System Architectural Design

Protocols

I2ISE

Protocols

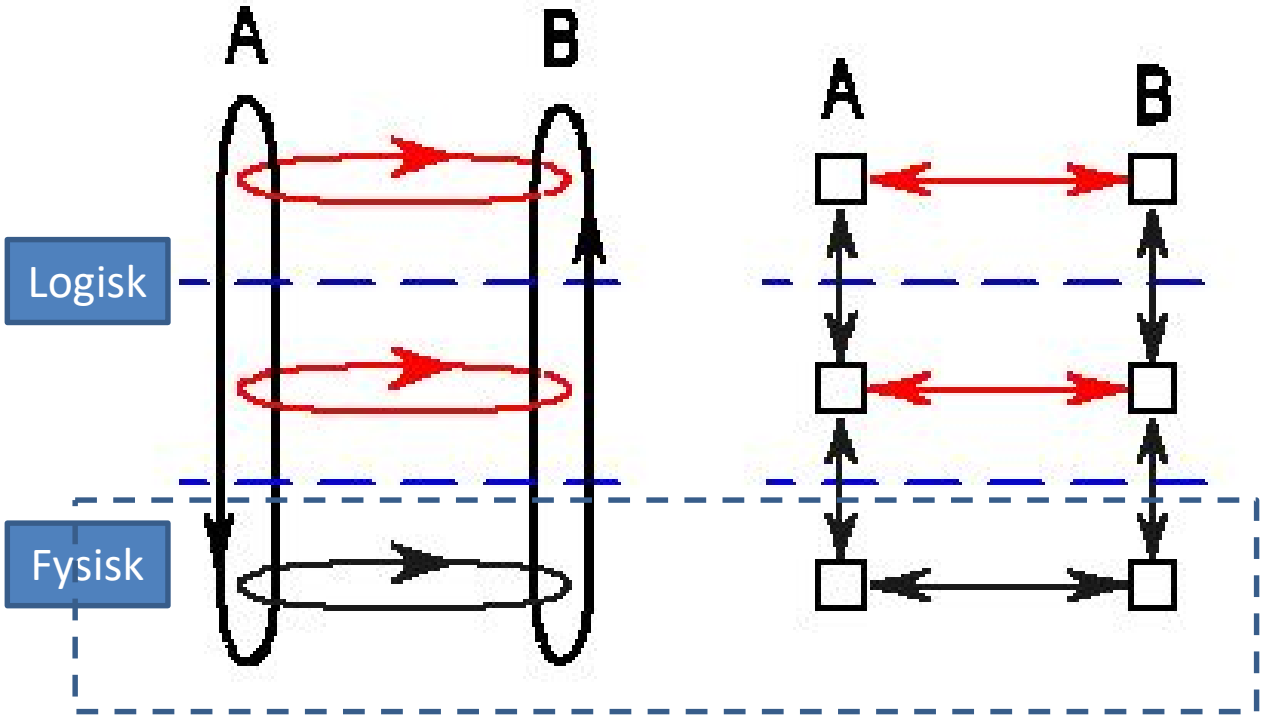
- Protocols are one ”step up” from the physical layer (signals, names, voltage levels etc.)

- 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.

- The protocol must specify the interface unambiguously

Physical – logical - software

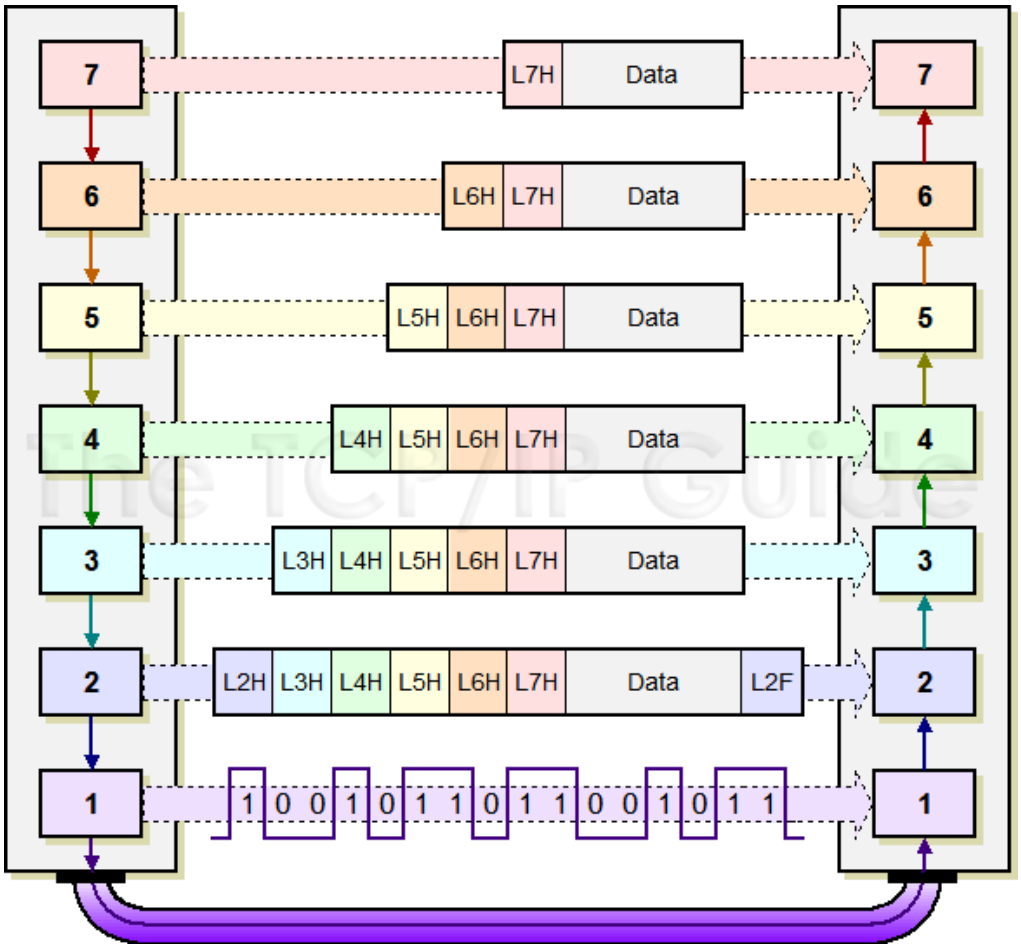


Protocol Stack

Physical – logical view

Software Implementation Layered Boundary Classes

Example: Data encapsulation in the OSI 7-layer model (Open Systems Interconnection)

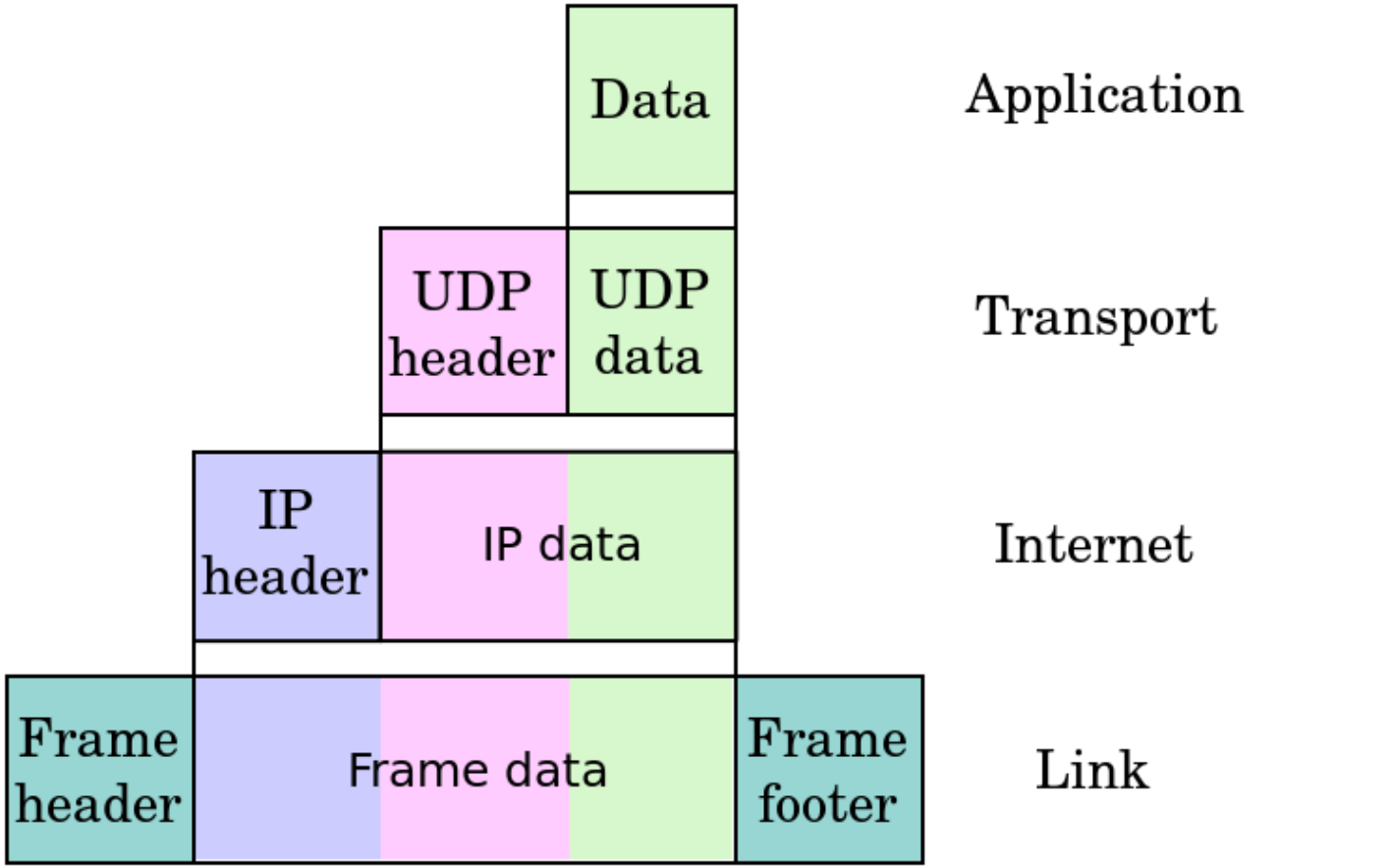


Navn
Application
Presentation
Session
Transport
Network
Data Link
Physical

OSI layers

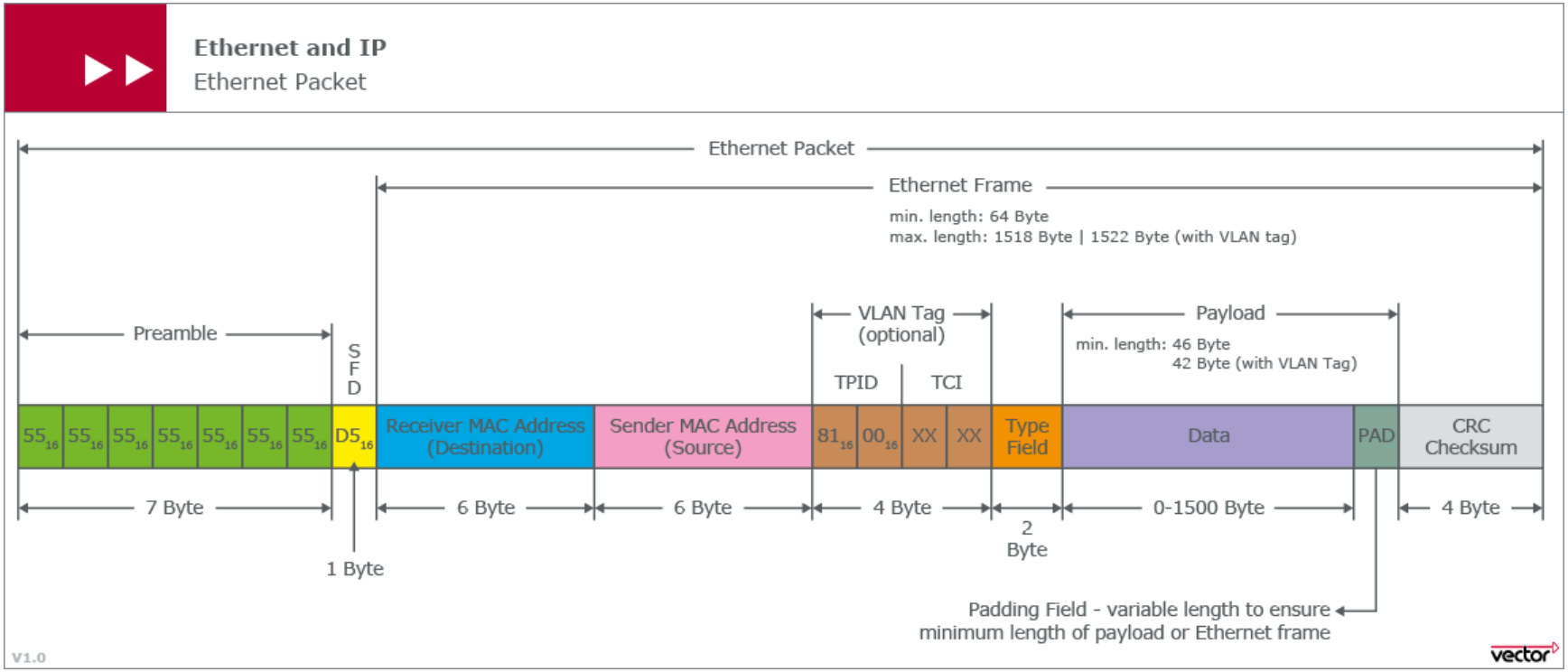
- Application Layer (7)
  - Communication between applications
  - Remote objects or functions
    (Egg. HTTP and web browsers)
- Presentation Layer (6)
  - Conversion between data representation
- Session Layer (5)
  - Handles connections like a phone call, TCP connection
- Transport Layer (4)
  - Splitting of data in packages/frames (Segmentation/de-segmentation)
  - Ex. Internet
    - Transmission Control Protocol (TCP)
    - User Datagram Protocol (UDP)
- Network Layer(3)
  - Translates logical to physical addresses
  - Routing of messages through intermediate nodes
  - May deliver messages by split into several frames
- Data Link Layer (2)
  - Moves frames as a collection of bits
  - Acknowledgement from receiver
  - Ensure error free transmission
- Physical Layer (1)
  - Mechanical and electrical interface
  - Moves bits over a communication channel
  - Concerns how the connection is established

IP Layers for UDP/IP



Unfortunately there is not an exact match to OSI!

Ethernet frame - logical level



UDP/IP embedded in Ethernet

Ethernet Data

IP Data

UDP Data

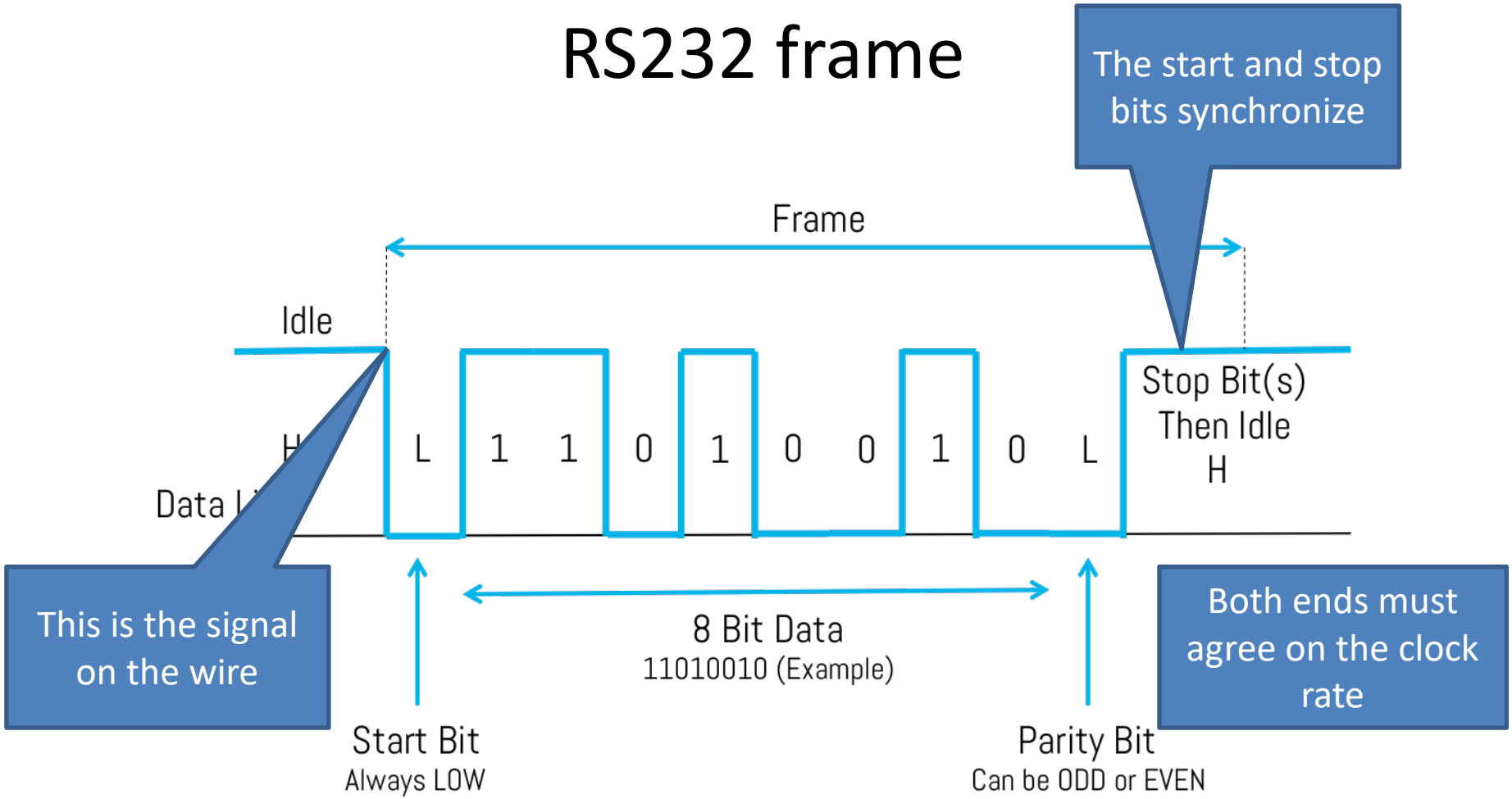
bit offset	0–3	4–7	8–15	16–18	19–31
0	Version	Header length	Differentiated Services	Total Length	
32	Identification			Flags	Fragment Offset
64	Time to Live		Protocol	Header Checksum	
96	Source Address				
128	Destination Address				
160	Options (if Header Length > 5)				
160 or 192+	Data				

bits	0 - 15	16 - 31
0	Source Port	Destination Port
32	Length	Checksum
64	Data	

Physical layer examples

- RS232
- X10
- Ethernet

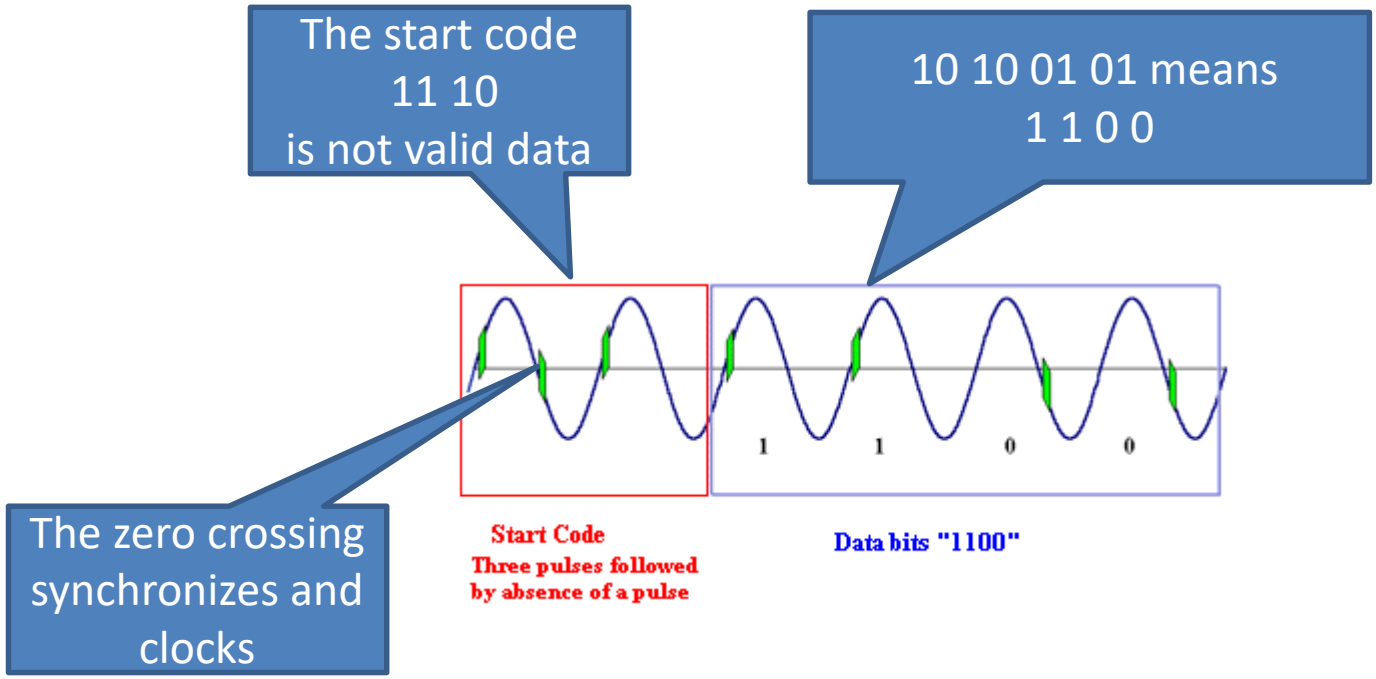
RS232 frame



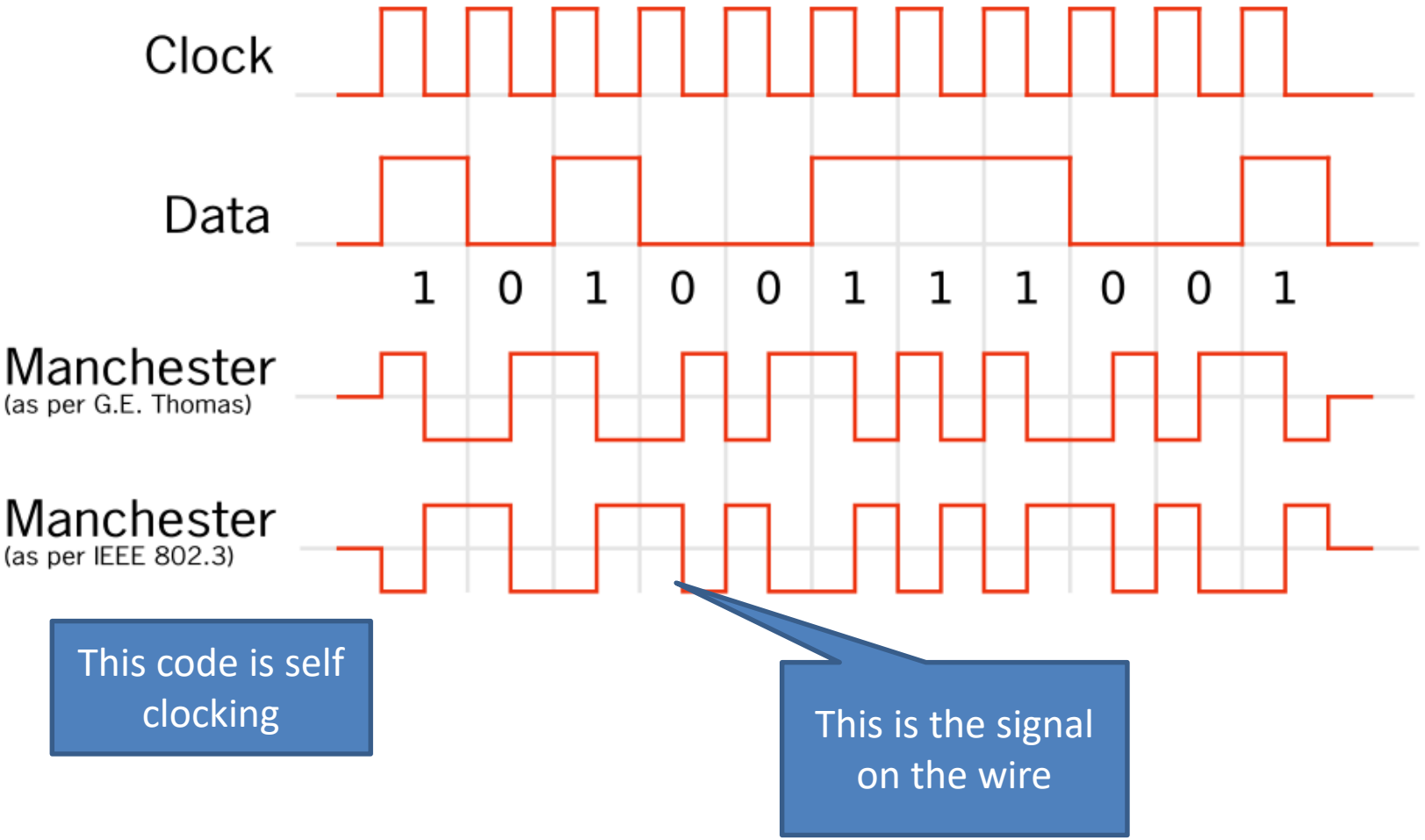
The AVR USART Frame Format

maxEmbedded.com
a guide to robotics and embedded systems

X10 – physical level



Ethernet – physical level



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

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

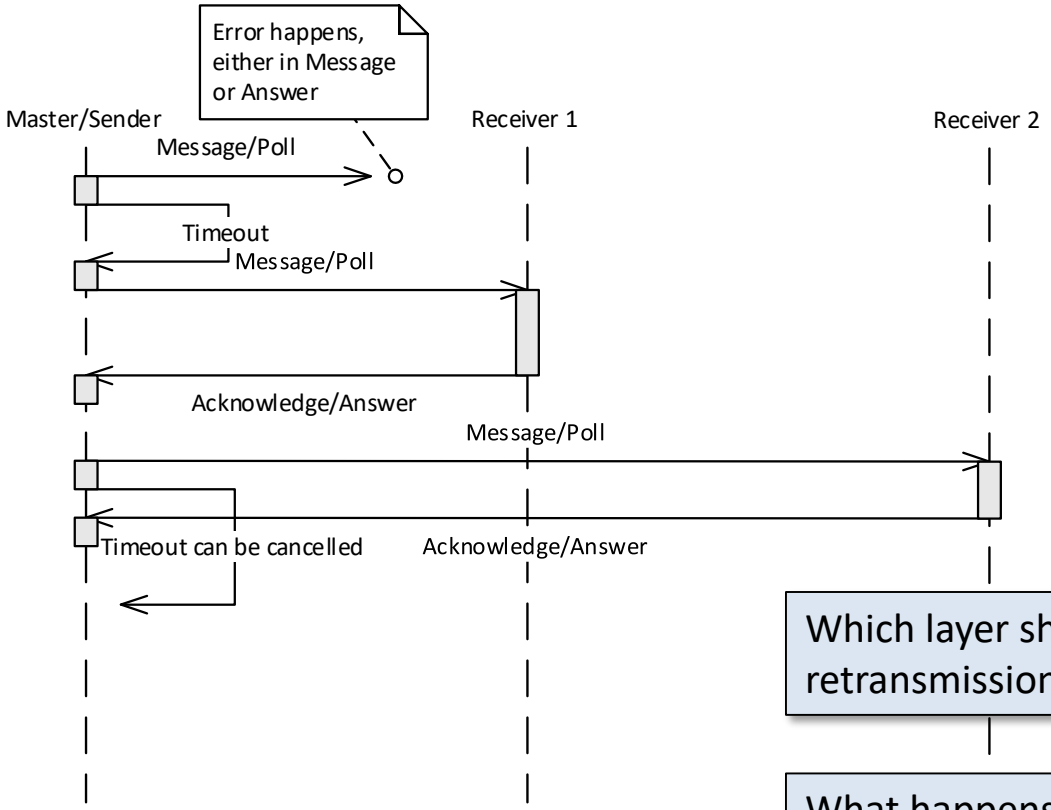
Examples of error correcting codes

Detect	Correct	Method
1	0	Parity
1	1	Triple modular redundancy (see previous slide)
1	1	perfect Hamming such as Hamming(7,4)
2	1	Extended Hamming
3	3	perfect binary Golay code
4	3	extended binary Golay code

Used by NASA for Deep Space Missions

Can correct 3 errors by using 24 bits to send 12 data bits

Typical Master/Slave setup
Half Duplex



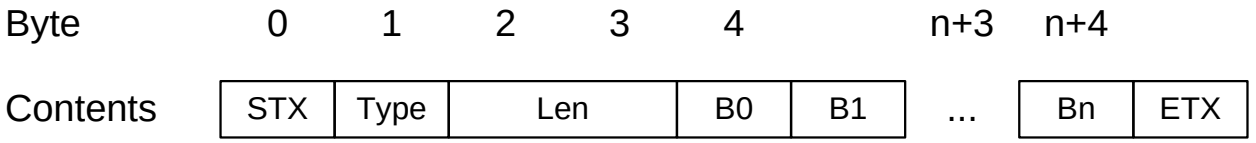
Which layer should handle retransmission?

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

Retransmission is not an option!

- https://voyager.jpl.nasa.gov/mission/status/

Example: Proprietary protocol



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

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

Your turn!

- Start the specification of a protocol between the PC program and the X.10 Controller in your semester project.
- Consider what information must flow (e.g. from System Sequence Diagrams)
  (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>)?
- Who is Master and Slave?
- Lessons learned in MSYS on 1st semester?
  - SendString, SendChar, SendInt
- Lessons learned from Application Models?
  - What Software modules could be feasible?