System Architectural Design

Interfaces and Protocols

I2ISE

Software interfaces

- Software interfaces, e.g. between SW components, can be specified using *contracts.*

- Contracts define an operation's interface in terms of prototype and conditions

- Contracts consist of (at least) three things:
  - Operation
  - Preconditions
  - Postconditions

- By knowing only the contract, other software components can interface the operation without knowing anymore about it

Software interfaces: Contracts

- **Operation**
  - the name of the operation
  - the name and type of each parameter
  - the return type of the operation

- **Precondition(s)**
  - What must be true before the operation is called

- **Postcondition(s)**
  - What will be true when the operation terminates, *if* the precondition was true

Software interfaces: Contracts: Example

Contract C01:	**Add**
Operation:	`double Add(double x, double y)`
Precondition:	None
Postcondition:	The sum of **`x`** and **`y`** is returned

Contract C02:	**Divide**
Operation:	`double Divide(double x, double y)`
Precondition:	**`y`** != 0
Postcondition:	The qoutient **`x/y`** is returned

Contract C03:	**SquareRoot**
Operation:	
Precondition:	
Postcondition:	

Protocols

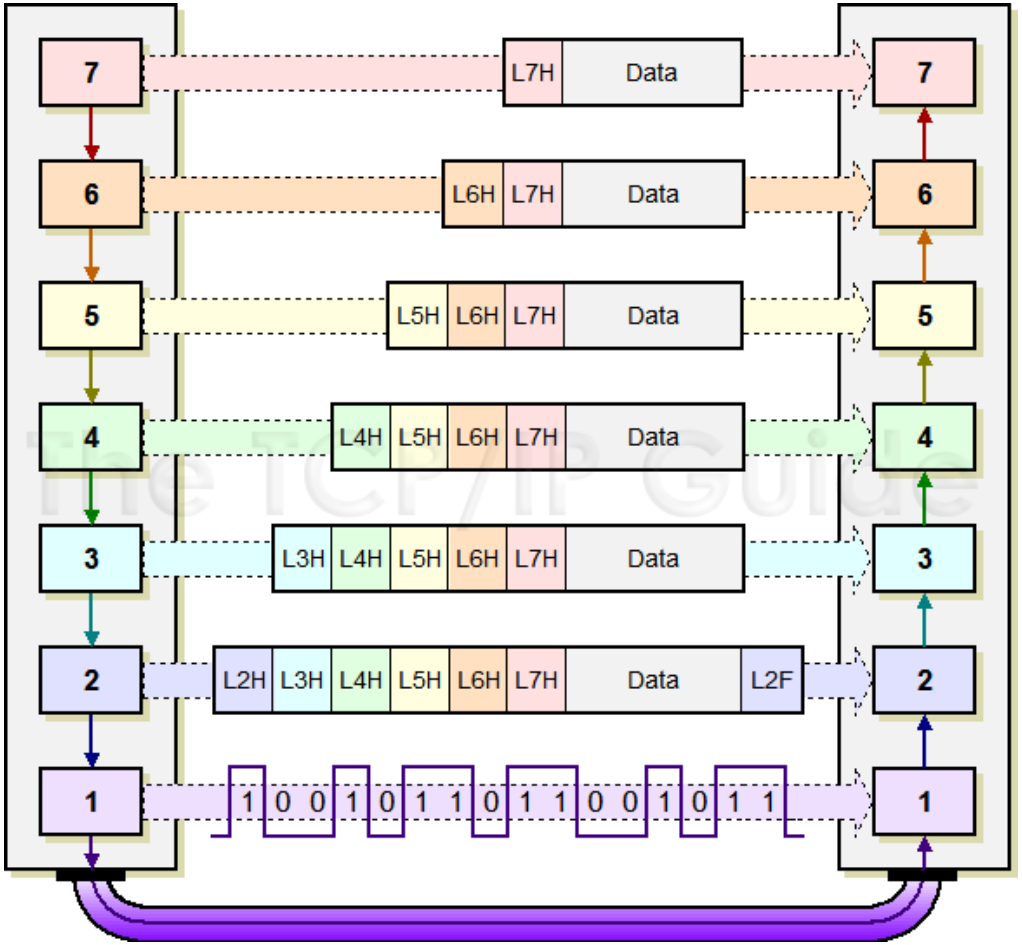
- Protocols are one ”step up” from the physical layer (signals, names, voltage levels etc.)

- Protocols define *how* the physical interface is used
  - E.g.: The *physical* interface is RS232 – 9 or 25 wires carrying data (Rx, Tx) and control (RTS, RTR, CTS, …) signals

  - The protocol defines how data transmitted/received shall be interpreted.

- Just like before, the protocol must specify the interface unambiguously

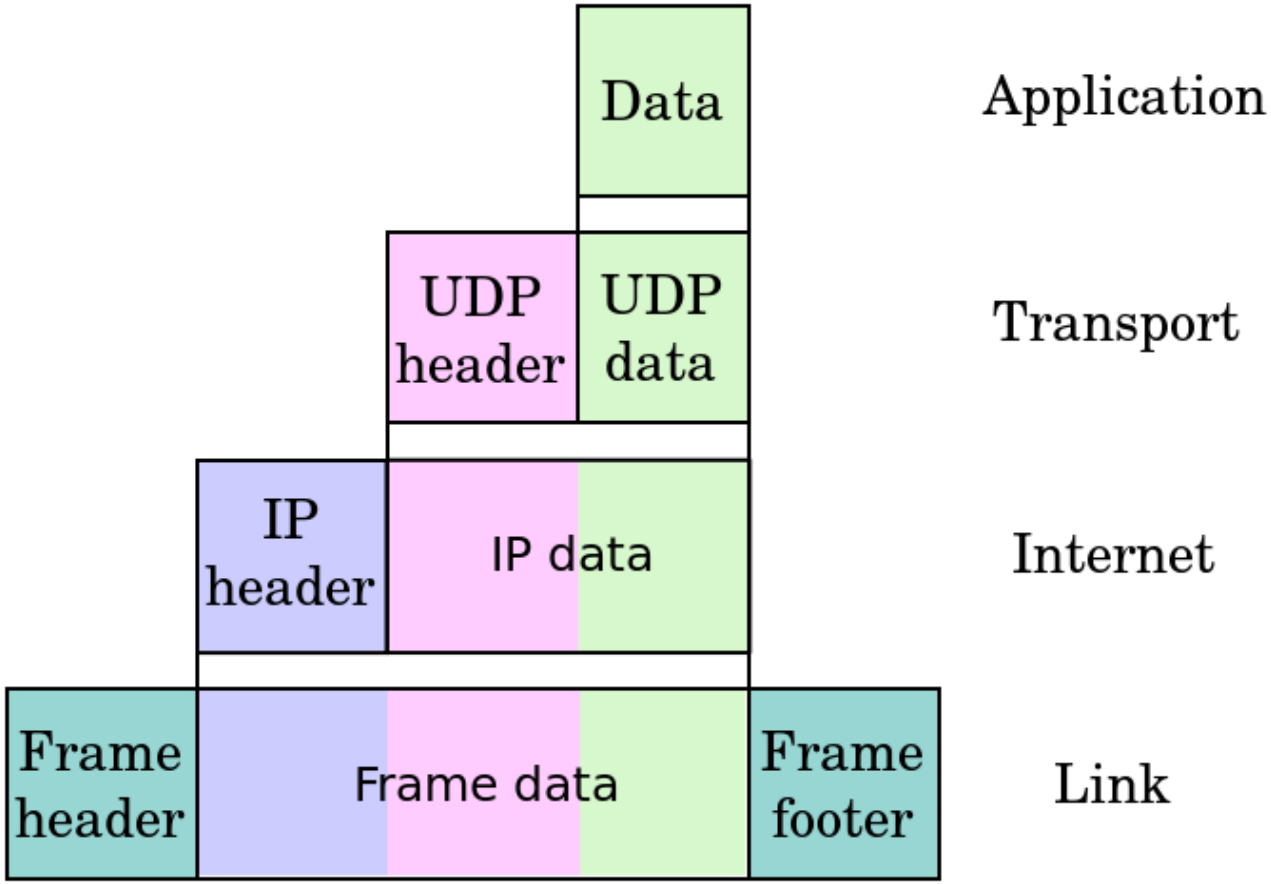
Example: Data encapsulation in OSI 7-layer model



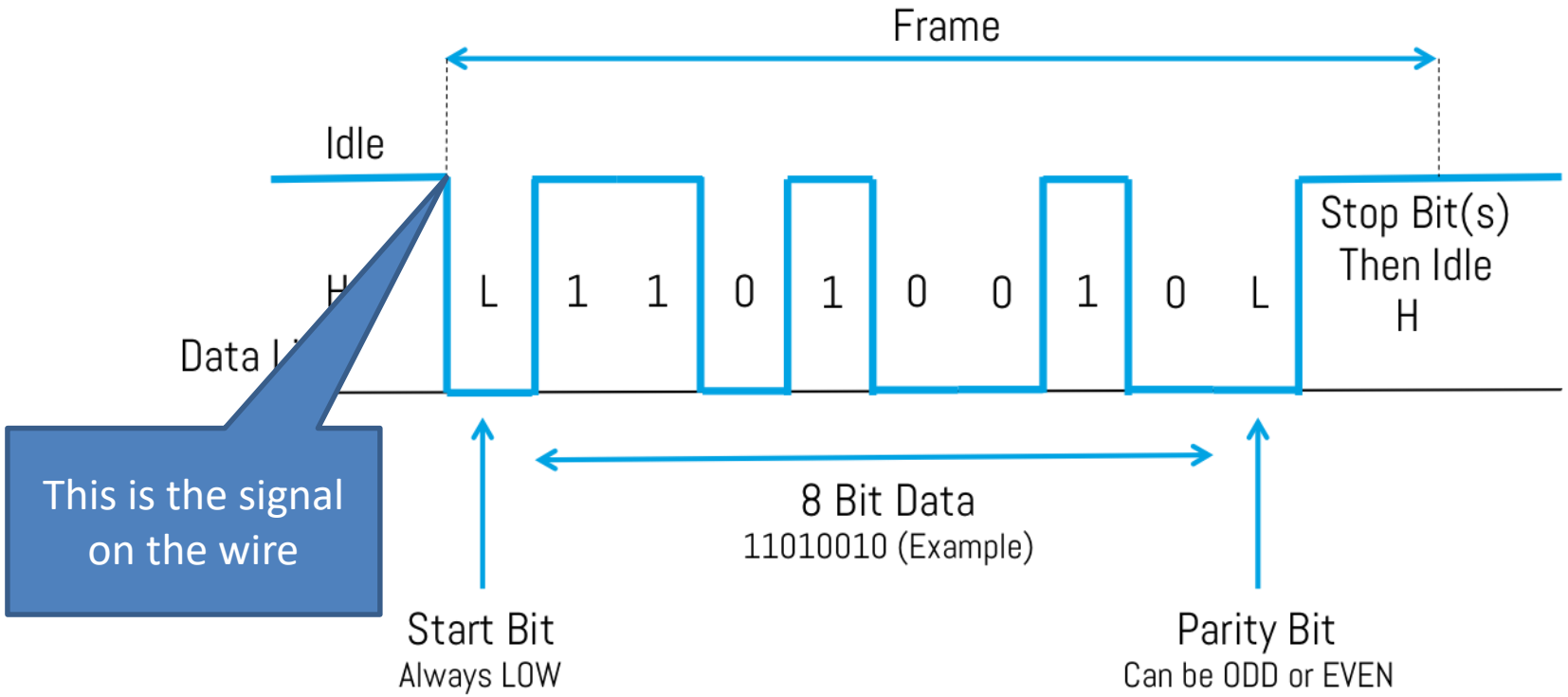
OSI layers

- Physical Layer (1)
  - Mechanical and electrical interface
  - Moves bits over a communication channel
  - Concerns how the connection is established
- Data Link Layer (2)
  - Moves frames as a collection of bits
  - Acknowledgement from receiver
  - Ensure error free transmission
- Network Layer(3)
  - Translates logical to physical addresses
  - Routing of messages through intermediate nodes
  - May deliver messages by split into several frames
- Transport Layer (4)
  - Splitting of data in packages/frames (Segmentation/de-segmentation)
  - Ex. Internet
    - Transmission Control Protocol (TCP)
    - User Datagram Protocol (UDP)
- Application Layers (6-7)
  - Communication between applications
  - Remote objects or functions
    (Egg. HTTP and web browsers)

IP Layers for UDP/IP



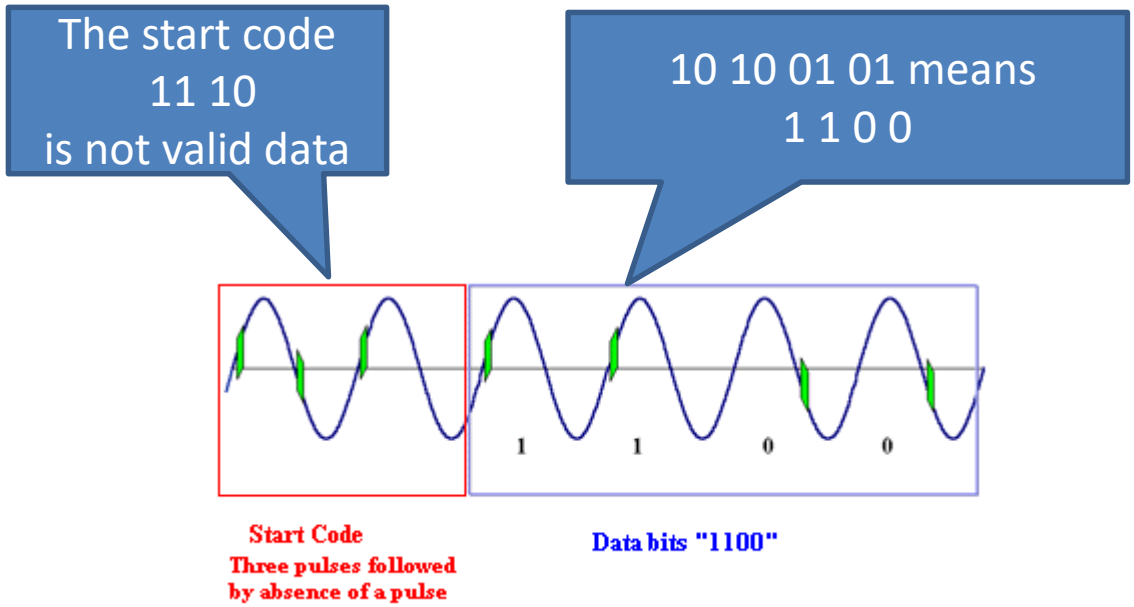
RS232 frame



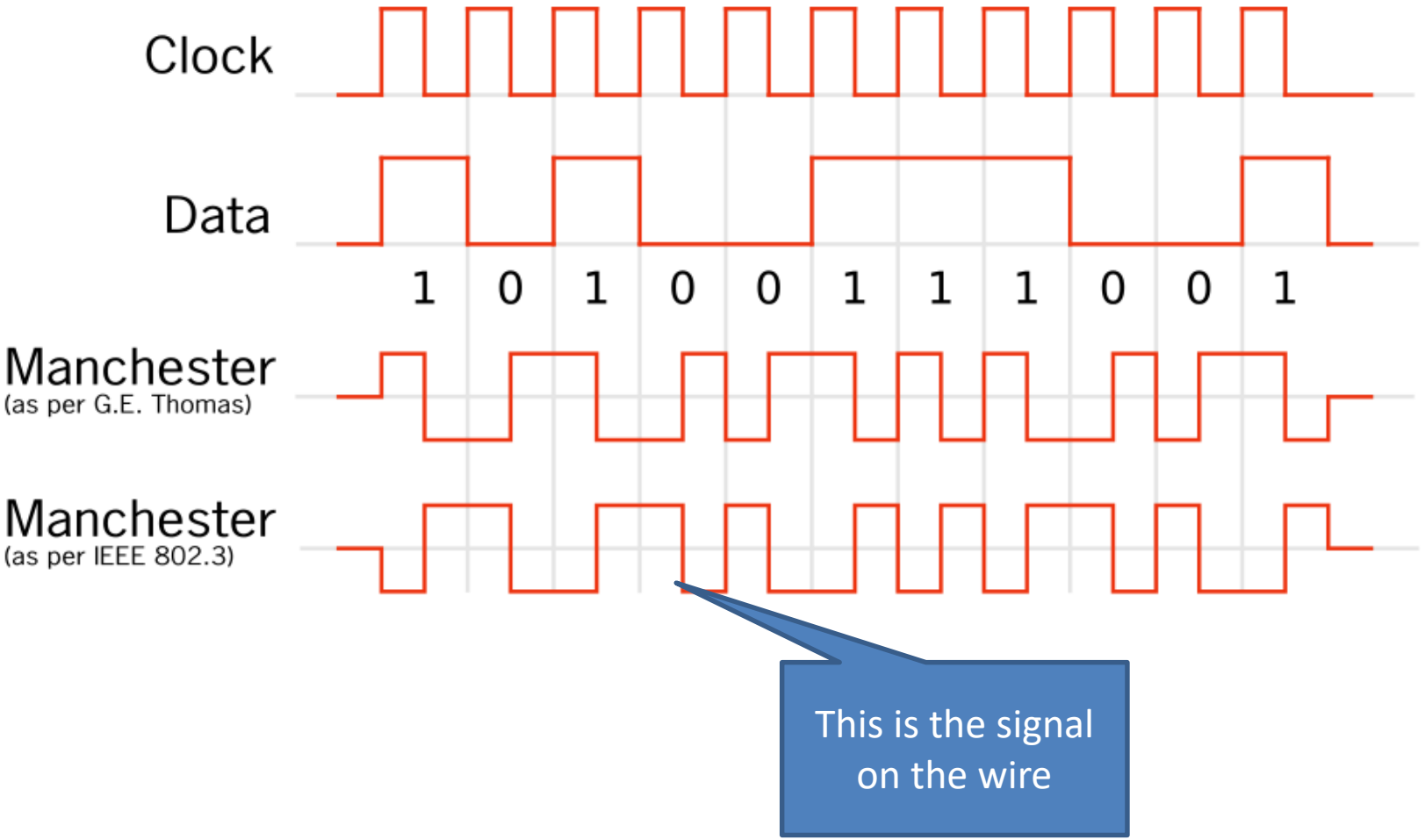
The AVR USART Frame Format

maxEmbedded.com
a guide to robotics and embedded systems

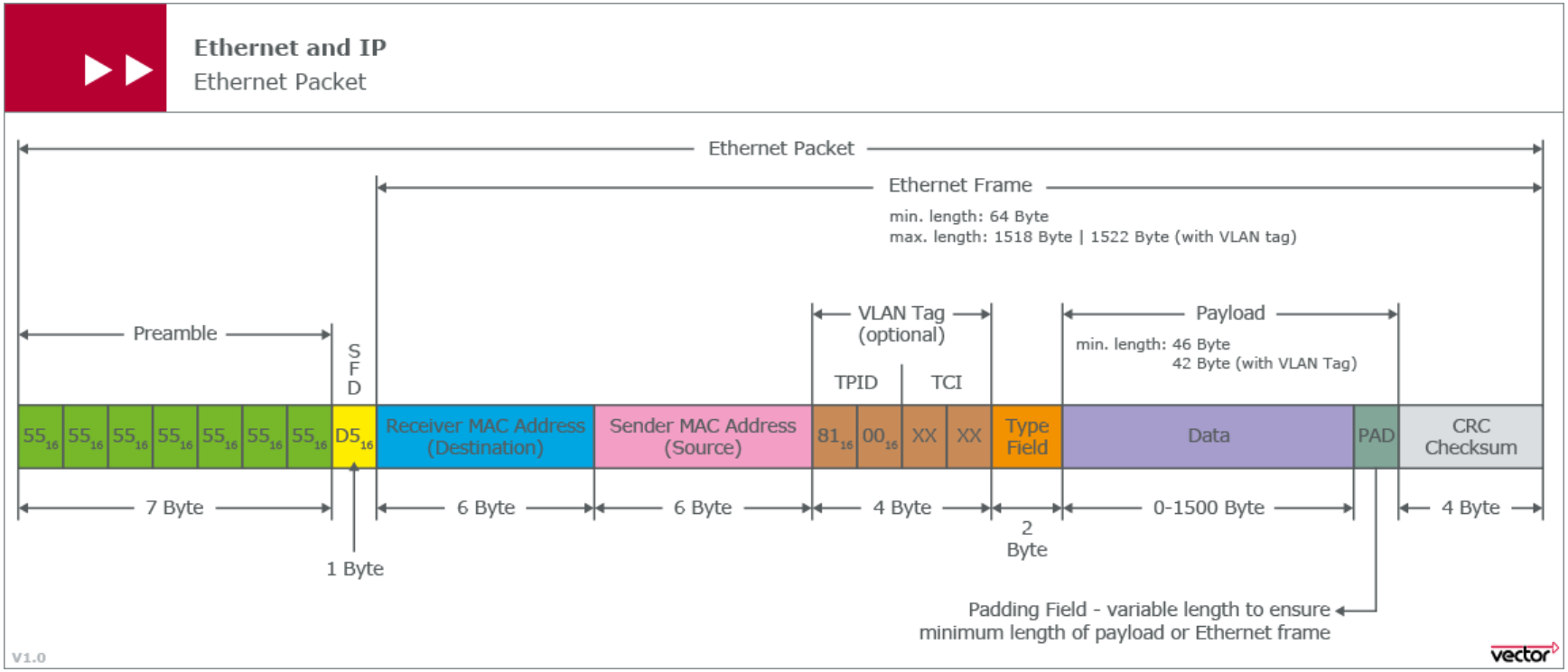
X10 – physical level



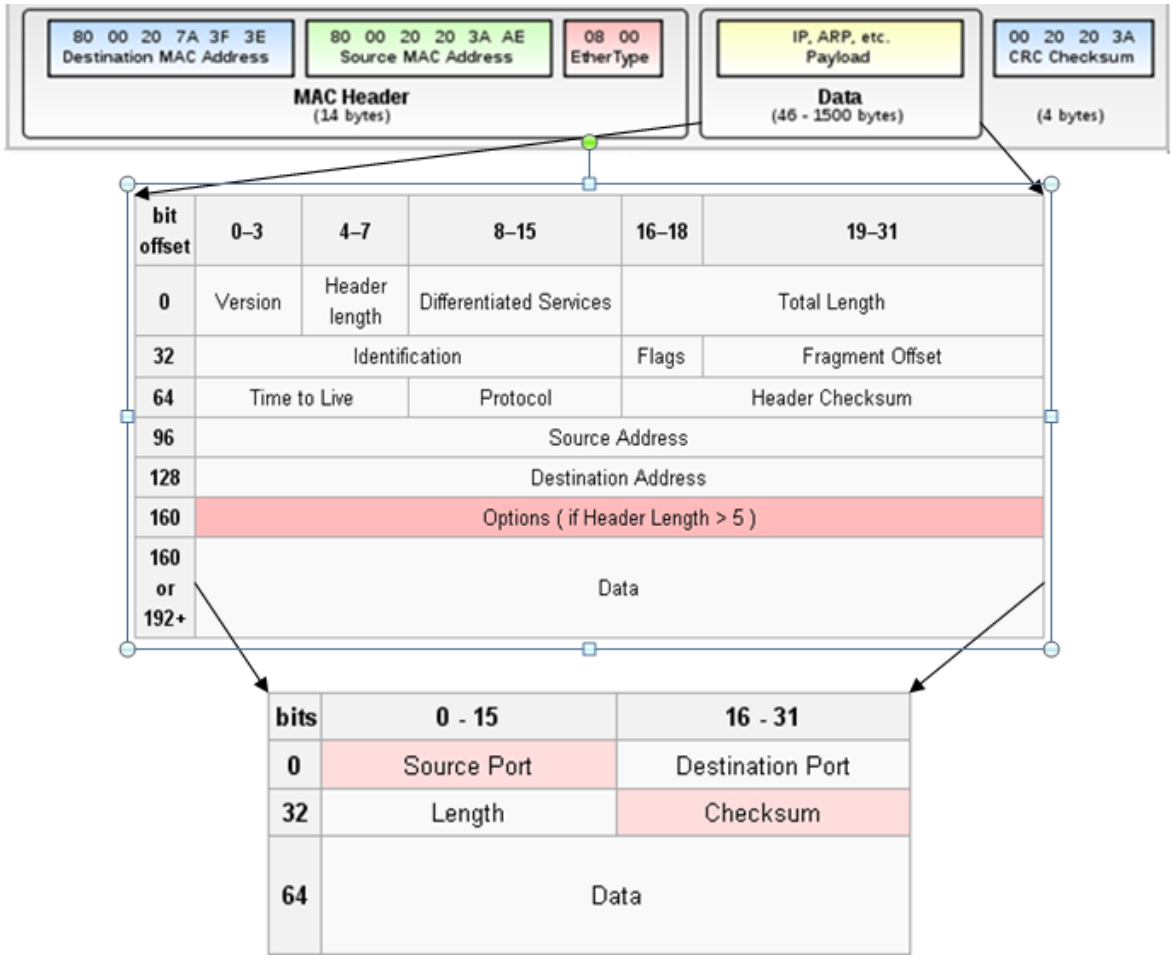
Ethernet – physical level



Ethernet frame - logical level



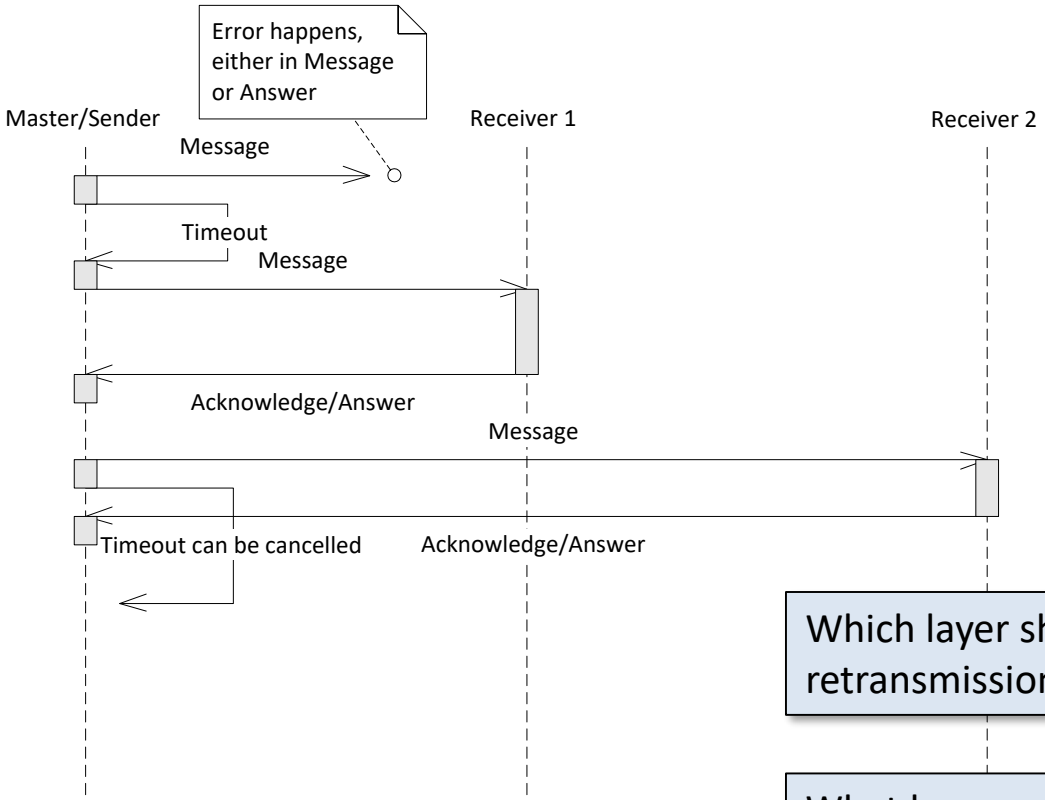
UDP/IP embedded in Ethernet



Handling of errors

- Error detection
  - Format: Header, Length, Payload (data), Footer
  - Checksums
  - Acknowledge
  - Timeouts
- Error correction/elimination
  - Retransmission
  - Resynchronisation
  - Error correcting codes
  - Robustness at the hardware level

Typical Master/Slave setup



Which layer should handle retransmission?

What happens to responsiveness of the system, as a function of timeout, retransmissions and receivers?

Simple error correcting scheme

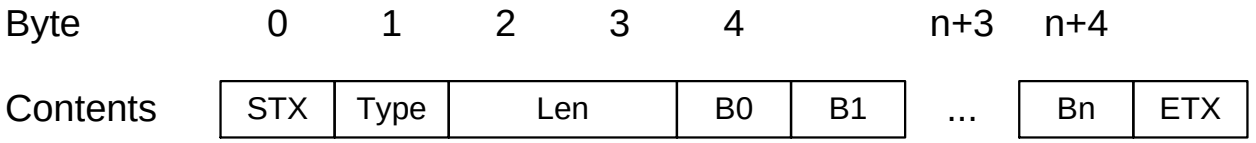
Triplet received	Interpreted as
000	0 (error free)
001	0
010	0
100	0
111	1 (error free)
110	1
101	1
011	1

Examples of error correcting codes

2 (single-error detecting)	Parity
3 (single-error correcting)	Triple modular redundancy
3 (single-error correcting)	perfect Hamming such as Hamming(7,4)
4 (SECDED)	Extended Hamming
5 (double-error correcting)	
6 (double-error correct-/triple error detect)	
7 (three-error correcting)	perfect binary Golay code
8 (TECFED)	extended binary Golay code

Kan rette 3 fejl ved at bruge 24 bits til at sende 12 databits

Example: Properitary protocol



Data request
Request for most recent data
Direction: Master > Slave
Type: ‘0’
Len: ‘00’
Data: -

Data response
Most recent data
Direction: Slave > Master
Type: ‘1’
Len: ‘04’
Data: B0: Sensor 1 LSB
B1: Sensor 1 MSB
B2: Sensor 2 LSB
B3: Sensor 2 MSB

Your turn!

- Start the specification of a protocol between the PC program and the X.10 Controller in your semester project.
  - Could you use your application model to help?

- Consider what information must flow.
  (Consider how to test it using a terminal program by sending only ASCII characters.)
  - Header (e.g. STX = ‘C’, message type, …)?
  - Data?
  - Footer (e.g. ETX = 0x13 <CR>?)