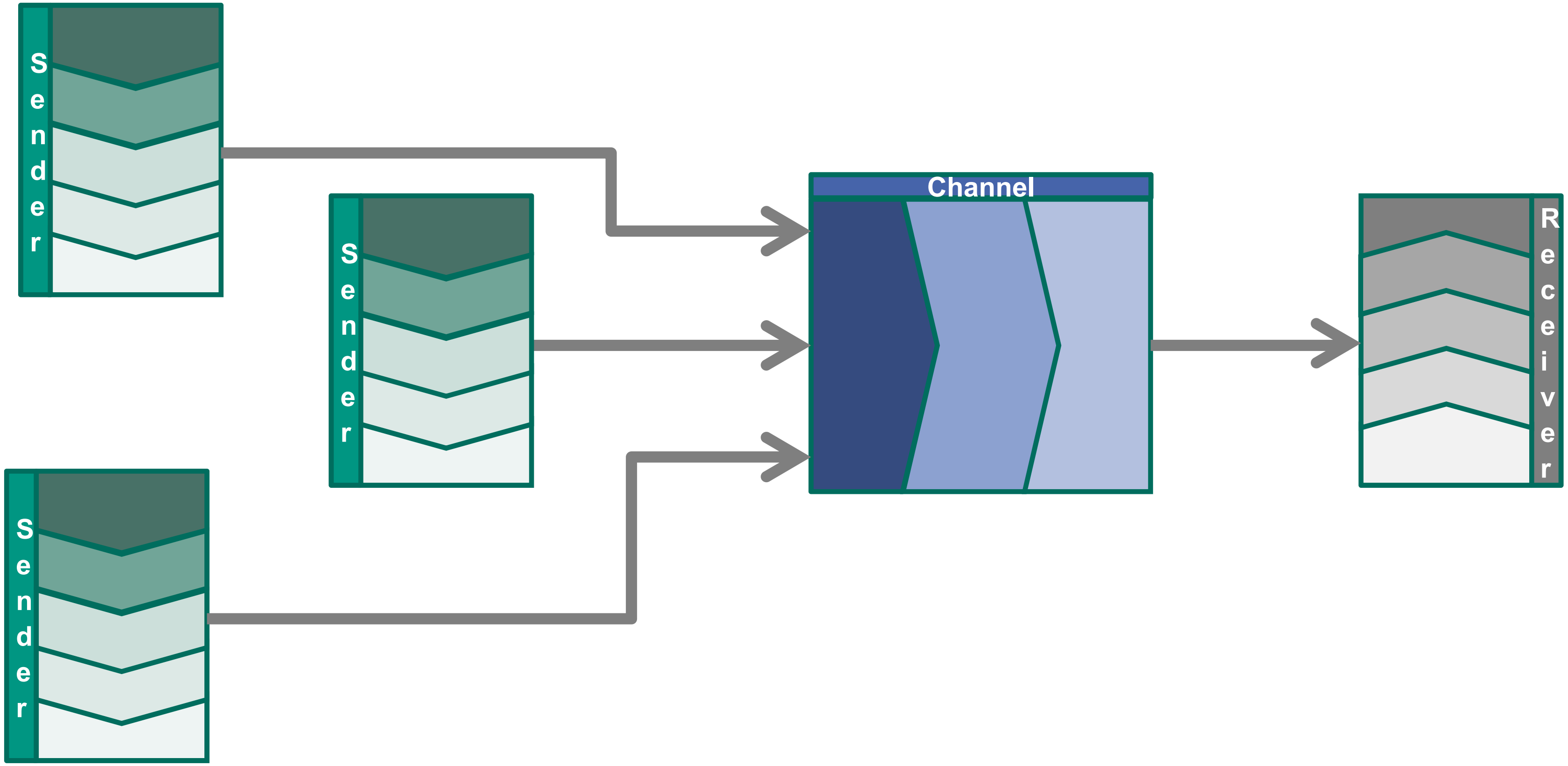
- For real data transmission a decision based on the exact value is not possible any more because of interferences and other disturbances that distort the signal. Therefore the signal has to be within in a predefined interval to be interpreted as “1” or “0”.

Transmission System - Arbitration

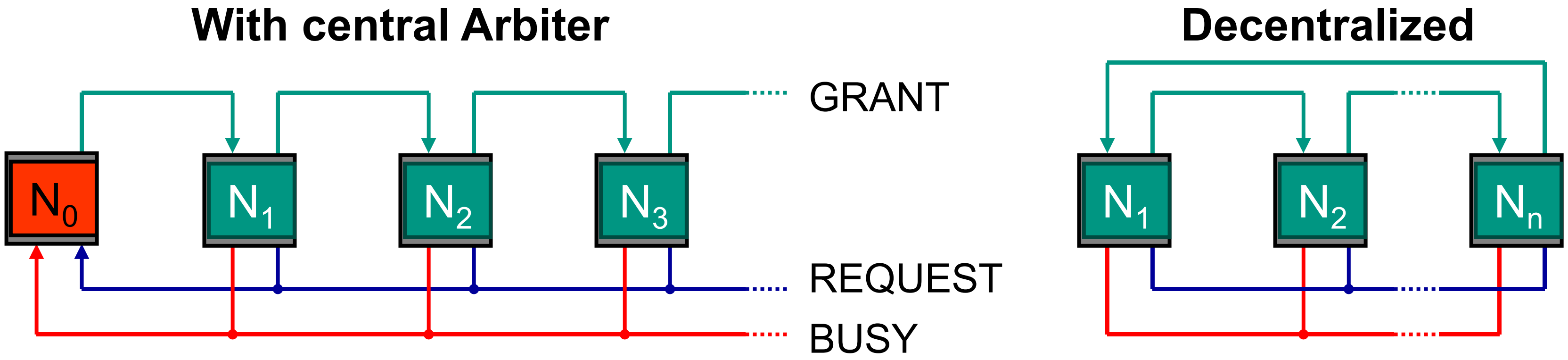


Dynamic Multiplexing: Arbitration

- Static multiplexing schemes can make bad use of available channel if transmissions are not constant
- Dynamic assignment of channel to one sender □ Arbitration



Daisy-Chain

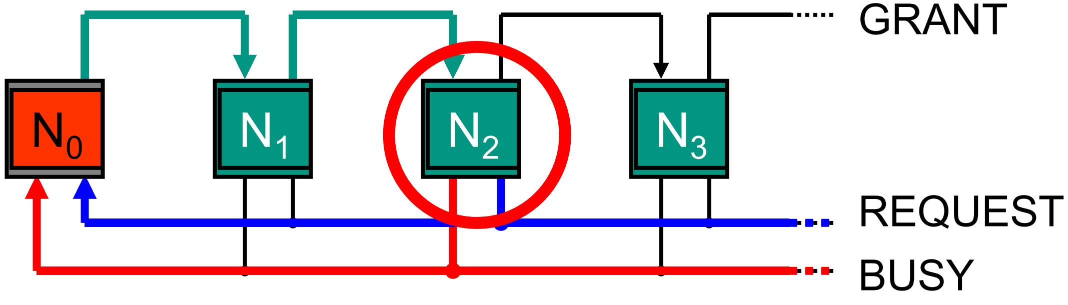


- All nodes that are “ready-to-send” activate the *REQUEST*-line
- If a node N_i receives the *GRANT*-signal it can take over the control of the bus
  - If N_i has data to send, N_i activates the *BUSY*-line and keeps the *GRANT*-signal
  - If N_i has no data to send, it passes the *GRANT*-signal to the next node N_{i+1}
- Centralized: One central arbiter N_0. Chain always starts with N_0.
- Decentralized: Chain always starts with the node that was the last bus master
- Properties:
  - relatively slow
  - easy extensions (scalable)
  - low hardware footprint

□ Centralized daisy-chaining unfair, as priorities are fixed by the wiring

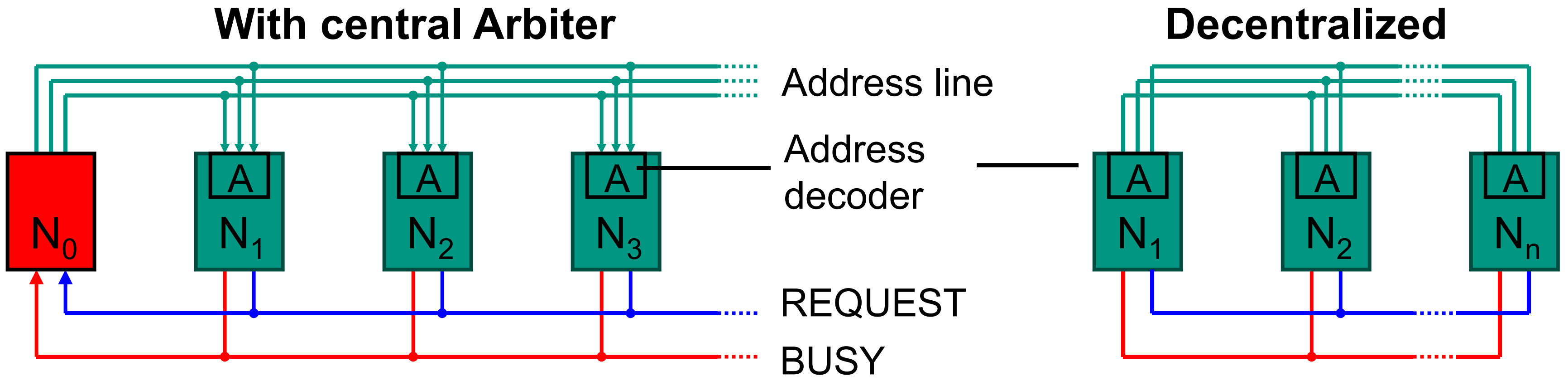
Daisy-Chain

- Example:



- N_2 has a transmit request (wants to send data)
- N_2 activates *REQUEST*-line
- Arbiter N_0 asserts *GRANT*-signal
- N_1 has no data to send and passes *GRANT*-signal
- N_2 asserts *BUSY*-signal, removes *REQUEST* and takes control of the bus
- N_2 de-asserts *BUSY*-signal thus freeing the bus for other transmissions

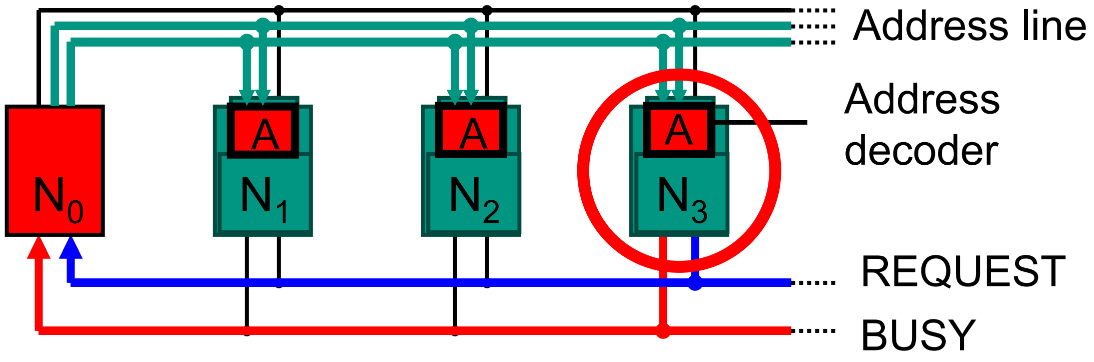
Polling



- Transmit requests are forwarded to an arbiter over a common signal line
- Arbiter then addresses the bus nodes sequentially following their priority
    - decentralized solution: last bus master acts as arbiter
- If the address of a node with a transmit request is set, the node asserts a *BUSY*-signal and becomes bus master.
- Properties:
    - fair and priority based arbitration possible
    - medium hardware resources
    - slow arbitration

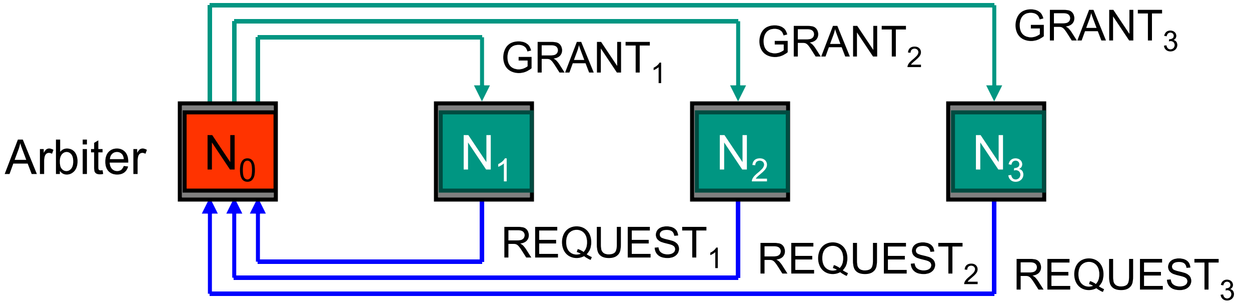
Polling

- Example:



- N_3 has a transmit request (wants to send data)
- N_3 activates *REQUEST*-line
- Arbiter N_0 polls N_1 (checks for transmit request)
- Arbiter N_0 polls N_2
- Arbiter N_0 polls N_3
- N_3 has issued a REQUEST and asserts now the BUSY-signal to take over the bus

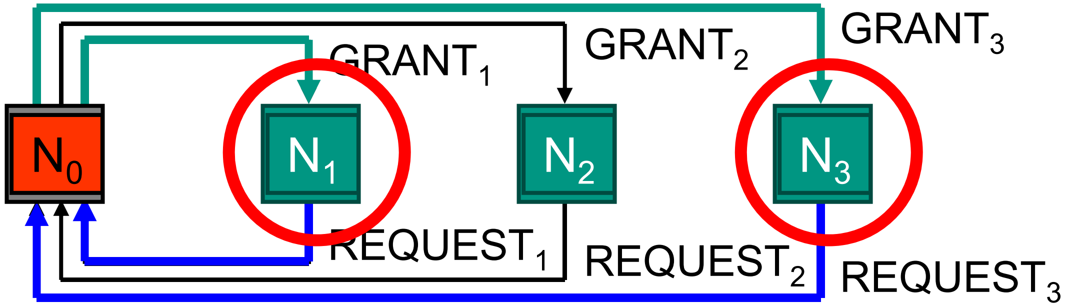
Tap Line



- All nodes are connected with a central arbiter over individual tap lines
- For a transmit request dedicated *REQUEST*-lines are asserted with an according signal
- The central arbiter issues *GRANT*-signals according to an arbitrary scheme
  - Priority table
- Properties:
  - fair and priority based arbitration possible (priorities can be changed easily)
  - high demand of hardware resources (expensive)
  - fast arbitration due to dedicate *REQUEST*- and *GRANT*-lines
  - Scalability to an arbitrary number of nodes somewhat limited

Tap Line

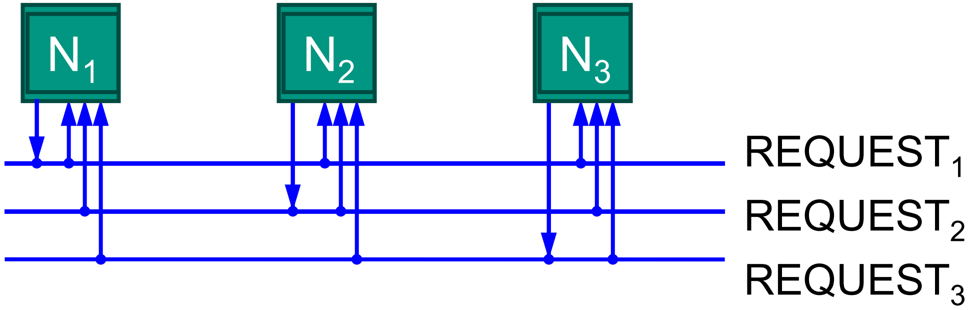
- Example:



- N_1 and N_3 each have a transmit request
- Both activate their *REQUEST*-line
- Arbiter N_0 signals a *GRANT* first to N_1
- N_1 takes over the bus to transmit data and subsequently removes its *REQUEST* signal
- Arbiter signals a *GRANT* to N_3
- N_3 takes control of the bus

Self-Selection

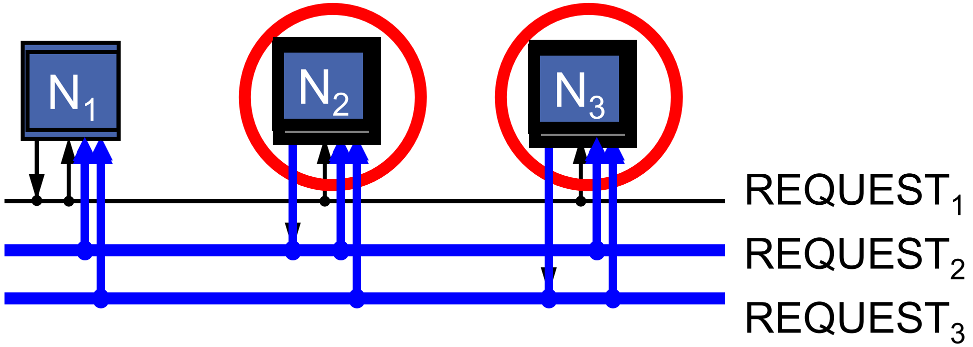
- Decentralized Arbitration by self-selection



- One REQUEST line per node that can be read/detected by all other nodes
- Shared arbitration function is implemented in every node (redundancy)
  - all nodes agree on what node takes control over the bus
- Properties:
  - Fair and priority based scheme possible
  - Fast due to dedicated REQUEST-lines
  - more expensive than daisy-chaining, cheaper than centralized arbitration over tap lines
  - Scalability to an arbitrary number of nodes somewhat limited

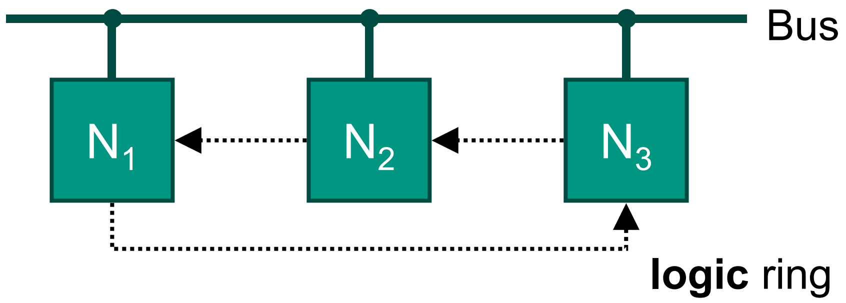
Self-Selection

- Example:



- N_2 and N_3 each have a transmit request
- Both activate their *REQUEST*-line
- All internal arbiters decide to grant the bus to N_3 first
- N_3 takes over the bus to transmit data and subsequently removes its *REQUEST* signal
- Afterwards all nodes have the common understanding that N_2 is granted the bus
- After finishing the data transfer N_2 removes its *REQUEST*-signal from the *REQUEST*-line

Token-Passing



- All nodes are connected to the same (serial) bus
- Every node has a unique address
- Every node has stored the address of its (logic) successor, thus creating a logic ring
- A “free-token” is rotating on the ring (by sending a “Token Packet”)
- A node holding the token is allowed to send data
- Is a node holding the token, but has no data to send, it will pass the token to its successor
- Procedures to inject and remove nodes have to exist

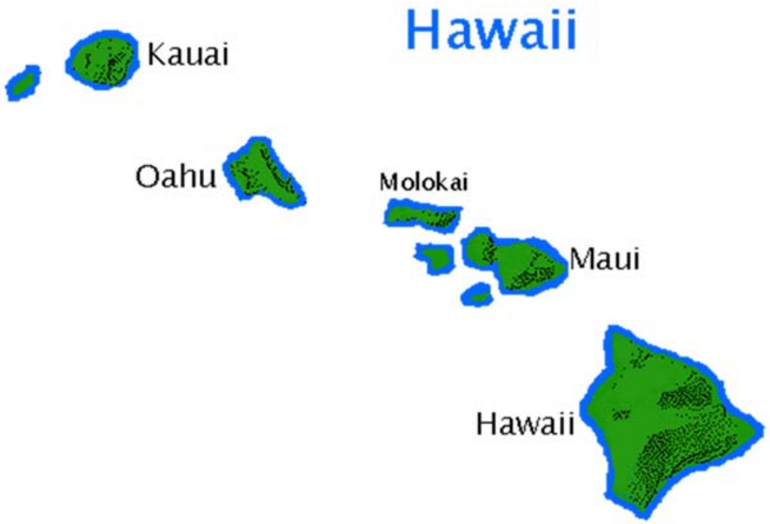
- Properties:
  - flexible, fair, and real-time capable scheme because each node can transmit after a maximal interval of ((`n-1`)·`max_data_length`) time units

Aloha

- Properties: - random bus access
  - not real-time capable (undefined idle time)
- First media access scheme based on a randomness-strategy
  □ first use in a (terrestrial radio) network on Hawaii
- Each node with a transmit request will output data on the bus immediately (without being granted permission and without checking if medium is free)
- Problem: signal distortion if two nodes are sending data simultaneously
  - Detection of conflicts: receiver does not send an acknowledge at the end of a message
  - Correction of conflicts: data packet is resend until an acknowledge is received

  □ large latencies possible

  □ channel utilization very high due to multiple data transmissions and acknowledge



□ Scheme is only adequate if channel utilization is lower than 20%

CSMA (Carrier Sense Multiple Access) I

- Extension to Aloha: before transmitting data the channel is sensed
- Two slightly varying schemes exits:

1. **Non-persistent Scheme**
    - For each Transmit Request the channel is checked to be free:
        - Channel free: transmit immediately
        - Channel occupied: wait for a random time span and retry (this reduces the probability of two nodes starting a transmission simultaneously)
    - Every iteration increases the random waiting time
    - Channel might be unused during the node’s waiting time

CSMA (Carrier Sense Multiple Access) II

- Extension to Aloha: before transmitting data the channel is sensed
- Two slightly varying schemes exits:

2. **P-persistent Scheme**
    - For each Transmit Request the channel is checked to be free:
        - Channel free: send immediately with a probability *p*, delay transmission for a time span *t* with a probability *1-p*

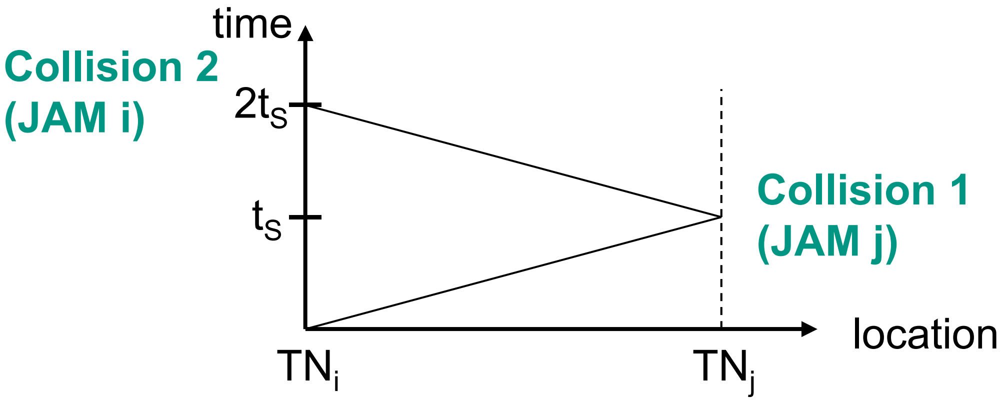
          t is the time for one bit to pass through the communication channel for a node to detect if a second node is trying to send as well
        - Channel occupied: wait until channel is free and restart from the beginning

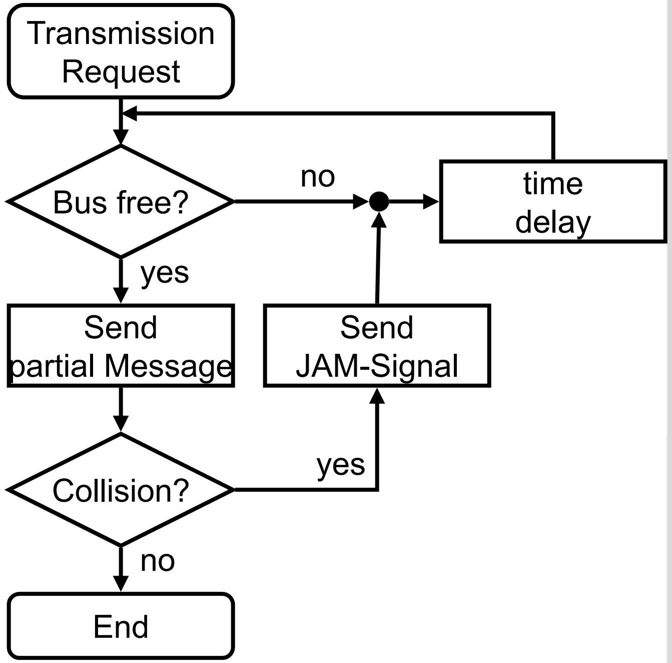
- Disadvantages of both schemes (as with Aloha):
    - No direct collision detection, collisions can only be identified through a missing acknowledge (only after a full data packet has been sent).

CSMA/CD (CD = Collision Detect)

- Direct detection of collisions
- Sender is always reading the channel and checks if the signal sent is identical to the one being read.
  - Fast detection if a collision has occurred
- Sender detects a collision:
  - Transmission of a JAM signal, transmission is ceased
  - Node that is transmitting simultaneously is detecting the JAM signal and ceases its transmission as well
- Receivers will discard data as soon as JAM signal occurs

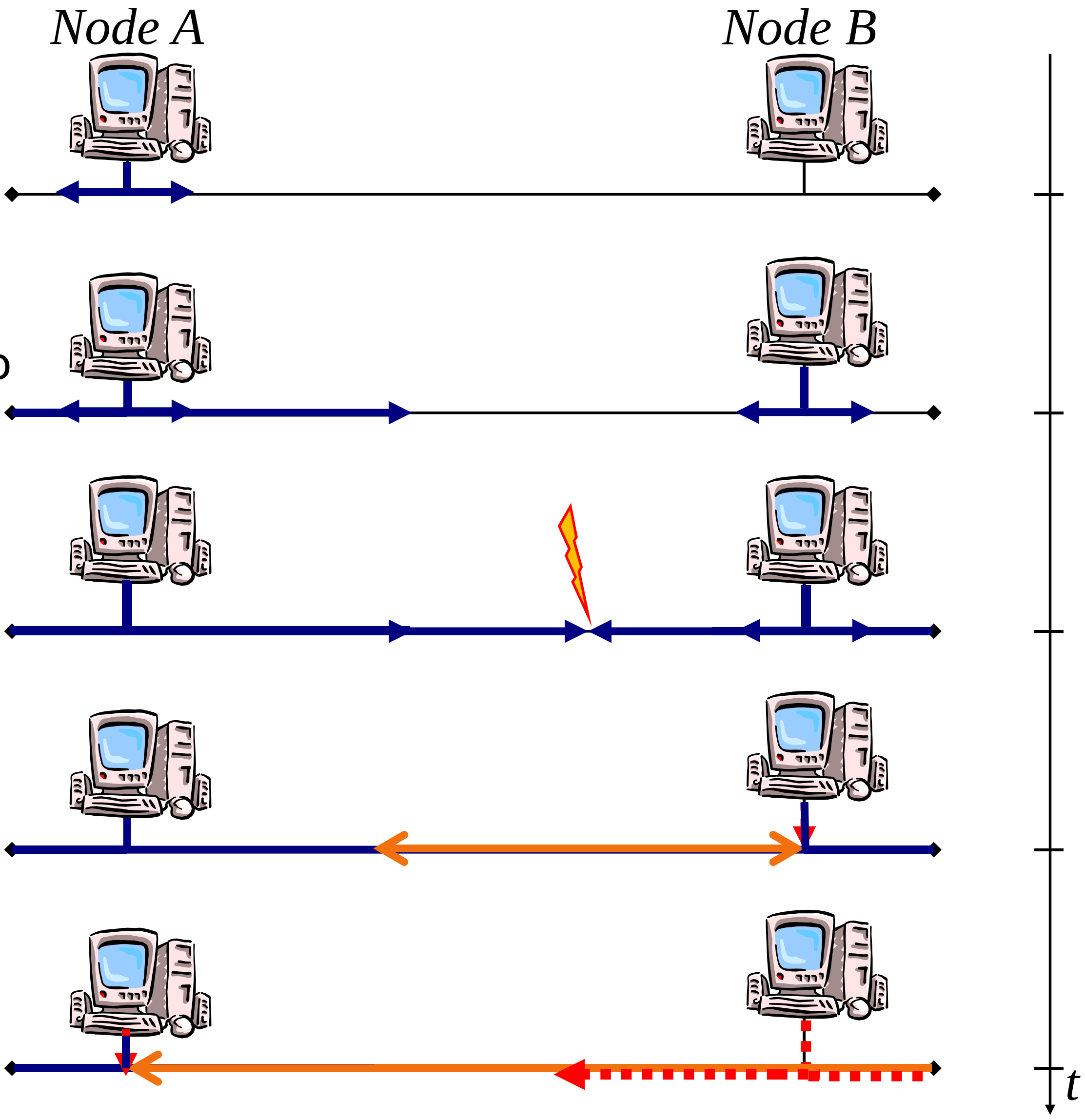
- Collision is detected after a maximal propagation time t_S
  - Minimal packet length: 2 t_S, maximum wire length must be defined.





Exemplary Sequence of CSMA/CD

- Node A starts to send, because Medium is free.
- Node B starts to send, because Medium seems to be free.
- A collision of data packets occurs.
- Node B detects the collision, ceases data transmission and sends JAM-signal.
- Node A detects collision and also sends a JAM-signal.



Properties of CSMA/CD

- Data destruction possible
- Need to discard already sent data after collision has occurred
- Bad channel utilization
  - Rule of thumb: 30%-70%
- No guaranteed Real-Time Capability

CSMA/CA (CA = Collision Avoidance)

- Avoidance of collisions by priority controlled bus arbitration.
- Every node is assigned an identifier (ID) that equals its priority.
- After completing a transmission on the bus, all nodes with a transmission request start to send their ID. All nodes are connected via wired-OR or via wired-AND respectively.
  - wired-OR: „1“ dominant, „0“ recessive
  - wired-AND: „0“ dominant, „1“ recessive
- A transmission starts with the most significant bit (MSB).
  - Each sender monitors the bus level during each bit being send
  - As soon as the bit currently being read from the bus is not identical with the bit send by the node, the node retreats and retries the transmission later.

Simultaneity

- **Problem:** It is necessary for all bus nodes in the cluster to read the bit being send (independent of the distance to the sending node), before (!) a new bit is put onto the bus.

- **Resulting requirement:** signal propagation time t_S is neglectably small compared to the digit length (bit time) t_B:

$$\left[t_s = \frac{l}{v}\right] << \left[t_B = \frac{1}{TR}\right]$$

$l = \text{length of the bus line}$

$v = \text{propagation velocity}$

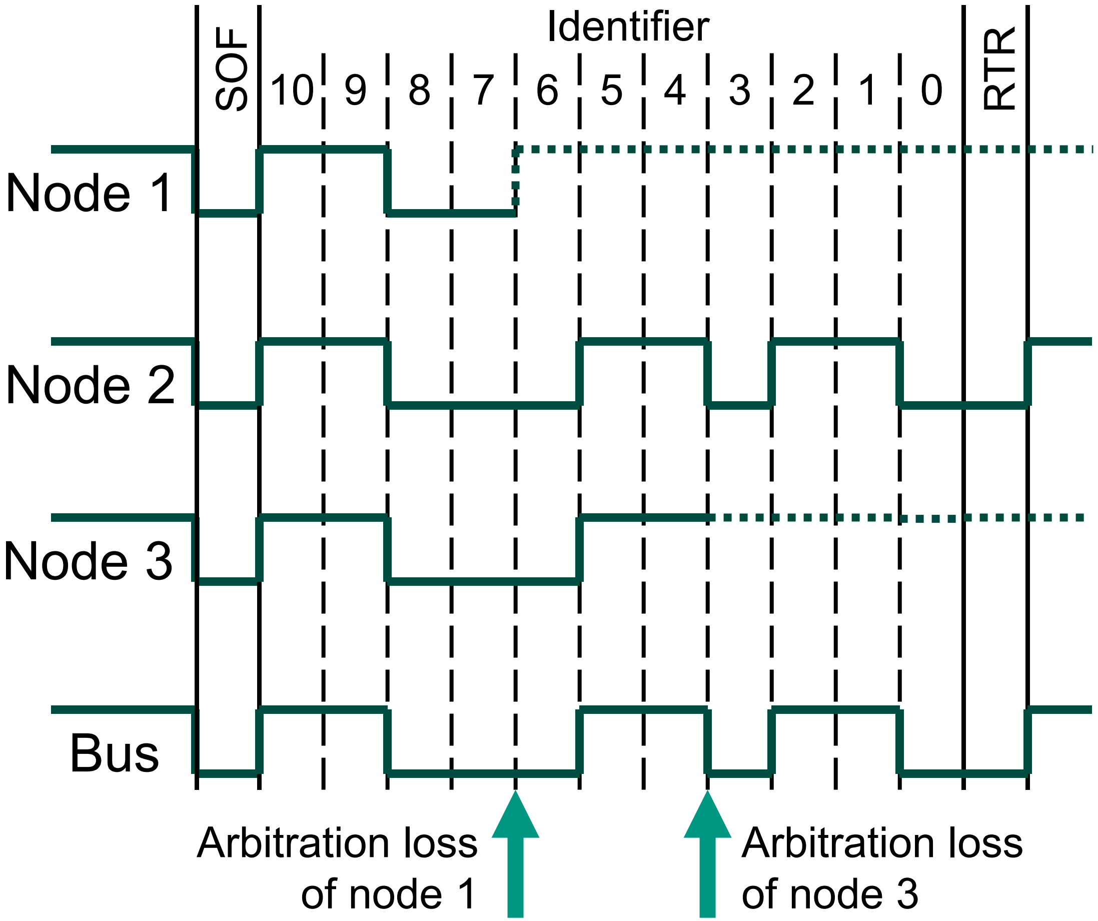
$TR = \text{transmission rate}$

Properties of CSMA/CA

- No data destruction
- No need to discard already sent data
- 100% channel utilization is possible
- Limited length of bus and/or transmission rate
  - due to simultaneity
- Real-Time Capability
  - If the packet length is finite, the node with the highest priority can adhere to real-time constraints.
  - Bus can be blocked if node with highest priority is constantly transmitting.
    - In general each node has to wait after a transmission for a predefined time before transmitting a new message.
  - Other nodes can adhere to real-time constraints as well if waiting time is long enough.

Example: CSMA/CA (CAN-Bus)

- Concurrent bus access of 3 bus nodes, dominant „0“

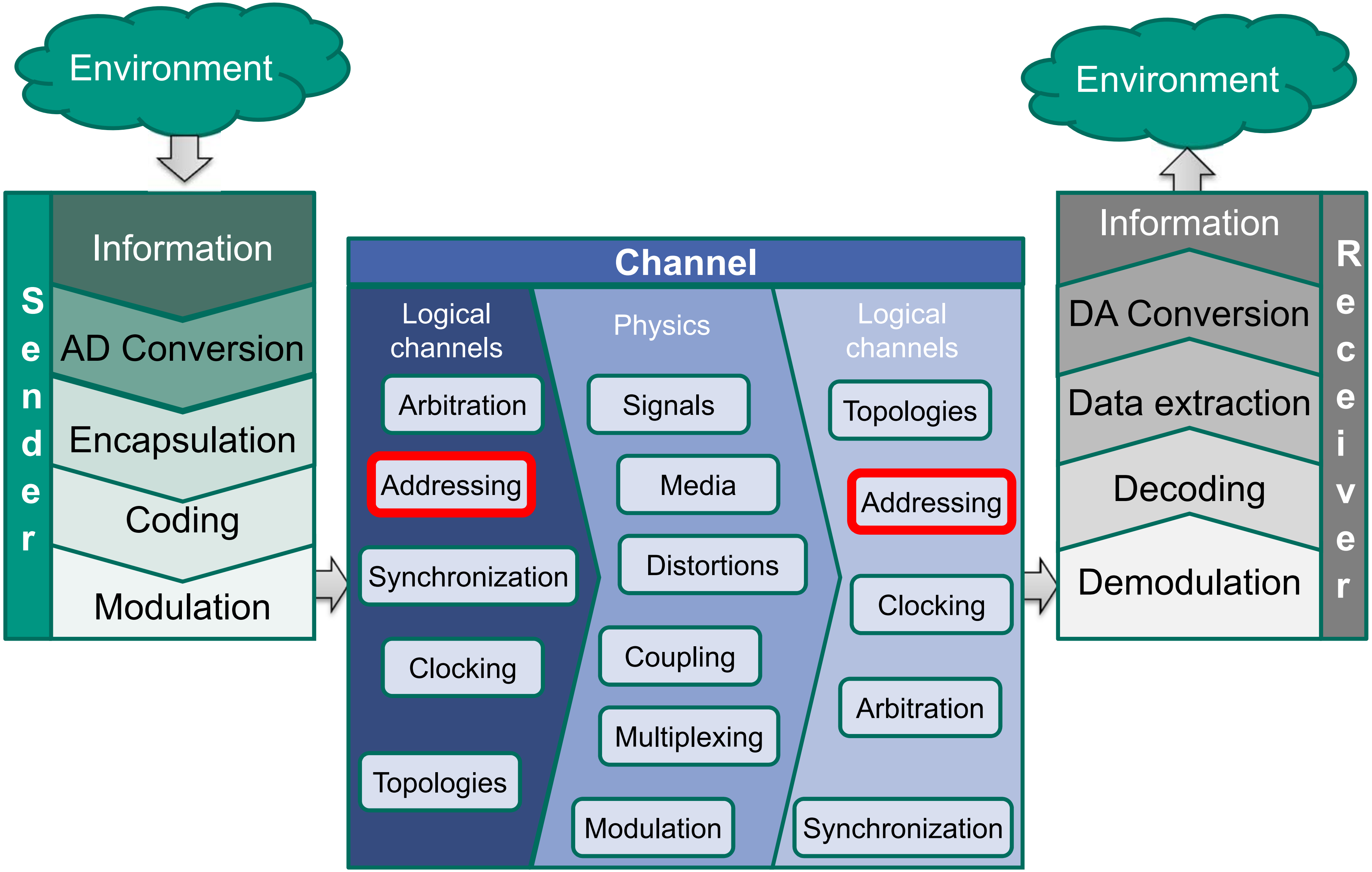


Node 2 is granted bus access sends its complete message

Clicker Session: Arbitration

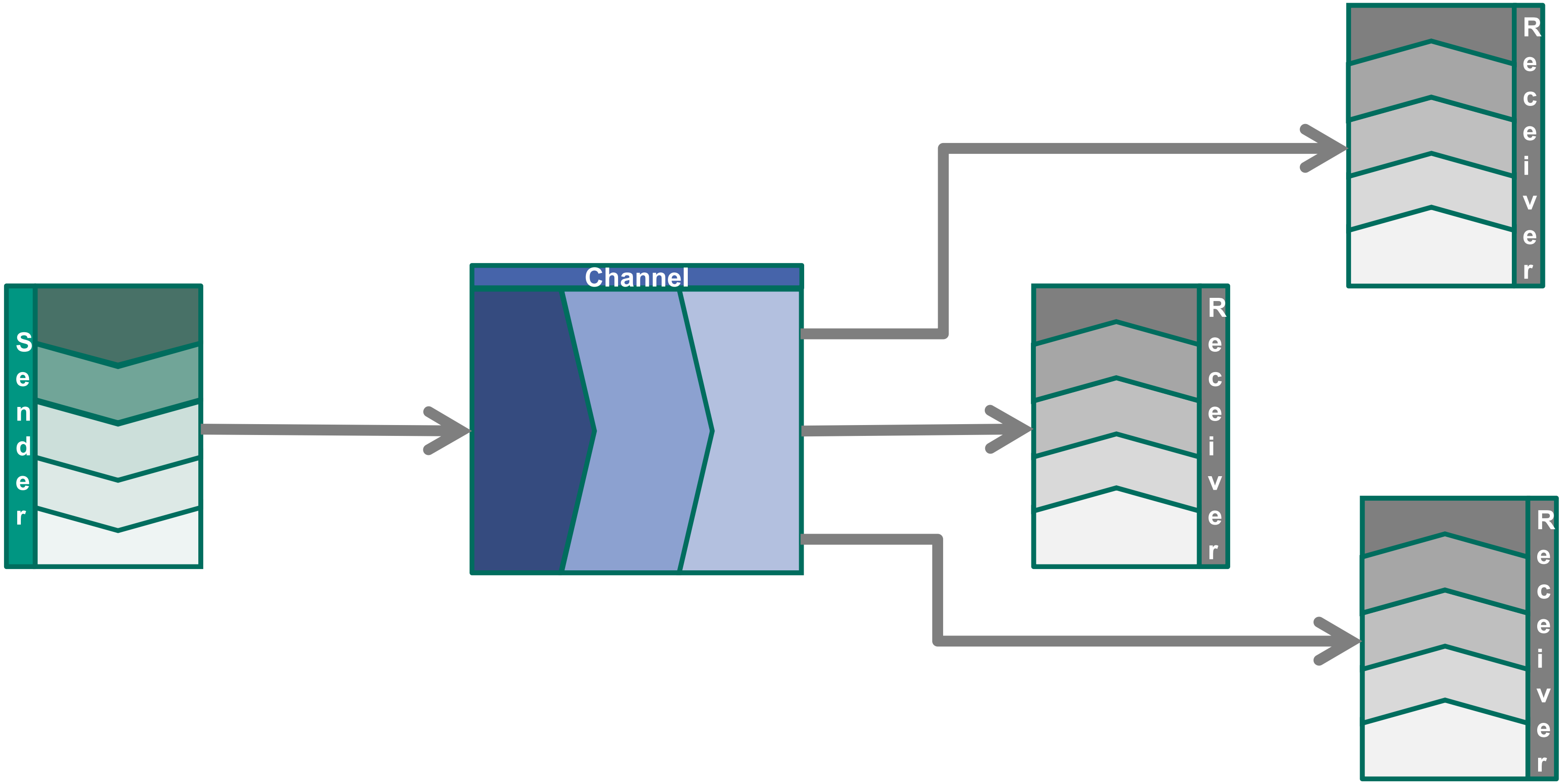
- https://arsnova.eu/mobile/#id/33969518

Transmission System – Addressing



Multiple Receivers : Addressing

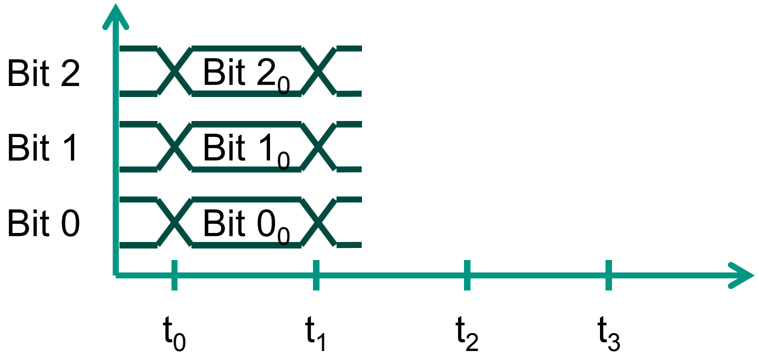
- Multiple receivers can be listening at the same channel
- It has to be distinguished which receiver should receive the actual transmission Addressing



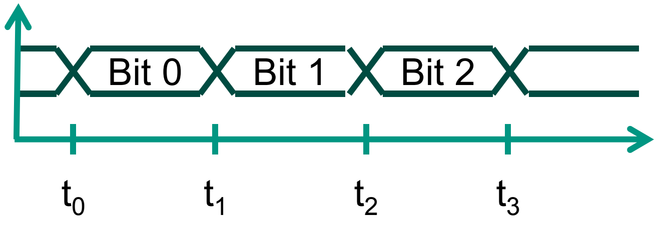
Node Addressing

- Every receiver in the communication system is assigned an individual and unique Identifier (ID, address)

- A transmission is initiated by sending the receiver’s address via
  - Additional signal lines (address lines)

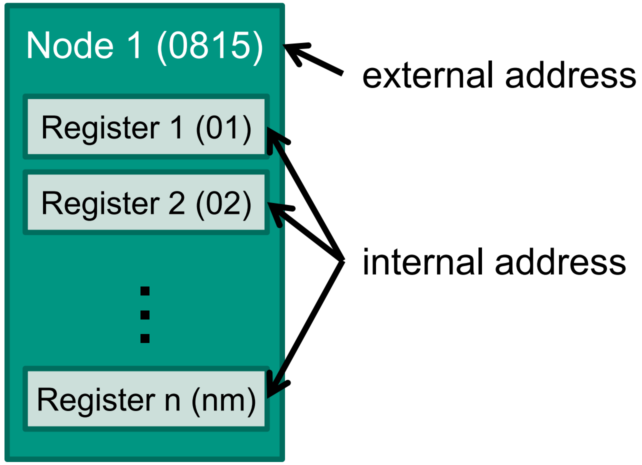


  - Sending the receivers address before sending the data if only one data line is available



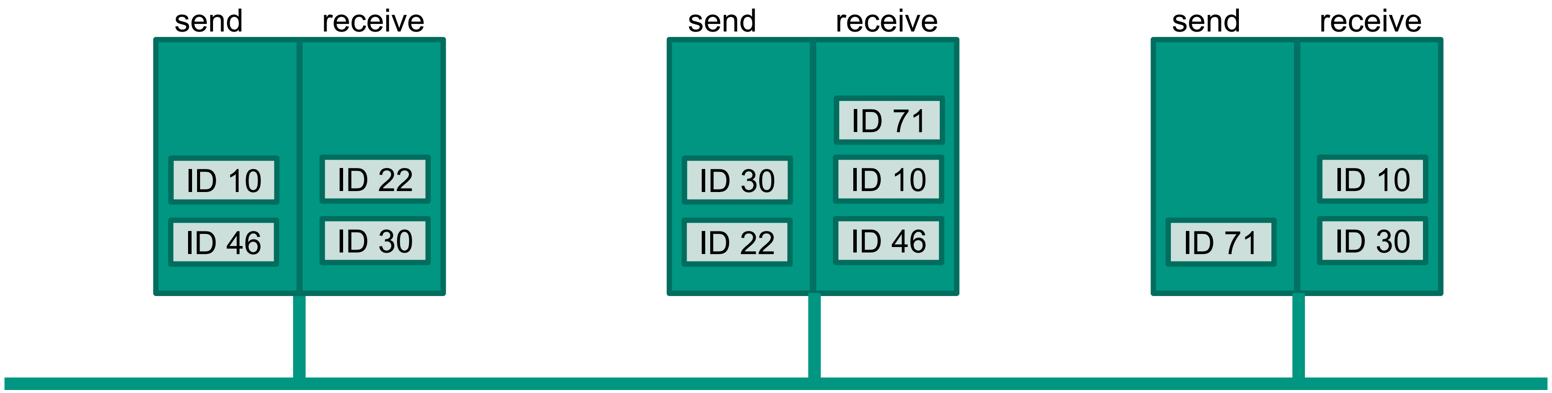
Internal Node Addressing

- There can also be additional internal addresses within the receivers that are also transmitted over the communication system
  - External Address: Identify the receiver
  - Internal Address: Identify receiver-internal memory or registers



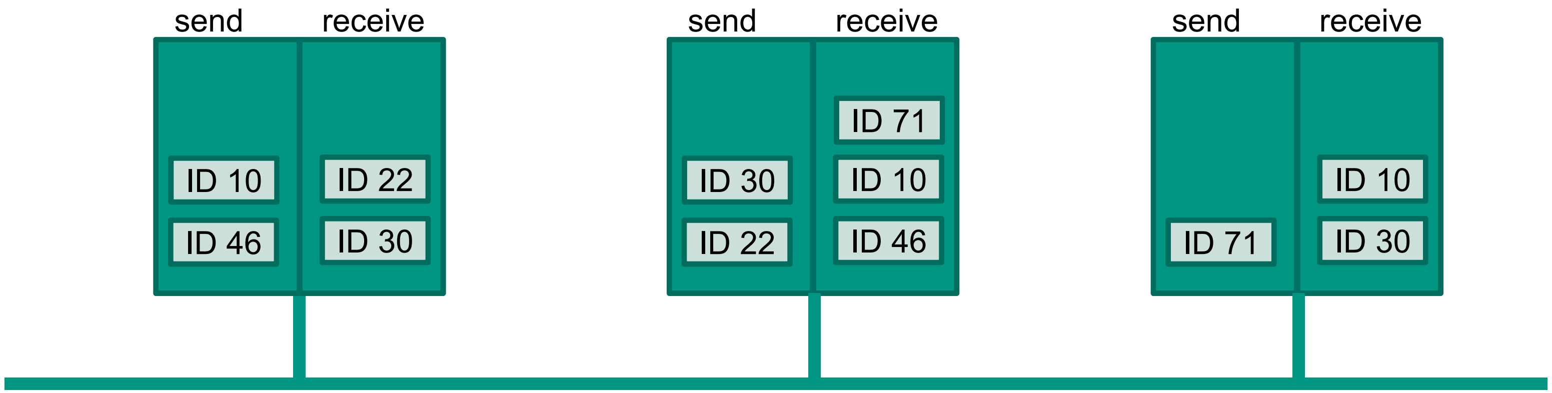
Message based addressing I

- The address is assigned to individual messages with defined content
  - E.g. temperature reading of a sensor, speed of a car
- Message ID is unique in a communication system
  - Each message can only be send by one single sender
  - Each sender can send an arbitrary number of different messages



Message based addressing II

- Every receiver can decide upon this message ID if the message is of interest to him or not
- No dedicated addressing of a receiver required/possible
  - Everyone in the communication system shares the same information □ Broadcast



send receive
ID 10 ID 22
ID 46 ID 30

send receive
ID 71
ID 30 ID 10
ID 22 ID 46

send receive
ID 10
ID 71 ID 30

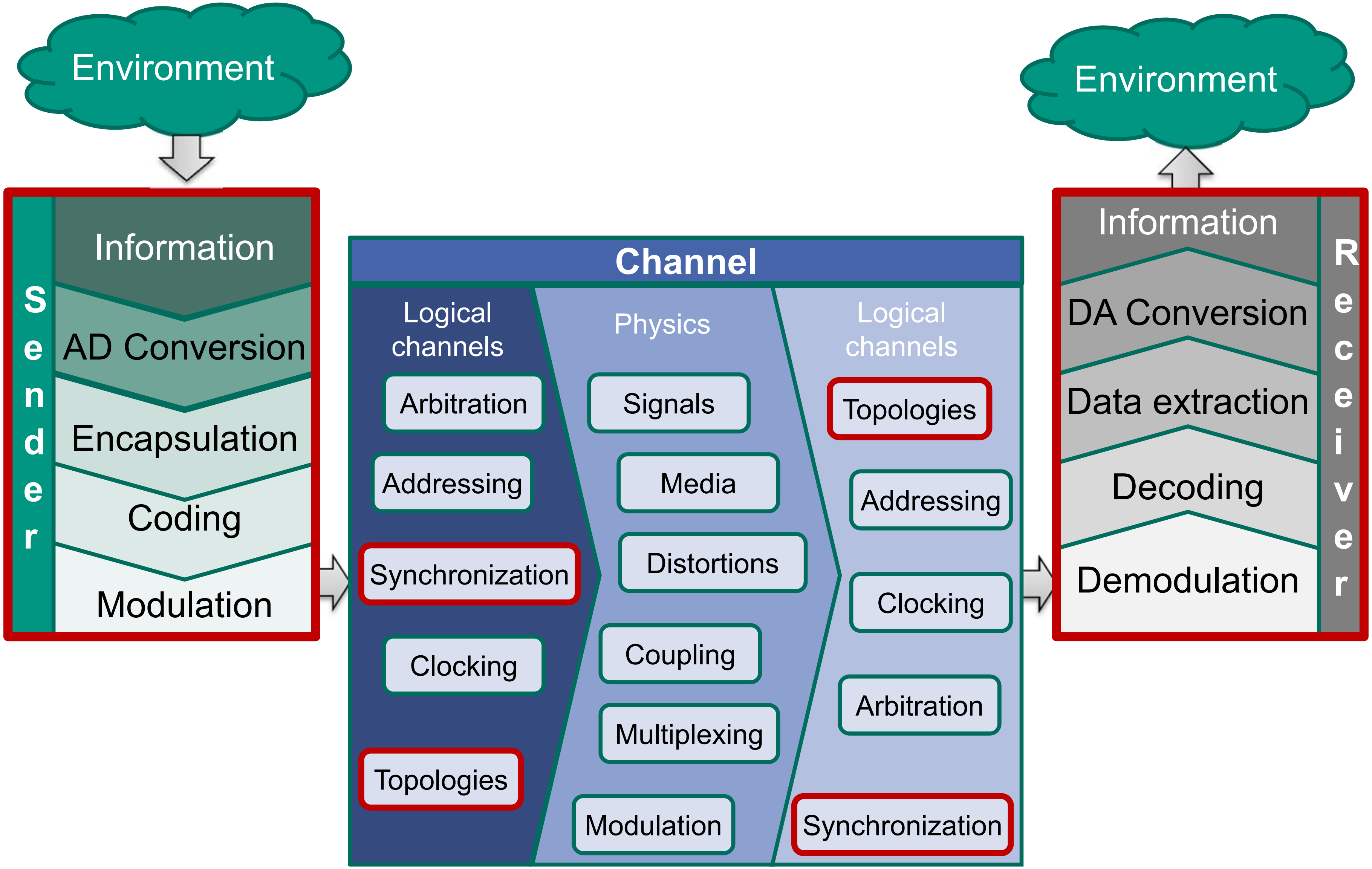
- Example: CAN

Possible setups for Communication Systems

- 1:1 Direct communication between two partners
  - Examples: RS-232, Telephone
- 1:n Only one sender, multiple Receivers
  - Examples: Radio, Television
- m:1 Multiple senders but only one receiver
  - Examples: Network printer
- m:n Multiple Senders, multiple Receivers
  - Examples: Telephone Conference

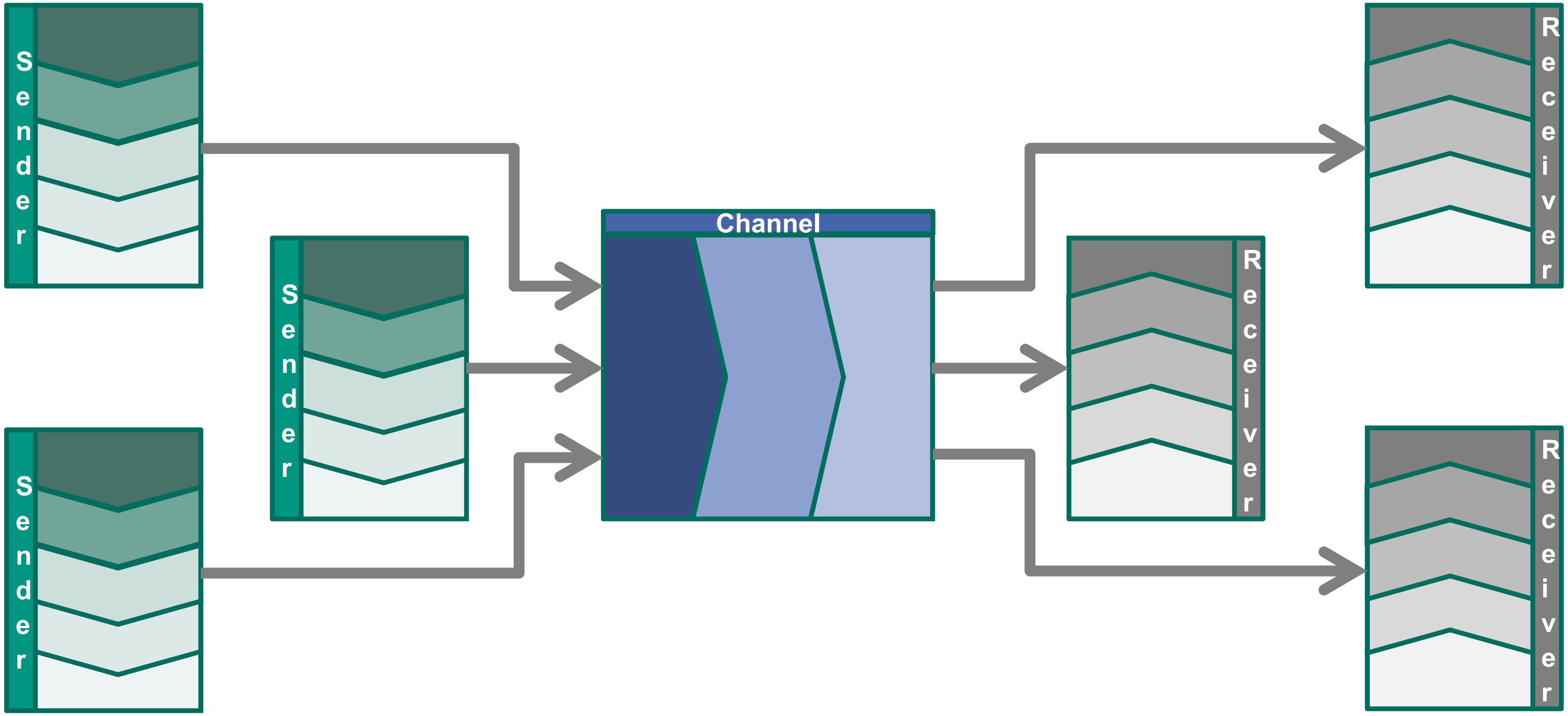
m: Number of senders
n: Number of receivers

Transmission System



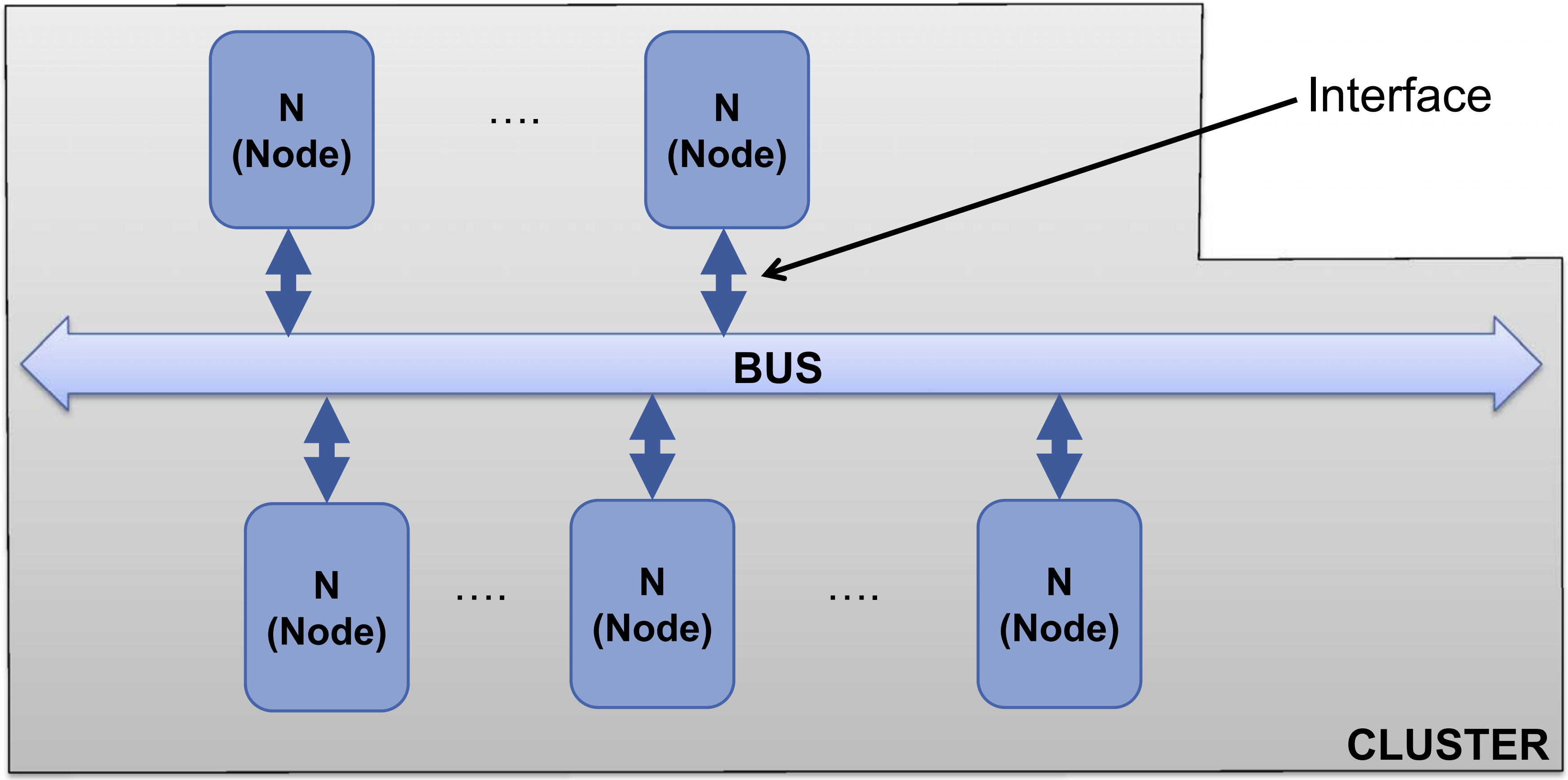
Bus System

- Multiple Senders and receivers share the same channel for communication
  - Arbitration and Addressing required



Simplified Overview

- Multiple Transmission Nodes (N) connected over the same wire bundle.



Interface

- Definition: Interface

  a : the place at which independent and often unrelated systems meet and act on or communicate with each other
  b : the means by which interaction or communication is achieved at an interface

  (Merriam-Webster dictionary)

- Pragmatic definition:
  - Defined data transmission connection between exactly two entities
  - Required Specifications:
    - Mechanical coupling
    - Electrical signals and timing
    - logic signals (coding, temporal sequence)
    - A complete definition is not always given!

Bus

- Definition: Bus

  A set of parallel conductors in a computer system that forms a main transmission path (Merriam-Webster dictionary)

  (1) A collection of wires through which data is transmitted from one part of a computer to another. […] When used in reference to personal computers, the term *bus* usually refers to *internal bus*. This is a bus that connects all the internal computer components to the CPU and main memory.

  All buses consist of two parts -- an address bus and a data bus. The data bus transfers actual data whereas the address bus transfers information about where the data should go. (webopedia)

- Remarks:
  - Bus Systems include the bus itself plus other entities (nodes, arbiter, power supply, API, etc…)
  - Nodes are connected to the bus over defined interfaces
    □ Interface definition is part of the bus definition
  - Current Sender can transmit to one ore more receiving nodes (□ Broadcast)
  - Bus management is necessary (e.g. to prohibit two nodes sending simultaneously)

Cluster & Node

- Definition: Cluster

  (1): a number of similar things that occur together: as

  (Merriam-Webster dictionary)

  (2): a group of loosely coupled computers that work together closely

  (webopedia)

- Pragmatic Definition:
  - A bus with a set of nodes connected over it.
  - Within the cluster it is possible to transmit information

- Definition: Node

  (1): : a point at which subsidiary parts originate or center

  (Merriam-Webster dictionary)

  (2): a centering point of component parts.

  (Dictionary.com)

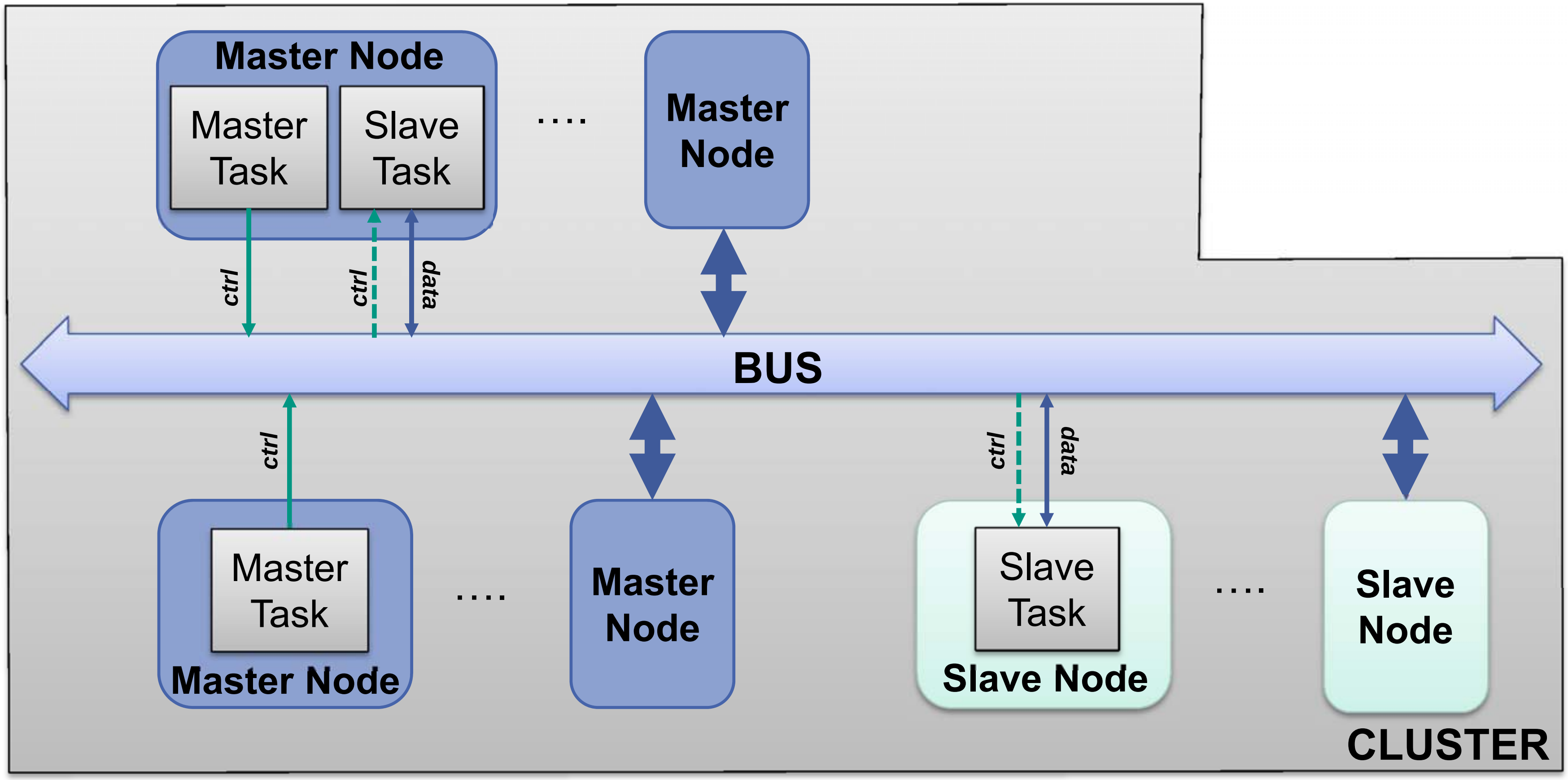
- Pragmatic Definition:
  - A terminal in a communication system

Bus Nodes

- Master Task
  - active entity
  - controls the bus
  - triggers communication
  - arbitration
- Slave Task
  - passive entity
  - is controlled by master
  - reads data off the bus
  - puts data on the bus, when triggered by a master task
- Master Task / Slave Task in the same Node (package) e.g. LIN Bus
- Possible Bus Types
  - Single Master / Multi Slave
  - Multi Master / Multi Slave

General Bus Structure (1)

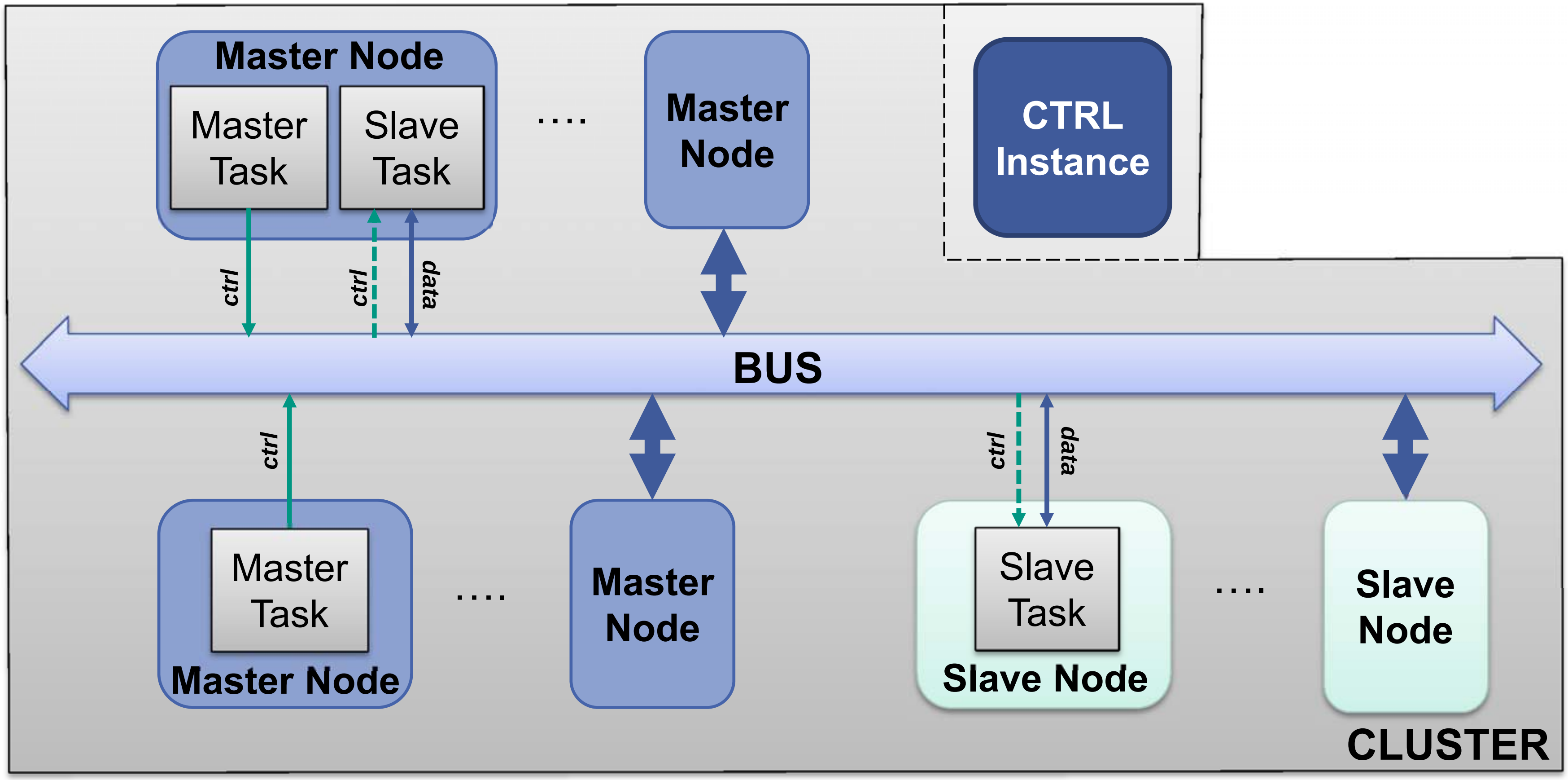
- Multiple Nodes connected over the same wire bundle
- One wire bundle is allotted to more than a single node



- Address of a node: coded with ld *n* control signals

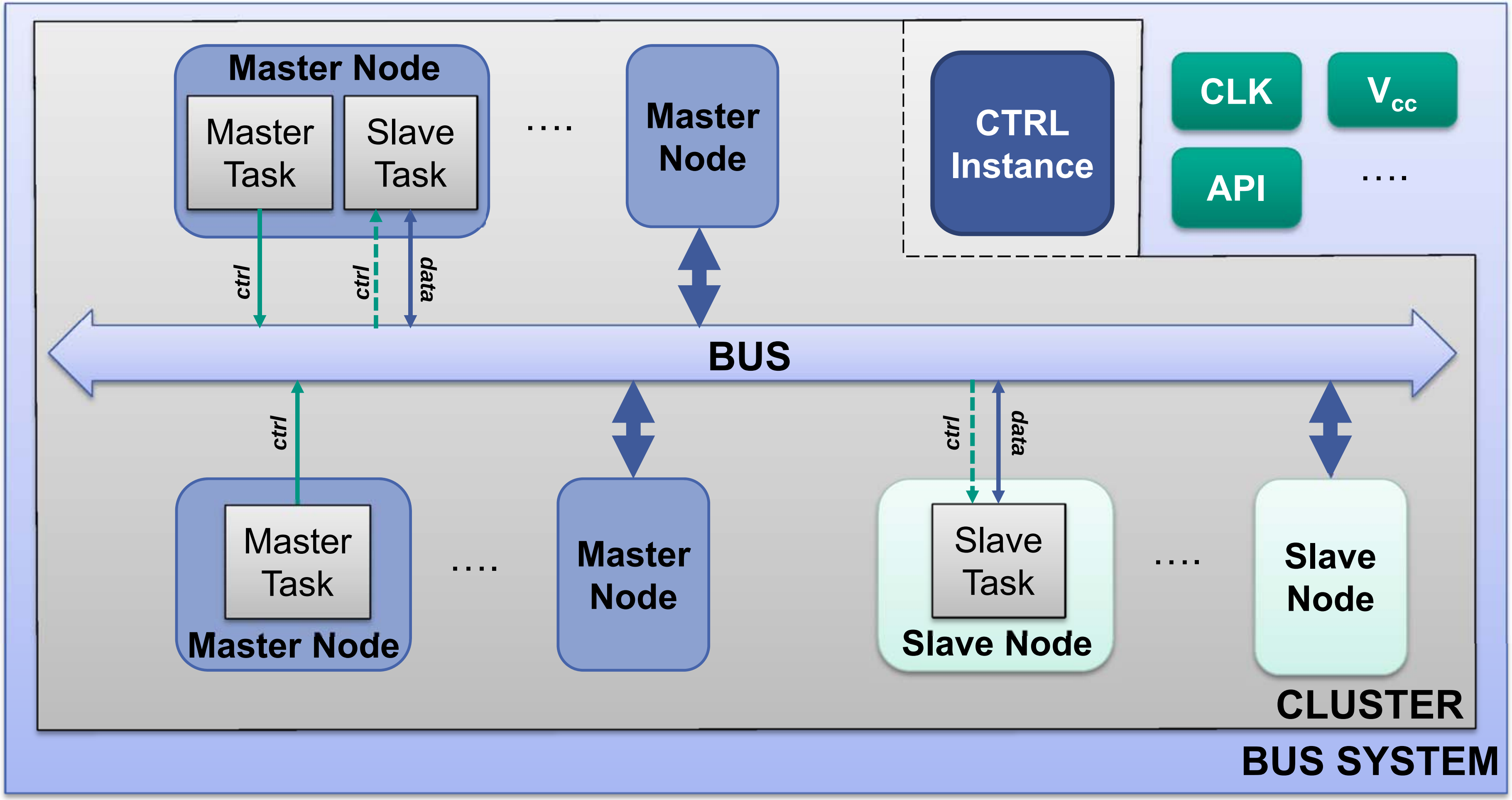
General Bus Structure (2)

- Multi-Master Busses require control instances (e.g. Arbiter) to regulate bus access
- Can be a node within the bus or an external controller

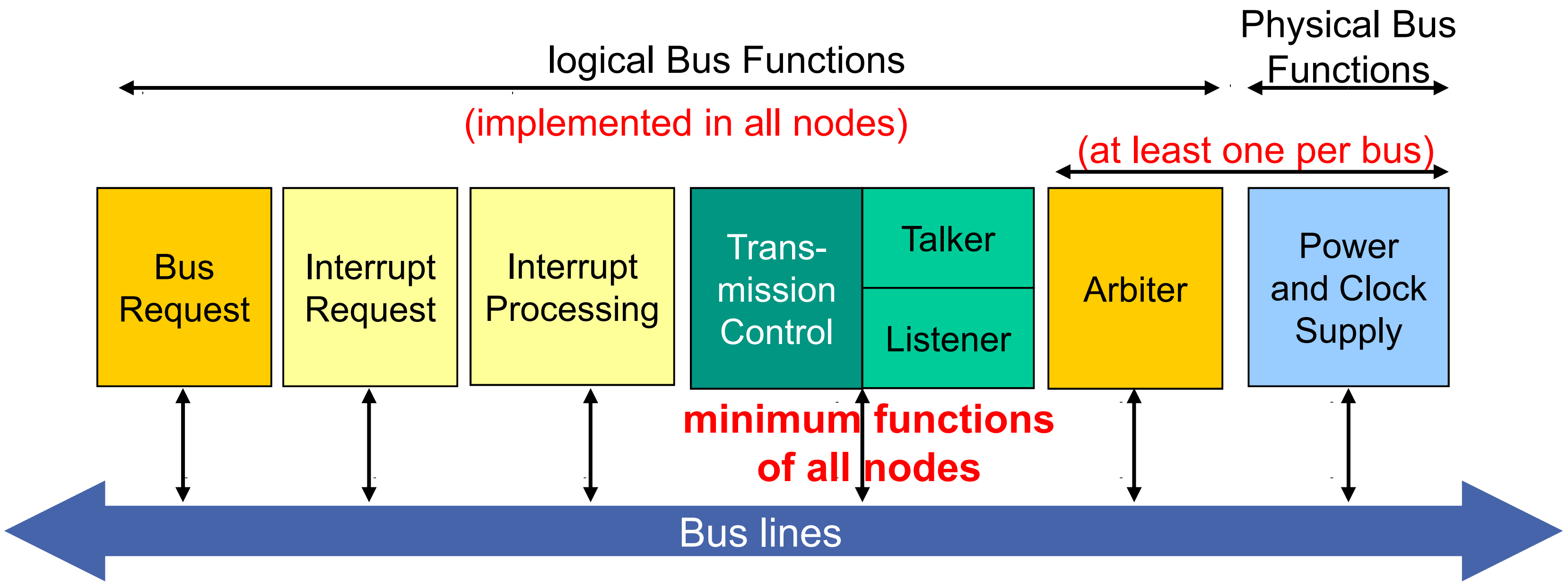


General Bus Structure (3)

- Additional components (clock, power supply, Application interfaces, … form the **bus system**.



Functional Setup of a Node



Bus Request:	Node wants to become bus master
Bus Arbitration:	Processing of Bus Request(s)
Interrupt Request:	Forwarding of a node's interrupt
Interrupt Processing:	Reception and processing of an interrupt
Transmission Control:	Control of the actual data transmission either as master or slave. A node can send data (Talker) or receive data (Listener)

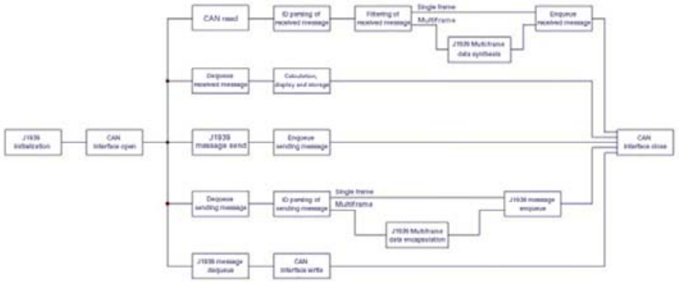
Description of a Bus System

- Technical parameters (on Technology Level)
  - Signals
  - Signal Lines
  - Levels
  - Interconnects (i.e. Plugtype, Footprints)

Source: http://www.etas.com

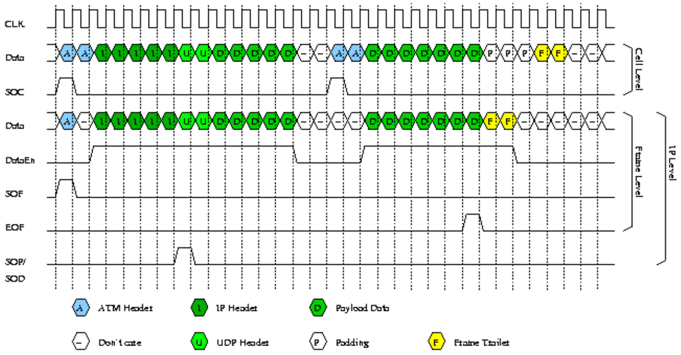
Source: http://www. leib-engineering.de

- Protocol
  - behavioral description
  - what happens, when does it happen
  - timely behavior



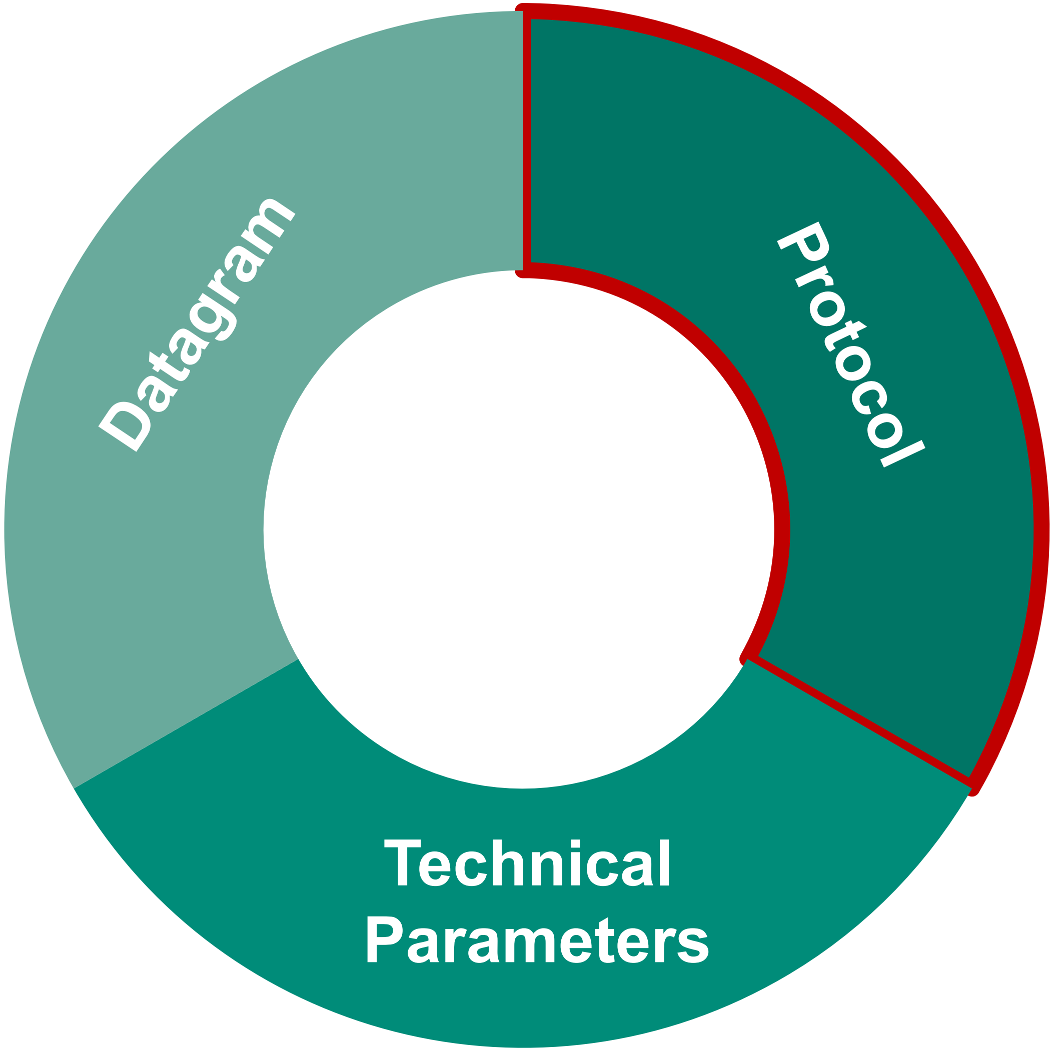
Source: http://sine.ni.com

- Datagram
  - Interpretation of signals (lines, bits, positions of bits)
  - Representation of Information



□ All of the above is forming the specification

Parts of a Specification



Protocols

Definition:

A protocol is a set of rules for communication amongst a set of communication partners.

- Requirements:
  - Completeness
  - Uniqueness / Unambiguousness
  - Transparency
  - Documentable
  - Adaptability
- A defined and eventually standardized protocol is the basis for save and reliable (and sometimes legally binding) functionality of open systems.

Protocol Specification

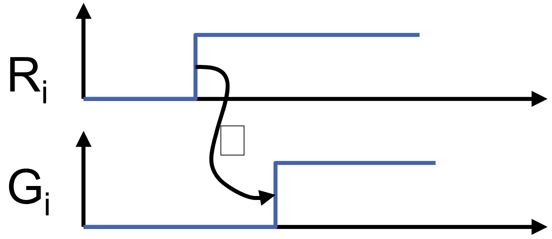
- Verbal representation
  „.. TN_i would like to be master and asserts signal R_i with 1. Subsequently the arbiter grants access to the bus G_i if priority of i is currently the highest...“

- Problems:
  - Representation is not unique
  - unclear, sometimes even confusing
  - Interdependencies hard to grasp

- □ Representation in a formulized procedure is useful

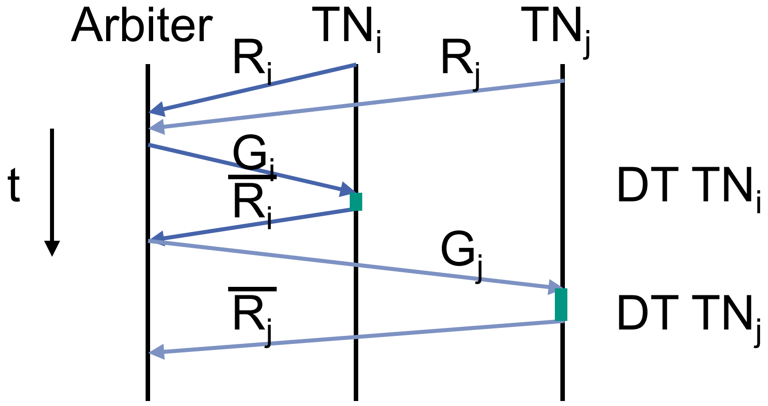
Protocol Specification

- Pulse diagram
  - Representation of temporal characteristics of signals
  - Cause and effects are marked with additional arrows



□If P_i maximal

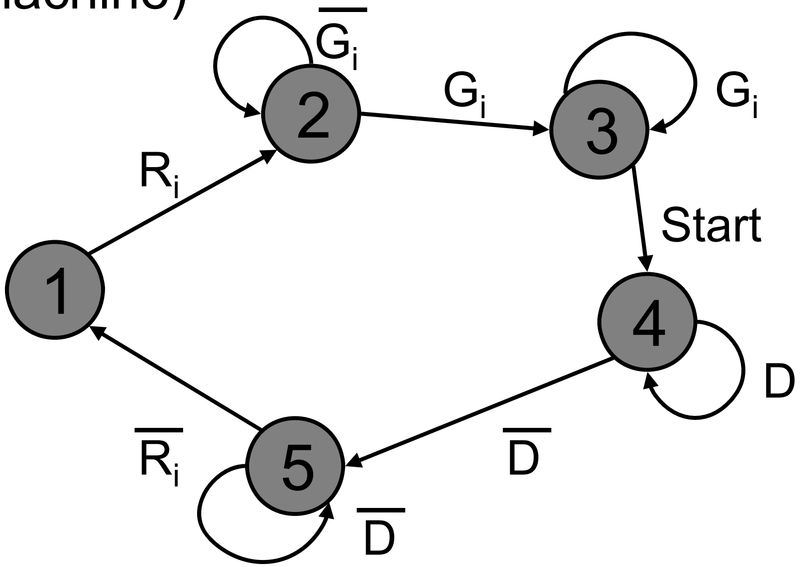
- Sequence diagram
  - Communication partners are marked with vertical lines
  - Exchange of messages are marked with descending lines



TN_i: Node i
DT: Data Transmission

Protocol Specification

- State Transition Diagram (finite state machine)
  - Graphical representation
  - Easy to read and understand

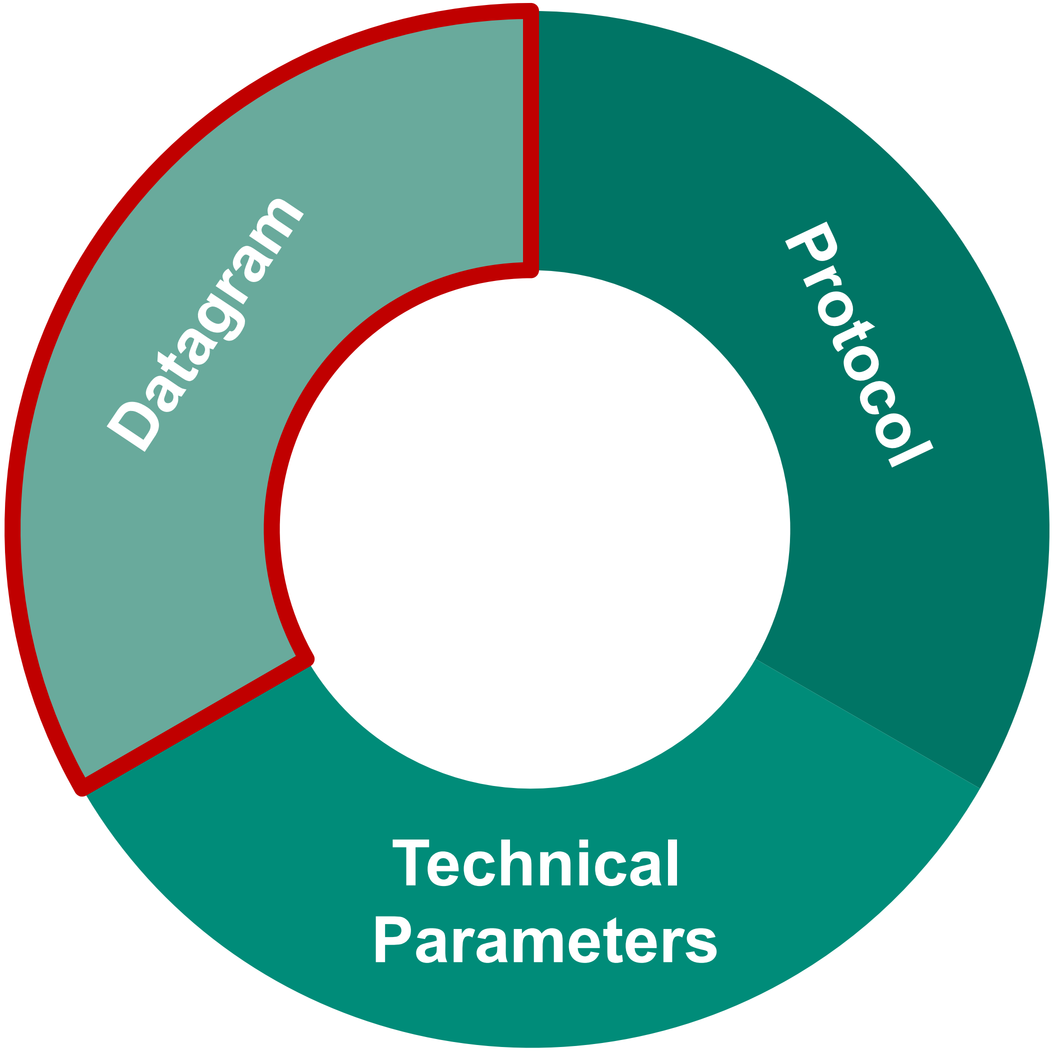


- Syntax notation in BNF-Notation (Backus-Naur-Form)

  Example: ARINC Protokoll

```
<language> ::= <prologue><frame_table><epilogue>
<prologue> ::= <gap_spec><delta_spec><ver_spec><other_spec>
<gap_spec> ::= GAP <digit><eol>
...
<binary> ::= <bit_value>[<binary>]
<bit_value> ::= 0|1
```

Parts of a Specification

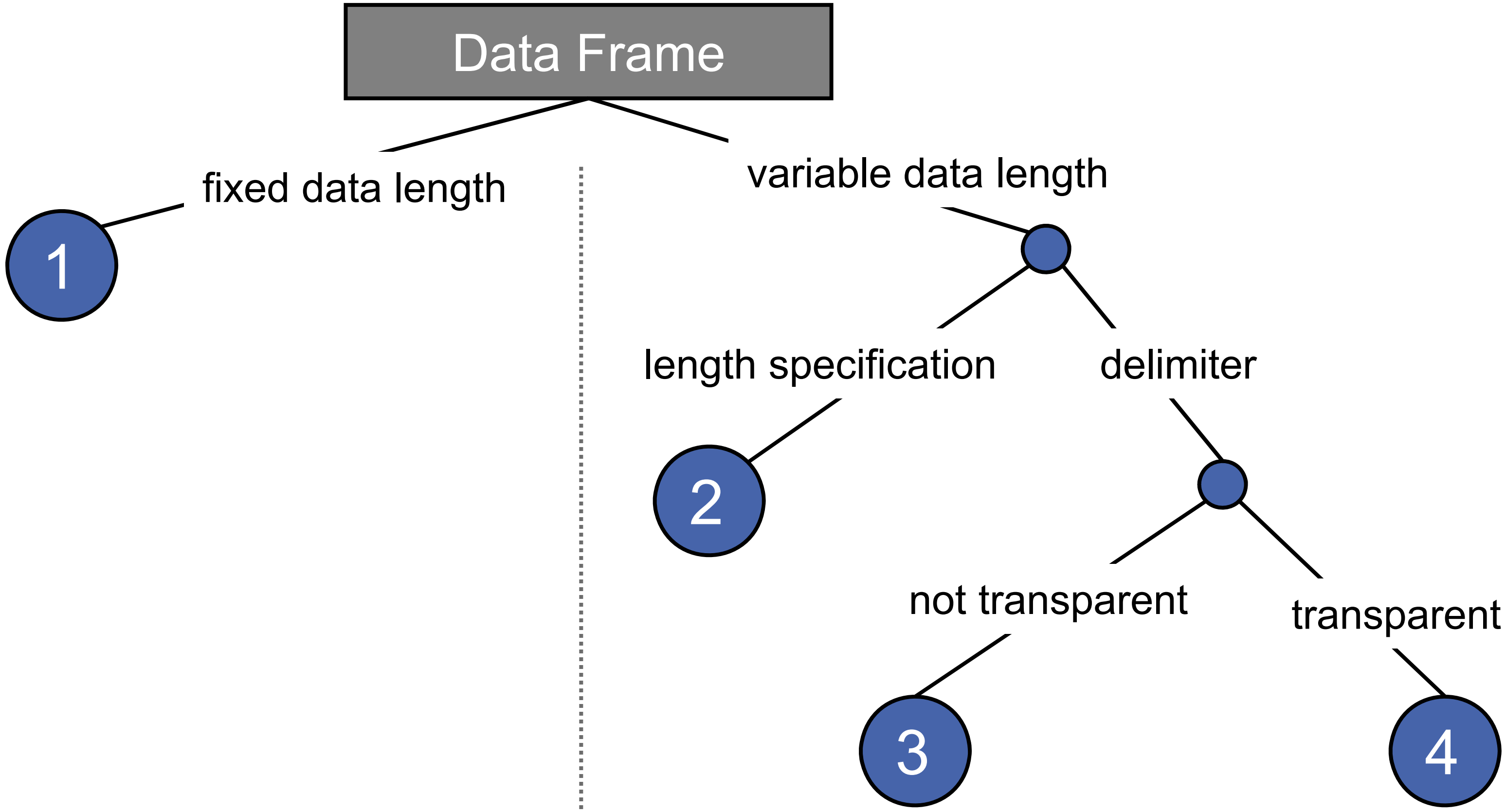


Dataframe - General Structure

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

- Dataframe is sometimes denoted as packet

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

Start	7/8 Bit	PTY	1-2 Stop

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

SOF	ID	RTR	ctrl field	data field	CRC	CRC del.	ACK Slot	ACK del.	EOF

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)

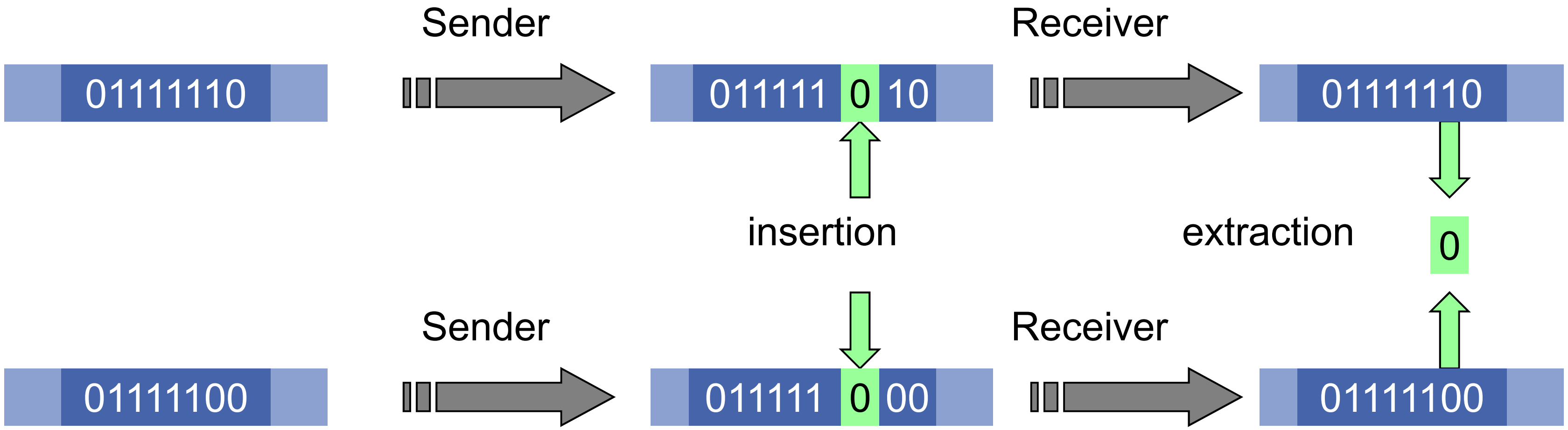
	DLE	STX	Datafield variable	DLE	ETX	

Bit Stuffing (Transparency)

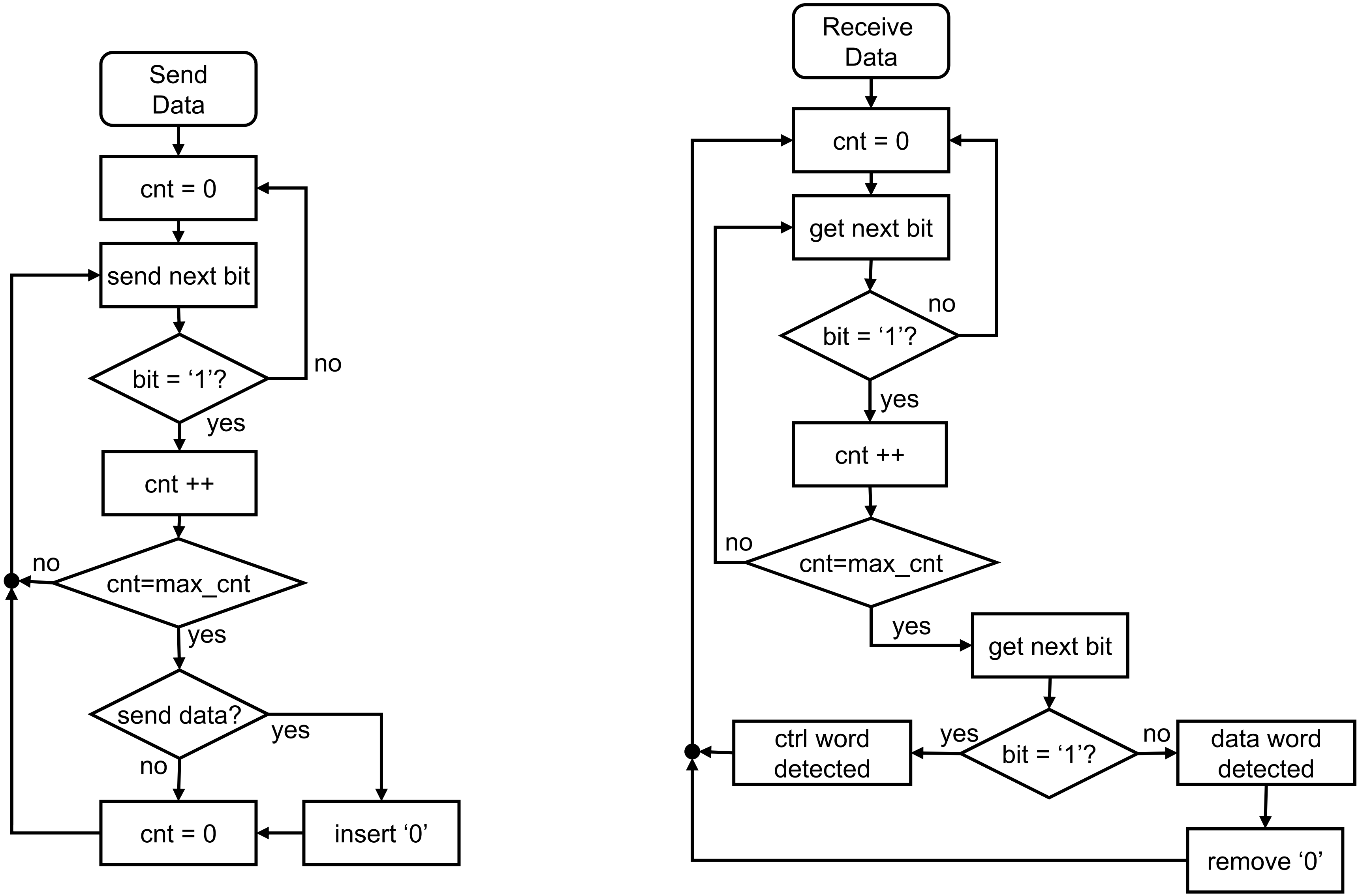
- Bit Stuffing
  - A sequence of *n* ‚0‘s or ‚1‘s is defined as control symbol for Data Start/End. This must not appear within the data stream.
  - Insertion of a ‚1‘ after *n-1* ‚0‘s and vice versa

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.

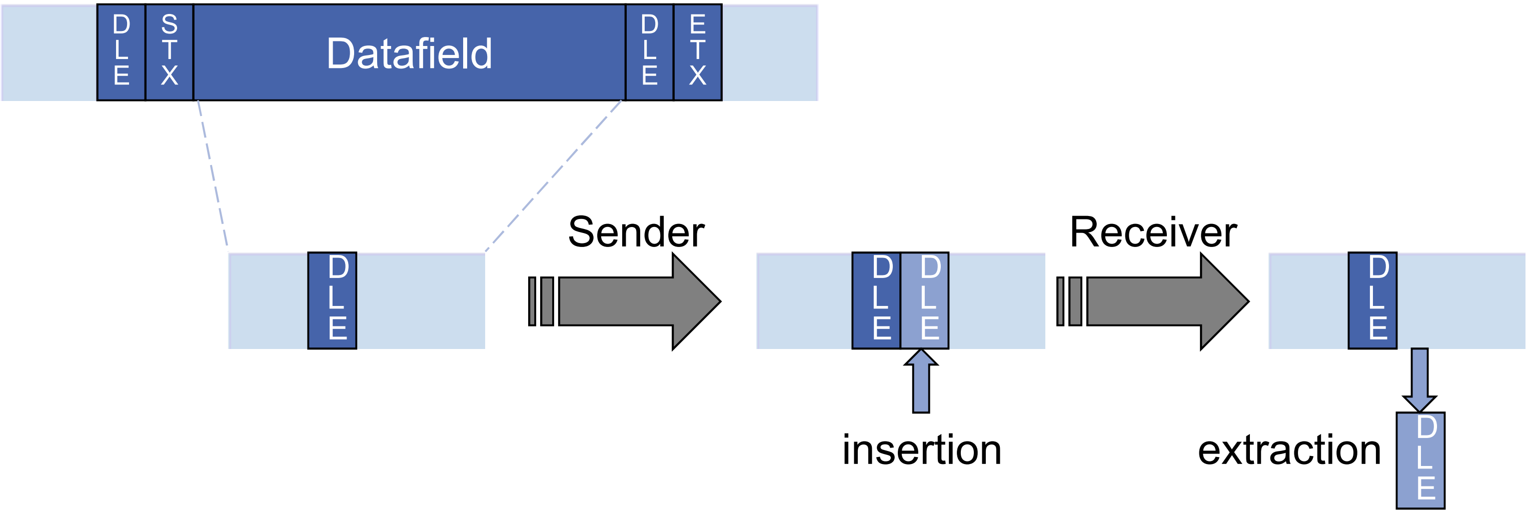


Bit Stuffing – Algorithm (Example)

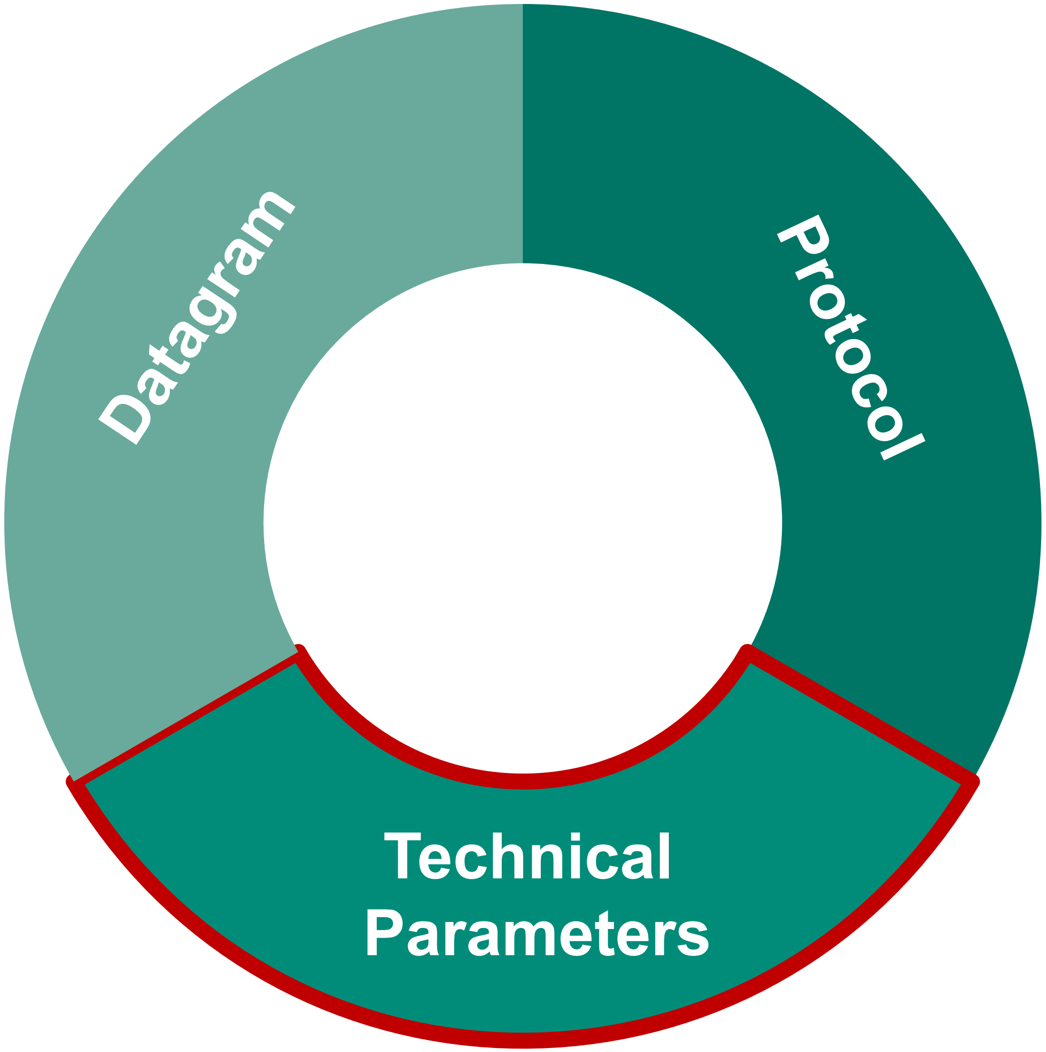


Symbol Stuffing (Transparency)

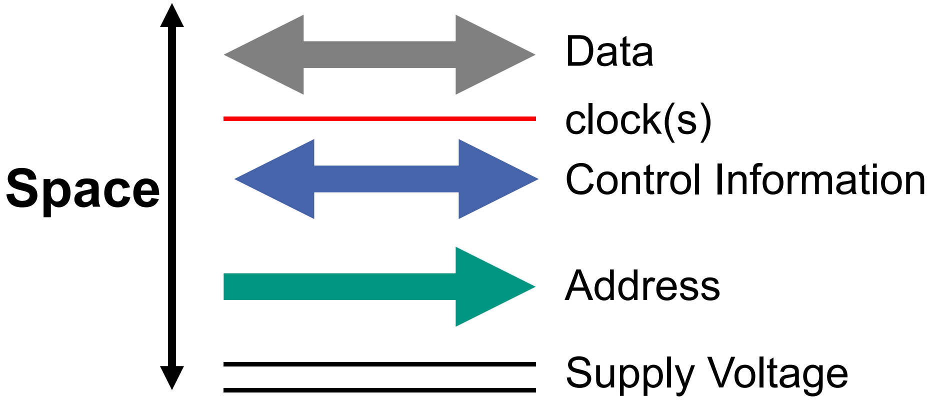
- Symbol Stuffing
  - When using frame formats with variable data field lengths, a symbol is necessary that marks the start and end of a data field:
    Data Link Escape (DLE)
  - This symbol can also be a regular symbol to be transmitted
    - □ Masking necessary
  - If DLE is part of the data field,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



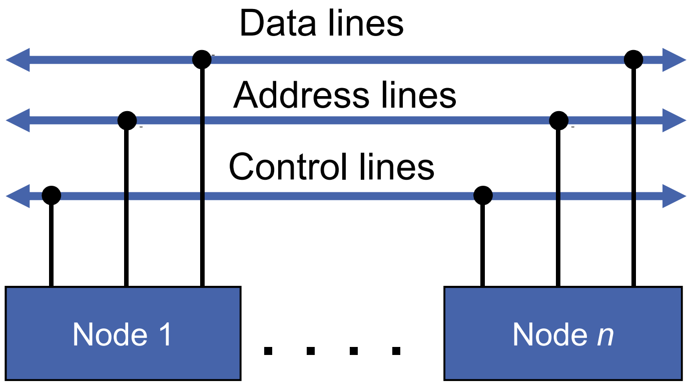
Parts of a Specification



Parallel Bus



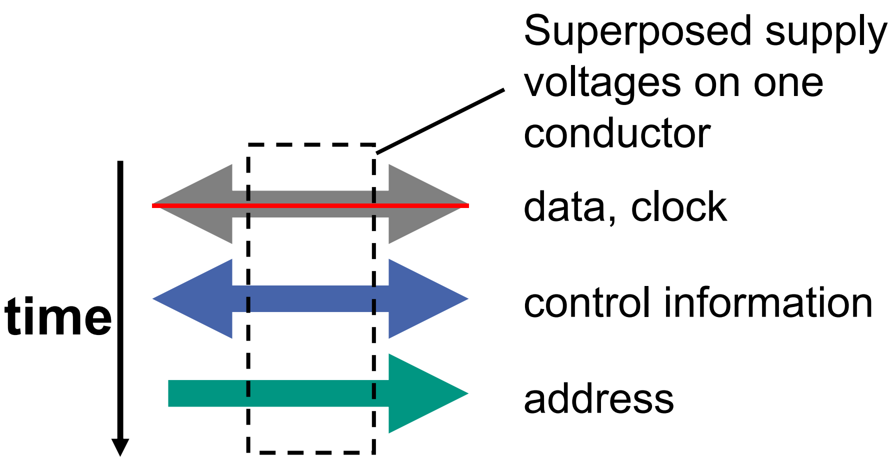
logical Setup



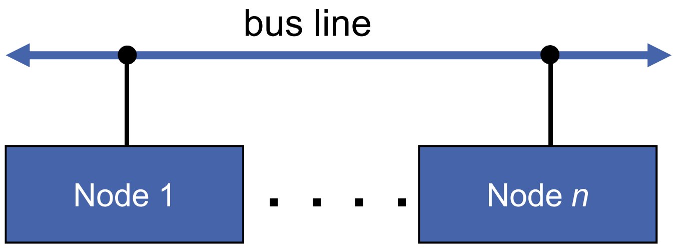
physical Setup

- Data Bus: Actual Data Transmission
- Address Bus: Selection of specific nodes and addresses
  □ sometimes data and addresses are transmitted over them same line at different points in time
- Control Bus: Bus requests, Arbitration, Interrupts
- Supply Bus: Power Supply and Clock Lines

Serial Bus



logical Setup



physical Setup

- Only a single line that is set up in a bus structure.
- Functions that are implemented via dedicated wires in parallel bus systems, are implemented in serial bus systems via (software-) protocols

Comparison Serial vs. Parallel Communication

Parallel Communication

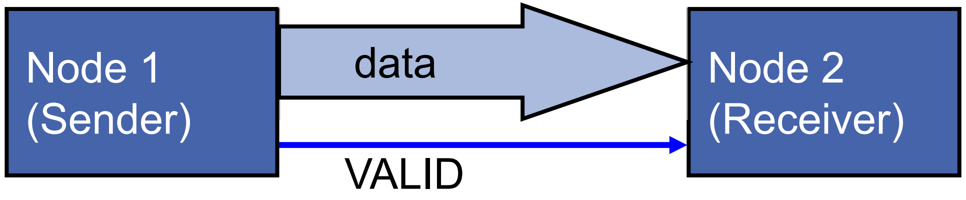
- Pros:
  - High throughput possible
- Cons:
  - Expensive
  - Skew of individual signal lines
  - Protocol changes become difficult because of predefined signal lines

Serial Communication

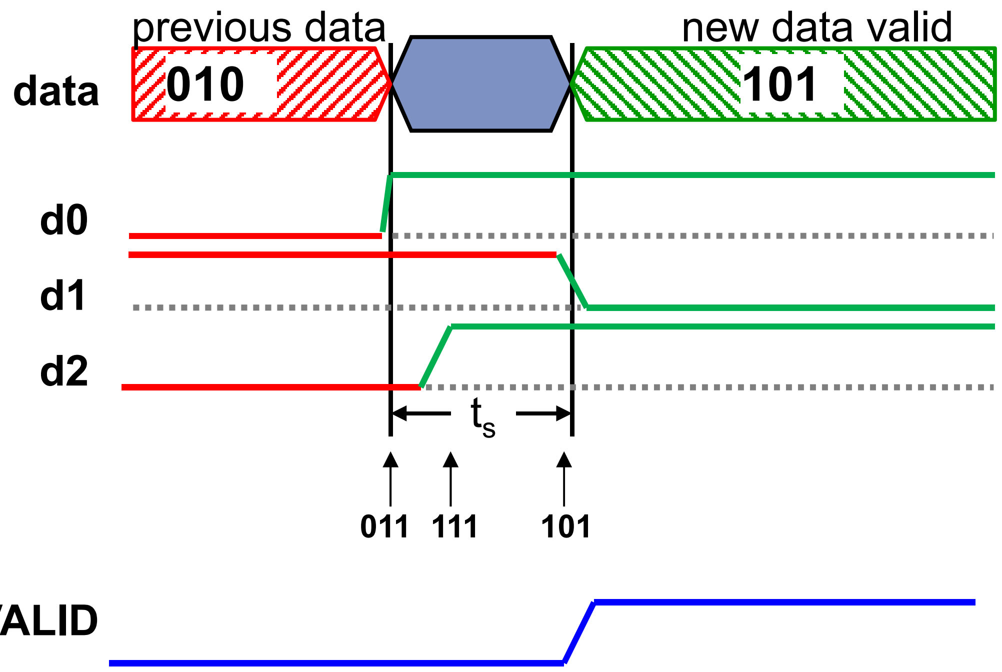
- Pros:
  - Cheap
  - High data rates with low effort
  - Flexible in terms of changes
- Cons:
  - less bits per clk can be transmitted

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

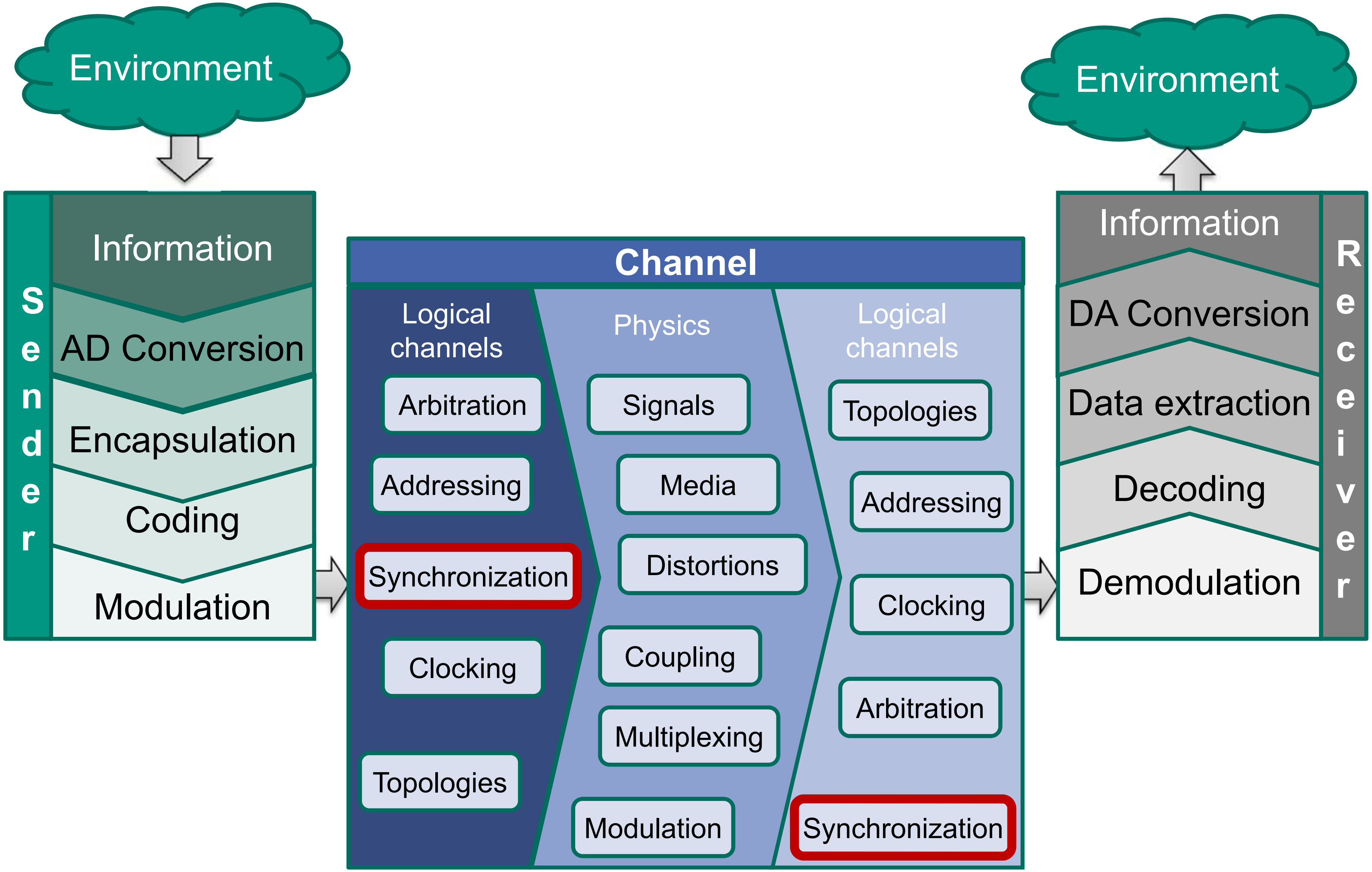


- Example:
  Signal change 010 □ 101



- Sender marks valid new data with a VALID signal

Synchronization



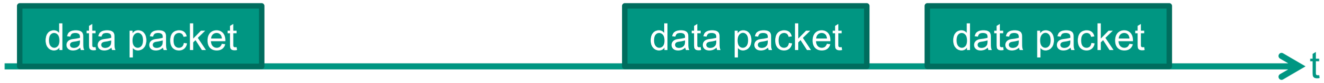
Different types of Synchronisation

- 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
    - …

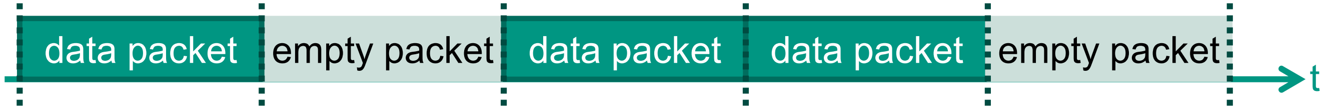
- High-Level Synchronization (e.g. hand shaking)
  - Logical Synchronization of communication process
    - When does a data frame start/end?
    - Is the receiver ready for reception?
    - …

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

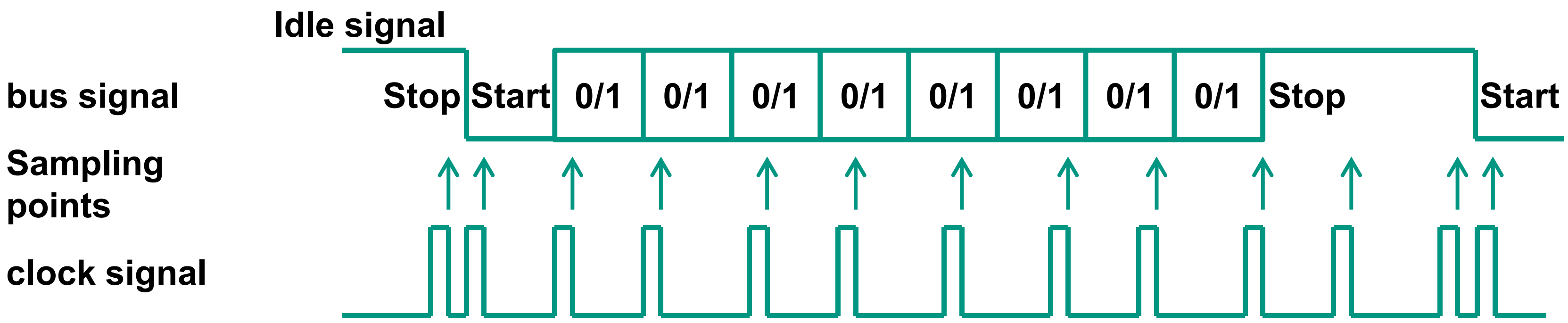


Synchronisation Schemes

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

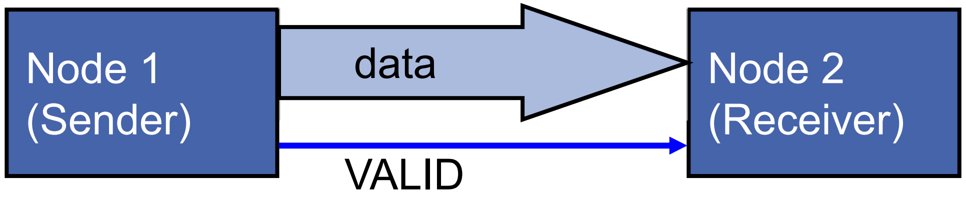
Start-Stop mode

- 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

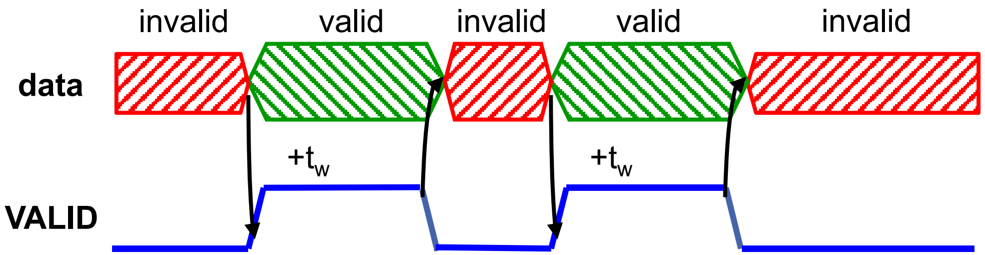


Simplex Handshake

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.



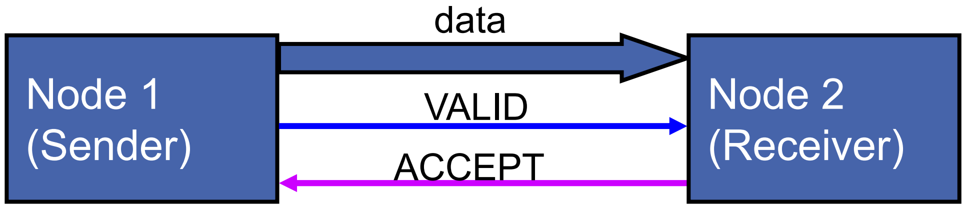
- One always waits for a fixed period of time t_w



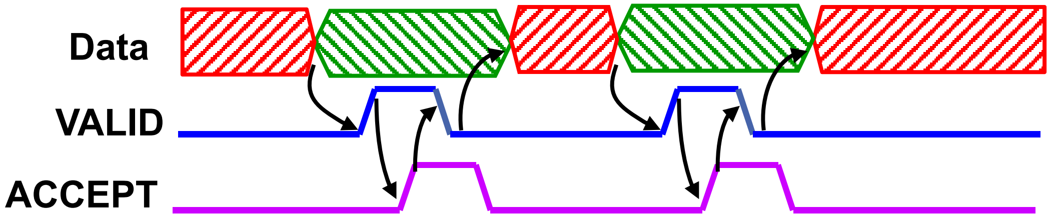
  - Disadvantage: Waiting time t_w is dependent on the slowest node on the bus and is always invariant in length (synchronous mode)

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.



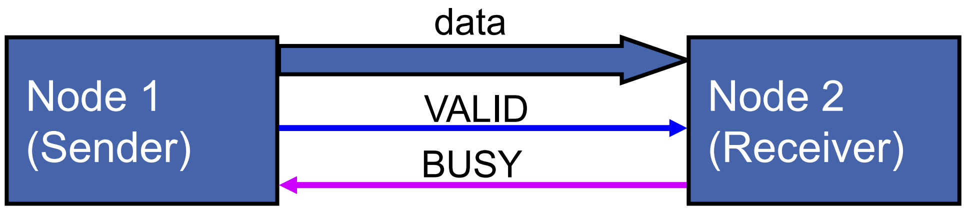
- Receiver asserts ACCEPT signal if data is not needed any longer.



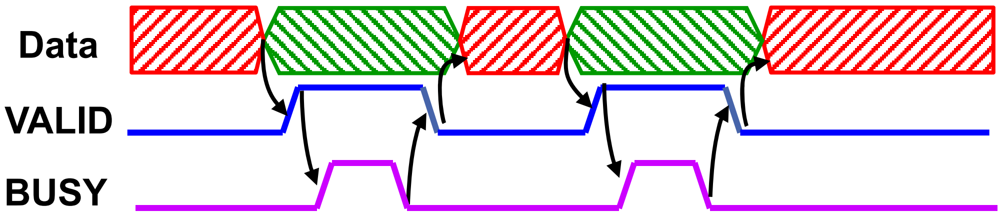
- Disadvantage: Problems with node synchronization if ACCEPT signal is asserted for too long (sender will remove subsequent data from the bus before receiver has read it)

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.



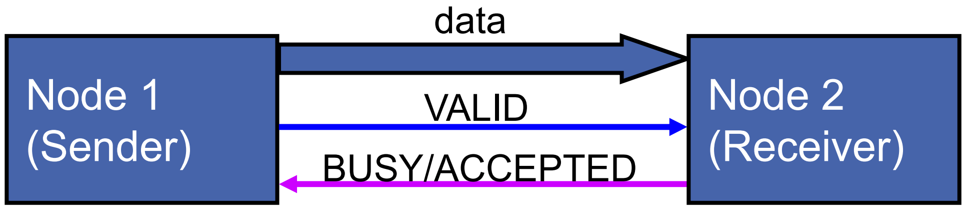
- Receiver asserts BUSY-Signal as long as data has to be available.

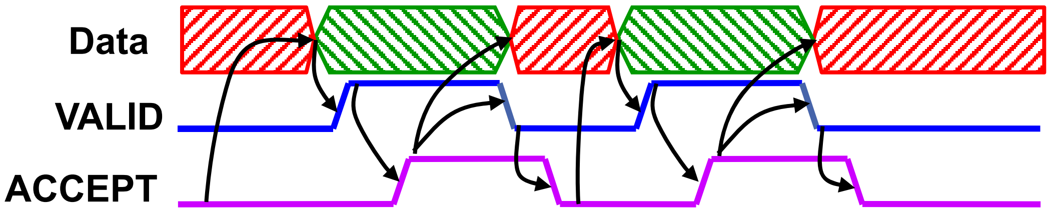


- Disadvantage: Problems with node synchronization if BUSY signal is asserted too late (data will be removed from the bus too soon)

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.

Interrupts

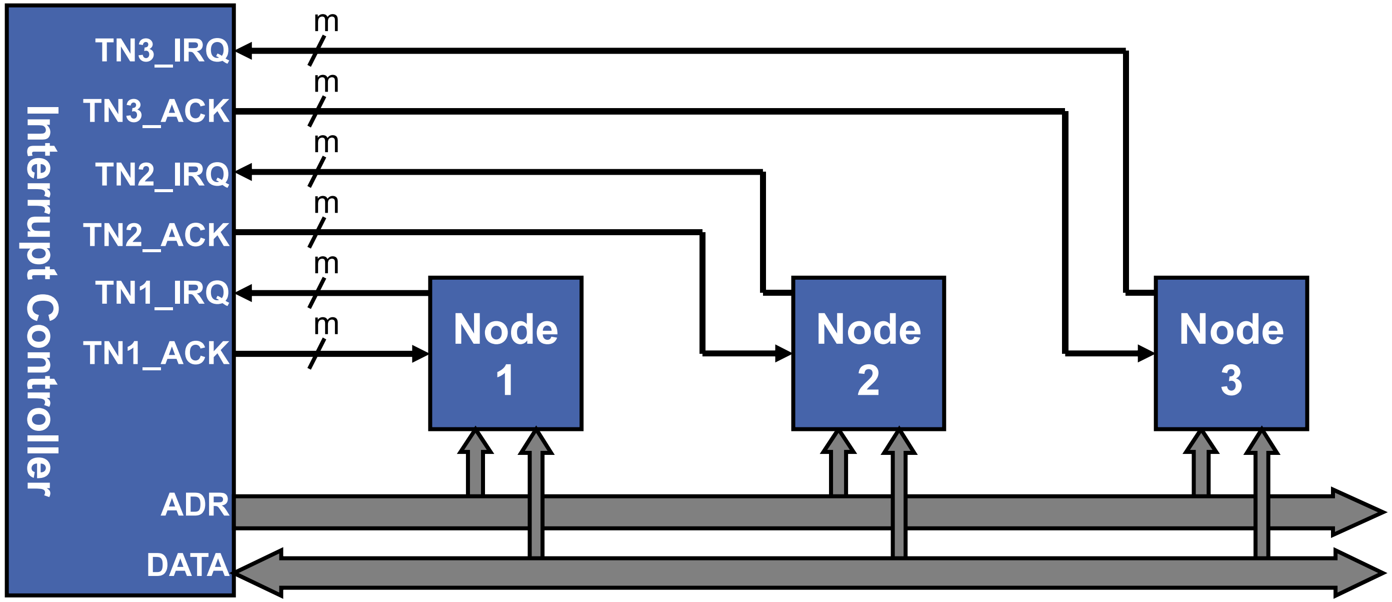
- In a master-slave bus system, only the masters can actively apply for the bus

- Problem: If a slave has to send important information, it has to wait for a master to be polled

- Solution: Interrupts
  - Dedicated lines that go from the slaves to the masters
  - For serial communication, interrupt handling has to be emulated by cyclic polling of every slave

Examplary Implementation

- If a slave has a communication request, it asserts an interrupt
- Interrupt Controller (can be implemented within Master Node)
  - starts a communication session with the slave to service interrupt request
  - acknowledges Interrupt to node
- Multiple interrupt priorities (levels) possible



Outlook

- Error Detection
- Automotive Busses
  - CAN
  - LIN
  - FleyRay