↓ LEC. 5

Block parity check

- To reveal burst errors (in the same block), one can break down the bit sequence into words and organize them into a matrix

k words, N bit/word, $k \times N$ matrix

- Parity bits are calculated on both rows and columns
- A *burst* of less length than the number of columns can be revealed → the errors are all in a row of the matrix.
- Lower efficiency, higher effectiveness
- Alternative: *interleaving*

Example (1)

$$\boldsymbol{m} = 01010010\ 00001100\ 00111010\ 11000100\ 00010110\ 01101010\ 10001100$$

0	1	0	1	0	0	1	0	1
0	0	0	0	1	1	0	0	0
0	0	1	1	1	0	1	0	0
1	1	0	0	0	1	0	0	1
0	0	0	1	0	1	1	0	1
0	1	1	0	1	0	1	0	0
1	0	0	0	1	1	0	0	1
0	1	0	1	0	0	0	0	0

$$k = 7$$
$$N = 8$$

$$\boldsymbol{x} = \boldsymbol{m}\ 1001101\ 01010000$$

Example (2)

The idea of interleaving is that of "making the errors sparse".

1	0	0	1	1	0	1	0	1	0	1	0	0	0	0

Detection: possible

If the error burst is in a single row

0	1	0	1	0	0	1	0	1
0	**1**	0	0	1	~~**0**~~ 1	0	0	~~0~~ 1
0	0	1	1	1	0	1	0	0
1	1	0	0	0	1	0	0	1
0	0	0	1	0	1	1	0	1
0	1	1	0	1	0	1	0	0
1	0	0	0	1	1	0	0	1
0	**0**	0	1	0	~~**1**~~ 0	0	0	0

Even # of errors → no detection

Detection: impossible

0	1	0	1	0	0	1	0	1
0	**1**	0	0	1	**0**	0	0	**0**
0	0	1	1	1	0	1	0	0
1	1	0	0	0	1	0	0	1
0	**1**	0	1	0	**0**	1	0	**1**
0	1	1	0	1	0	1	0	0
1	0	0	0	1	1	0	0	1
0	**1**	0	1	0	**0**	0	0	0

1's Complement Sum

- Same matrix organization used in block parity check
- Rows are summed together with 1-complement arithmetic
- The sum, possibly complemented, is the error detection code (checksum)
    - The possible complement is dictated by computational complexity issues
- Used by IP, UDP, TCP and other Internet protocols

Example

1									1				1				
1	0				0				1	0			1	0			
		1	1			1	0			1	0				0		
				1			1	0			1	1					
	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	
	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	0	
	0	0	0	1	0	1	0	1	1	1	1	0	1	0	1	1	
	1	0	1	0	0	0	0	0	0	1	0	0	1	1	1	0	
	0	0	0	1	1	1	0	1	0	0	1	1	1	1	1	0	
	1	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**1**	partial sum
															1	**0**	carry over
	1	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**1**	1's complement
	0	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**0**	**0**	**0**	checksum

2

10

PAYLOAD

(2nd col) 4 → 100

(3rd col) 3 → 11

(4th col) 6 → 110

Complement

Other Codes

$\{b_n\}$
bits that are to be transmitted

$b_1 \cdot x + b_2 \cdot x^2 + b_3 \cdot x^3 + \ldots$

- Polynomial codes, also called Cyclic Redundancy Check (CRC).
- The individual binary digits to be transmitted are treated as coefficients (of value "0" or "1") of a given polynomial
- The sending and receiving entities use a common polynomial, called the generator polynomial
- The parity bits and their checks are obtained from polynomial divisions in algebra modulo 2

For example, with block parity check (p.60) you can detect exactly 1 bit in error → correct the bit!

2 Error Correction — PHY layer

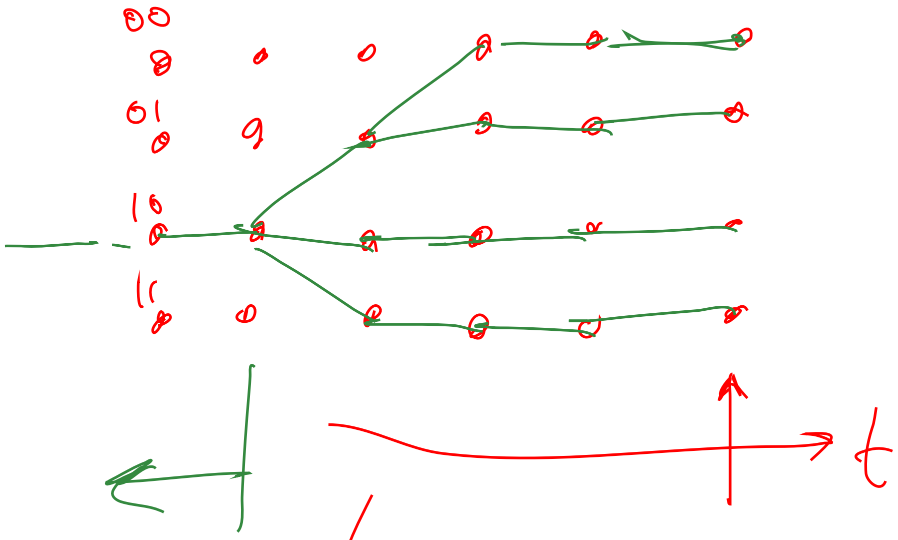
- Forward Error Correction (FEC) techniques → ex Viterbi algorithm.
- Redundancy is used to remedy erroneous bits (if small in number)
- No acknowledgement messages of correct reception are required
  - unidirectional communication
  - there is no need for buffering of sent IUs

  no need to retransmit
- Coding theory
  - added redundancy is generally greater than that used for detection alone
  - redundancy introduced to correct error is constant (fixed)

Example: repetition code

Viterbi Algm

Trellis



Convolutional Codes

message

1 0 0 1 1 0 1

coded message (3 repeat code)

111 000 000 111 111 000 111

Channel

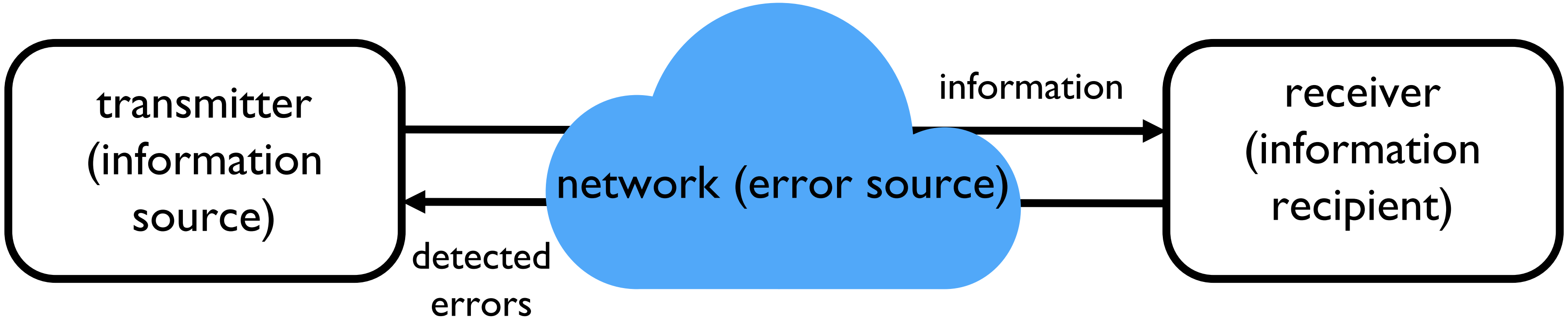
received message

1**0**1 000 0**11** 11**0** 111 **1**00 111

I can correct 1 error
or detect 2

③ # Error recovery

From Link layer (L2) up



- Error control by automatic retransmission (Automatic Repeat Request, ARQ)
- Different mechanisms used for retransmission (error detection, acknowledgements, timers, IU identifiers)
- ARQ procedures (differ in the size of the windows)
    1. stop and wait: positive acknowledgement mode with reissue
    2. sliding window, go-back-N: variable window mode with non-selective reissue
    3. sliding window, selective repeat: variable window mode with selective reissue

3.1

Stop and Wait

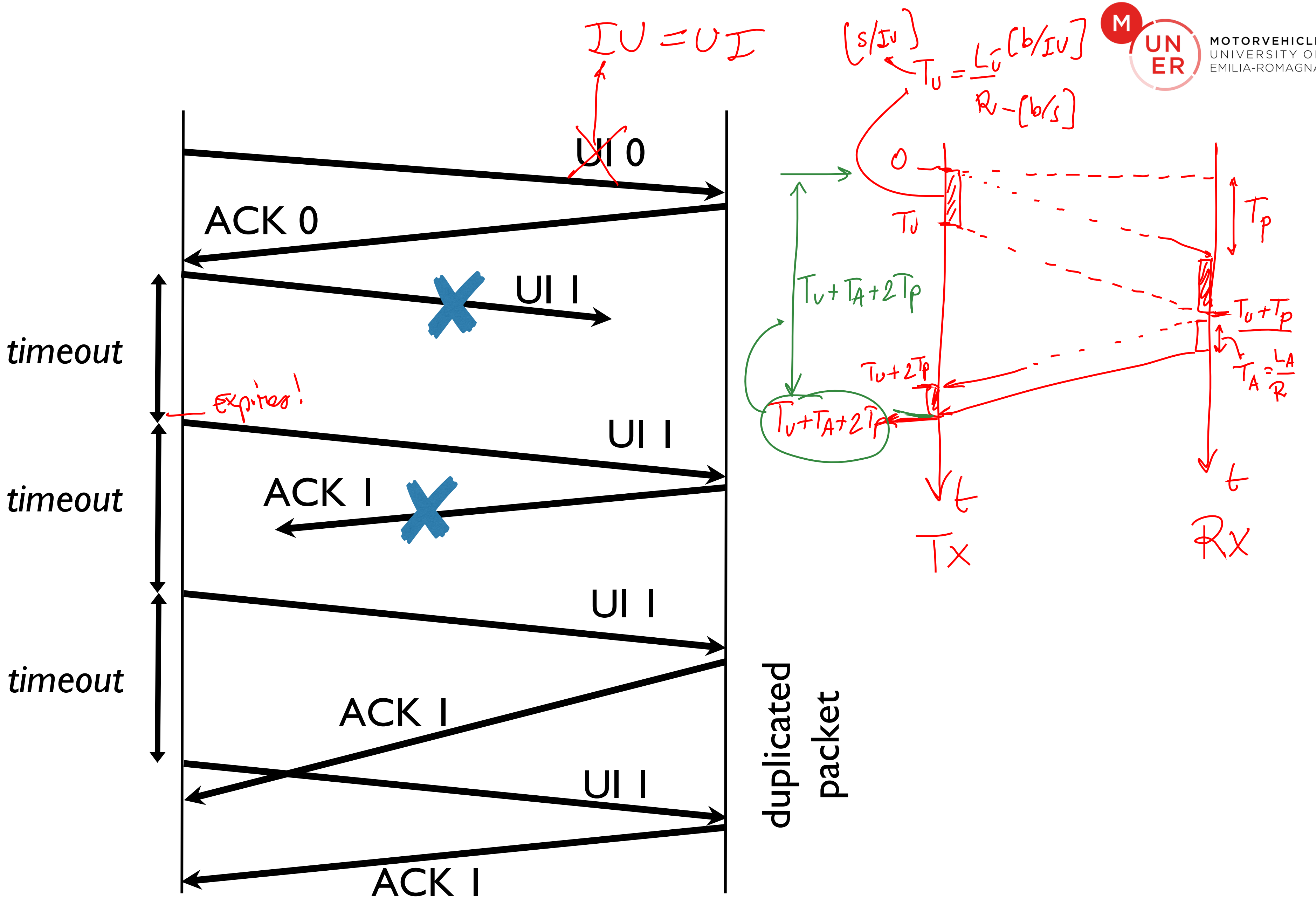
(Typically) The ACK packet is very short, much shorter than the IU.

- Data IUs are sent individually
- An acknowledgement (acknowledgment, ACK) is expected before sending the next IU
  - Empty IU or with data directed in the opposite direction
  - Acknowledgements are also subject to loss/errors
- It is necessary to activate for each sent IU a retransmission timeout
  - if no ACK has been received when the timeout expires, the IU is retransmitted
  - sufficiently large but not too large
- To avoid duplication, sequence numbers (SeQuence Number, SQN) are used to identify the IUs
  - entered into the PCI with a fixed number of bits, modulo numbering
- Cumulative ACKs: the SQN following the one encountered is reported
- If the sequentiality of the IUs is maintained during the transfer (no out-of-sequence), only one bit is needed to count the SQN

At Transport timeout is dynamically updated on the basis of RTT measurements

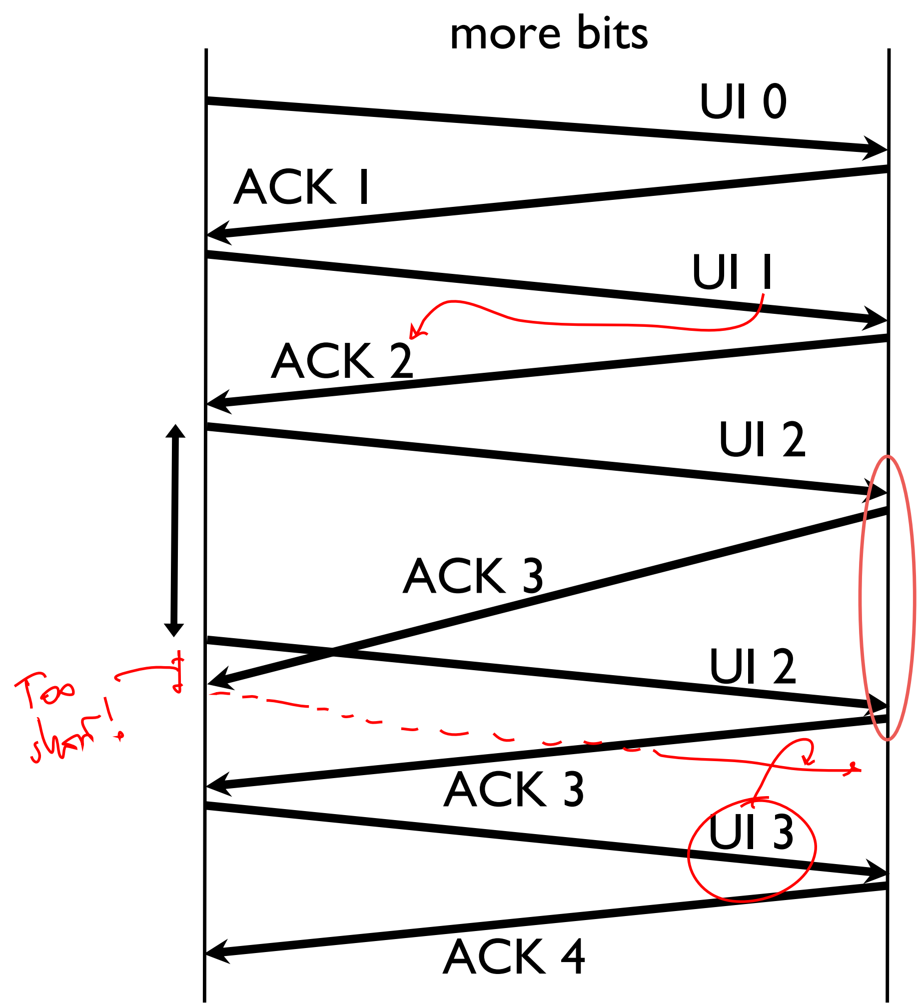
IMPOSSIBLE!

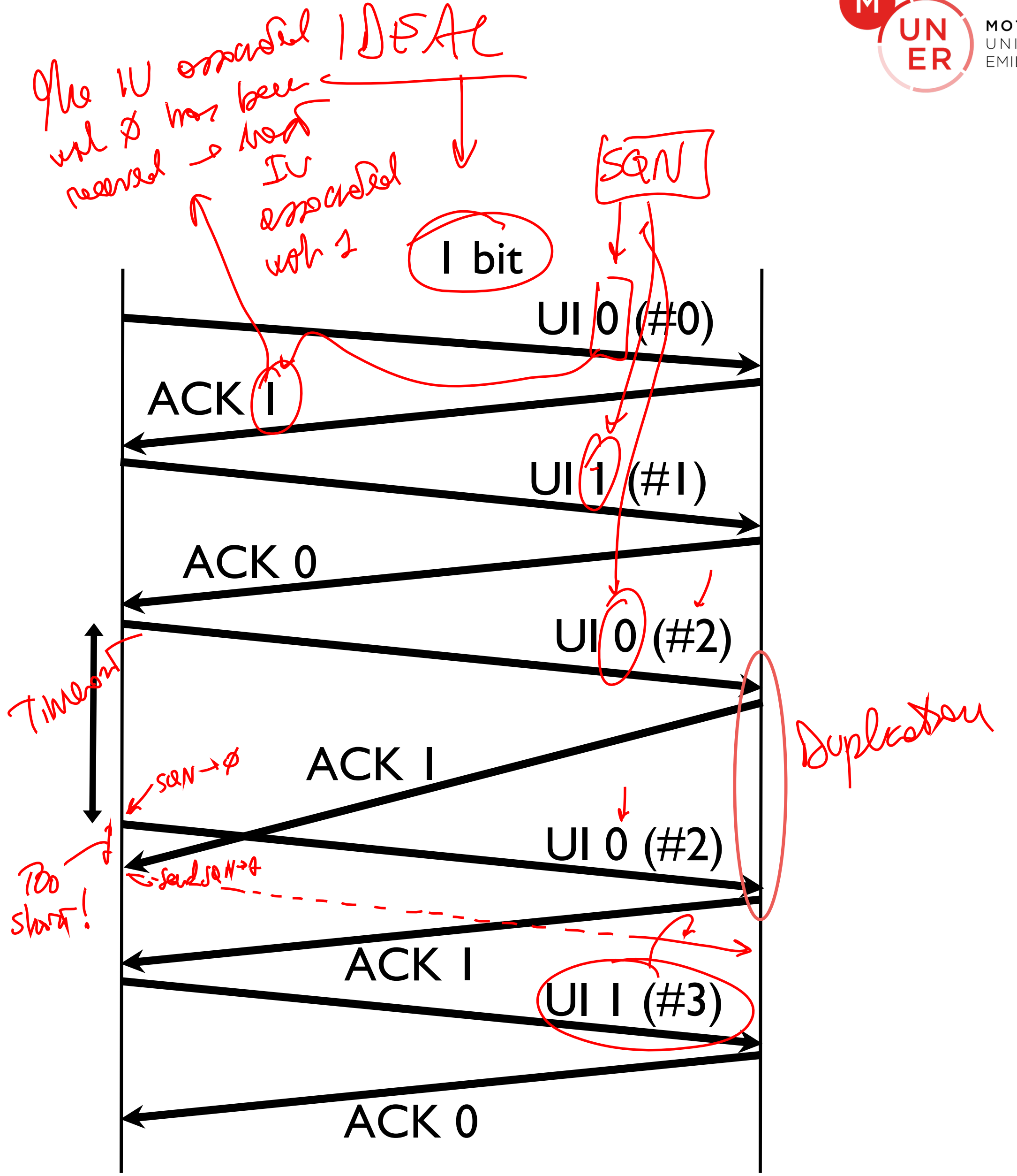
Example



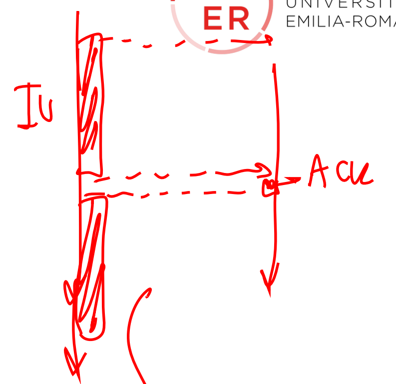
IU = UI
[s/IU]
$T_U = \frac{L_U [b/IU]}{R [b/s]}$
UI 0
ACK 0
UI 1
timeout
expires!
UI 1
ACK 1
timeout
UI 1
timeout
ACK 1
UI 1
ACK 1
duplicated packet
0
T_U
T_p
$T_U+T_A+2T_p$
T_U+T_p
$T_A = \frac{L_A}{R}$
T_U+2T_p
t
Tx
Rx

Example with SQN





Stop and wait: performance (1)



No error

- Total time required to send an IU and receive feedback

$$T_1 = T_U + T_A + 2T_p$$

See p.76

- Maximum degree of channel utilization

$\rho_0 < 1$

If $T_p << T_U$, then $\rho_0 \simeq 1$

$$\rho_0 = \frac{T_U}{T_1} = \frac{T_U}{T_U + T_A + 2T_p}$$

$$= \begin{cases} \dfrac{1}{2 + 2T_p/T_U} & \text{se } T_U = T_A \\ \dfrac{1}{2T_p/T_U + 1} & \text{se } T_U \gg T_A \\ 0 & \text{se } T_p \gg T_U \end{cases}$$

$$\frac{T_U}{T_U + 2T_p} = \frac{1}{1 + 2T_p/T_U}$$

Very common!

Network is too large & packets too short!

Stop and wait: performance (2)

$\# att$ is a Geometric RV

With errors

Average number of unsuccessful Txs

$= (1-p) \cdot \frac{1}{(1-p)^{\cancel{2}}}$

- Source does not receive ACKs (IU or incorrect/lost ACKs)
- Ass. 1 Statistical independence of IU and lost ACK events
- Ass. 2 Probability of error p
- Geometric distribution of the number of attempts

$P\{\# att = 1\} = 1-p$

$P\{\# att = 2\} = p(1-p)$

$P\{\# att = 3\} = p^2(1-p)$

$\vdots$

$P\{\# att = k\} = p^{k-1}(1-p)$

$= (1-p) \sum_{k=1}^{\infty} k \cdot p^{k-1}$

$\underbrace{\sum_{k=0}^{\infty} p^k}_{f(p)} = \underbrace{\frac{1}{1-p}}_{g(p)} \quad p<1 \quad = (1-p)^{-1}$

$$P\{\text{tx ok con } k \text{ tentativi}\} = p^{k-1}(1-p)$$

$$N_t = \sum_{k=1}^{\infty} \boxed{k p^{k-1}(1-p)} = \frac{1}{1-p}$$

$= \sum_{k=1}^{\infty} k \cdot P\{\# att = k\}$

$f'(p) = g'(p)$

$\sum_{k=1}^{\infty} k \cdot p^{k-1} = -1 \cdot (1-p)^{-2} \cdot (-1) = \frac{1}{(1-p)^2}$

$$\boxed{\overline{T}_1} = (N_t - 1)T_O + T_1$$

$$= \frac{p}{1-p} T_O + T_1 = \frac{p T_O + (1-p) T_1}{1-p}$$

Time-out

p. 78

$p \ll 1$

$$\simeq \frac{T_1}{1-p}$$

$$\rho = (1-p)\rho_0$$

$\frac{T_U}{T_1} = \frac{T_U}{T_1/(1-p)}$

$\frac{T_U}{T_1}$

LEC 5