Computer Networks

Data Link Layer - Logical Link Control

Prof. Dr. Oliver Hahm

2024-12-12

Agenda

Error Control

- How can errors occur?
- What are the consequences?
- What can be done?

Failure Causes

Failure Causes

During the transmission of bit sequences on the physical layer errors may occur

They are typically caused by…

- Signal deformation

  - Attenuation of the transmission medium
- Noise
  - Thermal or electronic noise
- Crosstalk
  - Interference by neighboring channels
  - Capacitive coupling increases with increasing frequency
- Short-time disturbances
  - Cosmic radiation
  - Defective or insufficient insulation

Typical BER values

POTS	$2 * 10^{-4}$
Radio link:	10^{-3} – 10^{-4}
Ethernet:	10^{-9} – 10^{-10}
Fiber:	10^{-10} – 10^{-12}

Burst errors are more common than single bit errors

The LLC sublayer ensures that errors are detected and handled

Error Detection

How can we detect errors?

Checksum

Checksum

The checksum is calculated by a pre-defined algorithm for a block of data. They are typically used for the verification of the data integrity.

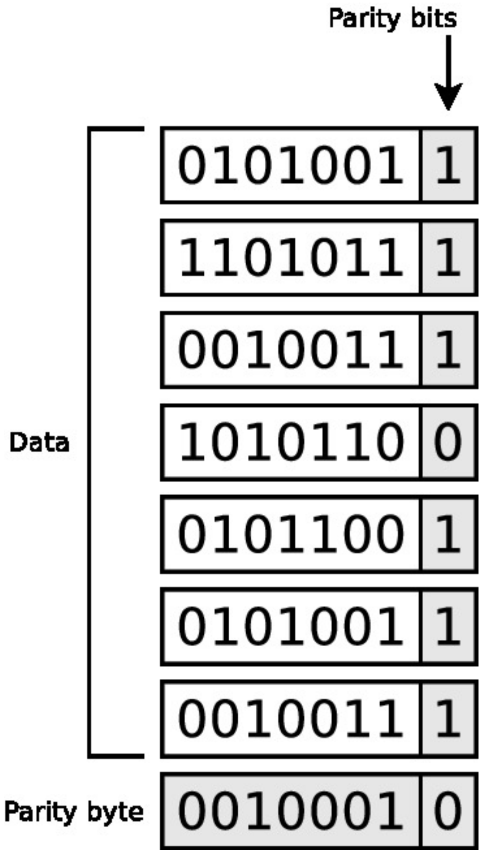
- For error detection, the sender attaches a checksum at each frame
- The receiver can now detect erroneous frames and discard them
- Possible checksums:
  - Parity-check codes
  - The polynomial code – Cyclic Redundancy Checks (CRCs)

How many bits do we require for the checksum?

Hamming Distance

- Each message (→ codeword) of n bits contains m bits of payload and r bits of checksum (with $n = m + r$ and $r > 0$)
- Typically all 2^m data messages are allowed, but not all 2^n codewords are valid
- The minimum distance between two valid codewords is called the Hamming distance
  - In order to detect d errors, the distance needs to be $d + 1$
    - → d *flipped* bits won't create another valid codeword
  - In order to correct d errors, the distance needs to be $2d + 1$
    - → The resulting word with d *flipped* bits is still closer to the original codeword than to any other

One-dimensional Parity-check Code

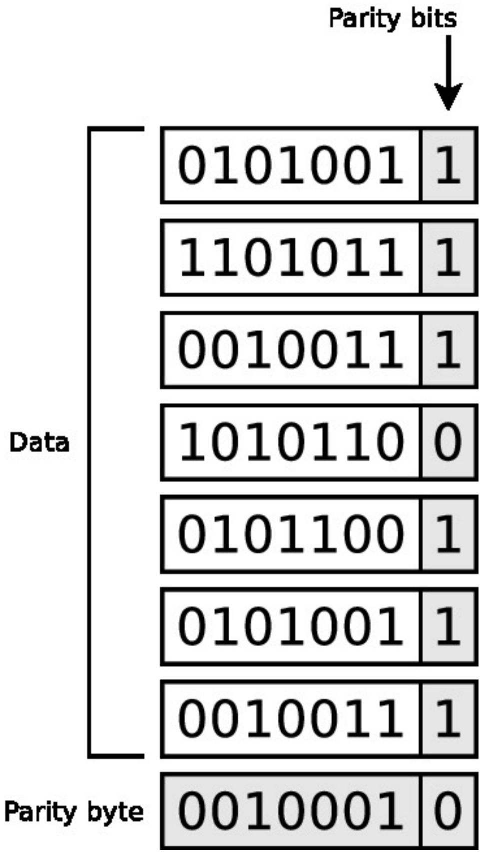


- Well-suited for short blocks of data, e.g., 7-bit US-ASCII characters
- For each 7-bit section, an additional parity bit is calculated and attached to balance out the number of 1 bits in the byte
  - If the protocol defines even parity, the parity bit is used to obtain an even number of 1 bits in every 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

What is the Hamming Distance here?

Two-dimensional Parity-check Code



- For all byte exists 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)

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:

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

Reminder

A polynomial is an expression which consists of variables and coefficients and non-negative integer exponents, e.g., $P(x) \quad = \quad a*x^2+b*x^1+c*x^0$

CRC Generator Polynomial

- The CRC specification defines a generator polynomial $C(x)$
  - The degree of the generator polynomial determines how many bit errors 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

Selection of common Generator Polynomials

- CRC-5
  Polynomial: $x^5 + x^2 + x^0$
  Representation: 0x05
  Application: USB
- CRC-8
  Polynomial: $x^8 + x^7 + x^5 + x^2 + x^1 + x^0$
  Representation: 0xA7
  Application: Bluetooth
- CRC-16-IBM
  Polynomial: $x^{16} + x^{15} + x^2 + x^0$
  Representation: 0x8005
  Application: Bisync, Modbus
- CRC-16-CCITT
  Polynomial: $x^{16} + x^{12} + x^5 + x^0$
  Representation: 0x1021
  Application: HDLC
- CRC-32
  Polynomial: $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x^1+x^0$
  Representation: 0x04C11DBB7
  Application: Ethernet

CRC Example: Computation

Generator polynomial:

Frame (payload):

$x^5 + x^2 + x^0 \Rightarrow$ `100101`

$21 \Rightarrow$ `10101`

- The generator polynomial has 6 digits $\Rightarrow$ five `0` bits are appended

Frame (payload):

Frame with appended 0 bits:

`10101`

`10101 00000`

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

Reminder: *XOR* operation

```
1 XOR 1 = 0
1 XOR 0 = 1
```

```
0 XOR 1 = 1
0 XOR 0 = 0
```

The sender calculates the checksum

```
1010100000
100101||||
------vv||
  111100||
  100101||
  ------v|
   110010|
   100101|
   ------v
    101110
    100101
    ------
      1011 = Remainder
```

Padding

- The remainder must consist of n bits if n is the degree of the generator polynomial
- If the remainder is shorter than n, it must be *filled* with zeros

- The checksum is appended to the payload

- **Result**: `01011` will be appended to the frame
- Transmitted frame including checksum (code polynomial): `1010101011`

Generator polynomial:	`100101`
Frame (payload):	`10101`
Frame with appended 0 bits:	`1010100000`
Remainder:	`1011`
Transferred frame (code polynomial):	`1010101011`

CRC Example: Verification Transmission without error

Transferred frame (code polynomial):	`1010101011`
Generator polynomial:	`100101`

- The receiver of the frame is able to verify, if the frame did arrive error-free
- By dividing (only via `XOR`) by 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

Verification (at the receiver)

```
1010101011
100101||||
------vv||
  111110||
  100101||
  ------v|
   110111|
   100101|
   ------|
    100101
    100101
    ------
         0
```

CRC Example: Verification Transmission with error

Transferred frame (code polynomial):	`1110101011`
Generator polynomial:	`100101`
Correct Transmission:	`1010101011`

- If the transmitted frame contains a defective bit, the remainder of the division via `XOR` not `0`
- CRC *cannot* detect all errors

Verification (at the receiver)

```
1110101011
100101||||
------v|||
 111110|||
 100101|||
 ------v||
  110110||
  100101||
  ------v|
   100111|
   100101|
   ------v
       101
```

Properties of CRCs

Most important characteristic

A polynomial code with r check bits will detect all burst errors of length $\leq r$

- If the error consists of a multiple of the polynomial code of the used CRC it will not be detected
- **CRC-16-CCITT** for example will detect
  - All single, double and three-bit errors
  - All error samples with odd number of bit errors
  - All error bursts with up to 16 bits (see above)
  - 99.997 % of all 17-bit error bursts
  - 99.998 % of all error bursts with lengths $\geq$ 18
- Calculation of CRC can be implemented by a simple shift register circuit in hardware

Error Correction

Forward Error Correction (FEC)

- Error correction requires more redundant information to be added compared to error detection
- Upon error detection the frame typically needs to be retransmitted
- ⇒ For somewhat reliable transmission channels simple error detection is cheaper
- ⇒ For error-prone transmission media (→ wireless communication) error-correction may be cheaper, because it reduces the amount of retransmissions
- (Forward) Error Correction can be realized via Hamming code
  - Named after the mathematician **Richard Wesley Hamming** (1915-1998)

Simple Example of Error Correction

Remember

In order to correct d errors a code needs a *Hamming distance* of $2d + 1$

- Assume a code with only four valid codewords
  - $w_1 = 0000000000$
  - $w_2 = 0000011111$
  - $w_3 = 1111100000$
  - $w_4 = 1111111111$
- ⇒ The Hamming distance is 5
  - It can detect up to four bit errors
  - It can correct up to two bit errors
- **Example**:
  - If 0000000111 is received, the original must be 0000011111 (correct)
  - If 0000000000 is changed to 0000000111, the error is not corrected properly

Flow Control

Reliable Transmission through Flow Control

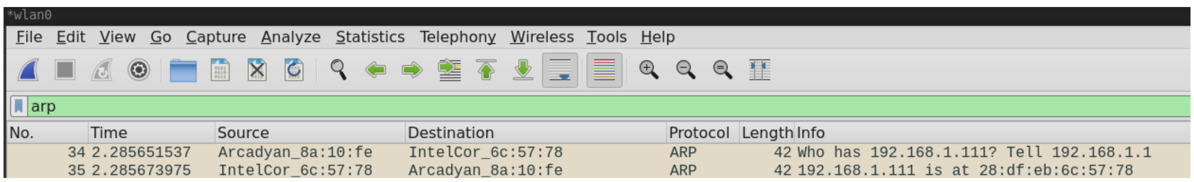
- Flow control allows the receiver to negotiate the transmission speed with the sender dynamically
  - Less powerful receivers or receivers under high load are not flooded with data

    - ∗ If a host receives data at a higher rate than it can handle it, data will get discarded and is lost
  - Concepts of flow control:
    - ∗ **Stop-and-Wait**
    - ∗ **Sliding-Window**

- Ethernet does not implement flow control mechanisms on Data Link Layer

Address Resolution

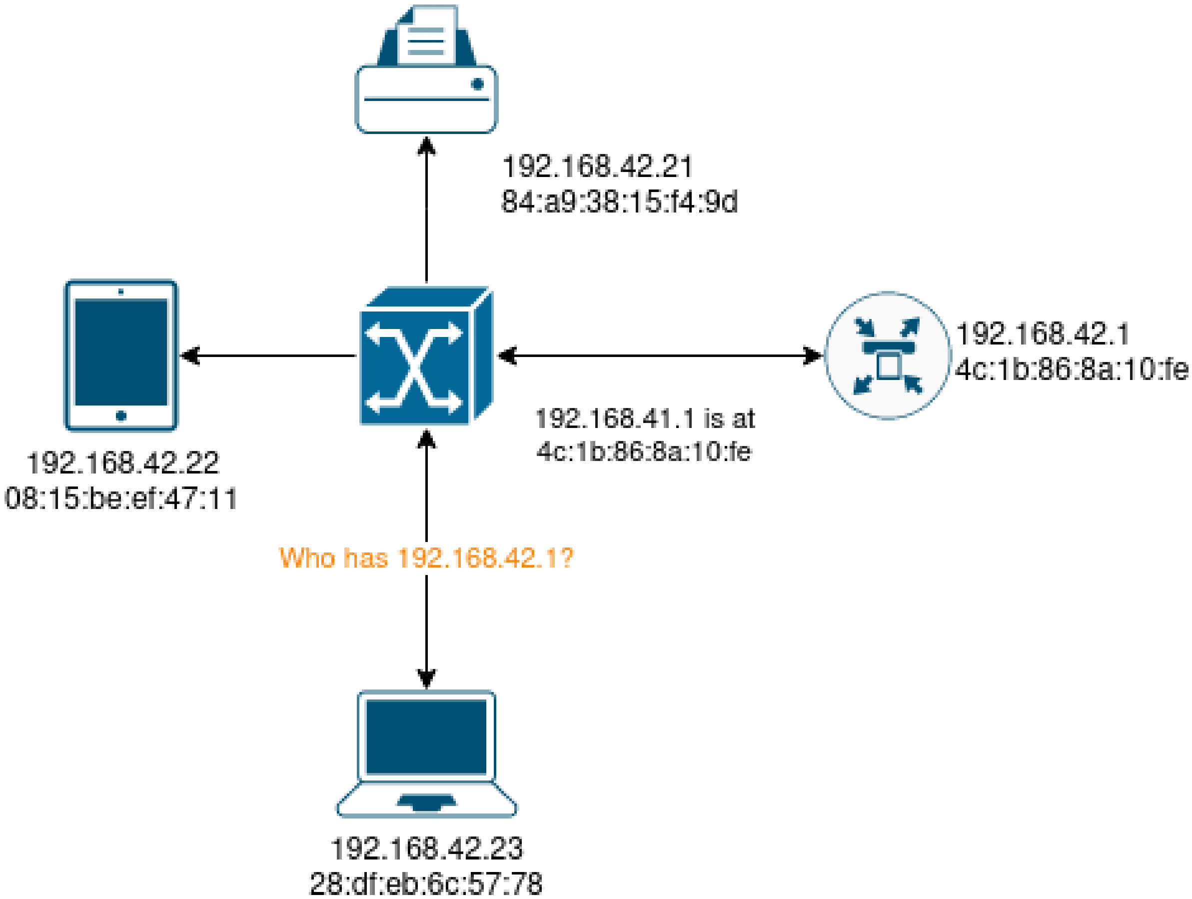
Address Resolution



- The network layer requires a mapping between physical and logical network addresses
- For IPv4 the Address Resolution Protocol (ARP) is used to resolve IPv4 addresses to MAC addresses [1]
- For IPv6 the Neighbor Discovery Protocol (NDP) accomplishes the same

[1] In fact, the original ARP specification, RFC 825, was written for IPv4 and Ethernet, but the functioning is not bound to IPv4 or any particular layer 2 protocol.

IPv4: Address Resolution Protocol (ARP)



Simplified ARP message flow

ARP uses **broadcast** messages:

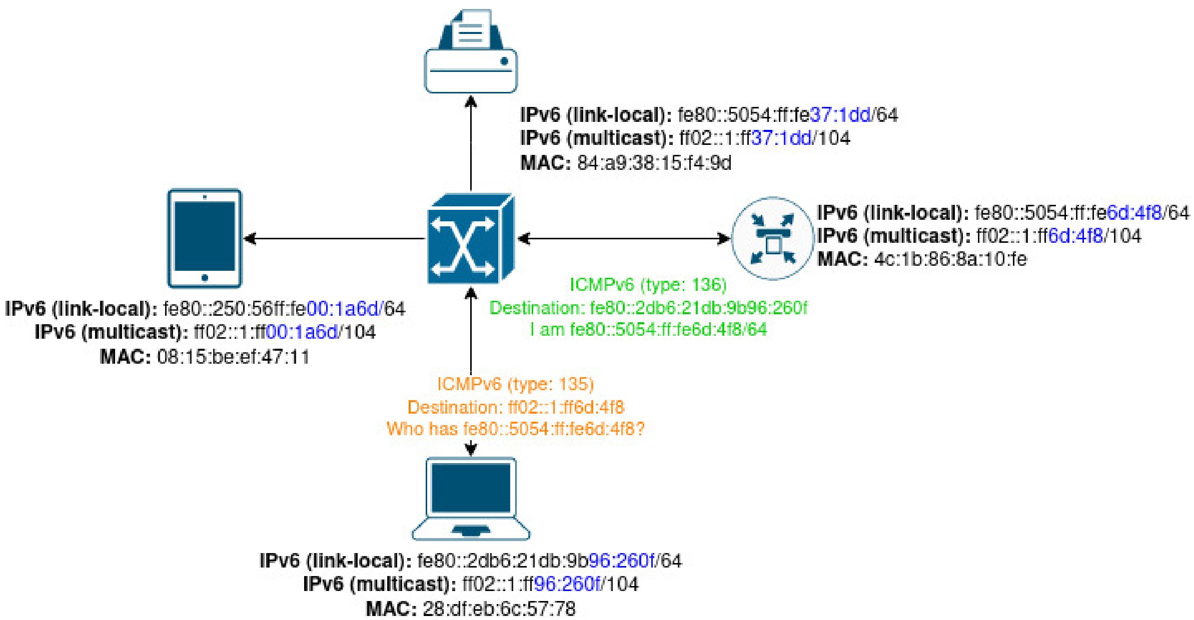
- To determine the MAC address of a network device in the LAN, it sends out a MAC broadcast frame containing the IP address
- Each network device that receives the frame compares this IP address to the address assigned to it
- If a network device has this IP address, it sends an ARP response to the sender via unicast
- The original sender can now map the source MAC address of the response to the searched IP address

IPv6: Neighbor Discovery Protocol (NDP)

In NDP routers and nodes can send proactively advertisements or be inquired via router and neighbor solicitations.

NDP uses **multicast** messages:

- NDP uses the **Solicited-node multicast address** → Prefix: `ff02::1:ff00:0000/104`
- Each node subscribes to this multicast group for each configured unicast address
- The remaining 24 bits are the final 24 bits of the corresponding unicast address
- Only nodes registered to this address will receive the **ICMP** message



Simplified NDP message flow

Neighbor Cache

- The Neighbor cache is used to speed up the address resolution
  - It contains a table with these information for each entry:
    - Layer 3 protocol type (e.g., IPv4)
    - Layer 3 address (e.g., its IP address)
    - Layer 2 address (MAC address)
    - Lifetime
  - The lifetime is set by the operating system
  - If an entry in the table is active, the lifetime is extended

The Neighbor cache can be displayed via `arp -n` or `ip neighbour`

```
# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.178.1            ether   9c:c7:a6:b9:32:aa   C                     wlan0
192.168.178.24           ether   d4:85:64:3b:9f:65   C                     wlan0
192.168.178.41           ether   ec:1f:72:70:08:25   C                     wlan0
192.168.178.25           ether   cc:3a:61:d3:b3:bc   C                     wlan0
```

Summary

You should now be able to answer the following questions:

- Which requirements need to be fulfilled to allow for error detection and correction?
- What is a CRC checksum and how does it work?
- For which purpose do we need ARP and NDP and how do they work?