Communication in Electronic Systems

Lecture 5: Introduction to Communication Systems

Lecturer: Petar Popovski

TA: Junya Shiraishi, João H. Inacio de Souza

email: petarp@.es.aau.dk

AALBORG UNIVERSITY
DENMARK

Connectivity

Course Overview: Part 2. Communication and Networking

- **MM5: Introduction to Communication Systems**
- MM6: Simple Multiuser Systems
- MM7: Layered System Design. Network Topology and Architecture
- MM8: Networking and Transport Layers
- MM9: Introduction to Security
- MM10: Packets and Digital Modulation
- MM11: Communication waveforms
- MM12: Workshop on modulation and link operation

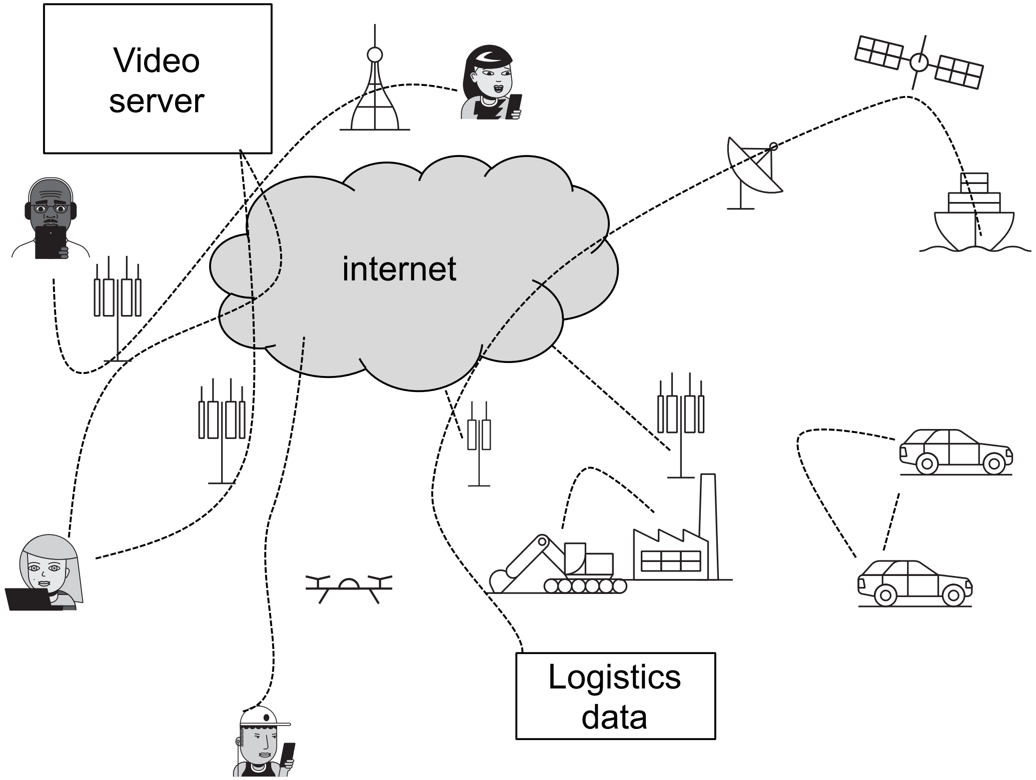
objectives

- learn the basic principles of communications and networks
- learn how to make mathematical models to analyze
  their performance and understand tradeoffs
- get insights into how some practical protocols operate

outline

- information, bits, and messages
- communication channels
- basic communication over ideal binary channel
- communication under random errors
- example of a RS232 link

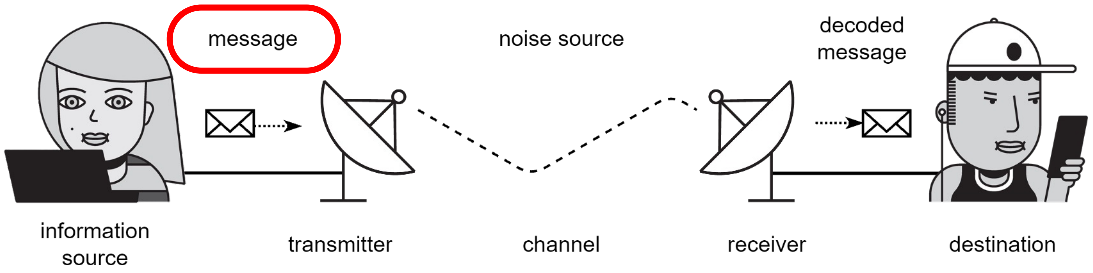
the complex connectivity ecosystem



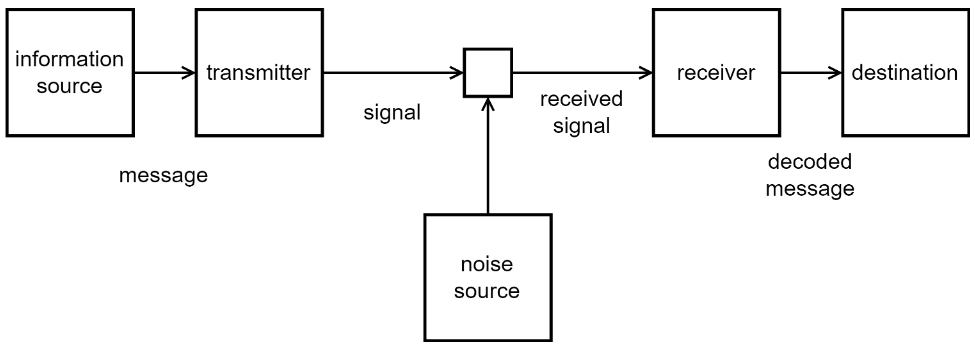
- many different types of connections
  - compare Bluetooth vs. satellite
  - data speed for streaming vs. data speed for texting
  - device capability
  - …

communication system: what does it take for it to work?

- system



- system model
  - a simplified version of reality
  - capable of capturing the important details



communication actors and messages

- the actors
  - introducing Alice, Bob and the friends (Carol, Dave, Eve, …)
  - also from the opposite side: Zoya, Yoshi, Xia, …

- what does it mean to communicate data from Alice to Bob?
  - Bob knows that Alice can be in one of two states (happy or sad), but does not know in which state Alice is now
  - the uncertainty of Bob can be resolved if Alice sends a message containing happy or sad

defining a bit of information

- in the previous example we did not speak about
  how much uncertain is Bob about the mood of Alice
  - in order to quantify this uncertainty, we can introduce a probability model for Alice's mood

    Prob(happy)=0.7 and Prob(sad)=0.3
  - the uncertainty for Bob is highest when

    Prob(happy)=Prob(sad)=0.5

- this brings us to a definition of a data bit

> a single bit of information corresponds to learning about
> the state of a system that can have two equally probable states

representing messages (1)

- in general, Alice can be in M different states
  - each state corresponds to a different message
- we can represent each of these states using the symbols 0 and 1
  - one possible representation

state 0	state 1	state 2	state 3	...
0	01	011	0111	...

  - but is not good, as we are striving to minimize the number of symbols that we should sent think of each symbol 0 or 1 having some (equal) cost
    - average number of symbols sent: $\frac{M+1}{2}$

representing messages (2)

- we can do better
  - for simplicity, assume $M=2^d$, where d is an integer
  - we can represent the states through a d-bit binary number representation

state 0	state 1	state 2	state 3	...
00...00	00...01	00...10	00...11	...

  - here we need on average $\log_2(M)$ bits

representing messages by a certain set of symbols
is called source coding

source coding (1)

- let us now assume that there are 8 different messages for Alice
- but they have different probability

Msg #	1	2	3	4	5	6	7	8
Prob	1/4	1/4	1/12	1/12	1/12	1/12	1/12	1/12
Coding	000	001	010	011	100	101	110	111

- probability that the first symbol is 0: 2/4 + 2/12 = 3/4
- probability that the first symbol is 1: 1/4
- the first symbol does not contain 1 bit of information, but less
- the average message length is 3 symbols

source coding (2)

Msg #	1	2	3	4	5	6	7	8
Prob	1/4	1/4	1/12	1/12	1/12	1/12	1/12	1/12
Code 1	00	01	1000	1001	1010	1011	1100	1101
Code 2	00	01	1000	1001	1010	1011	110	111
Code 3	00	01	0110	0111	1000	1011	110	111

- an idea known since the Morse code:
- use less symbols for more probable messages

- average number of symbols for Code 2 and Code 3:

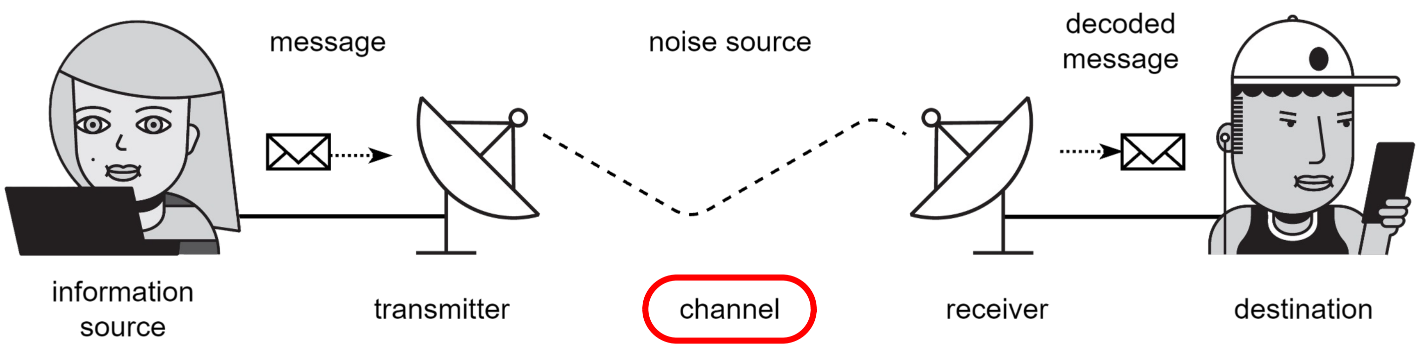
$$34 / 12 < 3$$

Yet, Code 2 is good, but Code 3 not.

exercises (5 min)

- calculate the average number of symbols per message
- why is Code 3 not so efficient?

communication channel



- communication channel is any physical or logical object that can represent more than one state

DNA

we can encode a message in chair arrangement

here we are interested in
electronic communication

limits on the communication channels

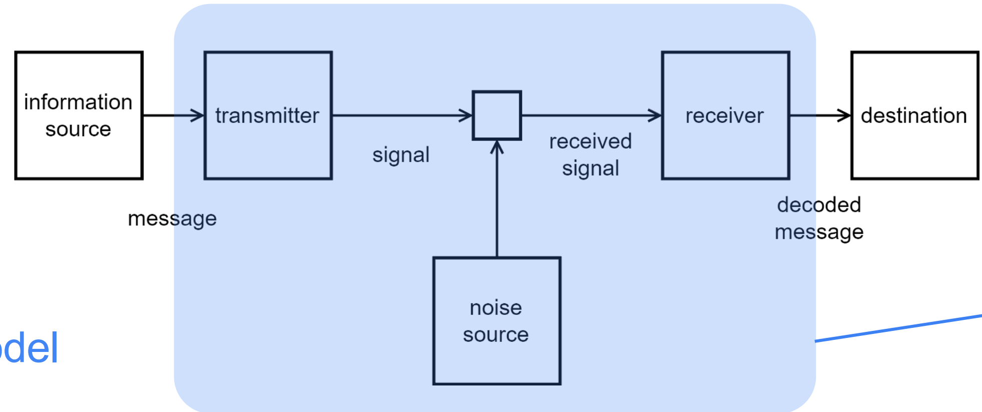
- the communication channel has physical limitations on the amount of bits it can carry
  - how many bits can we send by smoke signals

- two limits
  - speed by which symbols can be sent
    - limit on the shortest smoke signal duration

- the number of states that can be reliably differentiated when the symbols are received
  - smoke signals destroyed by wind

a model of a communication system

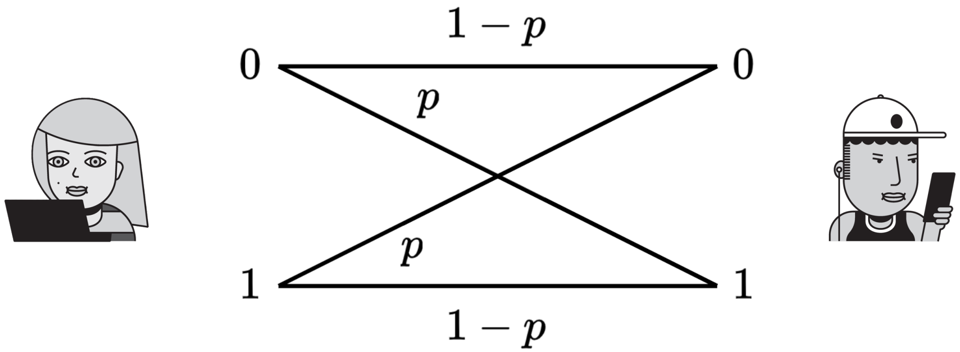
- introduced by C. E. Shannon in 1948
- there are also adversarial channels



noise is used to model all the random unpredictable factors that can affect the communication

communication channel for the (compressed) messages

a binary communication channel



- a black box that gets 0 or 1 as an input and provides 0 or 1 at the output
  - due to noise, sending 0 can result in receiving 1

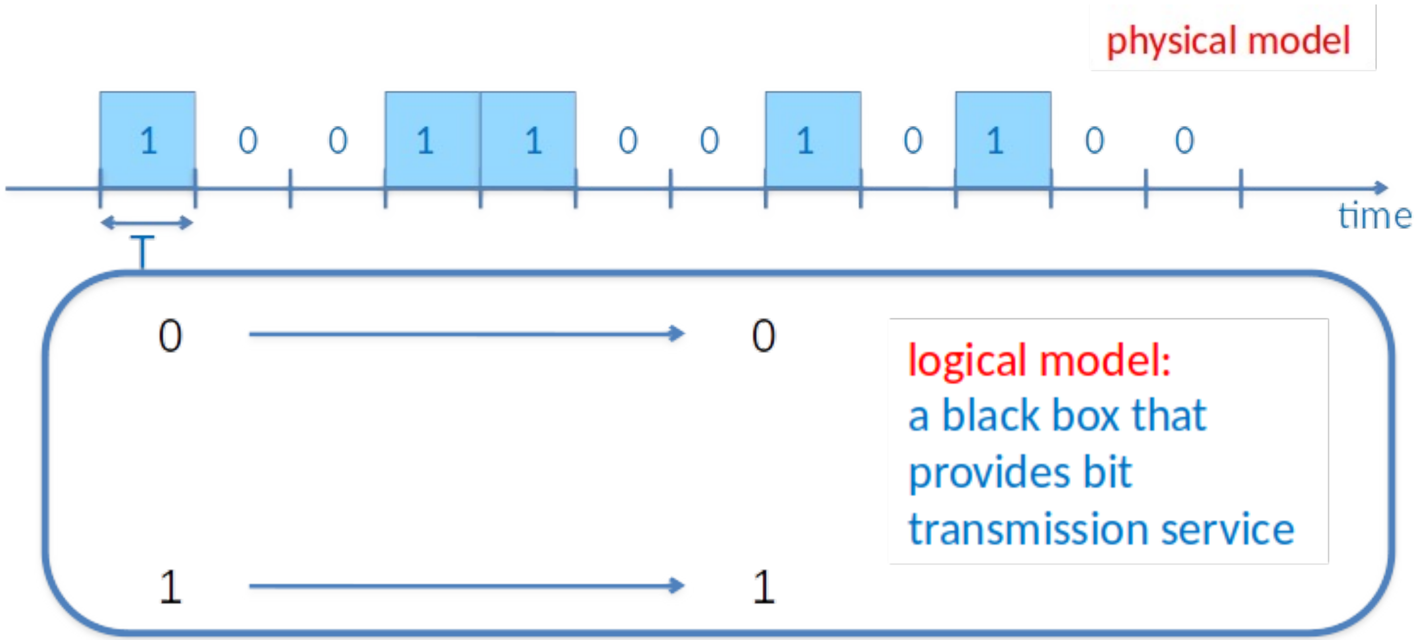
- this model does not say how fast we can send 0 or 1,
  it is decided by other parameters of the whole system
  - the diagram depicts how the channel behaves in a single channel use

ideal binary channel

- each T seconds Alice can decide to send or not to send a pulse to Bob

  pulse=1 no pulse=0

  - Bob receives the pulse perfectly



how to communicate messages over this channel

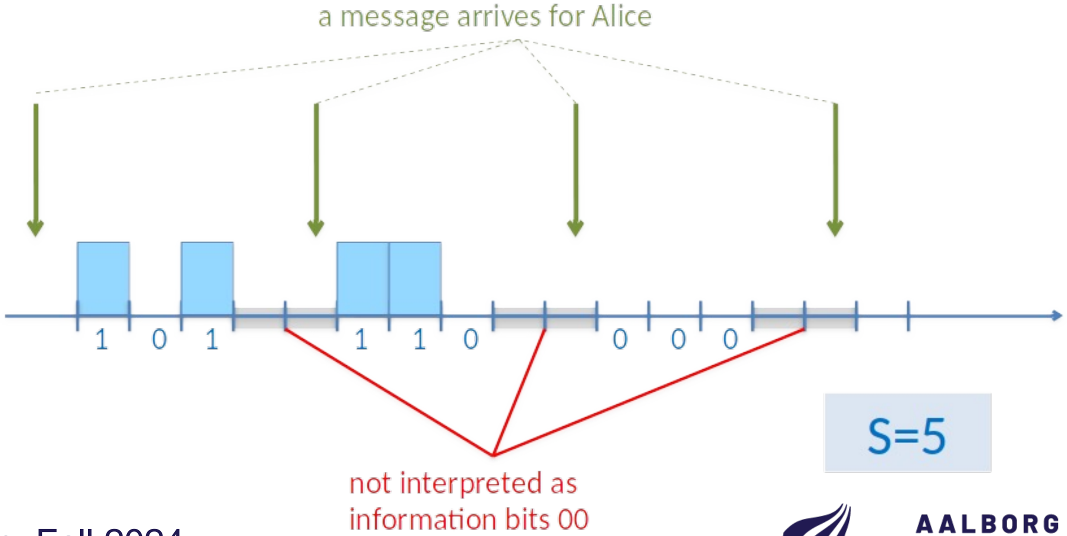
- let us assume that Alice has 8 different 3-bit messages

- two fundamentally different situations
  - Alice periodically transmits 3-bit messages
  - Alice transmits a new 3-bit message only upon being triggered by an event

Alice sends data to Bob periodically

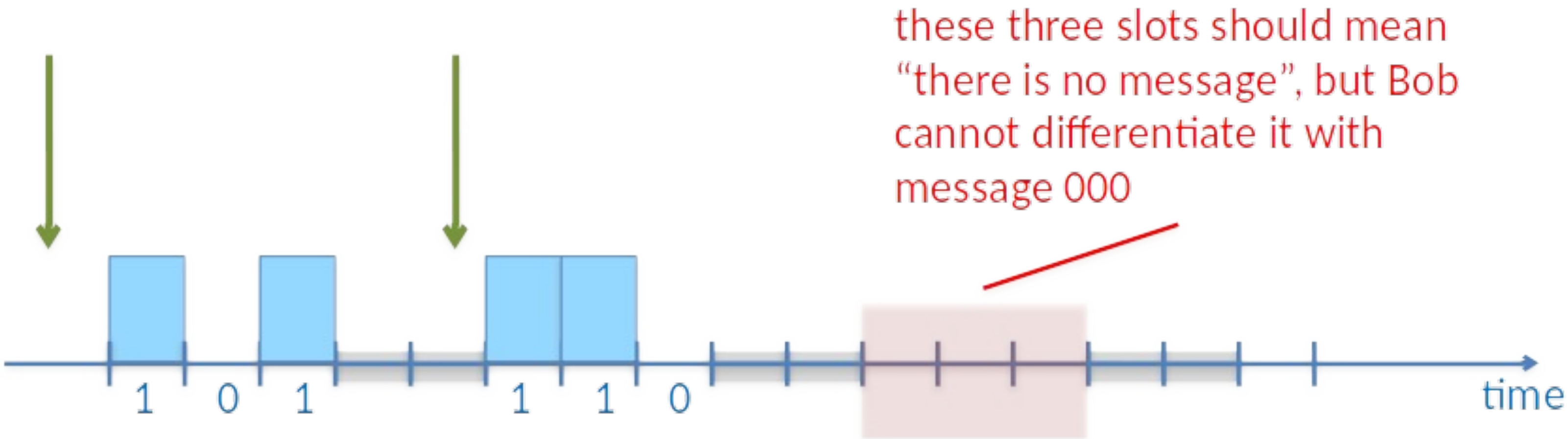
- here we assume that there has been a setup phase in the past through which Alice and Bob agreed several things:
  - common numbering of the slots used to send the bits
  - Alice sends messages to Bob with a period of S slots, where S>3
    - the first message is sent in slots 1,2,3
    - the second message is sent in slots S+1, S+2, S+3…

- with these agreements the only uncertainty for Bob is the content of the message bits



Alice sends data to Bob upon an event trigger (1)

- this is more complicated, as now there is an uncertainty also on the moment when the message is sent, not only on the message content
- we can still use the trick with predefined periodic structure moments when the message starts
  - however, there is a problem..



Alice sends data to Bob upon an event trigger (2)

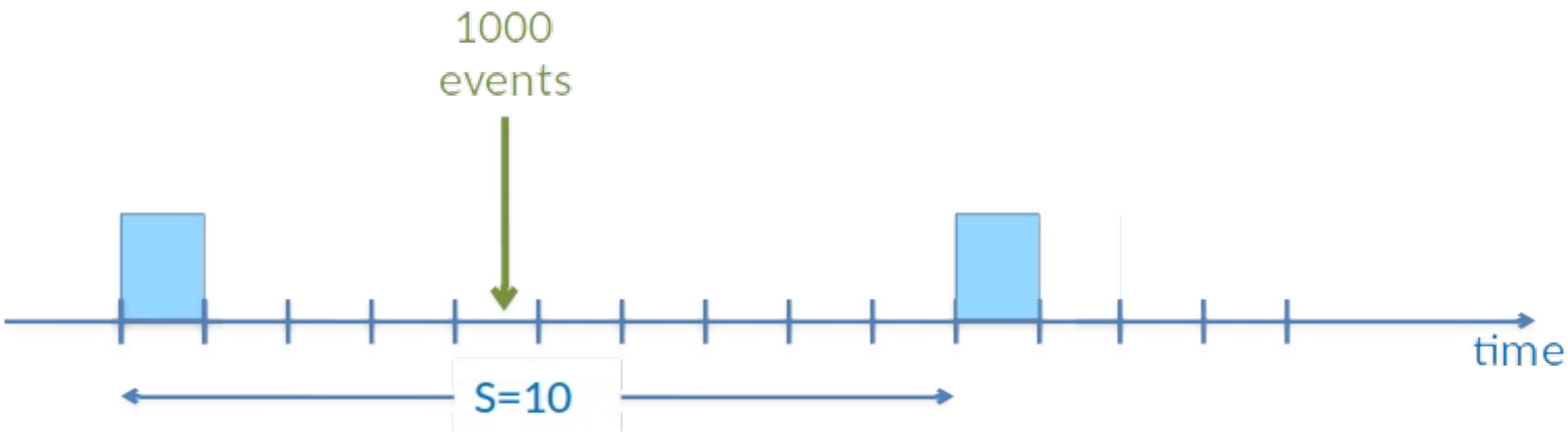
- we can solve this by expanding our set of possible messages!
  - the new set has 9 messages
  - messages 1-8 are the messages containing data
  - message 9 is “no data”
  - we can now encode the messages by prefixing each data message with 1

msg	1	2	3	4	5	6	7	8	9
code	1000	1001	1010	1011	1100	1101	1110	1111	0

Alice sends data to Bob upon an event trigger (3)

- the previous solution is not always good
  - let us assume that T=1 ms
  - thus, if we sent information bit in every slot, we get a data rate of 1 [kbps] kilobits per second
  - let us assume that the period of transmission is S=10
  - if 3 bits are sent each 10 ms, the data rate is 300 [bps] bits per second
- assume that on average one event occurs per second
  - a data rate of 3 [bps] should suffice!
- however, the events do not occur regularly and may occur in a burst
  - for example, 1000 events within an interval of 10 ms
    and then no event for the coming 1000 seconds

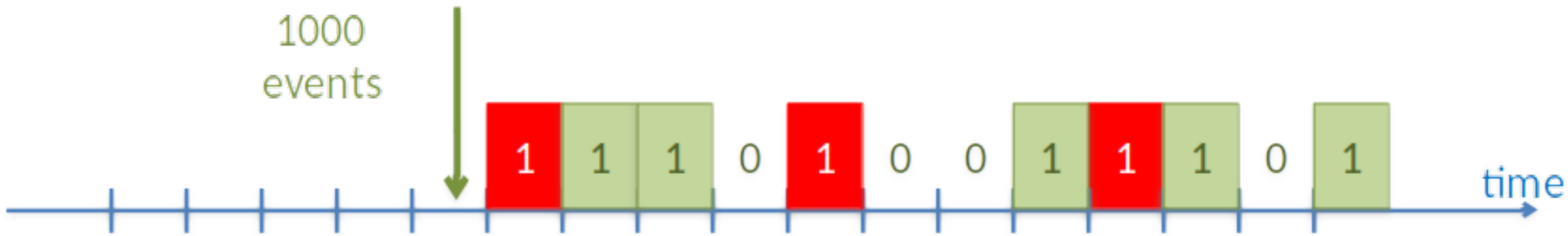
Alice sends data to Bob upon an event trigger (4)



- sending these events using transmissions with period 10 will take in total 10 seconds
  - this means that many of the events experience a large delay from the occurrence until being reported.
  - the highest delay is 10 seconds
  - what would be the expected/average delay?

Alice sends data to Bob upon an event trigger (5)

- an idea: start packet transmission at any time
  - use “1” to denote the packet start = a preamble; and send the 3 bits afterwards
  - this enables on-demand transmission
  - one packet takes 4 ms, all packets will be sent in 4 s
  - this decreases the maximal delay from 10 s to 4 s
  - what is the average delay?



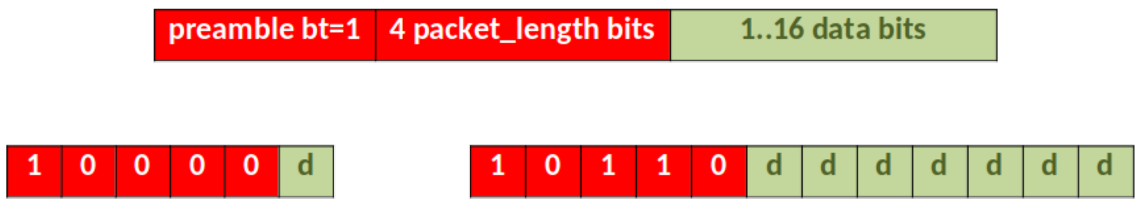
making data packets with variable length (1)

- the transmission methods described previously can be used whenever the packet length is fixed in advance
- packet length can vary
  - different applications using the same communication system
  - variable length compression (described before)
- we present one possible way to deal with it
  - introduce bits in the packet to tell what is the length of the data part

general principle:
any flexibility (starting time, packet length, etc.)
is paid by extra signaling

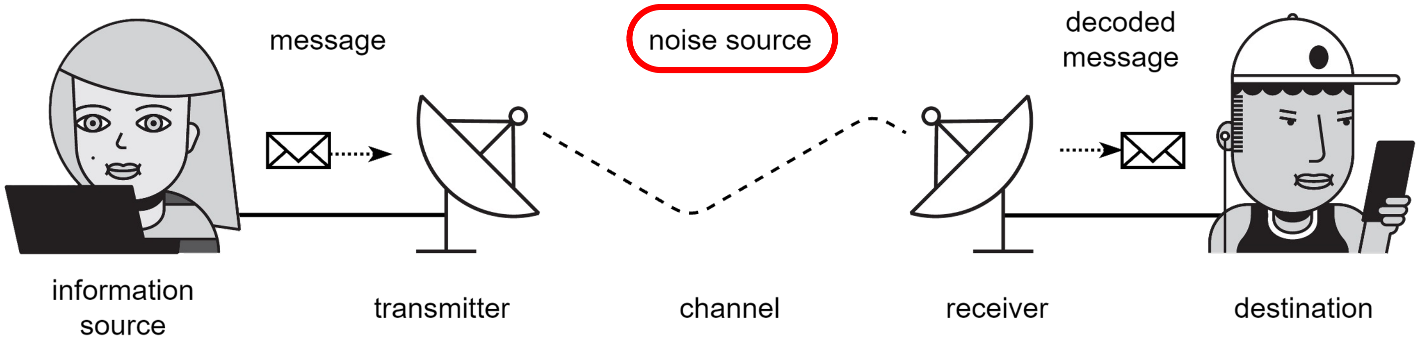
making data packets with variable length (2)

- we need to fix the assumption of the minimal and the maximal packet length
  - this will fix the number of bits required to indicate the packet length
  - example:
    - minimal number of data bits is 1;
      maximal number of data bits is 16.
    - this will fix the number of packet_length bits to 4

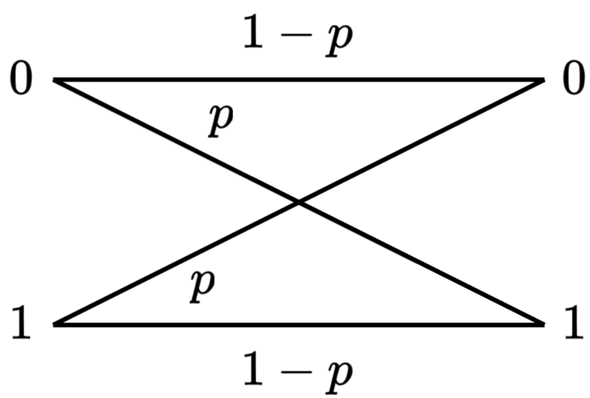


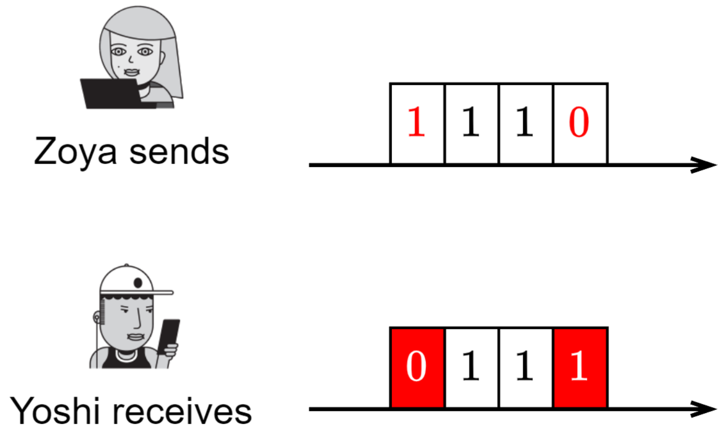
d=arbitrary data bit

communications under random errors



random transmission errors

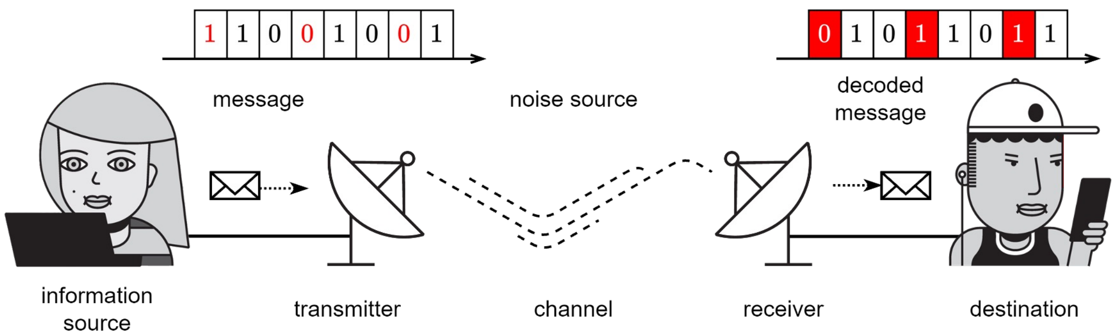




- in our model all bits are equal
- however, the effect of the errors are not equal for all bits in the packet
  - if the preamble bit is error, then the whole packet is lost
  - if Alice send nothing, but Bob erroneously receives 1, then he will erroneously decode a packet (likely 000)

counting the errors (1)

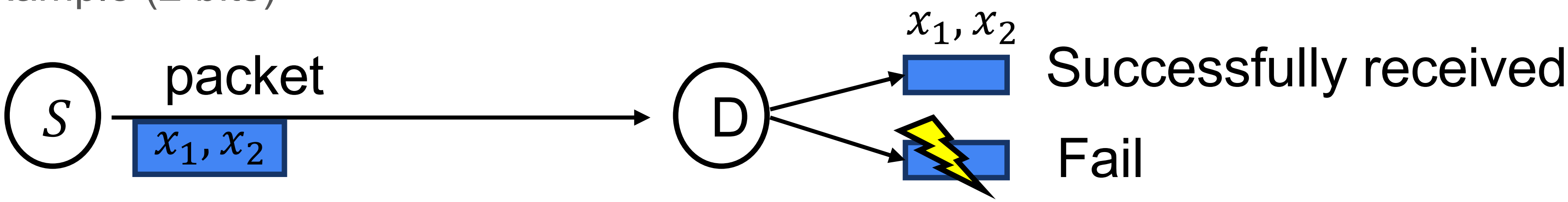
- Bit Error Rate (BER)
  - number of bits in error divided by the number of transmitted bits
  - key performance indicator for digital communication systems
  - affected by noise, interference, synchronization problems, channel fading



$$p_{\text{BER}} = \frac{3}{8} = 37.5\%$$

counting the errors (2) -Packet Error Ratio (PER)

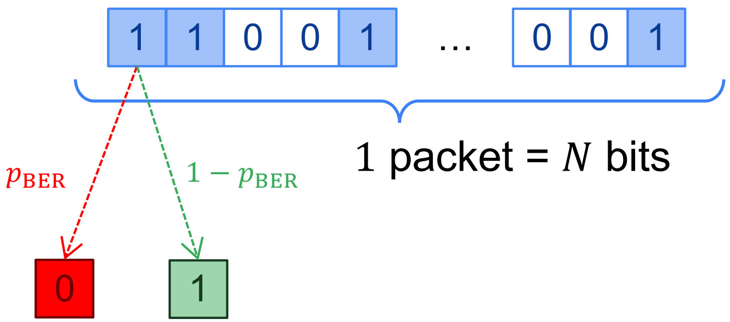
- packet error: at least one bit is flipped after delivering at the receiver
- specific example (2 bits)



1st bit	2nd bit	Received packet	Probability
x_1	x_2	Success	5/8 *5/8
$\overline{x_1}$	x_2	Failure	3/8 *5/8
x_1	$\overline{x_2}$	Failure	5/8 *3/8
$\overline{x_1}$	$\overline{x_2}$	Failure	3/8 *3/8

counting the errors (3)

- Packet Error Ratio (PER)
  - ratio of packets not successfully received
  - the PER depends on the BER
    - if one bit is erroneous, the entire packet is erroneous too



$$p_{\mathrm{PER}} = 1 - (1 - p_{\mathrm{BER}})(1 - p_{\mathrm{BER}}) \ldots (1 - p_{\mathrm{BER}})$$
$$= 1 - (1 - p_{\mathrm{BER}})^N$$

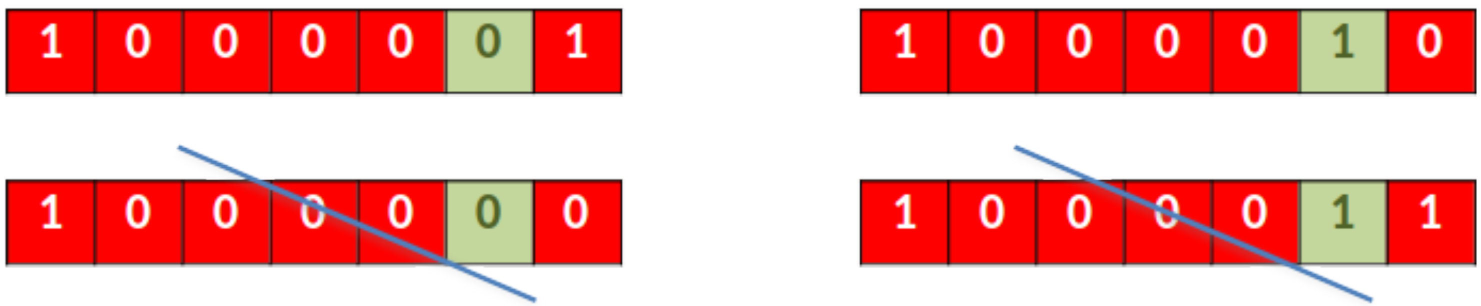
things we will do to deal with errors

- introduce a mechanism to detect errors
  - Bob will know if the received packet has errors or not
- introduce a feedback link
  - Bob can tell Alice if the packet has errors
- improve the reliability of the preamble
  - the packet is not missed if 1 is interpreted as 0
- enable Alice to retransmit a packet again
  - eventually the packet arrives at Bob

error detection (1)

- in order to detect errors, we must put additional redundancy to the packet
  - otherwise, any received packet is valid
  - we want errors to result in something that Bob recognizes as an invalid packet

- parity check
- the simplest way to detect one error
- a single redundant bit is required



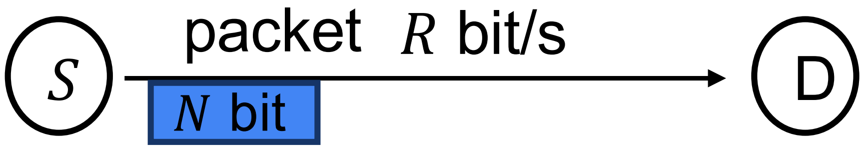
error detection (2)

- parity check is a very weak error detection code
  - if errors occur in two different bit positions, then the packet is accepted as correct

- CRC (Cyclic Redundancy Check) codes
  - come in different bit lengths, e. g. 8, 16, 32
  - very high probability to detect errors
    - the packet is in error if bit errors occur in the CRC bits or the other bits in the packet
  - yet, it is not perfect
    - there is always a probability of undetected error
  - the value of the K check bits is a function of the other M bits
    - it must happen that $2^{(M-K)}$ bit combinations have exactly the same parity check

throughput

- def: the expected number of messages successfully delivered to the receiver per unit time
  - Symbol/s or bits/s
- preliminary
  - the sender transmits packet with n bit with R bps
  - use the PER derived in the previous slides



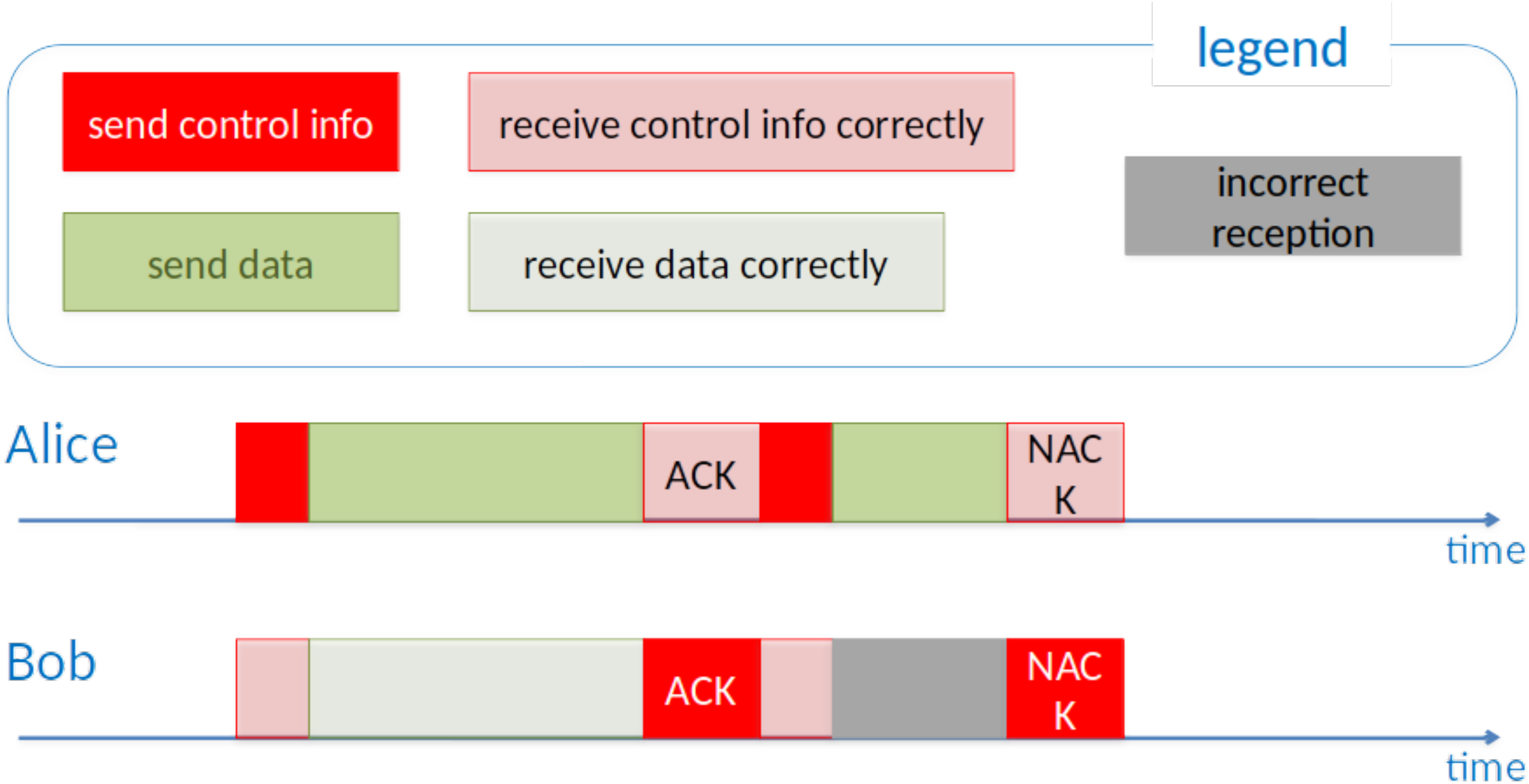
feedback link from Bob (1)

- assumption: data is only flowing from Alice to Bob
- Bob needs to be able to send
  - ACK (acknowledgement) if the packet is correct
  - NACK (not acknowledgement) otherwise
- how can the transmission by Bob be done?
  - a separate cable to carry the signals from Bob to Alice
  - transmission over the same cable, but not simultaneously
    - Time Division Duplex (TDD)
  - full-duplex transmission over the same cable with echo cancellation
- we will assume TDD operation
  - largely applicable in wireless communication

feedback link from Bob (2)

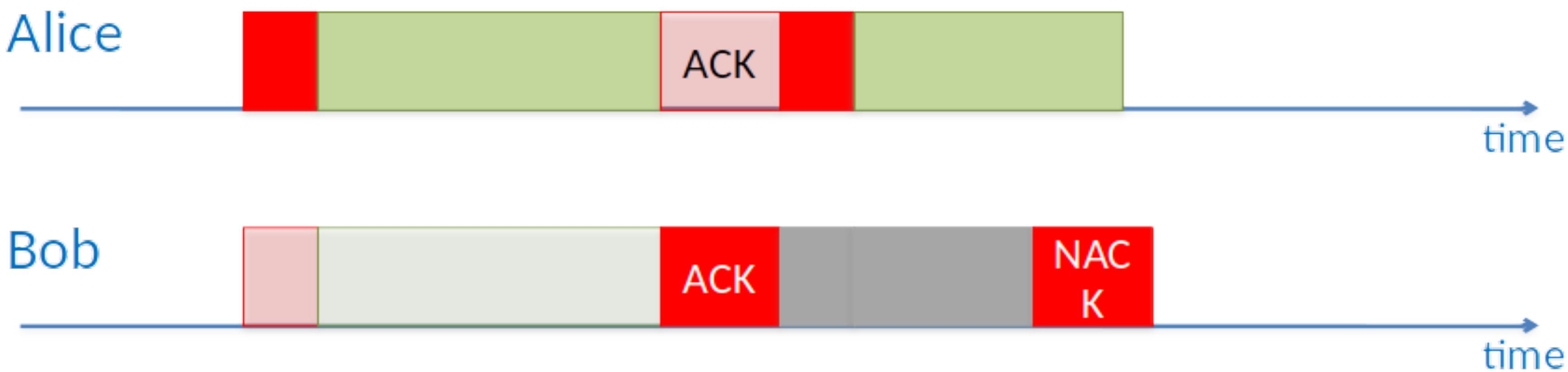
- in a TDD operation, Alice and Bob agree upon the following protocol
  - after each packet sent by Alice, Bob starts his transmission
- Bob needs to send 1 bit (ACK or NACK)
  - but this is highly unreliable, Alice is not even able to detect if there is an error
- we assume that the ACK/NACK packet sent by Bob has a length of 1 byte more reliable transmission (to be elaborated later)
  - still only 1 data bit for Alice (ACK or NACK)
    - Q: does the ACK/NACK bit carry one bit of information?

how should the TDD protocol work



feedback link from Bob (3)

- what can go wrong if Bob does not detect an error?
  - the obvious one – erroneous data will be passed on to Bob
    - the less obvious one: the protocol can break down
      - for example, Bob incorrectly detects the packet length



Automatic Retransmission ReQuest (ARQ) protocol (1)

- upon receiving NACK, Alice resends the same packet
  - repeats this until receiving ACK for this packet
  - this is a stop-and-wait ARQ protocol

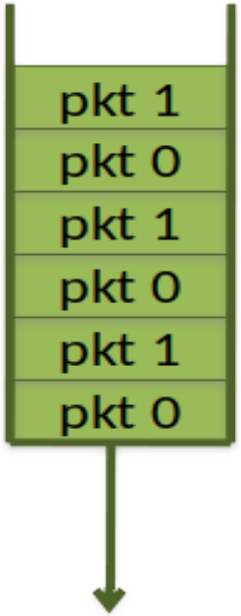
- note that NACK can be implicit
  - if no ACK arrives, Alice resends the same packet
  - this solves the situation when Bob erroneously detects the packet length

Automatic Retransmission reQuest (ARQ) protocol (2)

- a different challenge
  - error in ACK/NACK reception
- Example
  - Alice sends (1)(0101)(1)(011101)
  - Bob receives it correctly and sends ACK
  - error occurs and Alice receives NACK
  - Alice retransmits the packet (1)(0101)(1)(011101)
  - the problem: Bob thinks this is a new packet with data bits (011101), not a retransmission of the old one

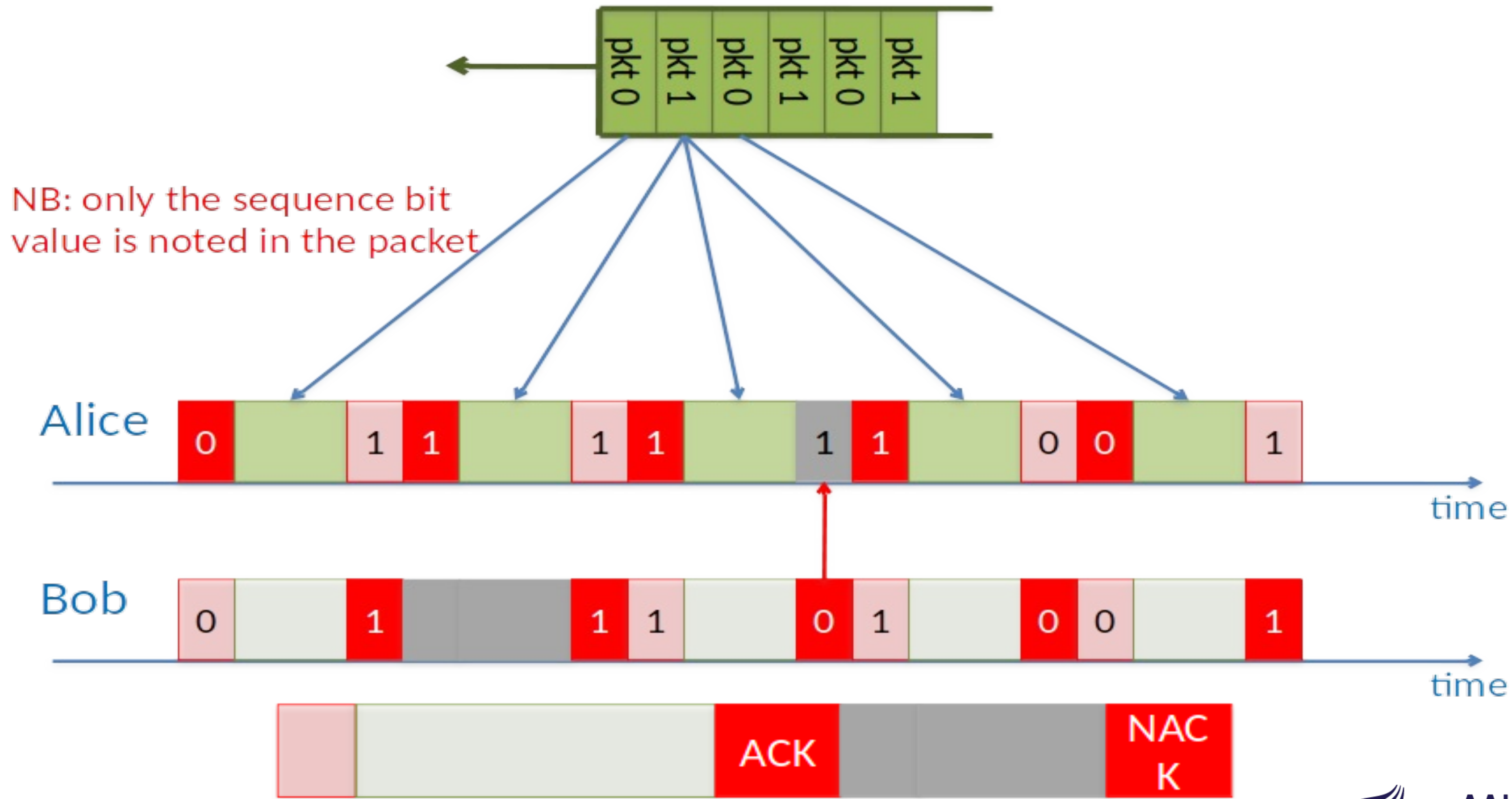
ARQ with sequence number (1)

- the problem is solved by a sequence number
- using a single-bit sequence number
  - Alice assigns modulo-2 numbers to the data packets in its queue



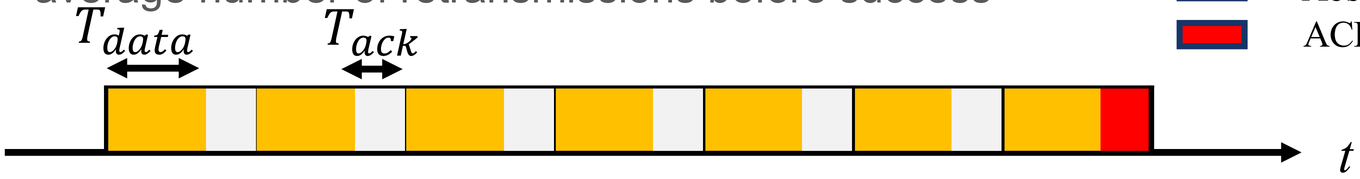
- Bob does not send ACK/NACK, but sends a bit to say which sequence bit value he expects next
- Alice increases the sequence number modulo 2 only when it is sure that Bob has received it

ARQ with sequence number (2)



ARQ performance

- counting the errors/retransmissions
  - average number of retransmissions before success



- probability of successful reception at the k-th attempt using the p_{PER}

$$p_k = \underbrace{p_{\text{PER}} \dots p_{\text{PER}}}_{k-1 \text{ times}} (1 - p_{\text{PER}}) = p_{\text{PER}}^{k-1}(1 - p_{\text{PER}})$$

- average number of delay

$$\bar{T} = (T_{data} + T_{ack})p_1 + 2(T_{data} + T_{ack})p_2 + \cdots = \sum_{k=0}^{\infty} k(T_{data} + T_{ack})p_k = \frac{T_{data} + T_{ack}}{1 - p_{\text{PER}}}$$

two-way connection and piggybacking

- we now make Alice and Bob "equal"
  - Bob is also able to send data
  - Alice is able to send ACK/NACK (or ARQ sequence bits) to Bob
- packet format
  - the other seq bit is piggybacked on the data

preamble	data length	own seq bit	other seq bit	parity bit	data

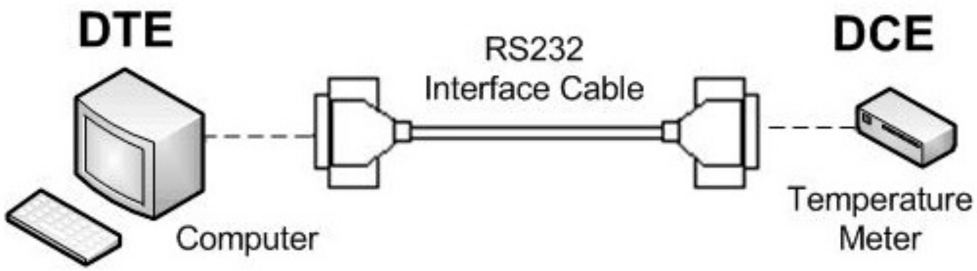
few remarks

- the described packet is usually called frame, thus differentiating from a data packet
- it is common that a packet contains an integer number of bytes
  - then a frame preamble is certain sequence of bits rather than one bit
  - packet length is given in number of bytes
  - stronger error detection code is added
- we fix now the frame structure as follows

preamble	pkt length	own seq number	other seq number	CRC error detection	data
1 byte	6 bits	1 bit	1 bit	1 byte	1-64 bytes

practical example - RS232

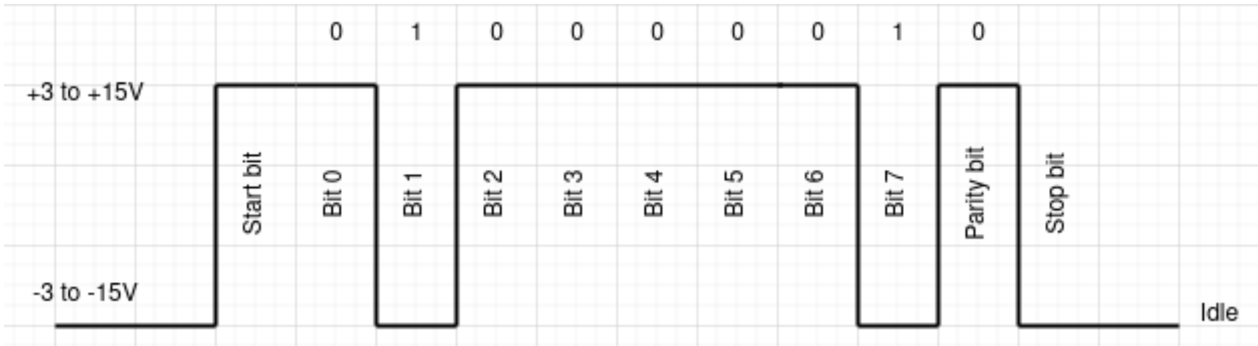
- invented in 1960
- based on serial communication, 1 to 1 communication
- made with modems in mind, but also used for other appliances like printer, memory devices, motors, and other serial connections
- Data Communication Equipment(DCE) and Data Terminal Equipment(DTE)



bit transmission and voltage levels

- binary communication
  - starts with a logical 0 for transmission
  - 3 to 15V for logical 0, -3 to –15 V for logical 1
  - sending an "A" (010000010) over RS232

- baud rate: the number of changes of the signal per second
- bud rate can be less, equal to or larger than the bit rate



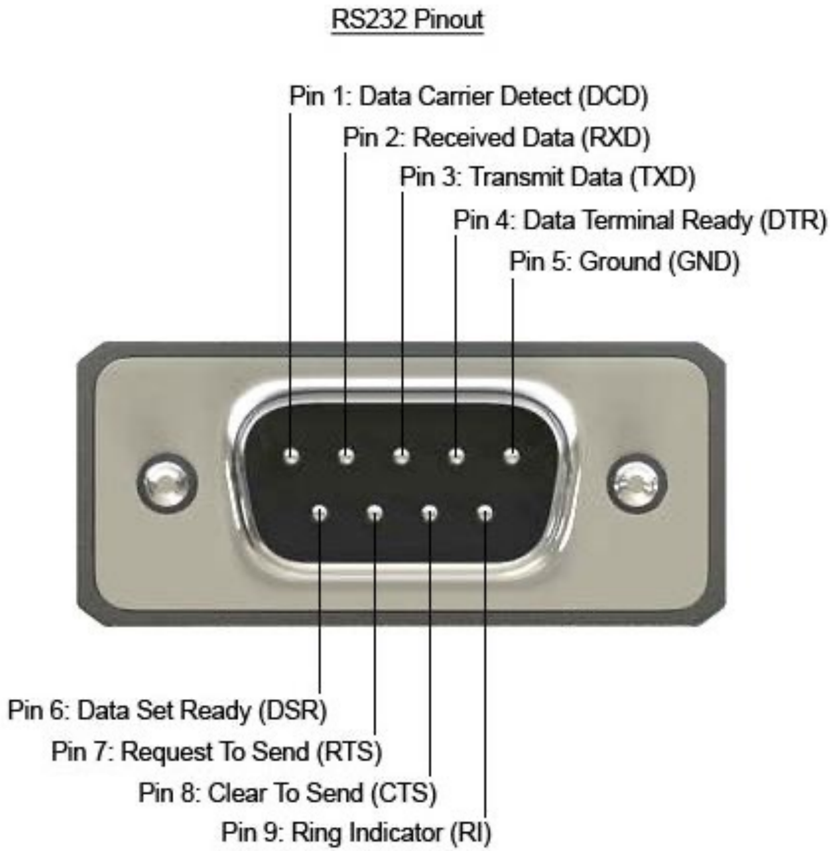
data rate and distance

- asynchronous protocol no clock reference needed at sender and receiver
  - Baud Rate (=bit rate in RS232) between the two must be the same
  - available capacity (Baud Rate) depends on distance

Baud Rate	Cable length
2400 bps	900m
4800 bps	300m
9600 bps	150m
19200 bps	15m
115200 bps	5m

notable pins (Male)

- flow control
  - Request to send(7) / clear to send(8)
- data pins
  - Received data(2) / transmit data(3)
- ground (5)
  - Common ground to infer voltage level from



summary and outlook

- model of a system is simplified, good-enough version
- information as an abstract entity
  that can be mapped to different physical carriers
- we have shown how to gradually build a serial link over a binary channel
- control information vs. data
- elementary protocol with retransmissions
- example of an RS232 links