5th Slide Set Computer Networks

Prof. Dr. Christian Baun

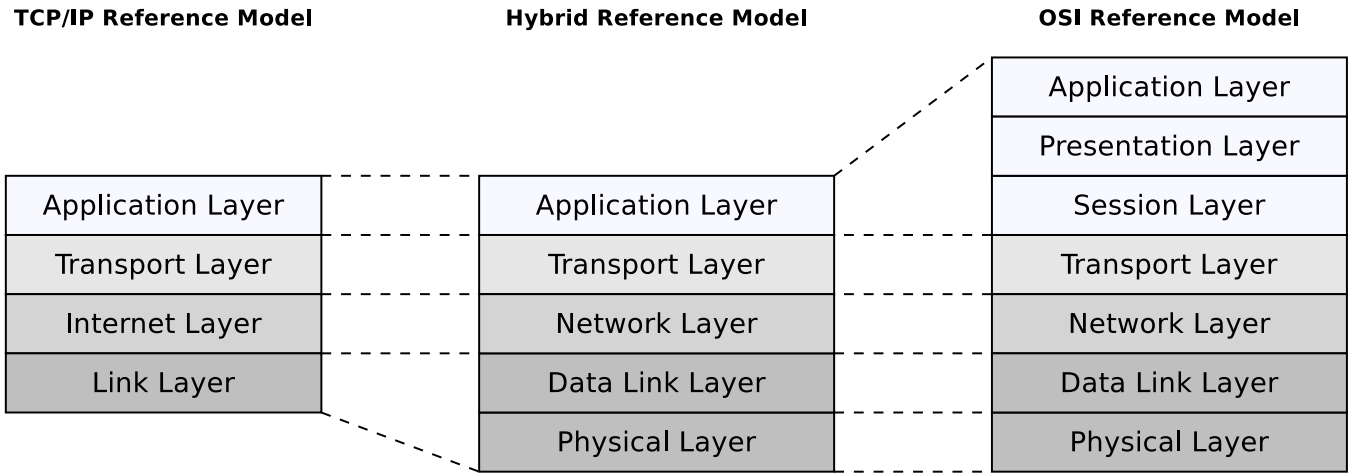
Frankfurt University of Applied Sciences
(1971–2014: Fachhochschule Frankfurt am Main)
Faculty of Computer Science and Engineering
christianbaun@fb2.fra-uas.de

Learning Objectives of this Slide Set

- Data Link Layer (part 2)
  - Framing (creating frames)
    - Character count in the header
    - Byte stuffing
    - Bit stuffing
    - Physical Layer coding violations
  - Framing in current computer networks
  - Error-detection codes
    - Two-dimensional parity-check code
    - The polynomial code – Cyclic Redundancy Check (CRC)
  - Error-correction codes
    - Hamming code
  - ~~Flow Control~~
    - ~~Stop-and-wait protocol~~
    - ~~Sliding window protocol~~

Data Link Layer

- Functions of the Data Link Layer
  - Sender: Pack packets of the Network Layer into frames
  - Receiver: Identify the frames in the bit stream from the Physical Layer
  - Ensure correct transmission of the frames inside a physical network from one network device to another one via error detection with checksums
  - Provide physical addresses (MAC addresses)
  - Control access to the transmission medium



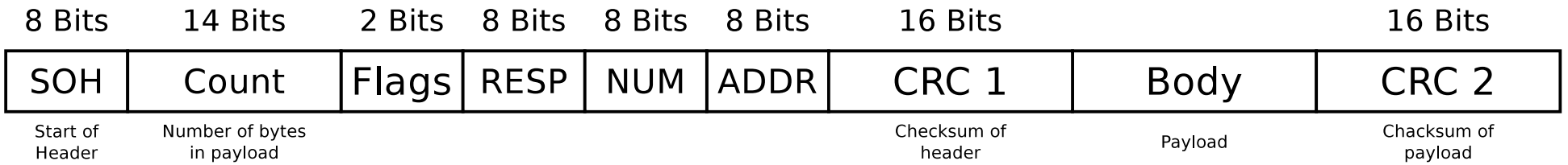
- Devices: Bridge, Layer-2-Switch (Multiport-Bridge), Modem
- Protocols: Ethernet, Token Ring, WLAN, Bluetooth, PPP

Framing (Creating Frames)

- The Data Link Layer does among others. . .
  - identify the frames in the bit stream of the Physical Layer
  - pack the data packages of the Network Layer in frames
- It is required that the beginning of each frame is highlighted
  - Without markings, the receiver cannot detect the frame boundaries
- Different ways exist to mark the frames' borders
  - **Character count in the header**
  - **Byte/Character stuffing**
  - **Bit stuffing**
  - **Line code violations** of Physical Layer with illegal signals
- All these different procedures have advantages and drawbacks

Character Count in the Frame Header

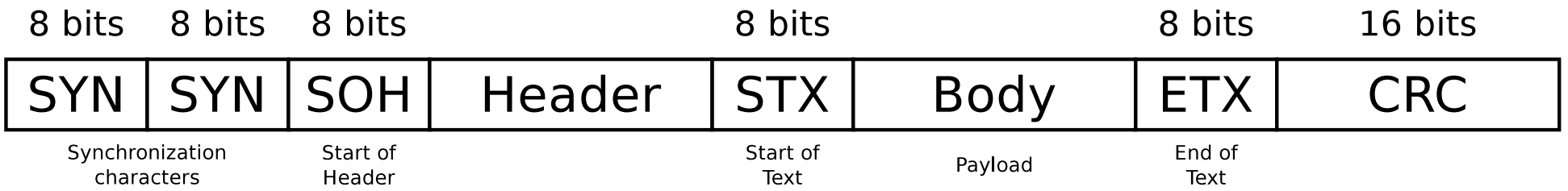
- The header of the frame contains among others the character count in the frame
- An example for a Data Link Layer protocol that specifies the frames border with a length information in the frame header is the byte-oriented **Digital Data Communications Message Protocol** (DDCMP) of DECnet
- In each frame, the field `Count` contains the number of bytes payload inside the frame



- Potential issue: If the field `Count` is modified during transmission, the receiver is unable to correctly detect the end of the frame

Byte/Character stuffing (1/2)

- A protocol, which highlights the frames border with special characters, is the byte-oriented (character-oriented) protocol **Binary Synchronous Communication** (BISYNC), which was invented by IBM in the 1960s



- Control characters ("**Sentinel characters**") highlight the structure of the frames

  - The start of a frame highlights the character `SYN`
  - The start of the header highlights the character `SOH` (Start of Header)
  - The payload is between `STX` (Start of text) and `ETX` (End of Text)

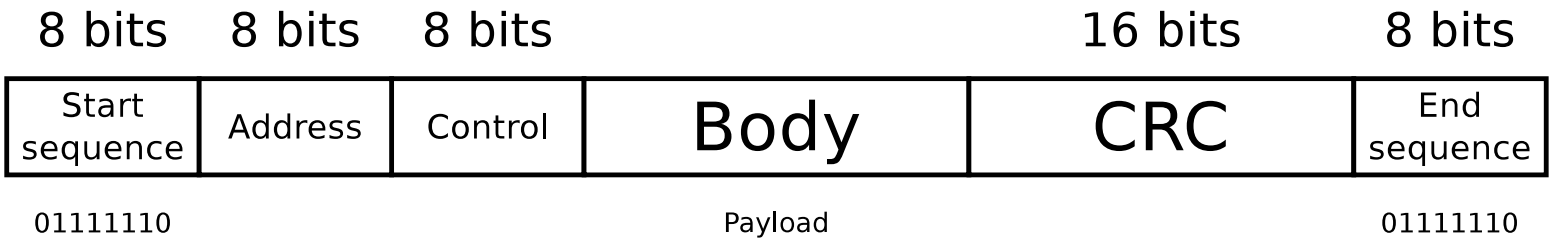
- If the payload (body) contains an `ETX` character, it it must be **protected** (*escaped*) by a stuffed `DLE` (Data Link Escape) in the Data Link Layer
- The `DLE` character is represented in the payload by sequence `DLE DLE`

Byte/Character stuffing (2/2)

- The method is called **Byte Stuffing** or **Character Stuffing**, because the...
  - sender **inserts** (*stuffs*) extra characters into the payload
  - receiver **removes** the *stuffed* characters from the received payload, before passing it to the Network Layer
- Drawback of this method:
  - Strong relationship with the ASCII character encoding
    - More recent protocols of this layer no longer operate byte-oriented, but bit-oriented because this allows using any character encoding

Bit Stuffing

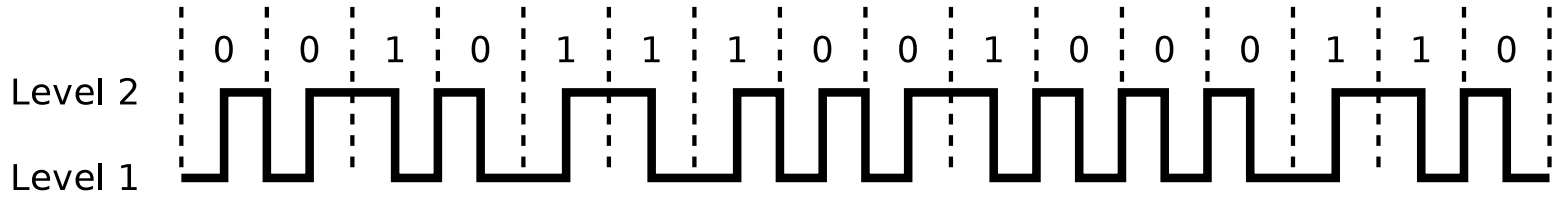
- When bit-oriented protocols are used, each frame begins and ends with a special bit pattern
- Examples: The protocol **High-Level Data Link Control** (HDLC) and the **Point-to-Point Protocol** (PPP), which implments HDLC
  - Each frame begins and ends with the sequence 01111110



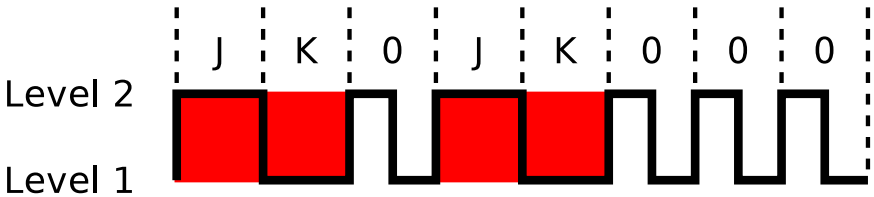
- If the HDLC protocol in the Data Link Layer...
  - of the sender discovers 5 consecutive 1-bits in the bit stream from the Network Layer, it *stuffs* a 0-bit in the outgoing bit stream
  - of the receiver discovers 5 consecutive 1-bits, followed by a 0-bit in the bit stream from the Physical Layer, it removes (*destuffs*) the 0-bit
- Advantages:
  - Ensures that the start/end sequence does not occur in the payload
  - Every character encoding can be used with this framing method

Line Code Violations

- Depending on the line code used in the Physical Layer, illegal signals can be used to highlight the frame boundaries
  - Example: Token Ring uses the Differential Manchester Encoding
    - With this line code, a signal level change occurs inside each bit cell



Starting delimiter of Token Ring

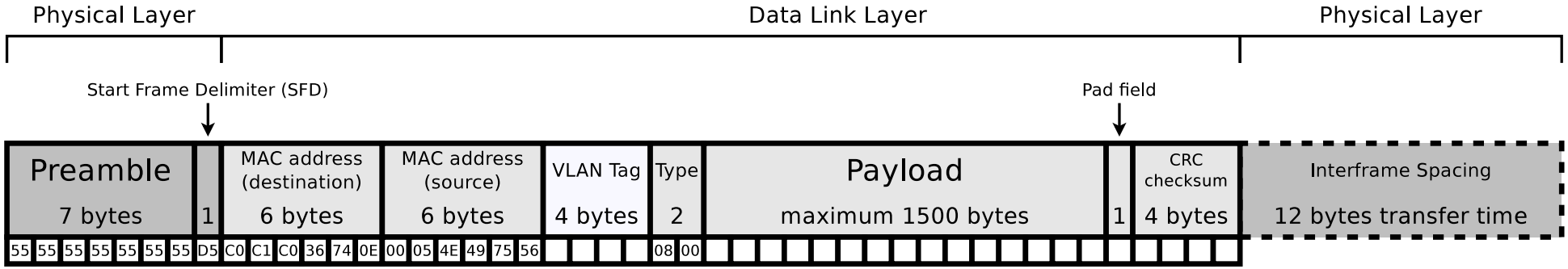


- If Token Ring is used, frames start with a byte (**starting delimiter**) which **contains 4 line code violations**

The second last byte (**ending delimiter**) of a Token Ring frame contains the same 4 line code violations as the starting delimiter

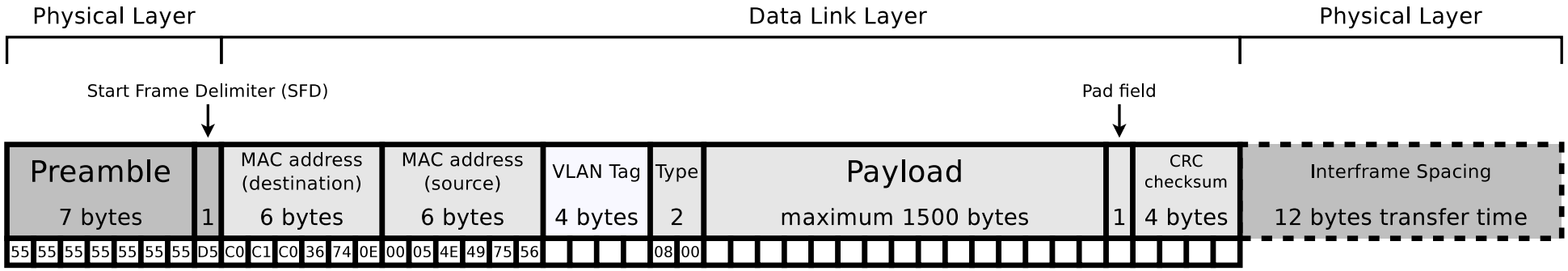
Framing in current Computer Networks (1/8) – Ethernet

- Up-to-date Data Link Layer protocols (e.g. Ethernet and WLAN) work bit-oriented and not byte-oriented
  - Reason: This way, every character encoding can be used



- Preamble is a 7 bytes long bit sequence `101010 ... 1010`
  - Is used in bus networks (topologies) to synchronize the receiver with the clock and to identify clearly the beginning of the frame
  - Is followed by the SFD (1 byte) with the bit sequence `10101011`

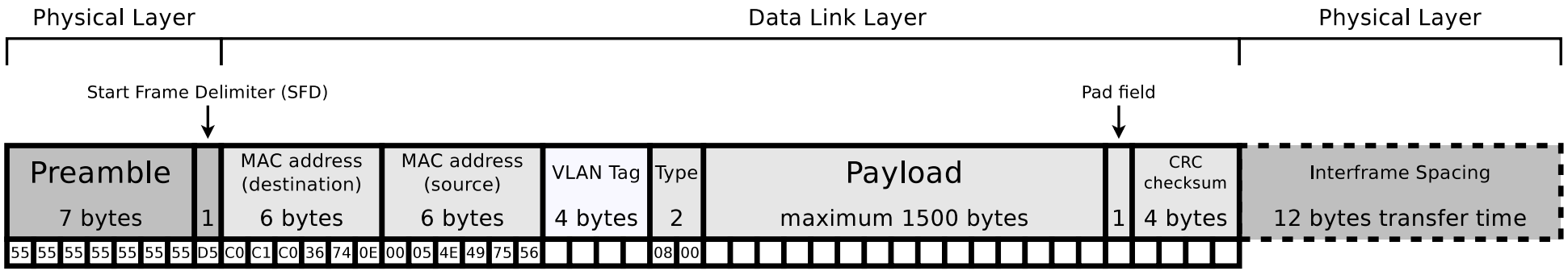
Framing in current Computer Networks (2/8) – Ethernet



- The fields for the physical addresses (MAC addresses) of sender and destination are 6 bytes long each
- The 4 bytes long optional VLAN tag contains, among others...
  - a 12 bits long VLAN ID ($\Longrightarrow$ see slide set 4)
  - and a 3 bits long field for the priority information
- The field `Type` contains the information what protocol is used in the next upper protocol layer

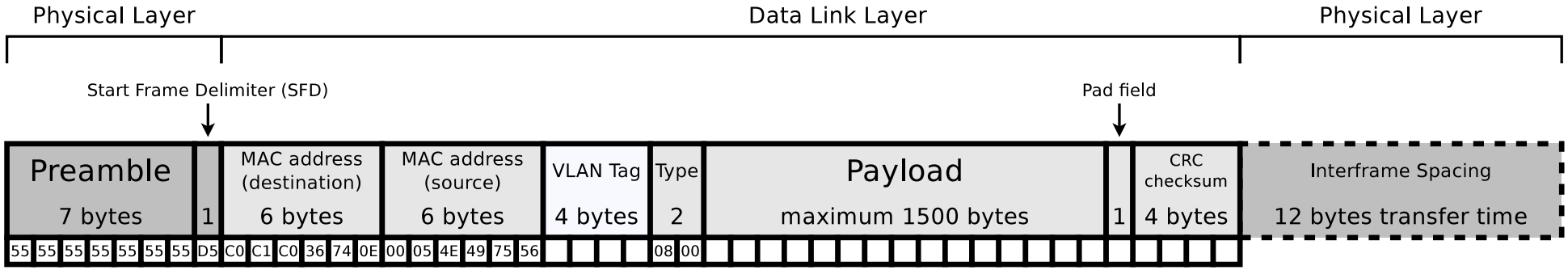
- If IPv4 is used, the field `Type` has value `0x0800`
- If IPv6 is used, the field `Type` has value `0x86DD`
- If the payload contains an ARP message, the field `Type` has value `0x0806`

Framing in current Computer Networks (3/8) – Ethernet



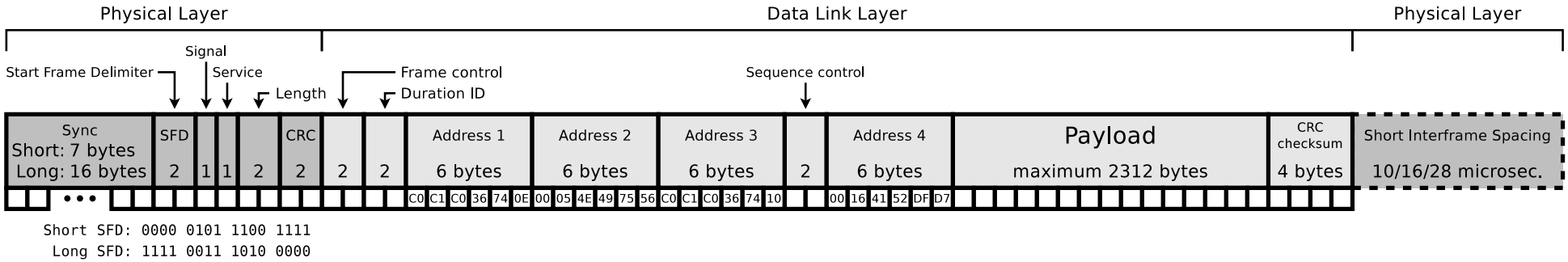
- Minimum size of an Ethernet frame: 72 bytes
  - Maximum size (including preamble and SFD): 1526 bytes
  - The VLAN tag increases the maximum size by 4 bytes
- Each frame can contain a maximum of 1500 bytes payload
  - With the Pad field, the frame length can be increased to the minimum frame size (72 bytes) when needed
    - This is required to get the collision detection via CSMA/CD working ($\Longrightarrow$ slide set 6)
- The last field contains a checksum (32 bits) for all fields, except the preamble and SFD

Framing in current Computer Networks (4/8) – Ethernet



- The **Interframe Spacing** or **Interframe Gap** is the minimum idle period between the transmission of Ethernet frames via the transmission medium
- The minimum idle period is 96 bit times (12 bytes)
  - It is 9.6 microseconds when using 10 Mbps Ethernet
  - It is 0.96 microseconds when using 100 Mbps Ethernet
  - It is 96 nanoseconds when using 1 Gbps Ethernet
- Some network devices allow to reduce the Interframe Spacing period
  - Benefit: Better data rate is possible
  - Drawback: For the receiver it may become impossible to detect the frames' borders ($\Longrightarrow$ the number of collisions may rise)

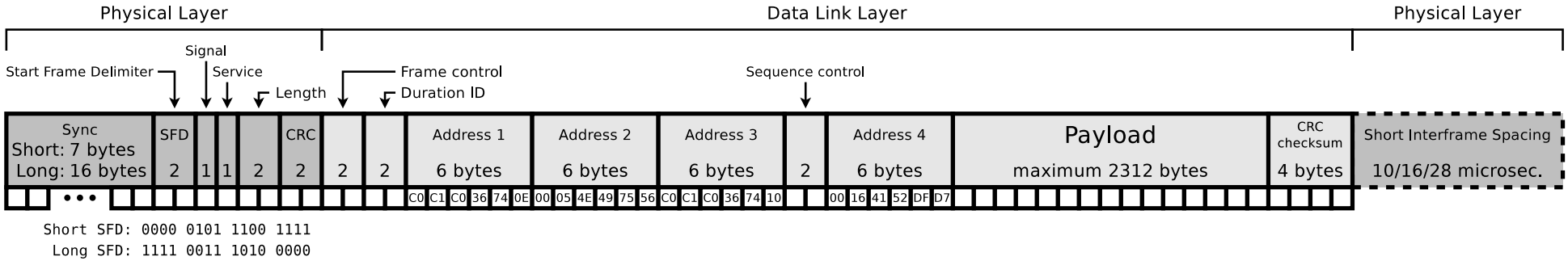
Framing in current Computer Networks (5/8) – WLAN



- In physical layer, the protocol adds. . .
  - a preamble to synchronize the receiver including a **SFD**
    - The **Short Preamble Format** is an optional standard which is not supported by all devices
  - a **Signal** field, which specifies the data rate (Mbps)
  - a **Service** field, which may contains some information about the modulation method used
  - a **Length** field, which specifies the time in microseconds which is required to transmit the frame
  - a **CRC** field, which contains a checksum over the fields Signal, Service and Length

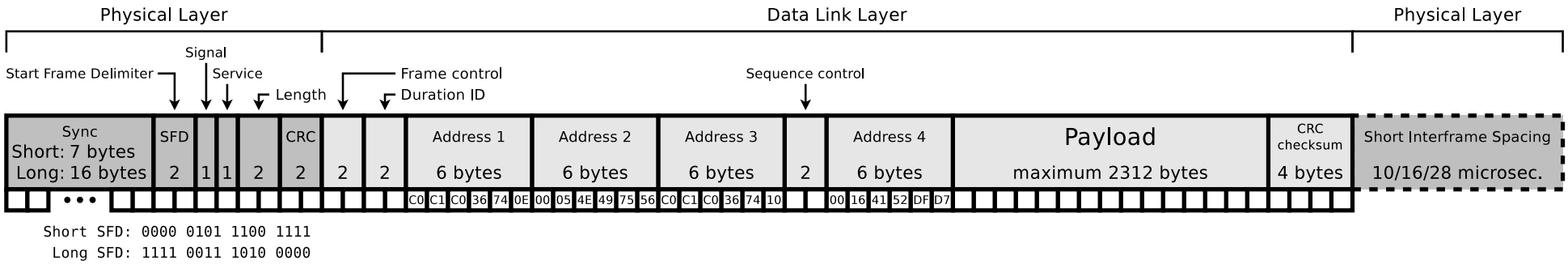
Framing in current Computer Networks (6/8) – WLAN

- Maximum size of a WLAN frame (only the DLL part): 2346 bytes



- The field Frame Control (2 bytes) contains several smaller fields
  - Among other things, here is the protocol version specified and it is specified what sort of frame (e.g. data frame or beacon) this frame is and if the frame is encrypted with the WEP method
- The field Duration ID (2 bytes) contains a duration value for the update of the counter variable **Network Allocation Vector** (NAV)

Framing in current Computer Networks (7/8) – WLAN

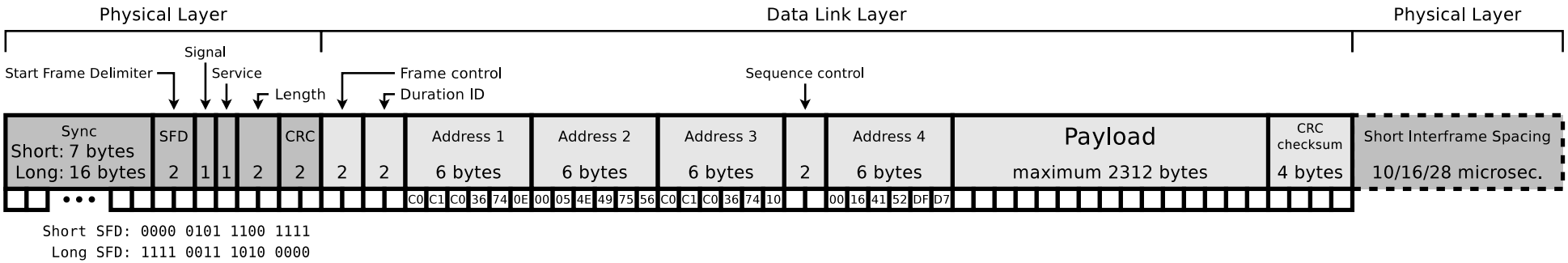


- The content of the 4 address fields depends on whether the WLAN network runs in **infrastructure mode** or **ad-hoc mode** and if the frame has been sent from a terminal device or an Access Point
- The MAC addresses are called **Basic Service Set Identifier** (BSSID) in this context

Possible values (MAC addresses) in the address fields

- MAC of sender (**Source Address**)
- MAC of receiver (**Destination Address**)
- MAC of next station on the route to the destination (**Receiver Address**)
- MAC of last station which forwarded the frame (**Transmitter Address**)

Framing in current Computer Networks (8/8) – WLAN



- The field Sequence Control (2 bytes) consists of a fragment number (4 bits) and a sequence number (12 bits)
  - If a frame has been split into several fragments, the sequence number is equal for all fragments
- The final field contains a CRC checksum (32 bits) for all fields, except the payload

Failure Causes

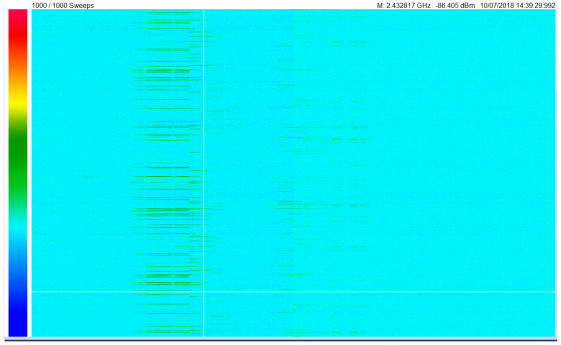
- During the transmission of bit sequences, errors may occur
- Failures are typically caused by...
  - **Signal deformation**
    - Attenuation of the transmission medium
  - **Noise**
    - Thermal or electronic noise
  - **Crosstalk**
    - Unwanted influence neighboring channels (e.g. capacitive coupling)
    - Capacitive coupling increases with increasing frequency

The next slide presents an example for such a cause of error

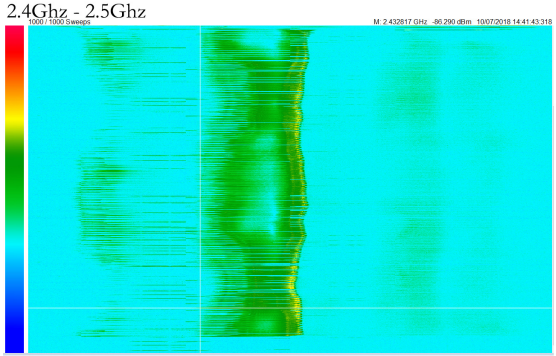
  - **Short-time disturbances**
    - Multiple bits in a row are incorrect (burst error)
    - Cosmic radiation
    - Defective or insufficient insulation
- It is important that errors are at least detected
  - Depending on the application, error correction may also be necessary

“Why my microwave makes me lose Wi-Fi connection”

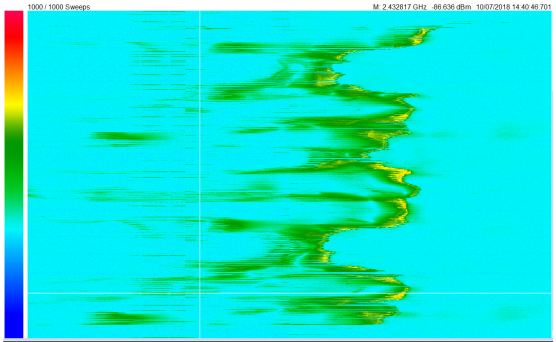
Background Wi-Fi
2.4Ghz - 2.5Ghz



Microwave on with cup of water placed in center of turntable
2.4Ghz - 2.5Ghz



Microwave on with cup of water placed on edge of turntable
2.4Ghz - 2.5Ghz



This is a graph of the 2.4Ghz to 2.5Ghz spectrum over a period of 20 seconds of microwave usage The horizontal direction is frequency, the vertical is time, color is power. Blue was around -80db and yellow around -60db.

The top graph is the baseline and on it you can see some Wi-Fi packets around the 2.3Ghz and 2.45Ghz-2.6Ghz range. Following is the first test I did where I placed a ceramic mug filled with water directly in the center of the turn table so that it didn't shift its xy position inside the microwave very much. The last graph is from when I put the ceramic mug of water on the very edge of the turn table so that it would have the most xy motion inside the microwave.

To address people concerned for my safety because my microwave is leaking: It isn't, microwaves can't be perfectly insulated and they have a fuck load of power to dampen. Inside the microwave, it's like 1000w of radiated power which is like 60db. Outside the microwave, I only measured at most -60db. That is about 0.000000001W of radiation power. Granted the antenna I used wasn't perfect so the actual ambient power level was higher. But still vastly below any amount of power that could cause harm.

Source: `https://www.reddit.com/r/dataisbeautiful/comments/8xu89b/why_my_microwave_makes_me_lose_wifi_connection/`

Error-detection Codes

- For error detection, the sender attaches a **checksum** at each frame
  - This way, frames with errors are detected and thrown away by the receiver

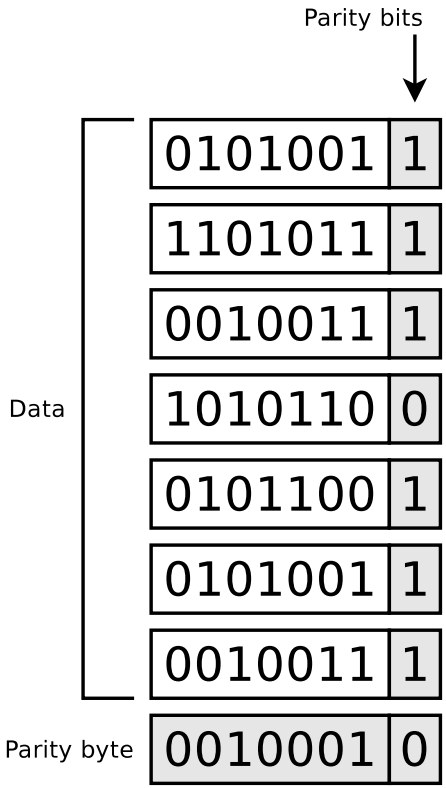
The Data Link Layer protocols do not provide the functionality, that the receiver can request previously discarded frames at the sender

This functionality provides the Transport Layer protocol TCP ($\Longrightarrow$ slide set 9)

- Possible ways to detect errors:
  - **Two-dimensional parity-check code**
  - The polynomial code – **Cyclic Redundancy Check** (CRC)

Two-dimensional Parity-check Code (1/2)

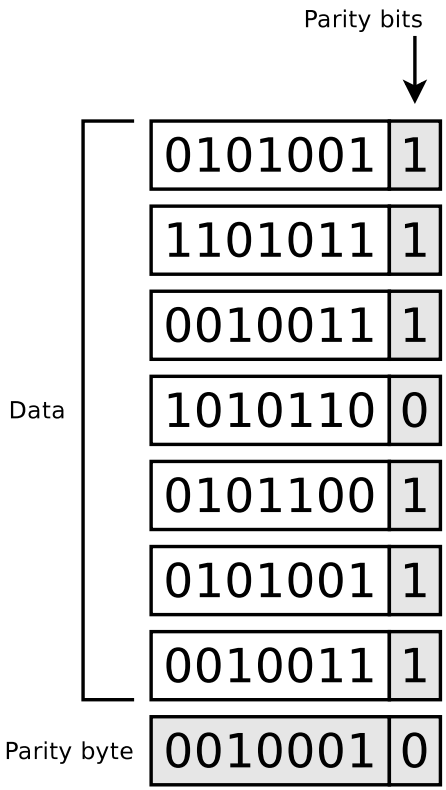
- This method is well-suited, especially when the 7-bit US-ASCII character encoding is used



- For each 7-bit section, an additional **parity bit** is calculated and attached to balance out the number of 1 bits in the byte
  - If the protocol defines even parity, the parity bit is used to obtain an even number of 1 bits in every byte
  - If odd parity is desired, the parity bit is used to obtain an odd number of 1 bits in every byte

$\Longrightarrow$ **one-dimensional parity-check code**

Two-dimensional Parity-check Code (2/2)



- For each byte exists, in addition to the parity bits, an additional **parity byte**
  - The content of the parity byte is calculated over each byte of the frame

  $\Longrightarrow$ **two-dimensional parity-check code**
- All 1-bit, 2-bit and 3-bit errors and most of the 4-bit errors can be detected via two-dimensional parity-check codes

Source: Computernetzwerke, *Larry L. Peterson, Bruce S. Davie*, dpunkt (2008)

- BISYNC is an example for a protocol, which uses this method for the transmission of data, encoded with the 7-bit US-ASCII character encoding

Cyclic Redundancy Check (CRC)

- Bit sequences can be written as polynomials with the coefficients 0 and 1
- A frame with k bits is considered as a polynomial of degree $k-1$
  - The most significant bit is the coefficient of x^{k-1}
  - The next bit is the coefficient of x^{k-2}
  - ...
- Example: The bit sequence 10011010 corresponds to this polynomial:

$$\begin{aligned} M(x) &= 1*x^7+0*x^6+0*x^5+1*x^4+1*x^3+0*x^2+1*x^1+0*x^0 \\ &= x^7+x^4+x^3+x^1 \end{aligned}$$

- Sending and receiving messages can be imagined as an **exchange of polynomials**

Polynomials in mathematics

A polynomial is an expression which consists variables and coefficients and non-negative integer exponents

Cyclic Redundancy Check (Method)

- The Data Link Layer protocol specifies a **generator polynomial** $C(x)$
  - The choice of the generator polynomial determines, which error types can be detected
- $C(x)$ is a polynomial of degree k
  - If e.g. $C(x) = x^3 + x^2 + x^0 = 1101$, then $k = 3$
    - Therefore, the degree of the generator polynomial is 3

The degree of the generator polynomial is equal to the number of bits minus one

- If for a frame, the CRC checksum need to be calculated, n 0 bits are appended to the frame
  - n corresponds to the degree of the generator polynomial

Selection of common Generator Polynomials

Name	$C(x)$	Application
CRC-5	$x^5 + x^2 + x^0$	USB
CRC-8	$x^8 + x^2 + x^1 + x^0$	ISDN
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + x^0$	Ethernet

Example of Cyclic Redundancy Check (1/4)

Generator polynomial:	100110

- The generator polynomial has 6 digits
  - Therefore, five 0 bits are appended

Frame (payload):	10101
Frame with appended 0 bits:	1010100000

- The frame with the appended 0 bits is divided from the left only via XOR by the generator polynomial
  - `1 XOR 1 = 0, 1 XOR 0 = 1, 0 XOR 1 = 1, 0 XOR 0 = 0`
  - Always start with the first common 1
  - The remainder (remainder polynomial) is the **checksum**

```
The sender
calculates
the checksum


1010100000
100110||||
------vv||
  110000||
  100110||
  ------v|
   101100|
   100110|
   ------v
     10100 = Remainder
```

Example of Cyclic Redundancy Check (2/4)

- The checksum is appended to the payload
  - The length of the remainder must be n bits
  - n is the degree of the generator polynomial
- Result: 10100 will be appended to the frame
- Transmitted frame including checksum (code polynomial): 1010110100

Generator polynomial:	100110
Frame (payload):	10101
Frame with appended 0 bits:	1010100000
Remainder (remainder polynomial):	10100
Transferred frame (code polynomial):	1010110100

fill the remainder

- The remainder must consist n bits and n is the degree of the generator polynomial
- If the remainder is shorter than n, it must be *filled* with zeros
- Example: The remainder is 101 and n is 5, then the checksum to appended is 00101

Example of Cyclic Redundancy Check (3/4)

Transferred frame (code polynomial):	1010110100
Generator polynomial:	100110

- The receiver of the frame is able to verify, if the frame did arrive error-free
- With a division (only via XOR) through the generator polynomial, transmissions with errors are detected
  - For division with XOR, always start with the first common 1
- If the remainder of the division is 0, then the transmission was error-free

```
The receiver
verifies if the
transmission
was error-free
```

```
1010110100
100110||||
------vv||
  110101||
  100110||
  ------v|
   100110|
   100110|
   ------v
         0
```

Example of Cyclic Redundancy Check (4/4)

Transferred frame (code polynomial):	1110110100
Generator polynomial:	100110
Correct Transmission:	1010110100

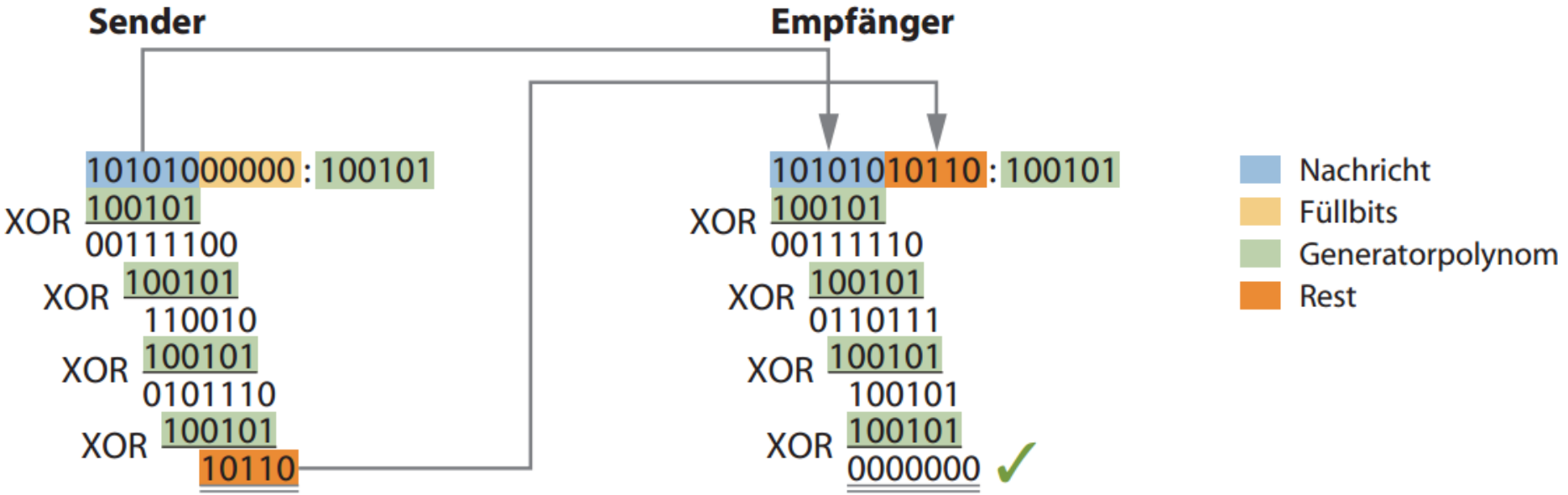
- If the transmitted frame contains a defective bit, the remainder of the division via XOR not 0
- This method identifies all 1-bit errors, every odd number of defective bits, and all burst errors of length n
  - n is the degree of the generator polynomial

```
1110110100
100110|||
------v|||
 111010|||
 100110|||
 ------v||
  111001||
  100110||
  ------v|
   111110|
   100110|
   ------v
    110000
    100110
    ------
     10110
```


Sources

- Hompel, Büchter, Franzke. *Identifikationssysteme und Automatisierung*. Springer. 2008. P.219-221
- Meinel, Sack *Digitale Kommunikation: Vernetzen, Multimedia, Sicherheit*. Springer. 2009. P.122-124

Summary of the CRC-Method (in German language)



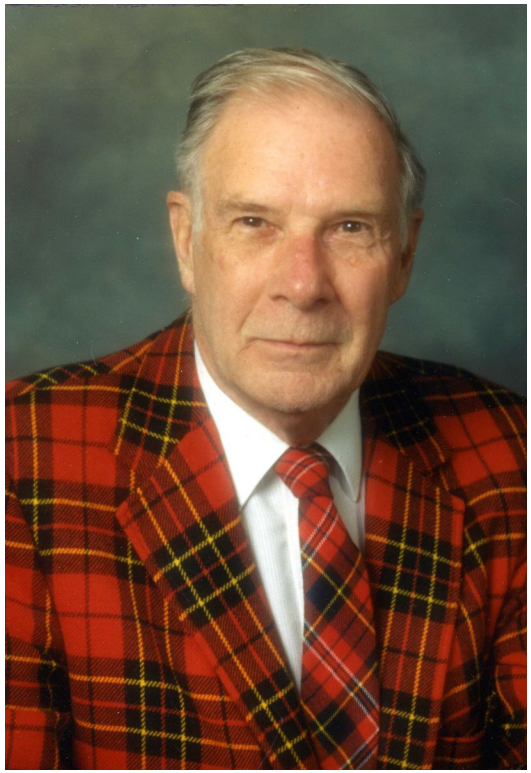
Der CRC-Prüfwert ist der Rest der Division der Nachricht durch das Generatorpolynom. Dieselbe Rechnung mit der Nachricht plus angehängtem CRC ergibt beim Empfänger 0, wenn kein Übertragungsfehler aufgetreten ist.

Source: CRCs berechnen mit Intrinsics. Oliver Lau. c't 9/2013. P.184-185

Error-correction Codes

Image Source: http://amturing.acm.org/images/hamming-3.jpg

- Transmission errors can be detected via CRC checksums
  - If it is important also to correct errors, the data to be transmitted must be encoded in a way, that error-correction is possible
- Error correction can be realized via **Hamming code**
  - Named after the mathematician Richard Wesley Hamming (1915-1998)



Richard Hamming (1915 – 1998)

None of the established Data Link Layer protocols provide error correction

They all provide only error detection!

Hamming Code – Procedure: Position of the Parity Bits

In this course, we discuss a simplified version of the Hamming Code which is sometimes called Hamming-ECC

`http://www.mathcs.emory.edu/~cheung/Courses/455/Syllabus/2-physical/errors-Hamming.html`
`http://de.wikipedia.org/w/index.php?title=Fehlerkorrekturverfahren`
Andrew Tanenbaum. *Computernetzwerke*. Pearson (2000). 3rd edition. P. 208-209

- The bits of a bit sequence are numbered starting with 1 from the left
  - Bits, which are powers of 2 (1, 2, 4, 8, 16, etc.) are parity bits
    - The remaining bits are the payload
- Example:
  - 8 bits payload: 01001100

```
Position:  1  2  3  4  5  6  7  8
 Payload:  0  1  0  0  1  1  0  0

Position:  1  2  3  4  5  6  7  8  9 10 11 12
Data to be transmitted:  ?  ?  0  ?  1  0  0  ?  1  1  0  0
```

- The sender calculates the parity bits values

Hamming Code – Procedure: Parity Bits Values (1/2)

- Each bit position in the transmission is assigned to a position number
- In this example, the position number is a 4-digit number, because we have 4 parity bits
- Examples:

$$
\begin{aligned}
&\text{Position: } 1 \quad \Longrightarrow \quad \text{Value: } 0001 \\
&\text{Position: } 2 \quad \Longrightarrow \quad \text{Value: } 0010 \\
&\text{Position: } 3 \quad \Longrightarrow \quad \text{Value: } 0011 \\
&\text{Position: } 4 \quad \Longrightarrow \quad \text{Value: } 0100 \\
&\text{Position: } 5 \quad \Longrightarrow \quad \text{Value: } 0101 \\
&\ldots
\end{aligned}
$$

Hamming Code – Procedure: Parity Bits Values (2/2)

- Next, XOR of the values of those positions (in 4-digit format), where the payload is a 1 bit in the transmission, is calculated
  - In the example it is position 5, position 9 and position 10

```
                Position:  1  2  3  4  5  6  7  8  9 10 11 12
Data to be transmitted:    ?  ?  0  ?  1  0  0  ?  1  1  0  0

     0101   Position 5
     1001   Position 9
 XOR 1010   Position 10
 --------
  =  0110
```

- The result are the values of the parity bits
  - These are inserted into the transmission

```
                Position:  1  2  3  4  5  6  7  8  9 10 11 12
Data to be transmitted:    0  1  0  1  1  0  0  0  1  1  0  0
```

Hamming Code – Procedure: Check Data (1/2)

- The receiver can verify if a bit sequence is correct
  - It calculates XOR of the calculated and received parity bits
    - The parity bits are position 1, 2, 4 and 8

```
Received data: 1  2  3  4  5  6  7  8  9 10 11 12
               0  1  0  1  1  0  0  0  1  1  0  0


    0101   Position 5
    1001   Position 9
XOR 1010   Position 10
--------
    0110   Parity bits calculated
XOR 0110   Parity bits received
--------
=   0000   => Correct transmission
```

Hamming Code – Procedure: Check Data (2/2)

```
   Received data: 1  2  3  4  5  6  7  8  9 10 11 12
                  0  1  0  1  0  0  0  0  1  1  0  0

    1001   Position 9
XOR 1010   Position 10
--------
    0011   Parity bits calculated
XOR 0110   Parity bits received
--------
=   0101   => Bit 5 is defective!
```

- Possible results of the calculation:
  - Position number of the modified bit
  - 0 if the transmission was correct
- If ≥ 2 bits have been changed, the only statement that can be made is, that bits have been changed at all
  - The positions can not be determined this way

Conclusion for Error Detection Codes

- Encoding data with error correction codes is useful, where transmission retries cannot be requested
  - Example: The network only consists of a simplex channel
- Typically, only error detection codes are used
  - If an error is detected, the frame is just thrown away
    - None of the established Data Link Layer protocols provide error correction
    - The Data Link Layer protocols provide only error detection!
    - The Transport Layer protocol TCP requests again transmissions with errors
- If only **error detection** is supported, and not **error correction**, **lesser parity bits need to be transmitted**
  - For this reason, **error detection is more efficient than error correction**

See the model calculation for error detection/correction on the next slide

Model calculation for error detection/correction

- Block size: 1,000 bits
- 1 Megabit data (1,000 blocks) shall be transferred
- **Error detection**
  - A single parity bit at the end of each block is enough to detect if the block contains errors
    - This increases the size of each block by 1 bit to 1.001 bits
  - After 1,000 blocks, an extra block (1.001 bits) need to be transmitted
  - Therefore, 1 Megabit data requires 2,001 parity bits
- **Error correction** (Hamming code)
  - To correct errors in 1.000 bits data, 10 parity bits are required
    - The parity bits are positions 1, 2, 4, 8, 16, 32, 64, 128, 256 and 512
  - Therefore, 1 Megabit data requires 10,000 parity bits

Source

Andrew Tanenbaum, David Wetherall *Computernetzwerke*. Pearson (2012). 6th edition. P. 253

Exercise

1. 8 bits (10011010) of payload exist
   - What is the data to be transmitted (including payload and parity bits)
2. Verify that the following messages were transmitted correctly
   1. 00111101
   2. 101110100010
   3. 001101100100
   4. 0001101100101101

$\Longrightarrow$ Exercise sheet 3

Reliable Transmission through Flow Control

- Via flow control, the receiver controls the **transmission speed of the sender** dynamically, and ensures this way the **integrity of the transmission**
  - Receivers with poor performance are not flooded with data, they are unable to handle
    - As a result data would be lost
  - During transmission, lost data is transmitted again
- Procedure: **Transmission retries** occur when they are needed
  - Basic mechanisms:
    - **Acknowledgements** (ACK) as feedback
    - **Timeouts**
  - Concepts of flow control:
    - **Stop-and-Wait** ($\Longrightarrow$ see slide set 9)
    - **Sliding-Window** ($\Longrightarrow$ see slide set 9)

- Ethernet does not implement flow control mechanisms on Data Link Layer
- In practice, if flow control is required, it is provided by the Transport Layer protocol TCP
- **Flow Control in the Data Link Layer will not be discussed in this course**