Computer Networks
The Data Link Layer

2025/2026, 1st semester

Dr. Szilveszter Kovacs

E-mail: szilveszter.kovacs@uni-miskolc.hu

www.iit.uni-miskolc.hu/~szkovacs

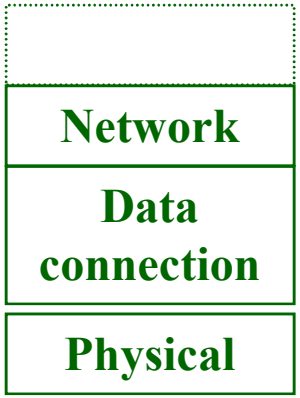
Institute of Information Technology 107/a.

Phone: +36 46 565-111 / 21-07

Content

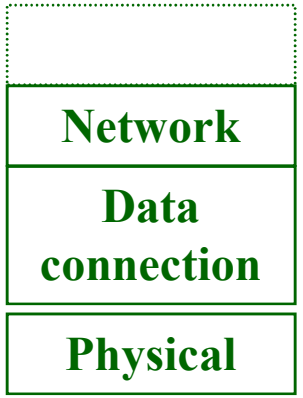
- General conceptss
- Framing – Frame recognition
- Error correction, could be
  - direct or indirect.
- Data flow control
- Elementary data link protocols

The data link layer



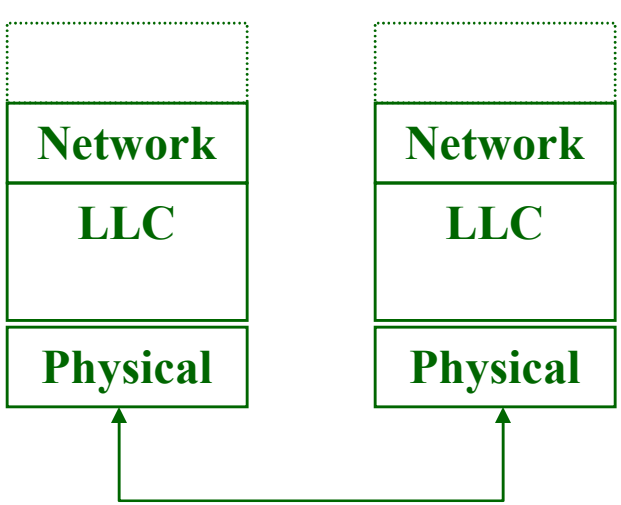
- **The second layer**
- **Task (general)**
  - **Well-defined interface and providing services to the network layer**
  - Ensuring **reliable, effective communication** between **two adjacent** stations connected by **"wire-like"** channel.

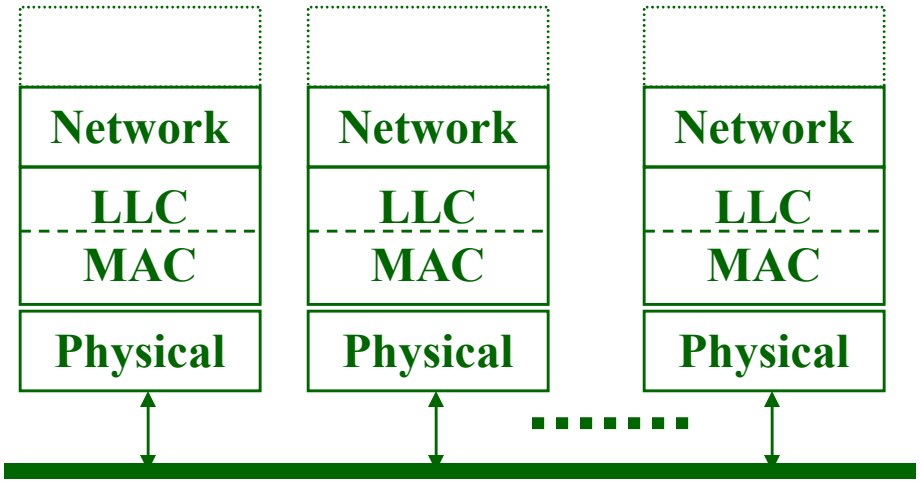
“Wire-like” channel



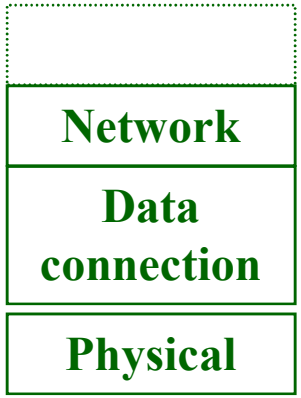
- **The bits arrive in exactly the same order as they were sent.**
- **A “wire-like” channel between two stations can be provided by**
  - **a point-to-point** connection, with the **physical layer,**
  - **broadcast** channel, with the **physical + media access sublayer.**

Logical Link Control
Media Access **Control**

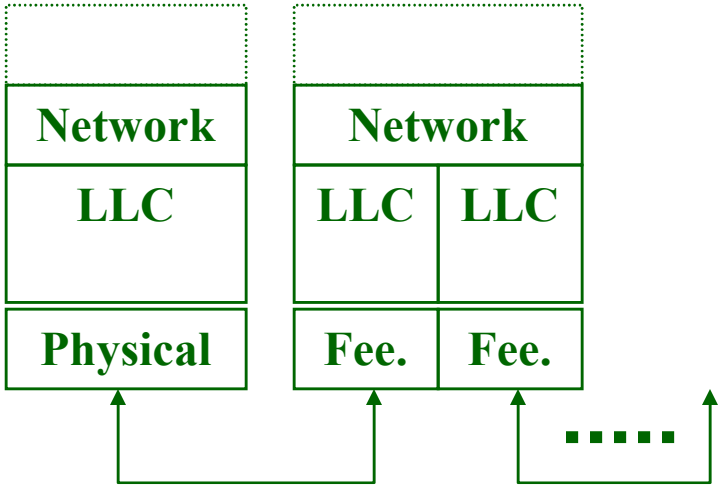


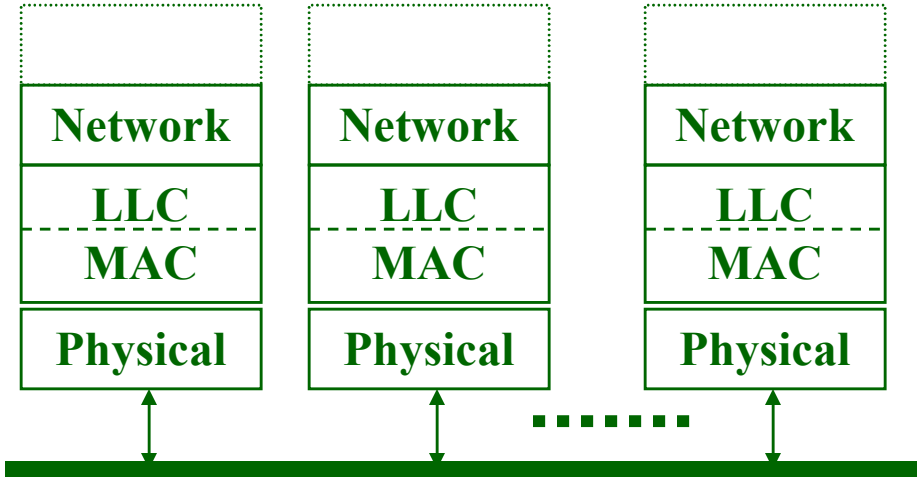


The data link layer



- **In case of point-to-point** connection, **only LLC**
  - **Logical Link Control**
- **In case of broadcast** channel **MAC + LLC**
  - **Media** Access **Control** + **Logical Link Control**





Problems to be solved by the data link layer

- Eliminating transmission errors
- The finite handling latency due to (possibly variable) data transfer speed
- Handling the finite processing capabilities of machines (possibly variable)
  - finite processing time,
  - finite reception capacity
    - fast transmitter can flood slow receiver (flooding) → traffic control

Services provided to the network layer

- **Reminder:**
  OSI **standard service primitives**:
  request; **indication**;
  response; **confirm**.
- **Possible service classes:**
  - **Connectionless service without acknowledgement**
  - **Acknowledged connectionless service**
  - **Acknowledged connection-based service**
- **The functions of the data link layer must provide these.**

Connection without acknowledgement - free service

- **Connection - free**
  - **Data units (frames) are sent independently of each other (frames are addressed independently)**
    (broadcast channels (MAC) are usually connection-free - LAN)
  - **Faster** (no connection setup, teardown)
- **Without feedback**

  ⇒ not necessarily error-free
  - **may be good for low error rates (lower overhead, no error handling required)**
  - **errors are handled by (some) higher layer (it notices, it repeats, etc.)**

Acknowledged connection-free service

- **Connection-free**
  - **Data units (frames) are sent independently of each other** (message broadcast channels (MAC) are usually connectionless - LAN)
  - **Faster** (no connection setup, disconnection)
- **Achieves error-free transmission**
  - **Receipts indicate the flawless arrival of frames.**
  - If there are typically more errors,
    it is better to handle error handling at a low level
    (less data needs to be repeated)
    (although introducing error filtering there means more overhead (loss) than not having it)

Handling transmission errors

- **Common errors, possibly caused by the MAC protocol itself** (e.g. CSMA contention) ⇒ **must be handled in the MAC protocol itself**
- **However, this is only part of the errors** (e.g. if the recipient does not exist, it will not be received even if there is no collision).
- **It is advisable to handle typical (frequent) errors at the lowest level where they can first be detected** (The higher the level at which error handling occurs, the greater the collateral loss resulting from errors)
- **But some level of error handling at lower levels does not exempt you from the need for error handling at higher levels! (They only improve efficiency.)**

Acknowledged connection-based service

- **The co-elements can have a “dialogue”.**
- **Data units** (frames) **arrive exactly once, in a controlled and correct order**
  - **Reliable frame stream with the same order of reception as the order of transmission**
- **Its phases:**
  - **establishing the connection** (call setup);
  - **transmission of frames** (mean communication);
  - **disconnection** (resource release).

Data Link Layer Functions

- **Framing and frame delimitation**
  - **Creating frames from network layer data and**
  - **delimiting frames from the physical bitstream.**
- **Fault protection**
  - **"Appropriate" coding of data, error detection,**
  - **sending and receiving receipts.**
- **Data flow control (flow control)**
  - **adjusting the transmission speed of the transmitter to the receiver's reception capacity (feedback → the transmitter should not unnecessarily load the channel)**
- **Relationship management**
  - **In the case of connection-based service, essential**
    - **establishing and breaking connections, reserving and releasing resources.**

Framing and frame delimitation

- **The usual operation of the data link layer:**
  - the bit stream received from the network layer **into discrete frames**, which **are provided with a checksum**;
  - then transmits the frames → bitstream to signal stream;
  - converting the received signal stream into a bitstream: **delimiting** the frames, checking the **checksum**;
  - transmits the bitstream to the network layer.
- **Framing and frame delimitation ≡ signaling frame boundaries and recognizing these signals** (transparently to the upper network layer)

Reasons for the framing

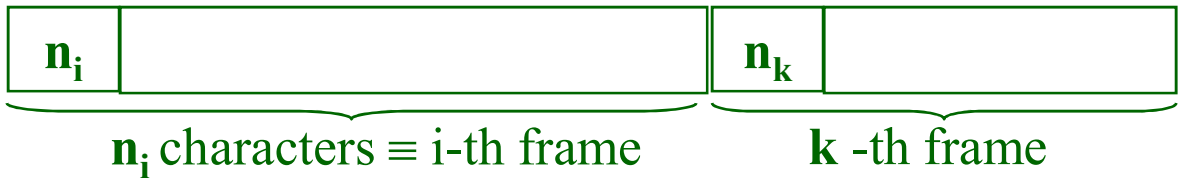
- Some MAC protocols require limited size (min., max.), or perhaps fixed size frames
- Physical transmission latency can be reduced if frames are shorter
- The probability of a frame errors can be reduced if the frames are shorter
- It is essential for error detection and limitation on a continuous bit stream.

Framing methods

- **The four methods examined**
  - **The character-count framing method**
  - **The character-insertion framing method**
  - **The bit insertion framing method**
  - **The invalid code pattern framing method**

Character-count framing

- **In a field of the frame header, we specify the number of characters in the frame (the length of the frame)**
- **Condition: there must be character synchronization, you must be able to recognize individual characters**
- **Problem: very vulnerable:**
  if it gets out of sync, it misinterprets frames
  - **if there is bit slippage, it may even misinterpret characters**
- **There is no option to re-sync in case of an error.**

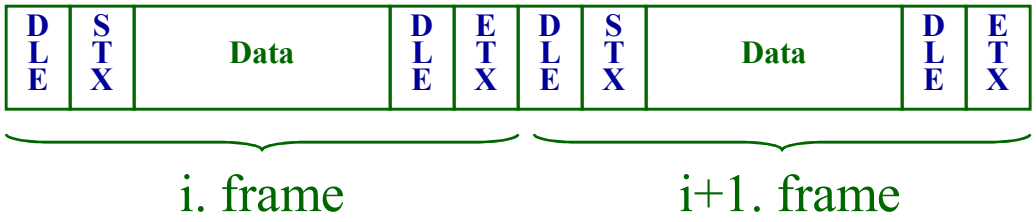


Character insertion framing

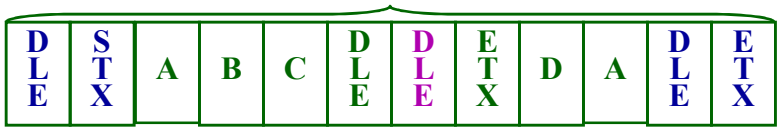
(Character stuffing)

- **The beginning and end of frames are marked with special character sequences.** (character-oriented method)
- **In case of an error, frame synchronization can be easily restored** (look for the beginning of the frame sequence).
- **Start of frame: DLE, STX (ASCII v EBCDIC character pairs)**
- **End of frame: DLE, ETX**
  - **DLE: Data Link Escape**
  - **STX: Start of Text**
  - **ETX: End of Text**
- **Usually used for terminal connections**
  E.g. **IBM BSC (**Binary **Synchronous Communication** protocol**)**

Character stuffing



- **Transparent transfer is a problem: the data should not contain DLE, STX, or DLE, ETX**
- **Solution: Before placing the frame boundaries, all DLEs between the data must be duplicated:**
  - **DLE+STX v. ETX pairs indicate frame boundaries,**
  - **and DLE+DLE is DLE in the text.**
- **For example, the text is: A+B+C+DLE+ETX+D+E; then the frame**
- **Problem: Character-oriented**



Bit insertion framing

(Bit stuffing)

- **Bit-oriented, not bound to characters**
- **Each frame begins and ends with a special bitpattern (flag)**

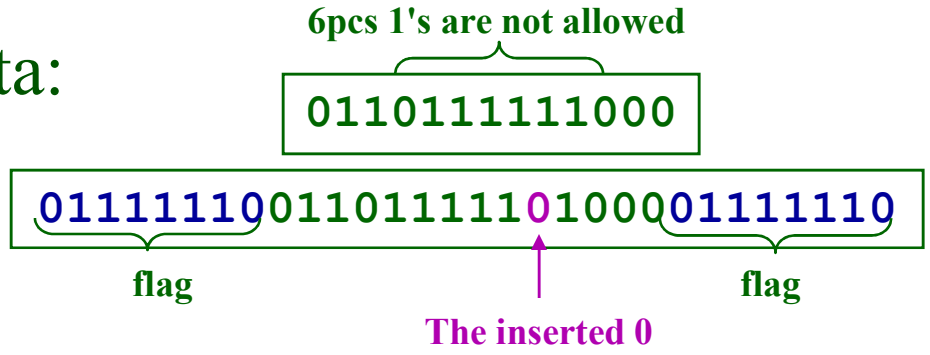
`01111110` **(6pcs of 1 bits)**

- **E.g. IBM SDLC** (Synchronous **Data** Link **Control Protocol**)
  ISO HDLC (High Level DLC , also used by X.25

- **Problem: such a bitpattern cannot appear in the bitstream**

Bit stuffing

- **Making it "transparent":**
  - **Before placing the frame boundaries, the transmitter inserts 1pcs of 0 into the data bit stream after every 5pcs consecutive 1 bits;**
  - **After recognizing and removing the frame boundaries, the receiver removes 1pcs of 0 after every 5pcs consecutive 1 bits.**
  - **So the 6-bit series 1 can only be the beginning or end of the frame.**
- For example, the data:
- Then the frame:



- Advantage: only 1 bit needs to be inserted, at most every 5 bits

Invalid code pattern framing

- **The beginning and end of frames are indicated by signals encoded differently from the data bits. "Encoding with violation".**
- **For example, in the case of Manchester encoding, the "standard"**
  - **HL⊓ or LH⊔ in place of**
  - **LL _ _ or HH ‾ ‾;**
  - For example, the 802.5 beacon ring is one such example.
- **Advantage: Transparent, efficient**
  The codes indicating the beginning and end of the frame cannot occur in the data (no bits need to be inserted)
- **Disadvantage: Can't always be done** (coding)

Combined

- **Often the character counting method is combined with some other method.**
  - **Specifying a frame length may speed up processing;**
  - **Another method can solve the problem of restoring frame sync.**

The fault protection

- **Fault protection can be direct**
  - **in case error correction coding from the received word.**
- **and can be indirect**
  - **in case of error indication coding by acknowledgement.**
- **Eg: The receiver directly verifies the integrity of the frame:**
  - **checking the formal rules of the framework;**
  - **verification of encoding and checksum.**
- **E.g.: The transmitter may be informed of the error indirectly via a acknowledgement**
  - **The lack of a positive acknowledgement indicates an error (There is usually no negative acknowledgement sent!)**

Directness - indirectness

- **The transmitter**
  - **in case of error-correcting coding, you can also directly correct the error,**
  - **in case of error indicating coding, it only detects it, but cannot repair it itself ⇒ indirectness (transmitter assistance)**
- **The transmitter**
  - **In case of an uncorrectable error** (feedback via the acknowledgement method) **you need to re-transmit the frame .**

Direct error handling

- **Error reporting, error correction**
- **basically based on redundancy:**
  more information needs to be transferred than actual (useful) information
- **Always applies to finite data units**
  (e.g. this is why frames are needed)

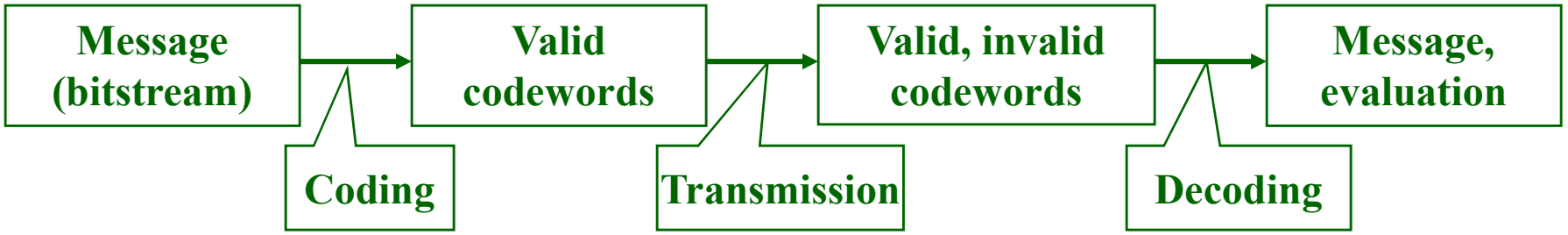
Model for coding

- **Message stream:** **m bits**
- **Redundant bits:** **r bits**
- **A code word (code word):** **n = m + r bits**
- **Coding: m + r → n (formation of code words)**
- **Code: the set of valid code words**
- **Possible number of messages:** **2^m**
- **Possible codeword number:** **2^n**

  (including invalid ones)
- **Valid code words:** **2^m pcs out of 2^n**

Model for coding

- **Data transfer**



- **As a result of decoding, we can get valid, or invalid codewords**
  - **in case of valid code – decoding – message;**
  - **in case of invalid code, evaluation:**
    - **error indication, or**
    - **error correction.**

Hamming distance

- **Hamming distance of codewords** (Hamming 1950) :
  - **The number of different bits in two codewords** (the number of 1's in the exclusive-or of two codewords).
  - **If the distance H between two codewords is d, then they can be converted into each other with d pcs of 1-bit errors, or**
  - **a valid codeword with d pcs of 1-bit errors can result in a valid codeword.**
- **Code Hamming distance:**
  - **The smallest Hamming distance between the codewords of the code.**
    (The code is the set of valid codewords.)

Theorems

- **Theorem 1: A code is capable of detecting d errors ⇔ if the Hamming distance of the code is d + 1**
  - **Any d unique errors can only result in an invalid codeword.**
- **Theorem 2: A code is capable of correcting d errors ⇔ if the Hamming distance of the code is 2 · d + 1**
  - **Even for an arbitrary d error, we are "closer" to the original codeword than to any other codeword.**
- **Theorem 3: An error-correcting code is also an error-detecting code. A code that can correct d errors can detect 2d errors.**

Items

- **Theorem 4: Perfect (direct) error protection, error correction code for all bits does not exist.**
  - **Let be an n-bit codeword, error correcting for all bits.**
  - **Then the H-distance of the code should be $2 \cdot n+1$**
  - **This is impossible, since $2 \cdot n + 1 > n$**
- **Comment**
  - **It can be seen that the use of error correction codes requires a lot of redundancy.**
    This reduces efficiency (increases overhead).
  - **In the case of data transmission** (where the error rate is relatively low), **it is more efficient to use a less redundant error-detecting code and retransmit the erroneous codes (data units)!**

Error codes

- **Parity code**
- **Parity per block**
- **Cyclic redundancy code (CRC)**

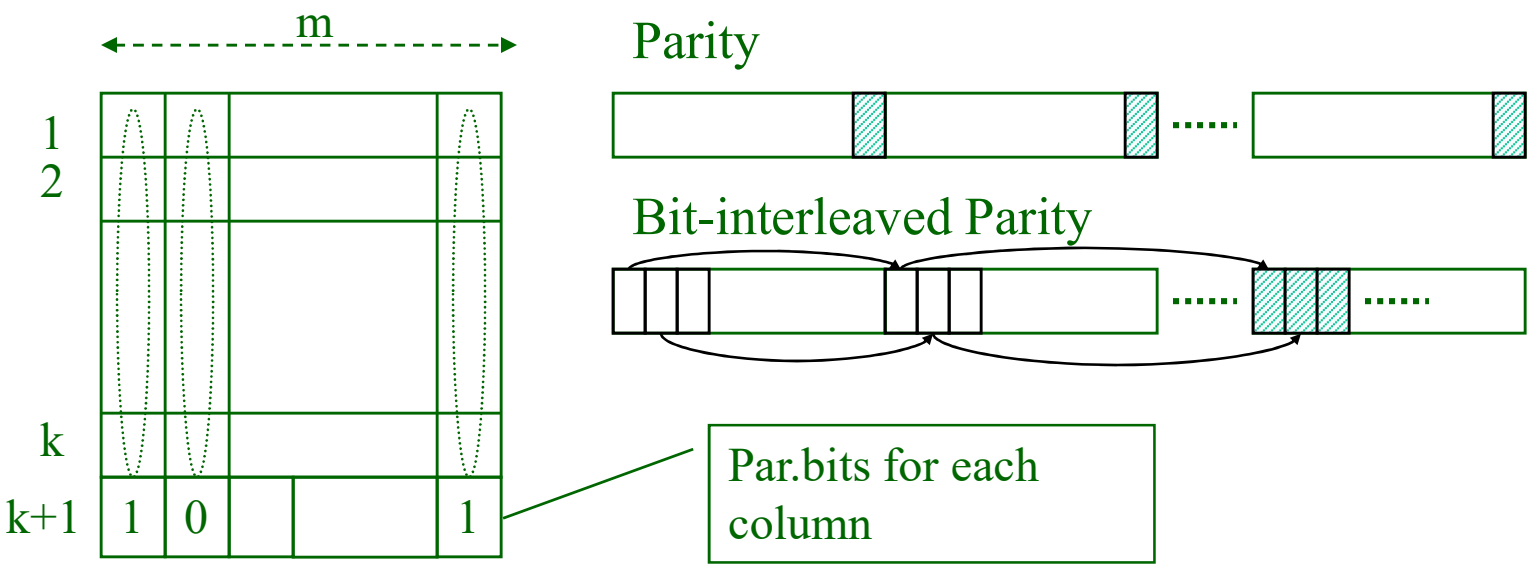
Parity code

- **A single parity bit is added to the end of the data. Its value must be chosen so that**
  - **In the case of even parity, the number of 1's in the codeword must be even (one's complement), or**
  - **In the case of odd parity, the number of 1's in the codeword must be odd.**
    - E.g. Even parity bit: exclusive-or (XOR) connection of data bits
- **The parity code increases the H-distance of the code from 1 to 2,**
  - **This allows us to detect 1 unique error.**
- **Problem: in case of group errors, the probability of error indication is 0.5 (group error: several adjacent bits are damaged)**

Bit-interleaved Parity (Parity per block)

- **Examines bits that are far apart and forms a parity bit from them** (detects group errors)
- **Training method:**
  - **we form a "matrix" of the data bits;**
  - **and forms parity column by column;**
  - **then makes the resulting parity bits the last row of the matrix;**
  - **the bits are sent row by row.**

Bit-interleaved Parity



- **In evidence: can detect a single group error of max. m bits;**
- **the m+1 long group error may go unnoticed;**
- **For example: a combined error in the i. and i+m. bits may remain unnoticed.**

Cyclic redundancy code

- **Cyclic Redundancy Check (CRC)**
- **Also known as: polynomial code**
- **Idea:**
  - **consider the bit strings as polynomials with coefficients 0 and 1,**
  - **i.e. an m-bit frame is a polynomial of degree m-1:**
  - $M(x)=a_{m-1}\,x^{m-1} + \ldots\, a_1\,x + a_0$

Example: $\overset{4\ \ 3\ \ 2\ \ 1\ \ 0}{1\ 0\ 0\ 1\ 1} \rightarrow x^4 + x + 1$

Cyclic redundancy code

- **Idea** (continued) **:**
  - **Let M(x) be the frame to be protected**
  - **Let G(x) be the r+1 bit, r-th degree generator polynomial,**
    - **so that m > r+1**
      (the frame is longer than the generator polynomial),
    - **and the generator polynomial G(x) is the same in both the transmitter and the receiver, given in advance.**
- **Principle:**
  - **Let R(x) be the checksum, which is the polynomial of degree r-1** (r bits) **which, when appended to the end of the frame to be protected, forms a polynomial that is divisible by G(x) without remainder. Where the division is based on modulo 2 arithmetic.**

0 + 0 = 0 0 - 0 = 0
0 + 1 = 1 0 - 1 = 1
1 + 0 = 1 1 - 0 = 1
1 + 1 = 0 1 - 1 = 0

Checksum calculation

- Let G(x) be of degree r.
- Insert r 0's behind the frame to be protected M(x) (so that it has m+r bits): $M(x)*x^r$
- Divide this (according to mod-2 algebra) by the generator polynomial G(x)
- The remainder of the division will be the desired checksum R(x)
- The codeword you are looking for is: $T(x) = M(x)*x^r + R(x)$

- Statement:
  T(x) is divisible by G(x) without remainder.

Proving the statement

- **For modulo-2 arithmetic:**
  (Exclusive or - XOR, no carry)

  0 + 0 = 0 0 - 0 = 0

  0 + 1 = 1 0 - 1 = 1

  1 + 0 = 1 1 - 0 = 1

  1 + 1 = 0 1 - 1 = 0

- **Proof:**

$$\frac{M(x)\cdot x^r}{G(x)} = Q(x) + \frac{R(x)}{G(x)} \quad \text{és}$$

$$\frac{T(x)}{G(x)} = \frac{M(x)\cdot x^r + R(x)}{G(x)} = Q(x) + \underbrace{\frac{R(x)}{G(x)} + \frac{R(x)}{G(x)}}$$

This is 0 in mod-2 arithmetic (A + A = 0)

In case of error:

- **$T(x) \rightarrow T'(x) = T(x) + E(x)$**
  - **where E(x) contains 1's, T(x) has changed!**
- **Because**

$$\frac{T'(x)}{G(x)} = \frac{T(x) + E(x)}{G(x)} = \underbrace{\frac{T(x)}{G(x)}}_{\text{There is no remainder here.}} + \underbrace{\frac{E(x)}{G(x)}}_{\text{The remainder of this indicates the error}}$$

- **From this theorem: It indicates an error for every E(x) that is not divisible by G(x)**
  - **For example, in the case of a single bit error, $E(x) = x^i$; where the i-th bit is incorrect**
- **$\Rightarrow$ If G(x) has two or more terms, it always indicates the single bit error!**

Standard generator polynomials

- **CRC-12 = $x^{12} + x^{11} + x^3 + x^2 + x + 1$**
- **CRC-16 = $x^{16} + x^{15} + x^2 + 1$**
- **CRC-CCITT = $x^{16} + x^{12} + x^5 + 1$**
- **Their properties (CRC-16; CRC-CCITT)**
  - **all errors 1 and 2 are detected;**
  - **all odd bit errors;**
  - **the 16, or shorter group error;**
  - **99.997% of the 17 group errors.**

- **A simple shift register circuit can be built to calculate CRC algorithms. This is HW, this is what is used.**

Elementary data link protocols

- **Data link layer protocol**
- **Functions of the data link layer:**
  - **Framing and delimitation**
  - **Fault protection**
    - **Direct (error detection, correction coding)**
    - **Indirect (ack)**
  - **Data flow control**
  - **Contact management**

} **Elementary data link protocols**

- **They implement indirect error protection and data flow control (transmitter-receiver synchronization).**
- **They build on the framework of framing and direct error detection.**

Elementary data link protocols

- **Prerequisites of the tests**
  - **The physical, data link and network layers**
    - **independent processes that communicate via messages ⇒ no interaction they are “transparent”**
  - **We consider two stations that want to send each other infinitely long messages received from the network layer.**
    - **the transmitter does not have to wait for the data to be sent, it is always available.**

General prerequisites

- **General structure of data connection frames:**

Head	Information section	Tail section
Header	Data	Trailer

In the header you can:

frame type:
-information frame (data)
-control frame
serial number,
ack information

In the tail section:
may contain error protection information

The data part:
network layer data

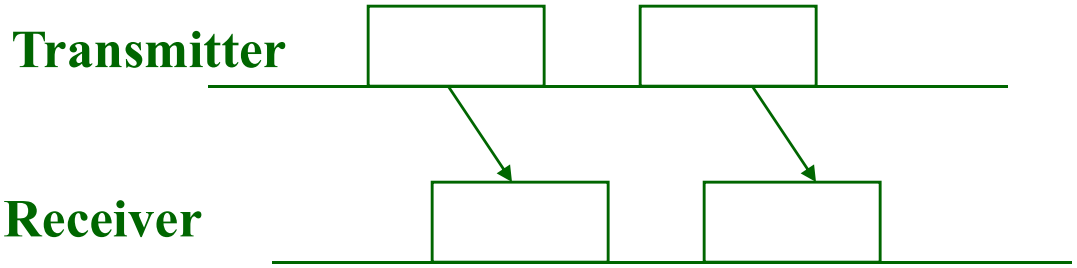
- **Requirement: error-free, in-order transmission**

The protocols discussed

- **Unconstrained simplex (utopia) protocol**
- **Simplex stop and wait protocol**
- **Simplex protocols for noisy channels**
- **Sliding window protocols**

Unconstrained simplex (utopia) protocol

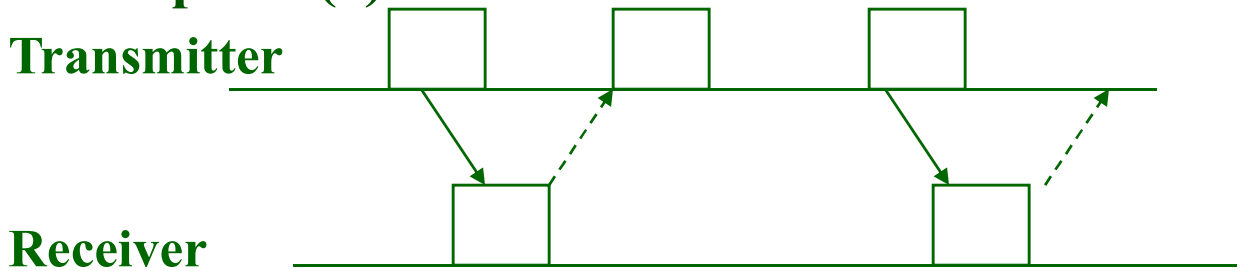
- **Additional assumptions**
  - **buffers are infinite (the receiver never fills up)**
  - **the channel is error-free**
- **Operation:**
  - **The seder is constantly sending**
  - **the receiver is constantly receiving**
- **One-way, no transmitter-receiver synchronization**



(Utopian, no error correction, or data flow control is needed)

Simplex stop and wait protocol

- **Additional prerequisites:**
  - **Finite buffers (receiver may fill up, traffic control is required);**
  - **The channel is error-free.**
- **Operation:**
  - **(1) transmitter sends a frame and then waits;**
  - **receiver receives the frame and then sends an acknowledgement;**
  - **Receiver takes the acknowledgement and then continues from point (1)**



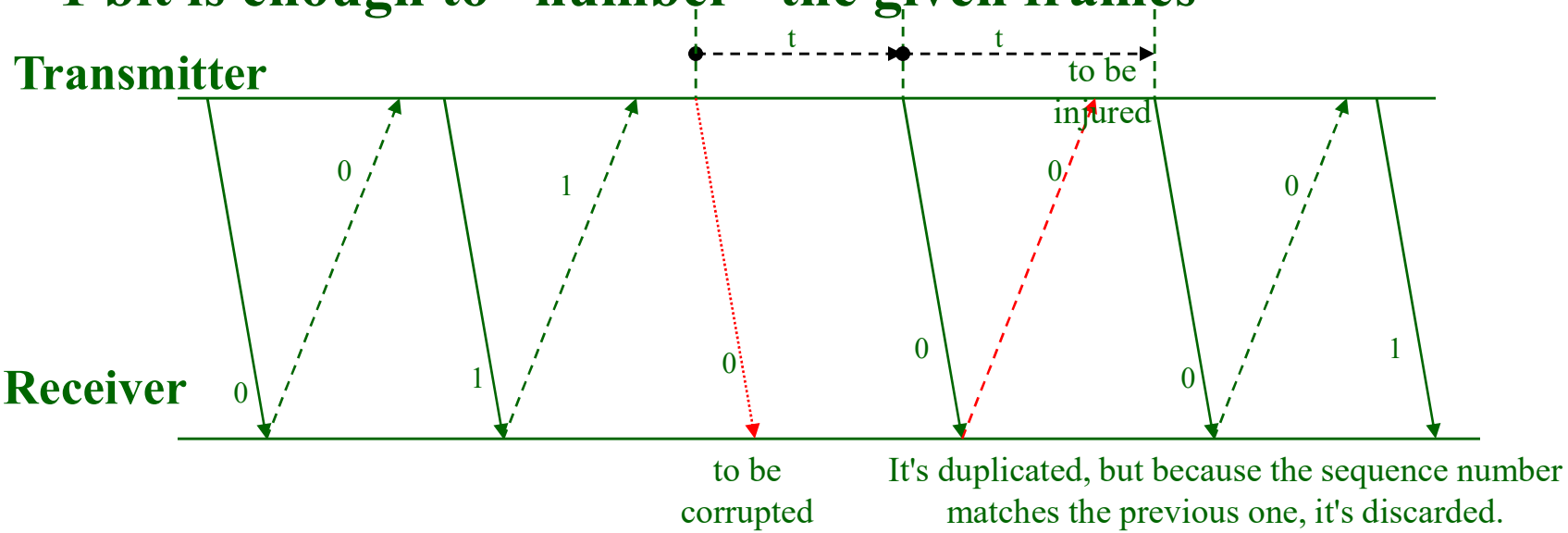
transmitter must wait for receiver's acknowledgement ⇒ feedback
⇒ synchronizes transmitter speed to receiver speed ⇒ traffic control
Problem : Not good for noisy channels (gets stuck on loss of acknowledgement)
error-free is a prerequisite

Simplex protocol for noisy channel

- PAR: Positive Acknowledgment with Retransmission
- Additional prerequisites:
  - finite buffers,
  - noisy canal.
- Operation
  - (A1) the transmitter sends a frame, then
  - (A2) sets a timer and waits
  - (A3) the transmitter retransmits the frame when the timer expires, then goes to (A2)
  - (A4) goes to (A1) when the acknowledgement is received
  - (V1) If the receiver receives a valid frame, it forwards it to the upper layer and then sends an acknowledgement.
  - (V2) the receiver waits if an invalid frame is received
- The PAR protocol
  - Traffic control and
  - provides error protection (with retransmission in case of error)

Simplex protocol for noisy channel

- **Problem :**
  - **In case of acknowledgement corruption, the same frame is sent by the transmitter several times**
  - **Therefore, it is necessary to be able to distinguish between consecutive frames**
  - **1 bit is enough to “number” the given frames**



The acknowledgement also needs a serial number (late repeat receipt)!

Sliding window protocol

- **Additional prerequisites:**
  - **buffers are finite,**
  - **The channel is noisy.**
- **Still a problem to be solved:**
  - **With high transmission latency, it is too long to wait for the "round-trip time" (for the acknowledgement to arrive before sending another frame).**
- **Solution :**

  It is necessary to allow sending multiple unacknowledged frames, while ensuring frame transmission in the correct order.

Sliding window

Sliding window protocol

- **The sent frames are numbered "cyclically"**
  - **E.g. for an n-bit number, 0, 1, … (2^n-1), 0, 1, ... (2^n-1), 0, ...**
- **The transmitter creates a "window" (with a maximum $w_{transmitter}$ width) for the sequence numbers of the frames sent but not yet acknowledged.**
- **The receiver creates a "window" (of constant $w_{receiver}$ width) containing the sequence numbers of frames which it is willing to receive.**

Sliding window

How the rotating window protocol works

(A1) transmitter receives a frame from the network layer, stores it in its buffer, sends it, at the same time sets a corresponding timer, and → increases the upper limit of the transmission window $\mathbf{w_{transmitter}}$ by 1,
Repeats all this until it reaches the max. window width (if it reaches it at all).

(A2) The transmitter, if the timer belonging to a frame has expired, → it retransmits that frame from its buffer.

(A3) When the transmitter receives a receipt, the lower limit of the transmitter window
→ sets it to the value following the receipt number.

Receiver receives a frame.

(V1) If the received frame is incorrect, → does nothing (does not acknowledge),

If the received frame is intact, then

(V2) If outside the receiving window: does nothing (discards and does not send a receipt),

If it falls within the receiving window, it checks whether the sequence number of the received frame is the same as the lower limit of the receiving window.

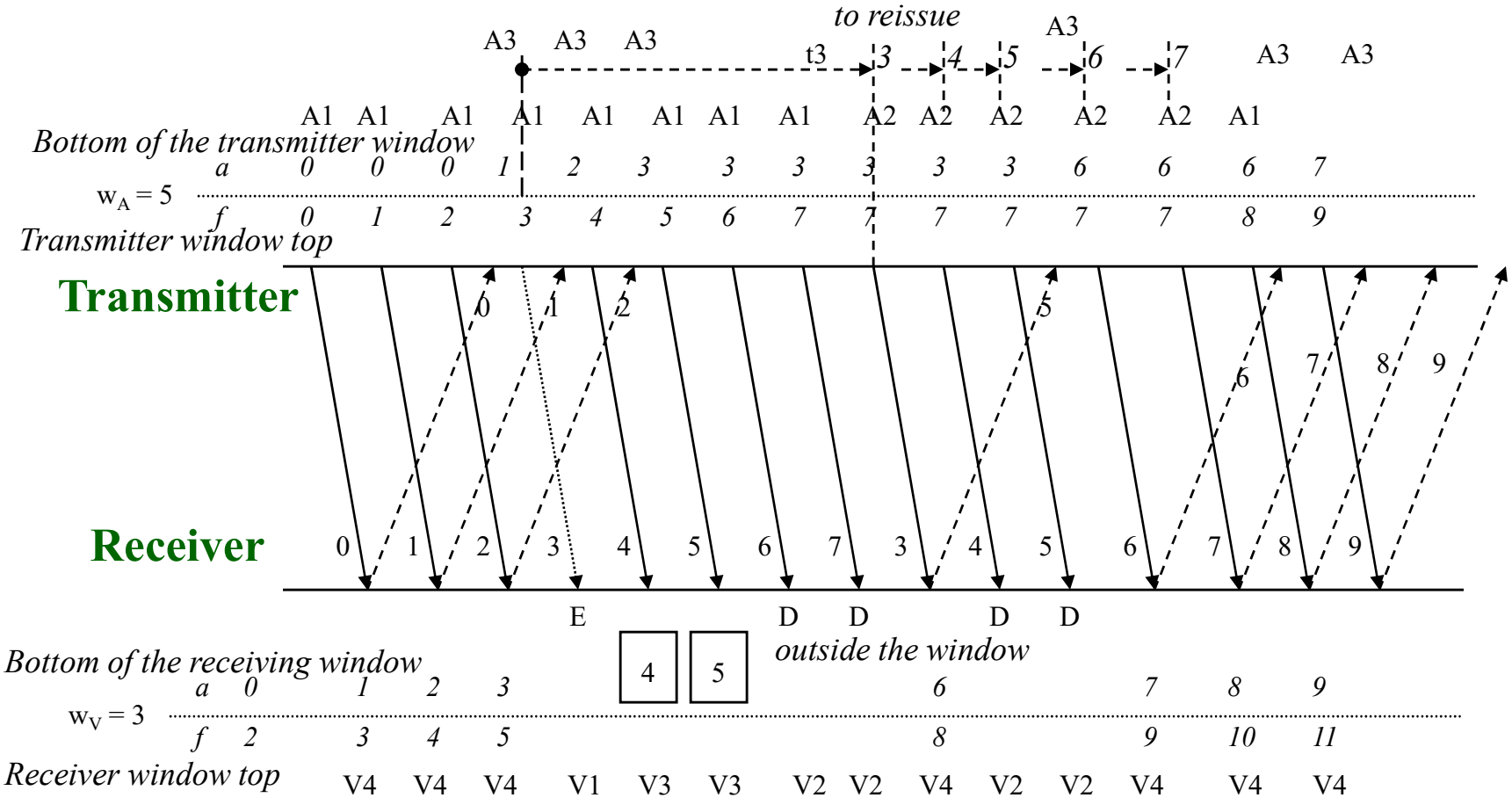
(V3) Not the same: it takes the frame and stores it in its buffer.

(V4) Same: receives the frame (forwards it to the network layer, if there are more frames in its buffer, it forwards them as well).
→ At the same time, it acknowledges the frame with the highest consecutive number and sets the lower limit of the receiver window to the next number. The receiver window has a fixed size $\mathbf{w_{receiver}}$ → the upper limit also shifts by this much.

Addition: *If the last acknowledged frame is retransmitted, the receiver repeats the acknowledgement!*

Example for the operation of the sliding window protocol

- Let $w_{transmitter} = 5$; $w_{Receiver} = 3$;
- Frame number 3 arrives incorrectly

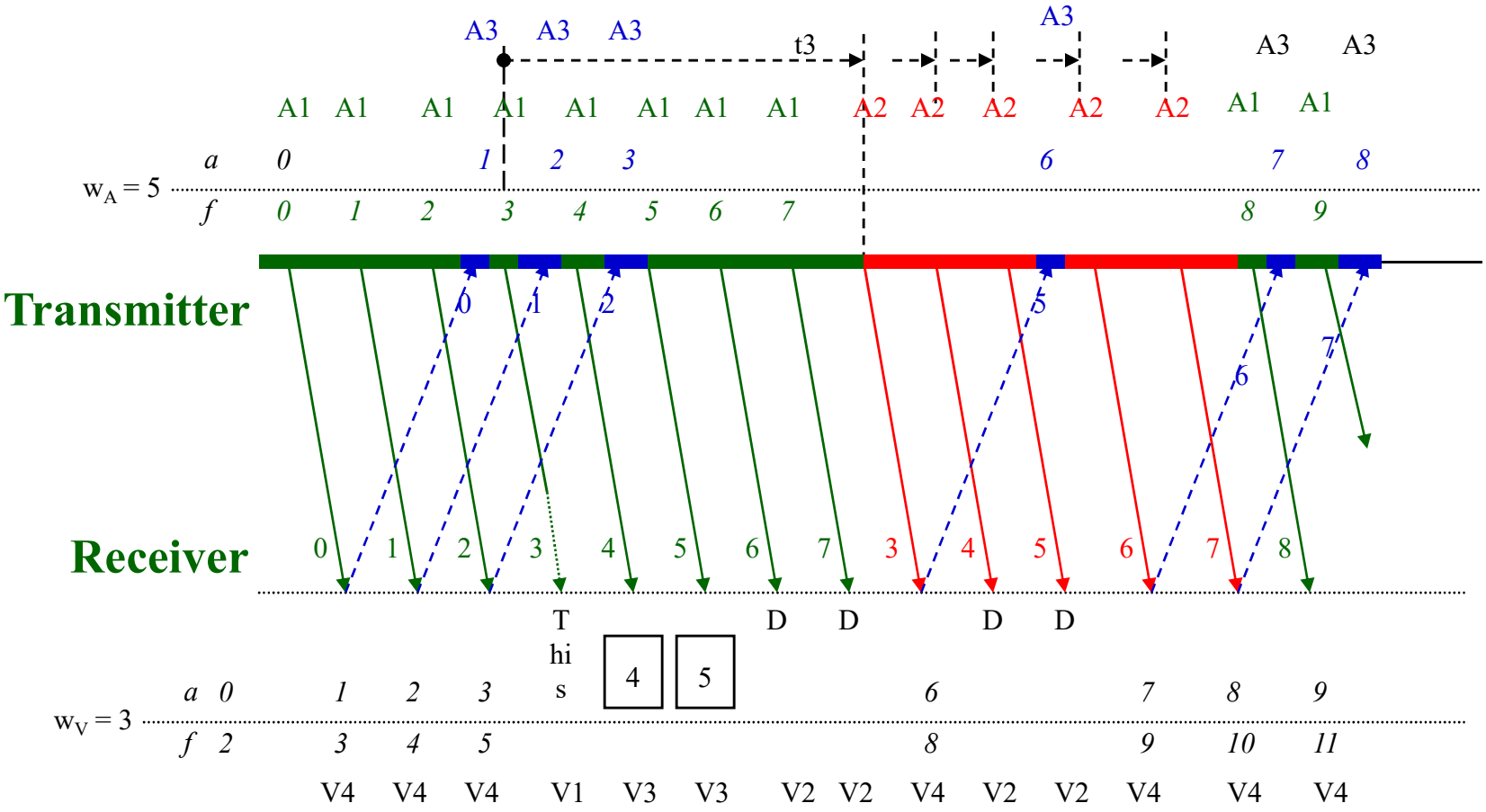


From the transmitter perspective

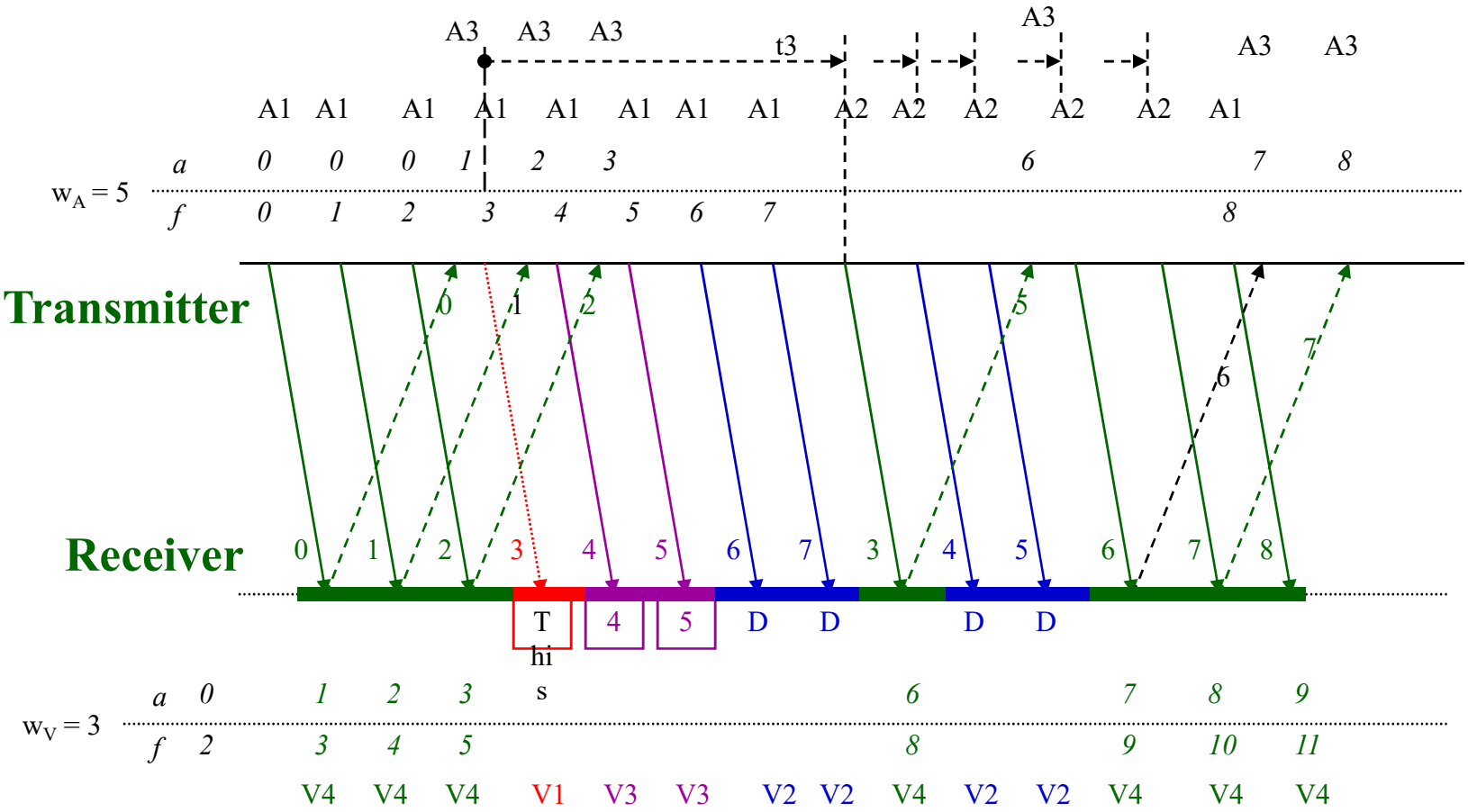
(A1) Transmitter transmits, sets a timer, increases the upper limit of the window (timer setting is not always visible)

(A2) If the timer expires, it retransmits a frame from its buffer.

(A3) When the transmitter receives a receipt, the lower limit of the window is set to the value following the serial number of the receipt.



From the receiver's perspective



(V1) If the received frame is incorrect, it does nothing (does not acknowledge),

(V2) If outside the receiving window: does nothing (discards),

(V3) Not the same as the lower bound: it takes the frame and stores it in its buffer.

(V4) Forwards a frame, if there are any subsequent frames in its buffer, forwards them as well. Acknowledges the frame with the highest sequence number, sets the lower limit of the window to the next frame, and moves the upper limit by the same amount.

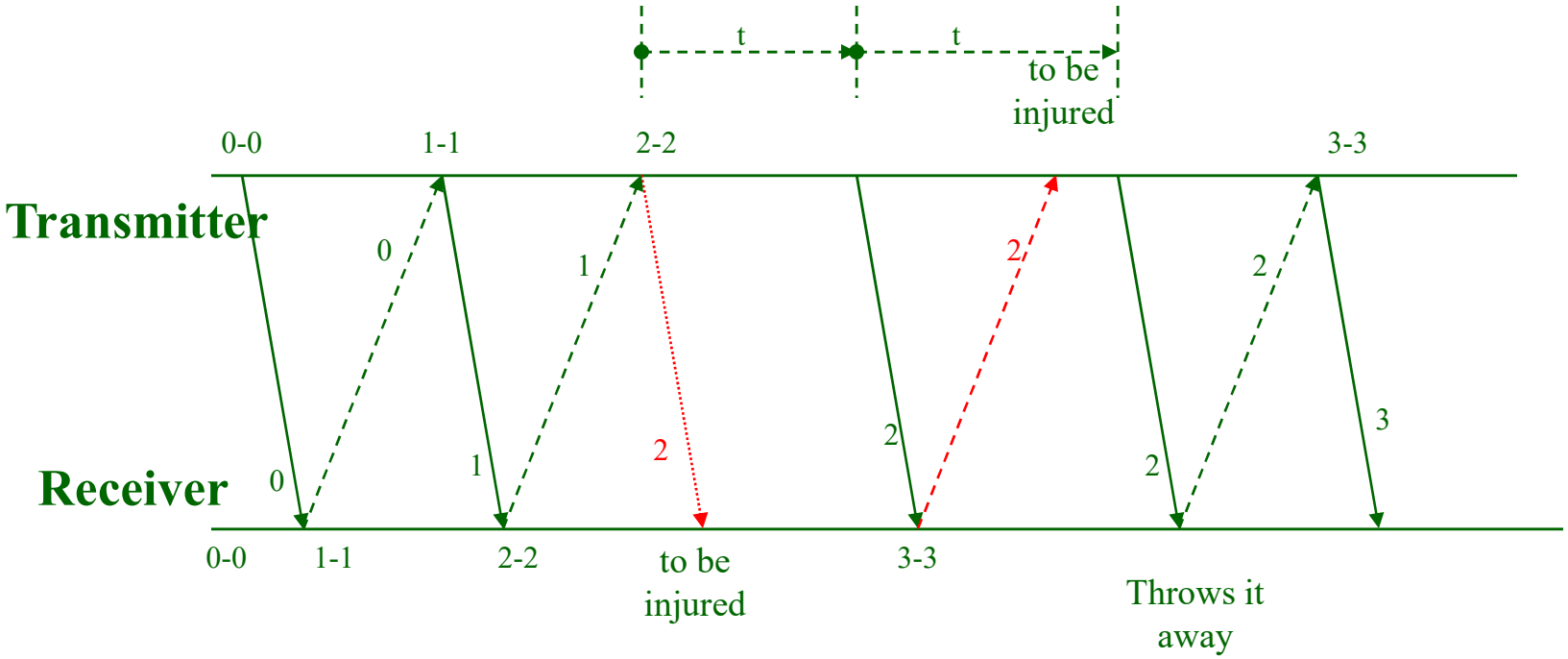
Sliding window

Choosing the window size

- The choice is made by
  - propagation delay (transmitter window)
  - and can be determined by the error rate (receiver window).
- The greater the delay, the wider the transmitter window:
  - The transmitter should not have to stop until the receipt is returned (the window should last at least the duration of the round trip)
- The larger the receiver window, the less repetitions are required in case of an error:
  - After re-receiving the faulty frame, the receiver also acknowledges the subsequent good frames received earlier.
  - The size of the receiver window is max. the size of the transmitter window

The PAR protocol

- **Sliding window protocol with window size $w_T = 1$; $w_{R=1}$**



Note: in the "original" PAR, the packet numbering was 0, 1.
Here the counting field is wider

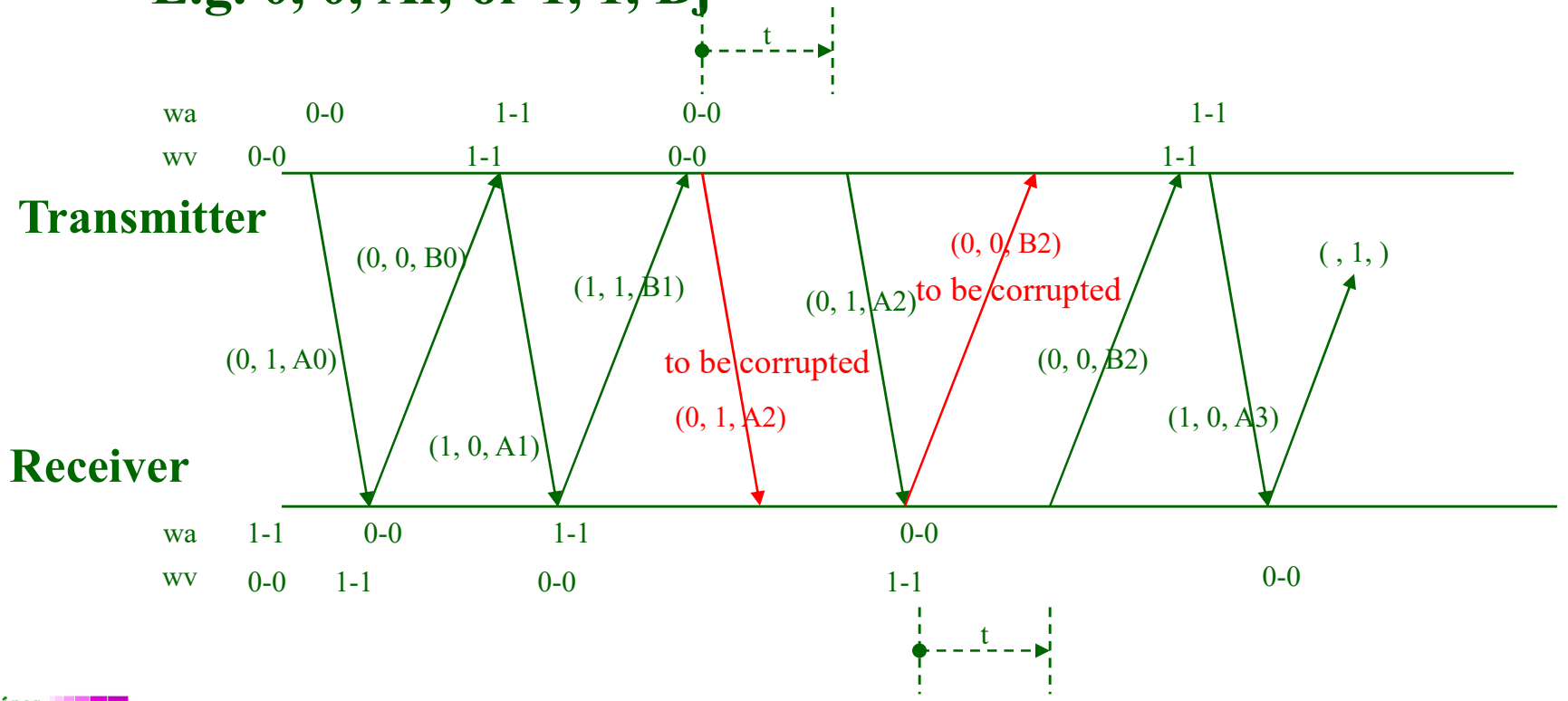
Piggybacking

- **In case of duplex (two-way) channel:**
  - **⇒ piggybacking acknowledgement is possible :**
  - **the receiver sends the acknowledgements by inserting them into the data frames in the opposite direction (some bits in the data frame are reserved for the acknowledgement)**
  - **Problem: if there is no oncoming traffic for a long time ⇒ the receipt is delayed**
  - **Solution: if the receiver were to send a receipt, it would set a timer and if this expires before it has anything to send (which it could attach the receipt to), ⇒ it would send a separate receipt frame.**

piggyback = (US) pick - a - back = *a child* sitting on sb's back/shoulder

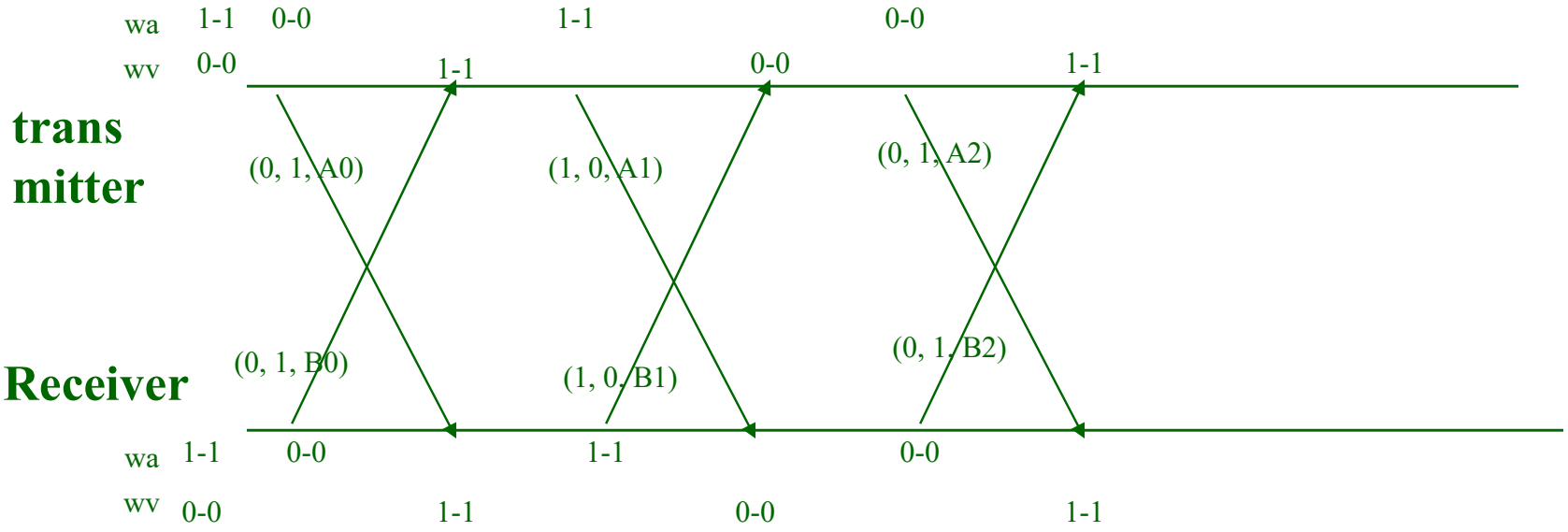
PAR with piggybacking acknowledgement

- **Stations A and B, both transmitter and receiver (bidirectional data traffic)**
- **Notation: frame sequence, acknowledgement sequence, frame (sequence number 1 bit)**
  - **E.g. 0, 0, Ai; or 1, 1, Bj**

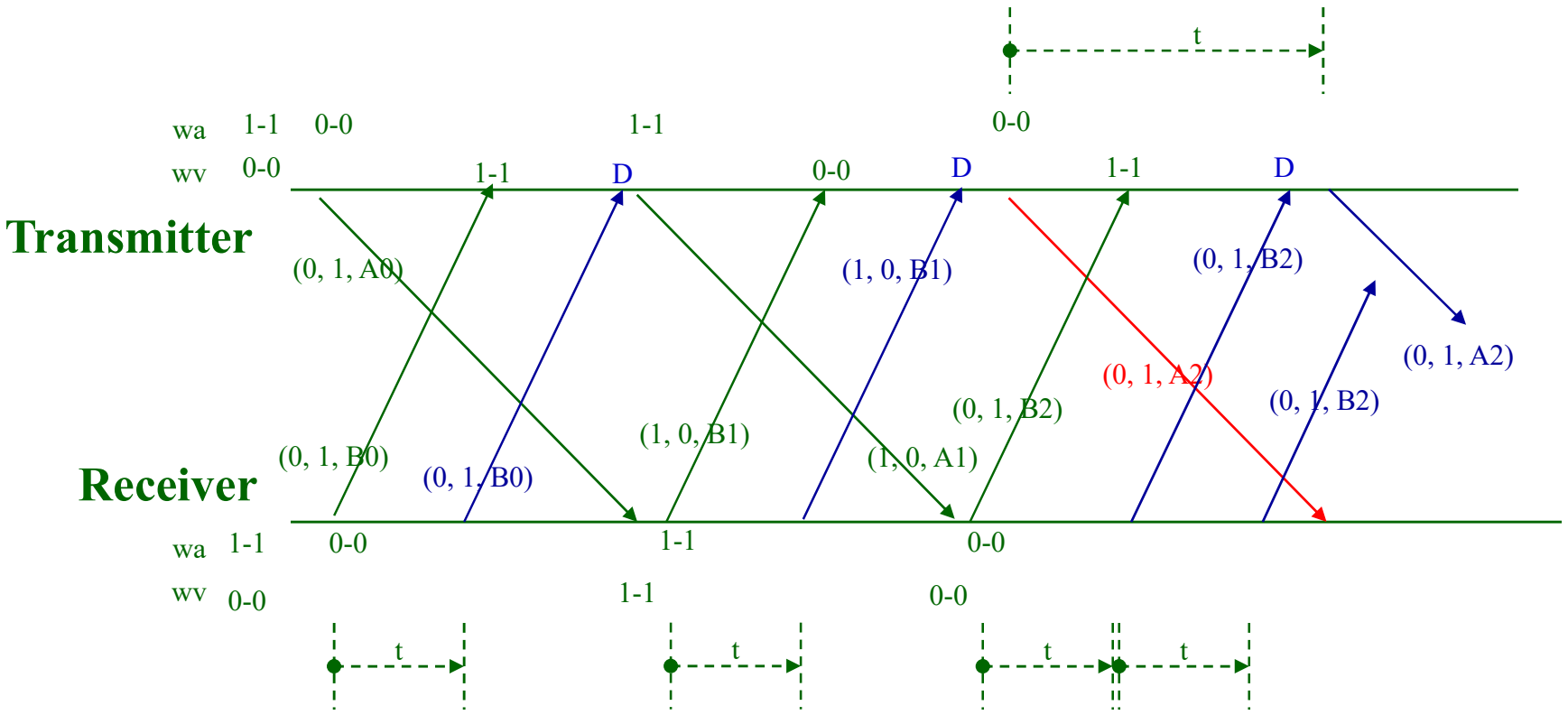


PAR with piggybacking acknowledgement

- **Stations A and B, both transmitter and receiver (bidirectional data traffic)**
- **Marking: frame number, receipt number, frame (the number is 1 bit)**
  - **E.g. 0, 0, Ai; or 1, 1, Bj**



If the timing is too short at B

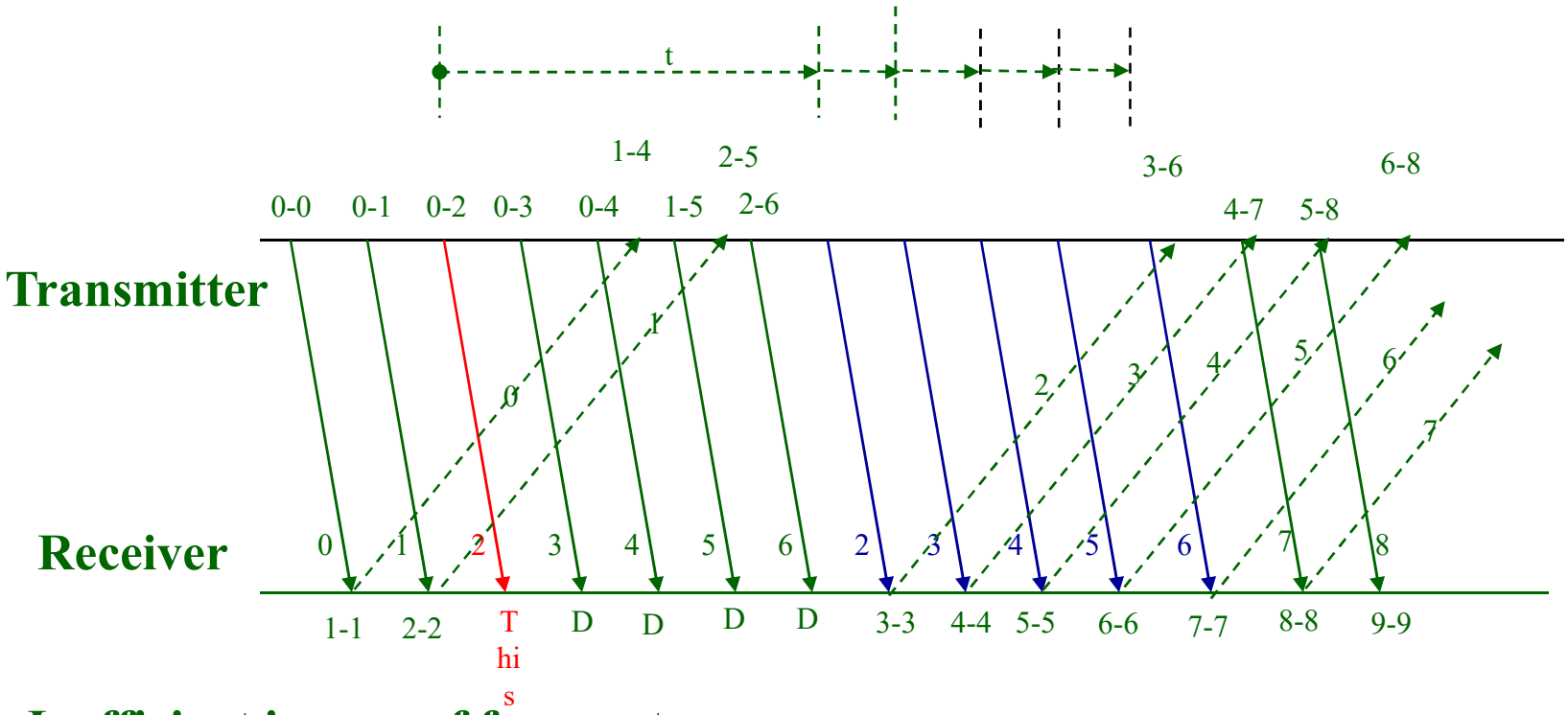


Problem: There is a lot of unnecessary repetition, but it still works (in order) correctly (it reduces efficiency)

Go back n error handling protocol

- **Let**
  - **$w_a = n$ (large transmitter window),**
  - **$w_v = 1$ (receiver acknowledges one by one).**
- **In the event of an error, the receiver discards all subsequent frames until the transmitter repeats the faulty frame** (hence the name "backtracking").

Go back n protocol



Inefficient in case of frequent errors.

With a larger receiver window, the receiver stores the well-received frames up to the window size, so when the transmitter repeats the faulty one, it also acknowledges the good ones (fewer need to be retransmitted).

That is why the sliding window protocol is also called selective *repetition* .

Size of the cyclic counter field

- **The window sizes determine**
- **In the case of an n-bit cyclic counter field, the sequence numbers can be between 0 - (2^{n-1})**

a) Then the maximum window size of the transmitter can be 2^n - 1.

b) Then the maximum size of the receiver window can be 2^{n-1} .

Suppose we do not comply with a)

- Let n = 3 (serial numbers 0 - 7), $w_T = 8$
- Transmitter sends frames numbered 0-7 (8 pcs)
- The taxpayer receives receipt number 7, therefore
- sends another 8 frames (with sequence numbers 0-7 again).
- The eceiver will again receive receipt number 7:
  - it is not possible to decide whether this is the receipt of the next 8 frames,
  - or a repeat of the previous 8 receipts!

max. $2^n - 1$

Suppose we do not comply with b)

- Let n = 3 (serial numbers 0 - 7), $w_T = 7$, $w_R = 5$
- Transmitter sends frames numbered 0-6 (7 pcs)
- The sender accepts them all, acknowledges them, sets his window to 7 - 3, but the acknowledgements are lost!
- The transmitter retransmits the first 7 (with numbers 0-6 again, because its timing has expired).
- Of these, the receiver accepts and buffers the 4 pieces numbered 0-3, considering them to be the next series...
- The problem is caused by “interleaving”. At the receiver, the sequence numbers must not overlap. At the receiver, the window size:
  max = (transmitter max sequence number + 1) / 2
  The frame with sequence number i is placed in the i mod w_R buffer, there must be no overlap! max. 2^{n-1}

Summary

- General findings
- Framing – frame recognition
- Error handling, which can be
  - direct or indirect.
- Data flow control
- Elementary data link protocols

Exercise

- **Task 1 : CRC calculation for (transmission)**
  - **Let the message be: 1 1 0 1 0 1 $M(x) = x^5 + x^4 + x^2 + 1$ (m = 5)**
  - **Let the gen. polynomial be m:1 1 0 1 $G(x) = x^3 + x^2 + 1$ (r = 3)**
- **Transmission , formation of R(x) :**

$M(x)*x^r = x^8 + x^7 + x^5 + x^3$

$(M(x)*x^r)/G(x) = (x^8 + x^7 + x^5 + x^3) : (x^3 + x^2 + 1) = x^5 + 1$

$- (x^8 + x^7 + x^5)$

x^3

$- (x^3 + x^2 + 1)$

$x^2 + 1 \qquad = R(x) \rightarrow R: 1\ 0\ 1$

- **Transmission , T(x) composition is :**

T(x) = 1 1 0 1 0 1 1 0 1 $\qquad T(x) = x^8 + x^7 + x^5 + x^3 + x^2 + 1$

m r

Exercise

- **Task 1 : Check CRC (receive)**
- **Receiving, checking the code word:**

$(x^8 + x^7 + x^5 + x^3 + x^2 + 1) : (x^3 + x^2 + 1) = x^5 + 1$

$-(x^8 + x^7 + x^5)$

$x^3 + x^2 + 1$

$-(x^3 + x^2 + 1)$

0; okay, no remainder.

- **Same with binary numbers :**

M(x)*x^3 = 1 1 0 1 0 1 | 0 0 0

G(x) 1 1 0 1 shift right to the first 1, then subtract

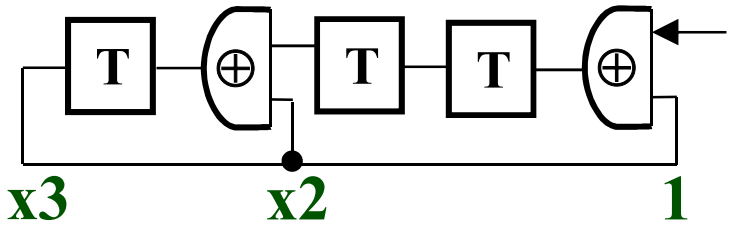
0 0 0 0 0 1 | 0 0 0

1 1 0 1 shift right to the first 1, then subtract

1 0 1 = R (cannot divide by o further b)

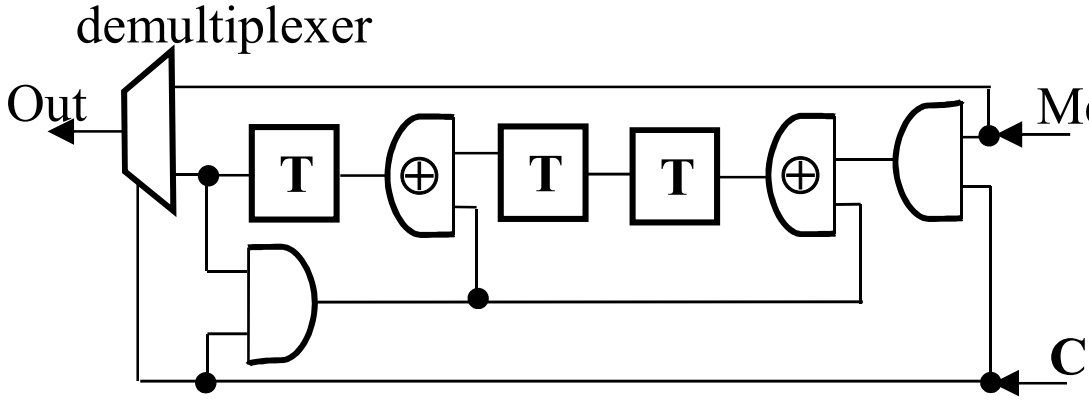
Exercise

- **CRC in HW implementation – latches and antivalence (XOR)**
- **E.g: G(x) : 1 1 0 1 ($x^3 + x^2 + 1$), r = 3**
- **Initially ∀ storage □ 0**



There is a bond and antvalence, where G(x) contains 1

- **If there is a 1 at the beginning ⇒ subtract G(x) and then move**
- **If all the bits have entered ⇒ the remainder is in the latches**
- **The complete circuit (transmitter):**
  (the receiver is like the above, only m+r enters and checks the latches to see if ∀ is 0)



Control :
up to m steps: V = 1
(message exits + remainder in registers)
Up to step r: V = 0
(checksum exits)