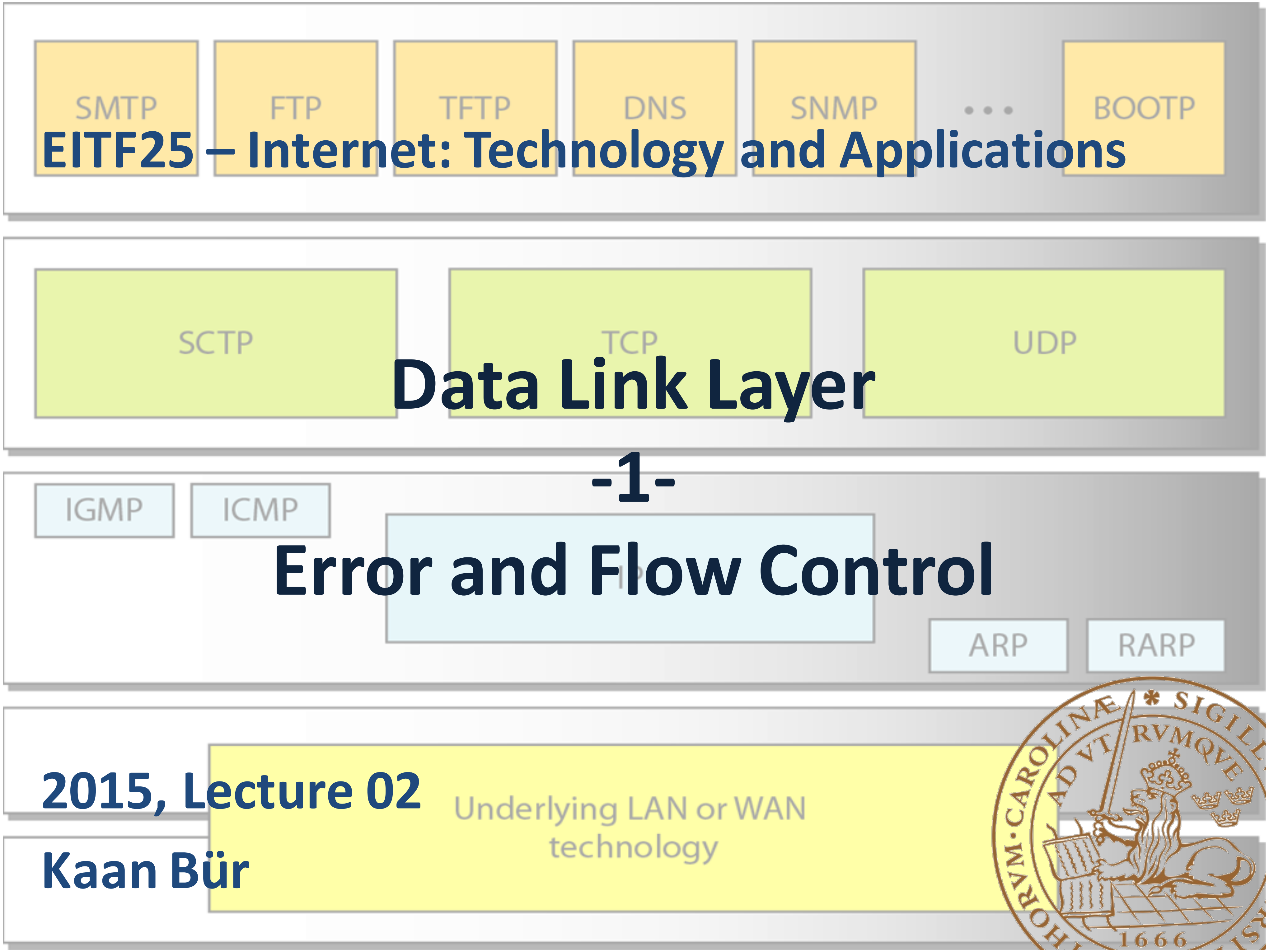
EITF25 – Internet: Technology and Applications

Data Link Layer

-1-

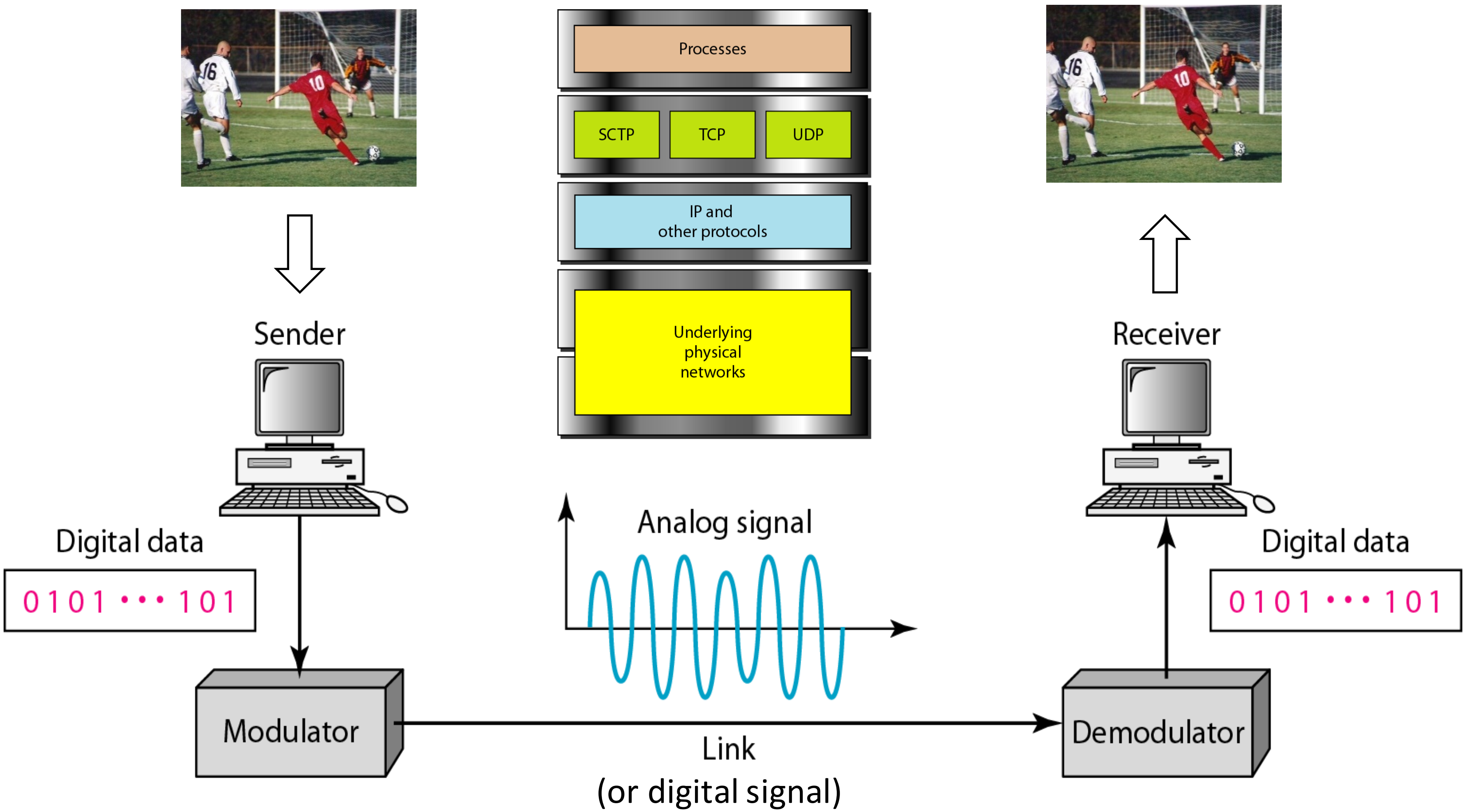
Error and Flow Control



2015, Lecture 02

Kaan Bür

Previously on EITF25



Data Link Layer

- Medium Access Control
  - Access to network
- Logical Link Control
  - Node-to-node error and flow control

Link layer protocols

Link layer protocols

- Error detection
  - All errors must be detected
- Error correction
  - Receiver must get correct data
- Flow control
  - Receiver must not be overloaded

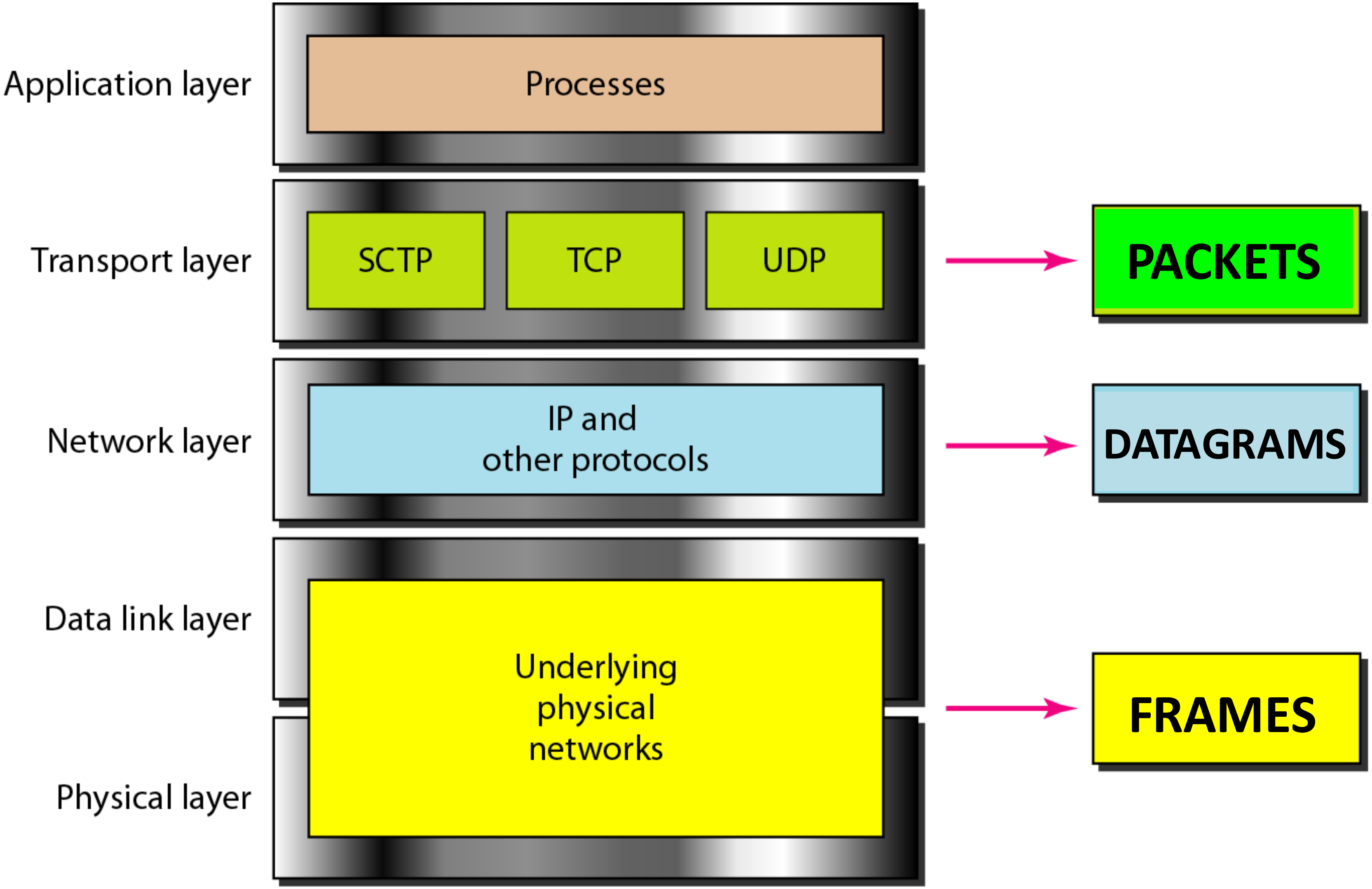
Internet: Data Link Layer (1)

Logical Link Control Sublayer

- Error detection and correction
  [S6.1-6][F10.1-5]
- Data link control, go-back-N
  [S7.1-2][F11.1-2, F23.2]
- Point-to-point protocol
  [F11.4]

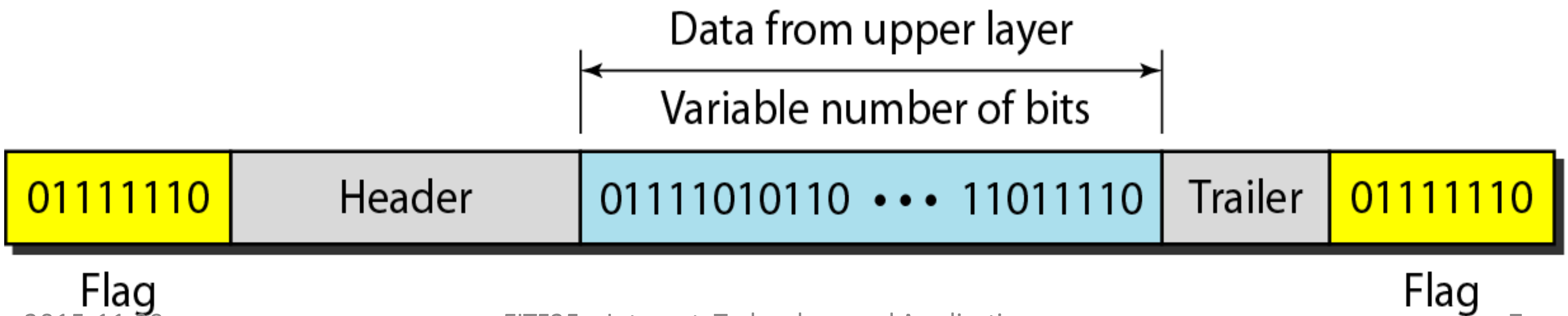
**[Kihl & Andersson: 4.1, 4.2, 4.3, 4.5]*

TCP/IP model and data units



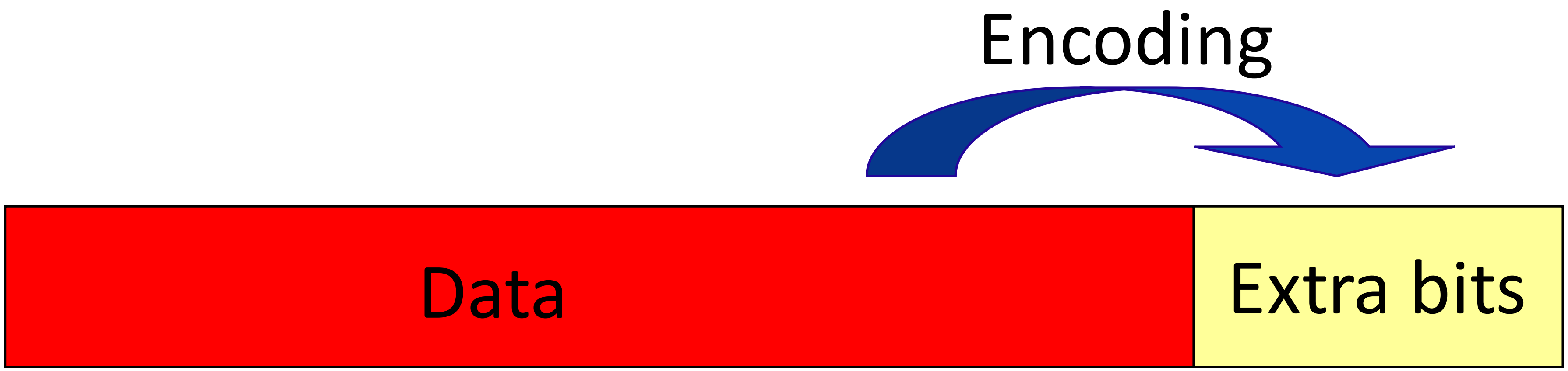
Framing

- Physical layer → bitstream
- Link layer → frames
- We need logical transmission units
  - Synchronisation points
  - Switching between users
  - Error handling



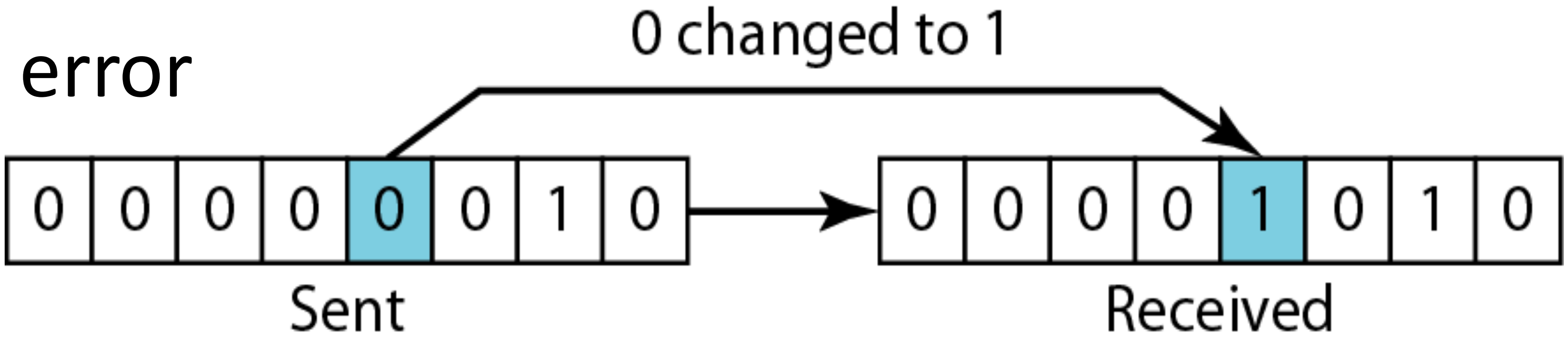
Error control

- Data assumed error-free by higher layers
  - Errors occur at lower layers (physical)
  - Job for LLC layer
- Extra (redundant) bits added to data
  - Generated by an encoding scheme from data

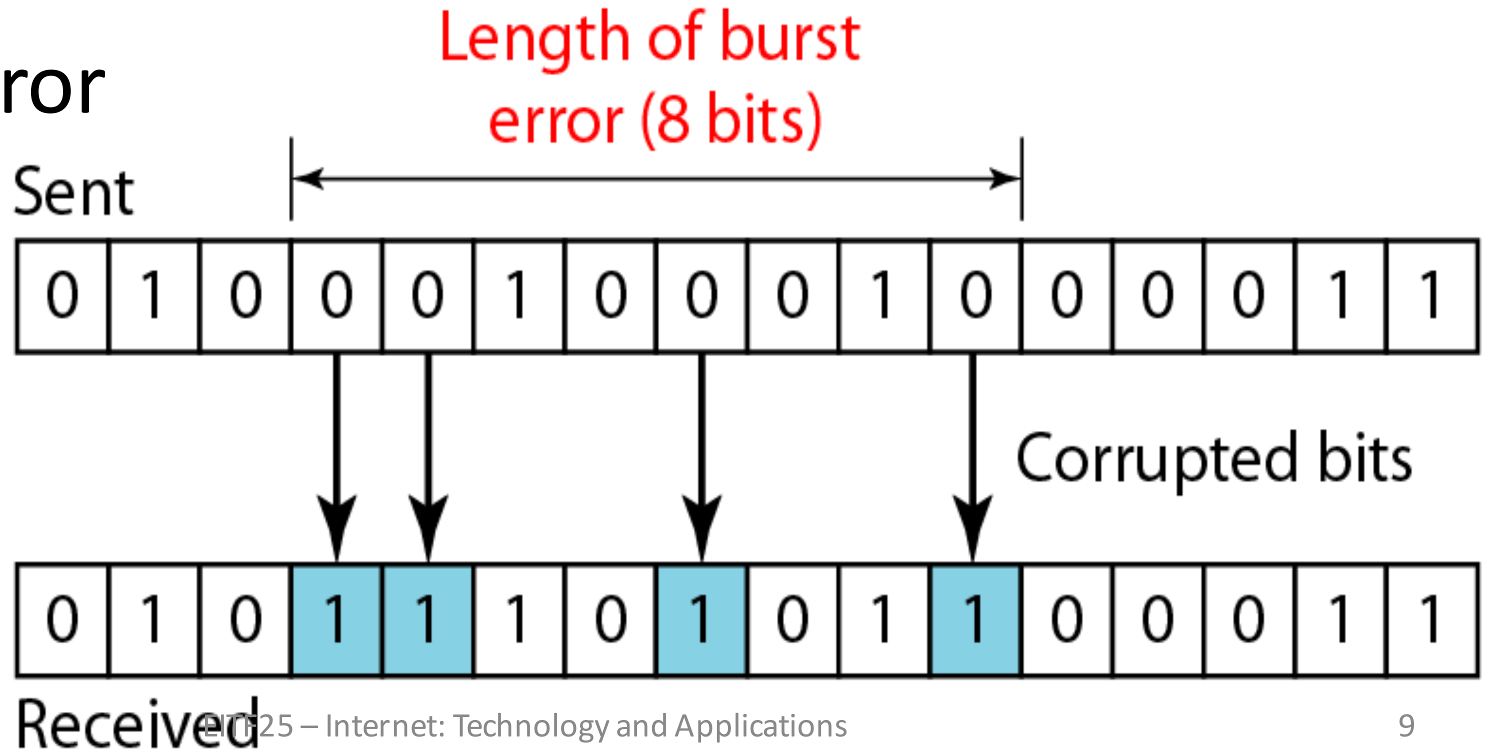


Error types

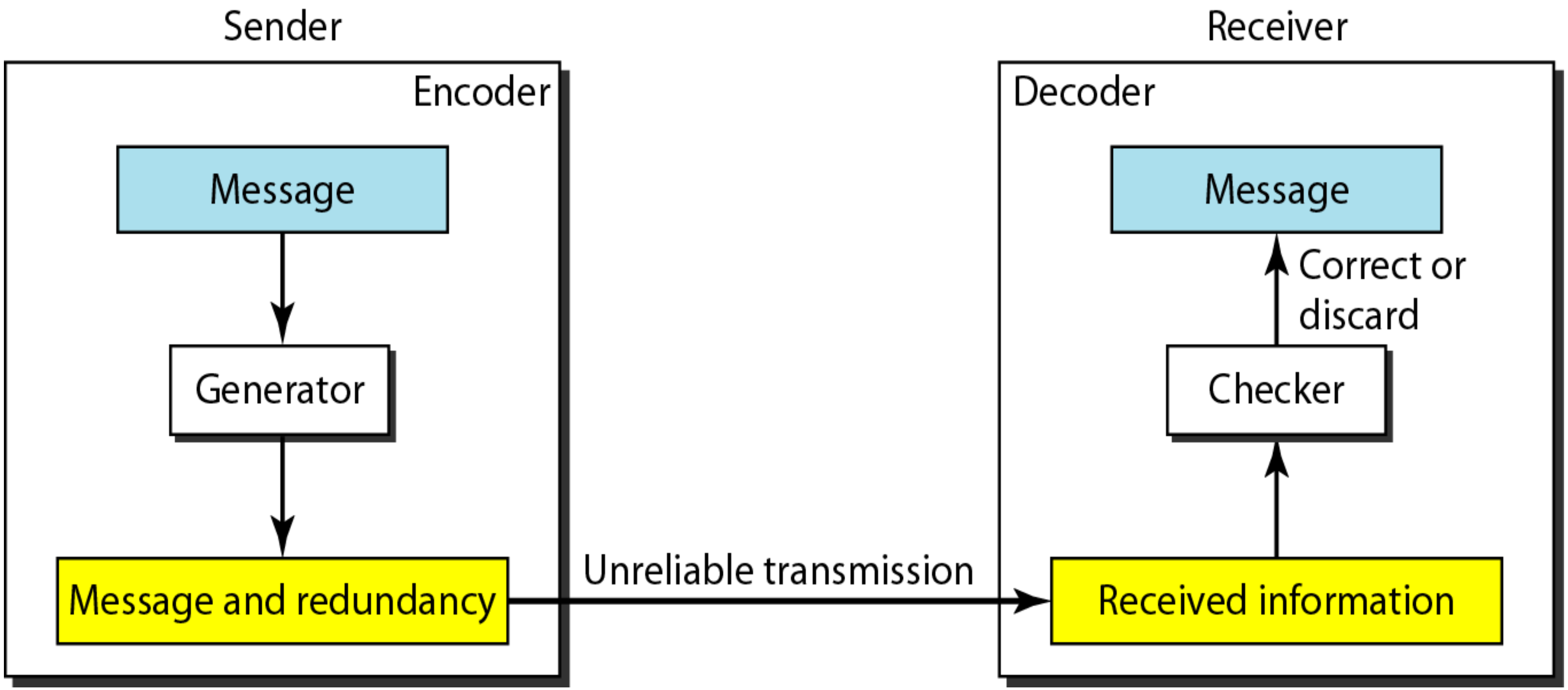
- Bit error



- Burst error



Error detection process



Error detection schemes

- Simple parity-check code
- Cyclic Redundancy Check (CRC)
- Checksum

Simple Parity-Check Code

- Extra bit added to make the total number of 1s in the codeword
  - Even → even parity
  - Odd → odd parity

dataword | | **codeword**

10011100 + 0 = 100111000

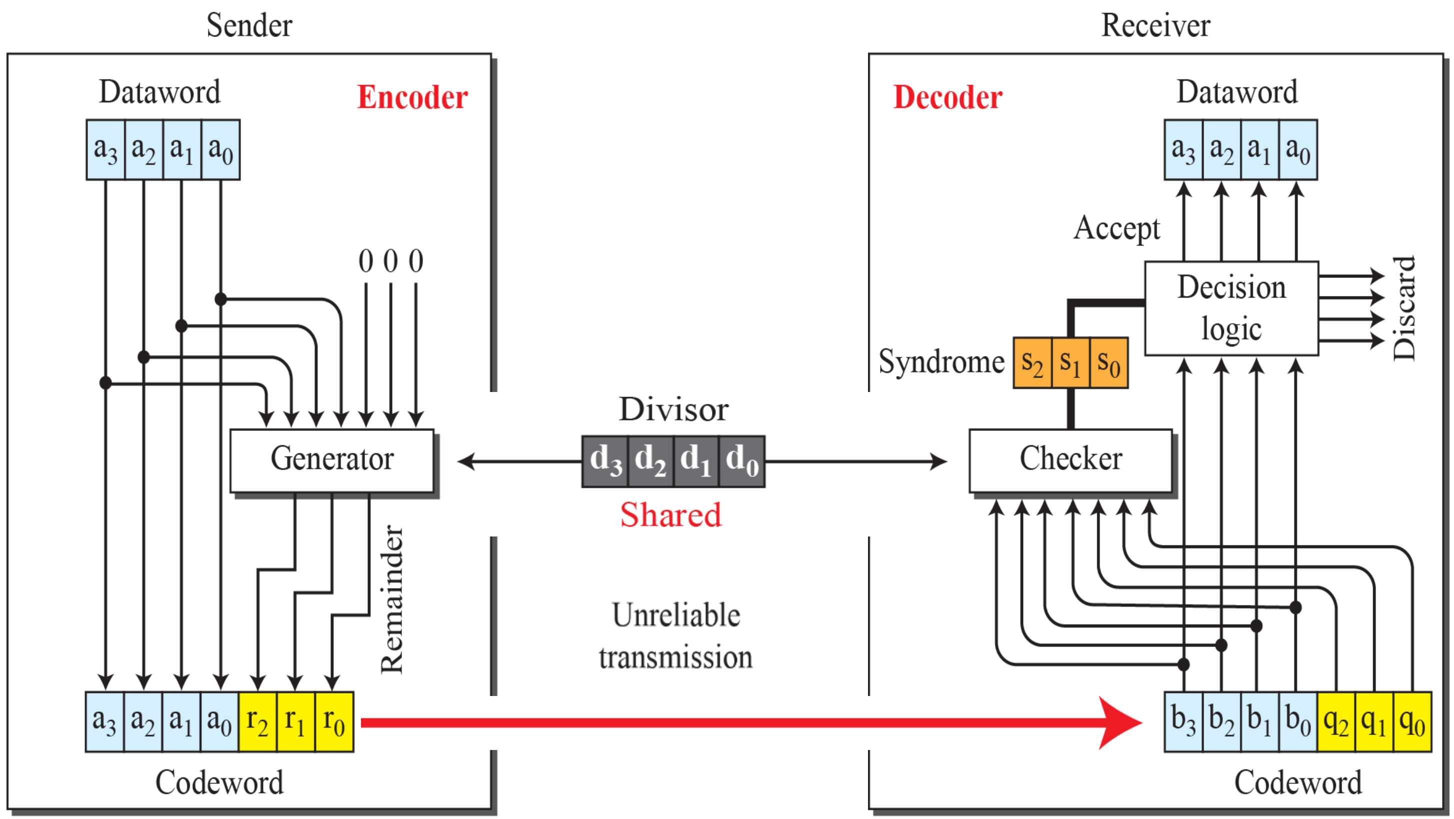
- Can detect an odd number of errors

Block coding

- Divide the message into k-bit blocks, called **datawords**.
- Add r redundant bits to each block. The resulting n-bit blocks ($n=k+r$) are called **codewords**.
- The code rate is $R=k/n$.

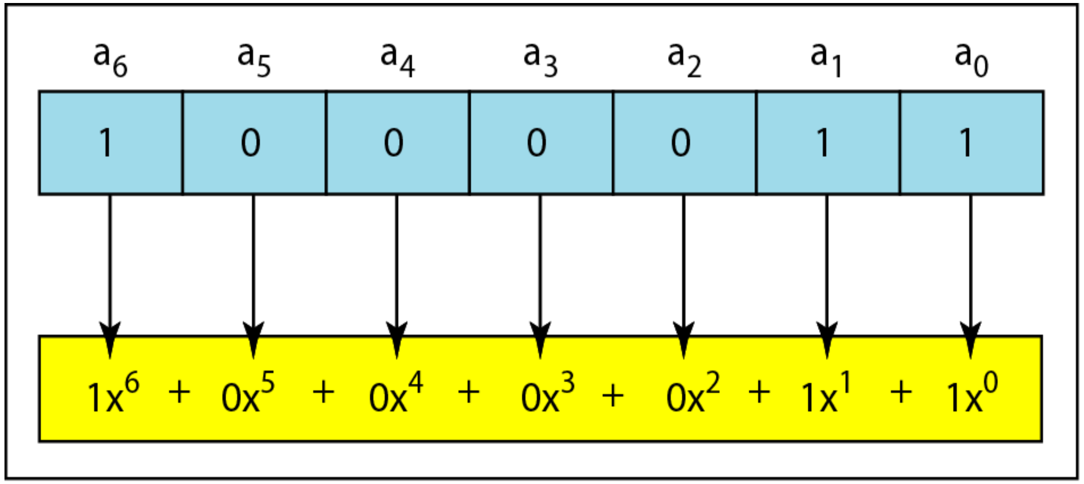
Cyclic Redundancy Check (CRC)

- Predefined shared *divisor* to calculate codeword

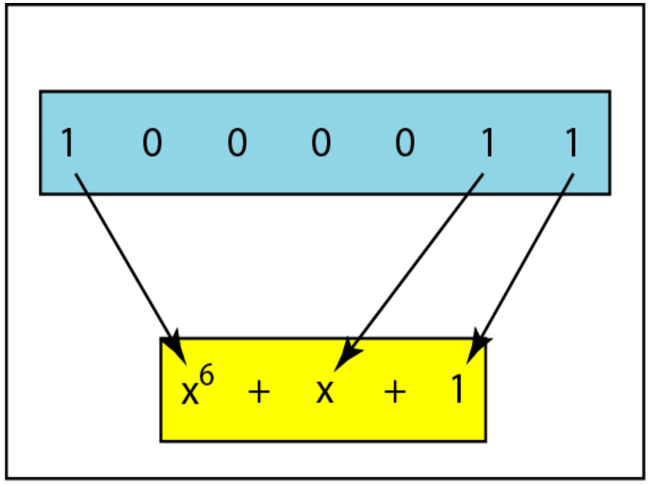


CRC: Polynomial representation

- The dataword of *k* bits is represented by a polynomial, *d(x)*.
- The degree of the polynomial is *k*-1.



a. Binary pattern and polynomial



b. Short form

CRC: The principle

- **Objective**: Send a dataword ***d(x)*** of k bits represented by a polynomial of degree ***k*-1**.
- **Given**: Generator polynomial ***g(x)*** of degree ***m***.
- **Find**: Remainder polynomial ***r(x)*** such that:

  $$c(x) = d(x) \cdot x^m + r(x)$$

  can be divided by g(x) without remainder.
- Codeword ***c(x)*** will then be sent to the receiver.
- ***r(x)*** has degree ***m*-1** or less, and CRC has ***m*** bits.

CRC: How it works

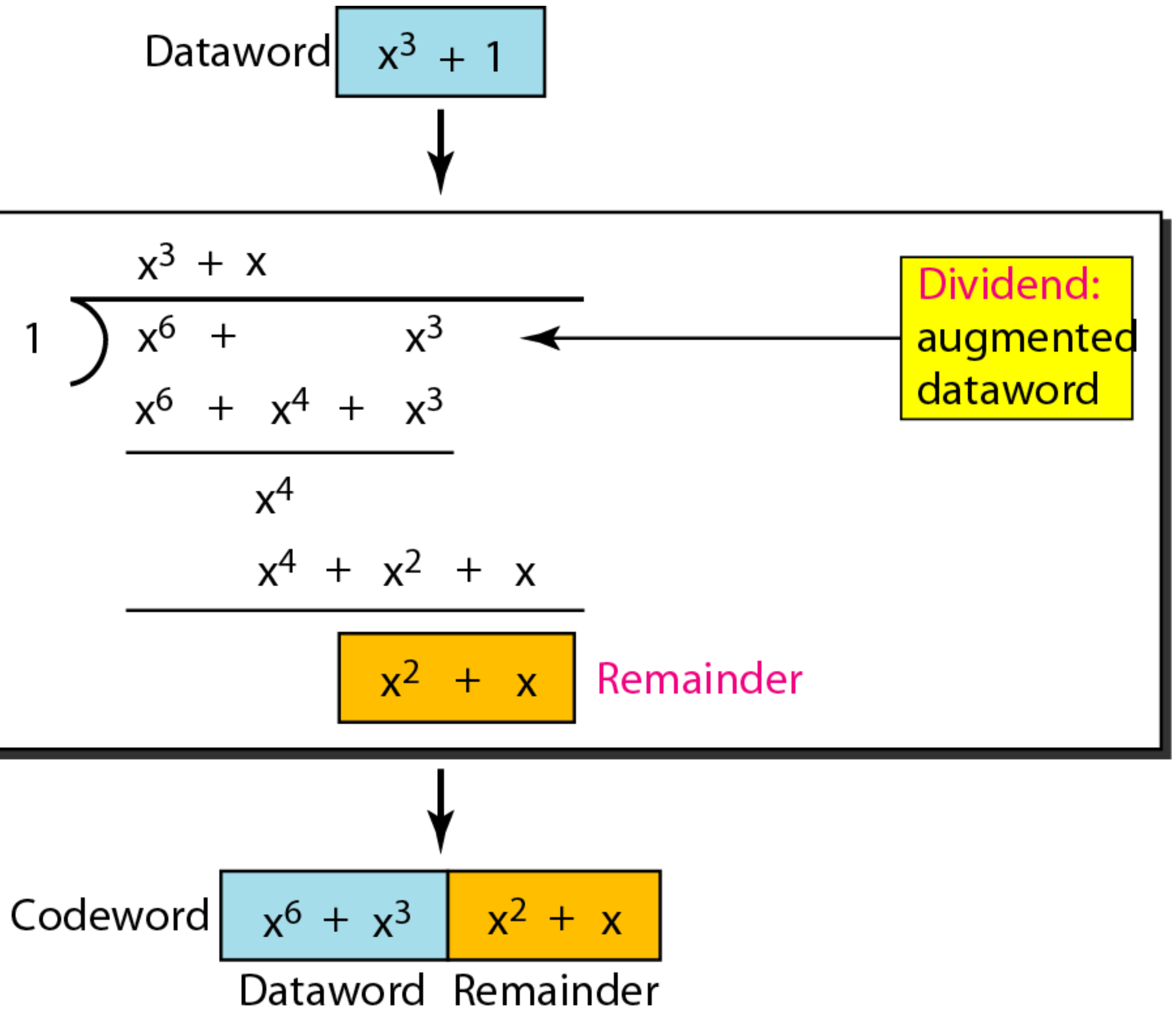
- Sender:
  1. Generate $b(x) = d(x) \cdot x^m$
  2. Divide $b(x)$ by $g(x)$ to find $r(x)$
  3. Send $c(x) = b(x) + r(x)$
- Receiver:
  1. Divide $c'(x) = c(x) + e(x)$ by $g(x)$
  2. Check remainder $r'(x)$ → if 0 data correct, $c(x) = c'(x)$
  3. Remove CRC bits from codeword to get dataword

Example: CRC derivation

- For dataword 1001, derive CRC using generator 1011.

- Data polynomial: $d(x) = x^3+1$
- Generator polynomial: $g(x) = x^3+x+1$
- Dividend: $b(x) = d(x)\cdot x^3 = x^6+x^3$
- Codeword polynomial: $c(x) = d(x)\cdot x^3 + r(x)$
- CRC polynomial: $r(x) = ?$

Example: CRC derivation



Error detection capabilities

- Single errors: $e(x)=x^i$ is not divisable by $g(x)$
- Double errors: $e(x)=x^j+x^i=x^i(x^{j-i}+1)$
  - Use primitive polynomial $p(x)$ with $deg=L$. Then if $n-1<2^L-1$ it is not divisable and all double errors will be detected
- If $x+1 \mid g(x)$ all odd error patterns will be detected
- In practice, set $g(x)=(x+1)\cdot p(x)$

Some standard CRC polynomials

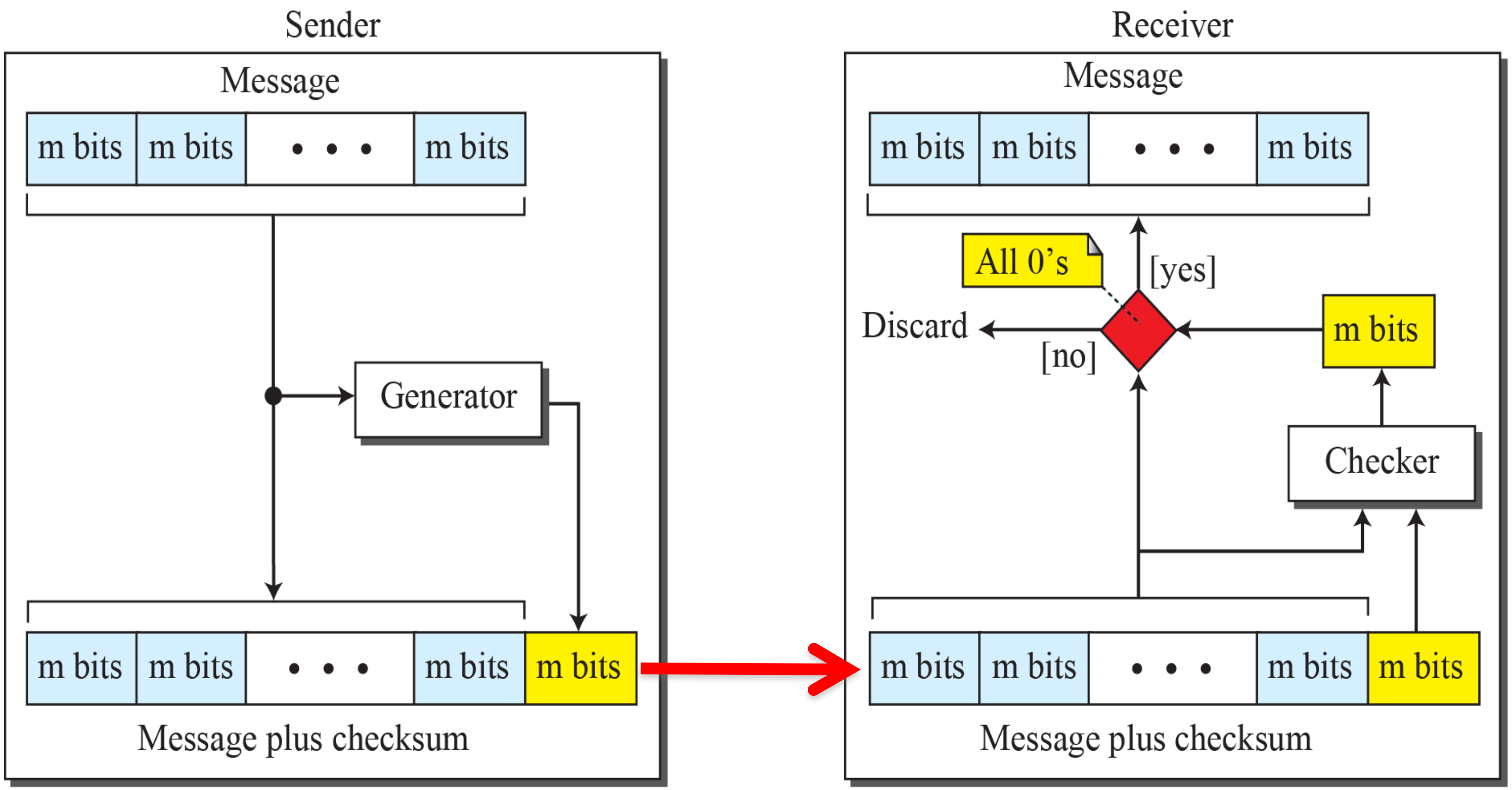
Name	*Polynomial*	*Used in*
CRC-8	$x^8 + x^2 + x + 1$ 100000111	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$ 11000110101	ATM AAL
CRC-16	$x^{16} + x^{12} + x^5 + 1$ 10001000000100001	HDLC
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ 100000100110000010001110110110111	LANs

Checksum

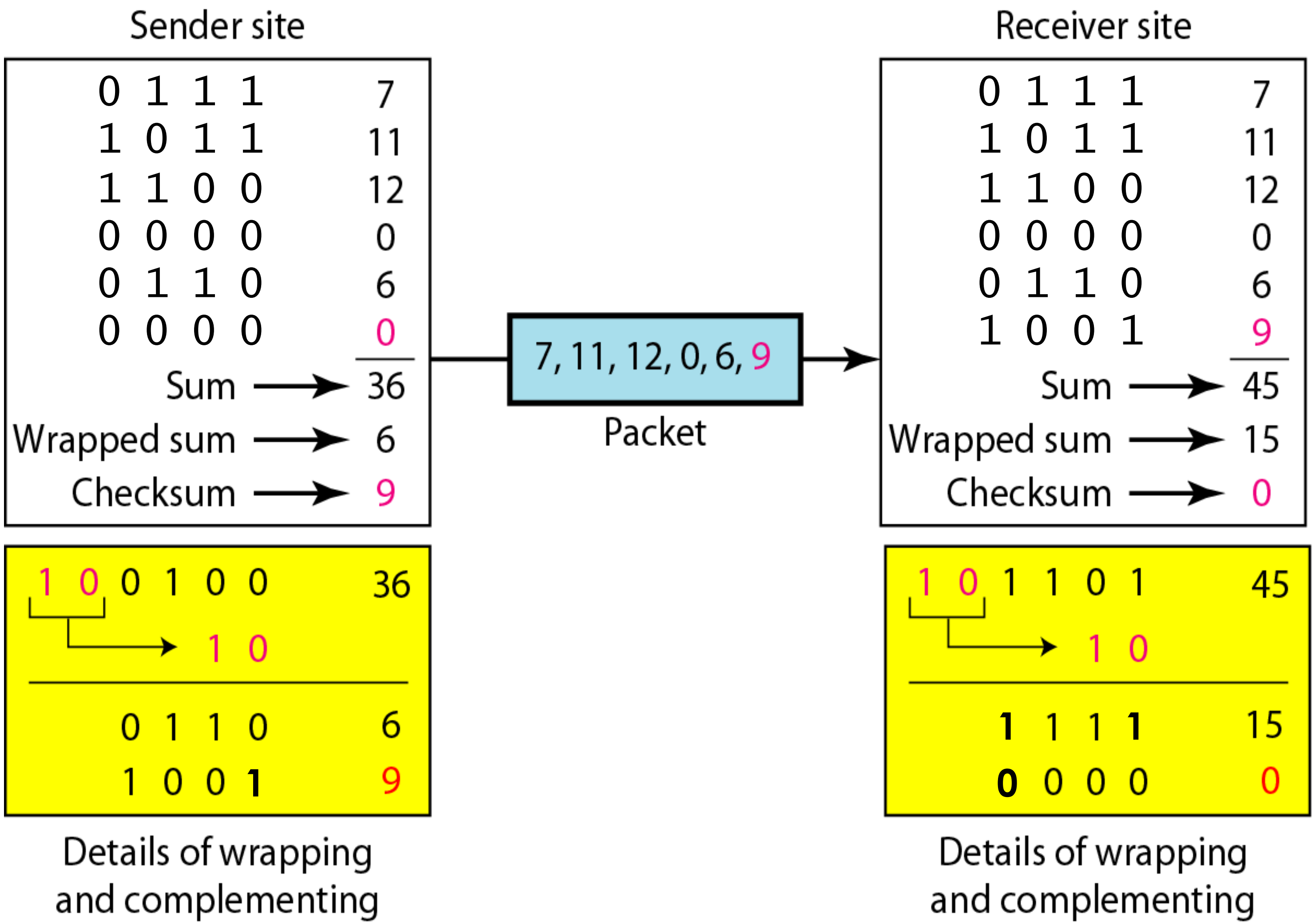
- The checksum is used in the Internet by several protocols although not at the data link layer.

- The main principle is to divide the data into segments of n bits. Then add the segments and use the sum as redundant bits.

Checksum process



Example: Checksum



Error Correction

Two alternative ideas

- Forward Error Correction (FEC)
  - Send each bit multiple times
  - Decode according to majority decision
- Retransmission
  - Resend the entire frame
- In most communication systems, both error detection and error correction occur.

See you in 15’ :)

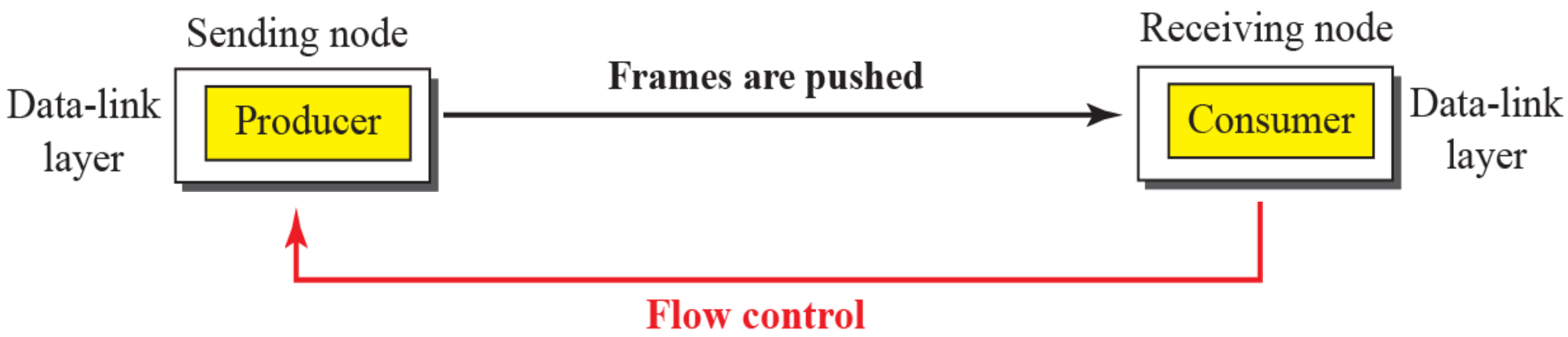
- After the break
  - Data link control protocols
  - Point-to-point protocol

Error and flow control

- The basic principle in error and flow control is that the receiver **acknowledges** all correctly received packets.

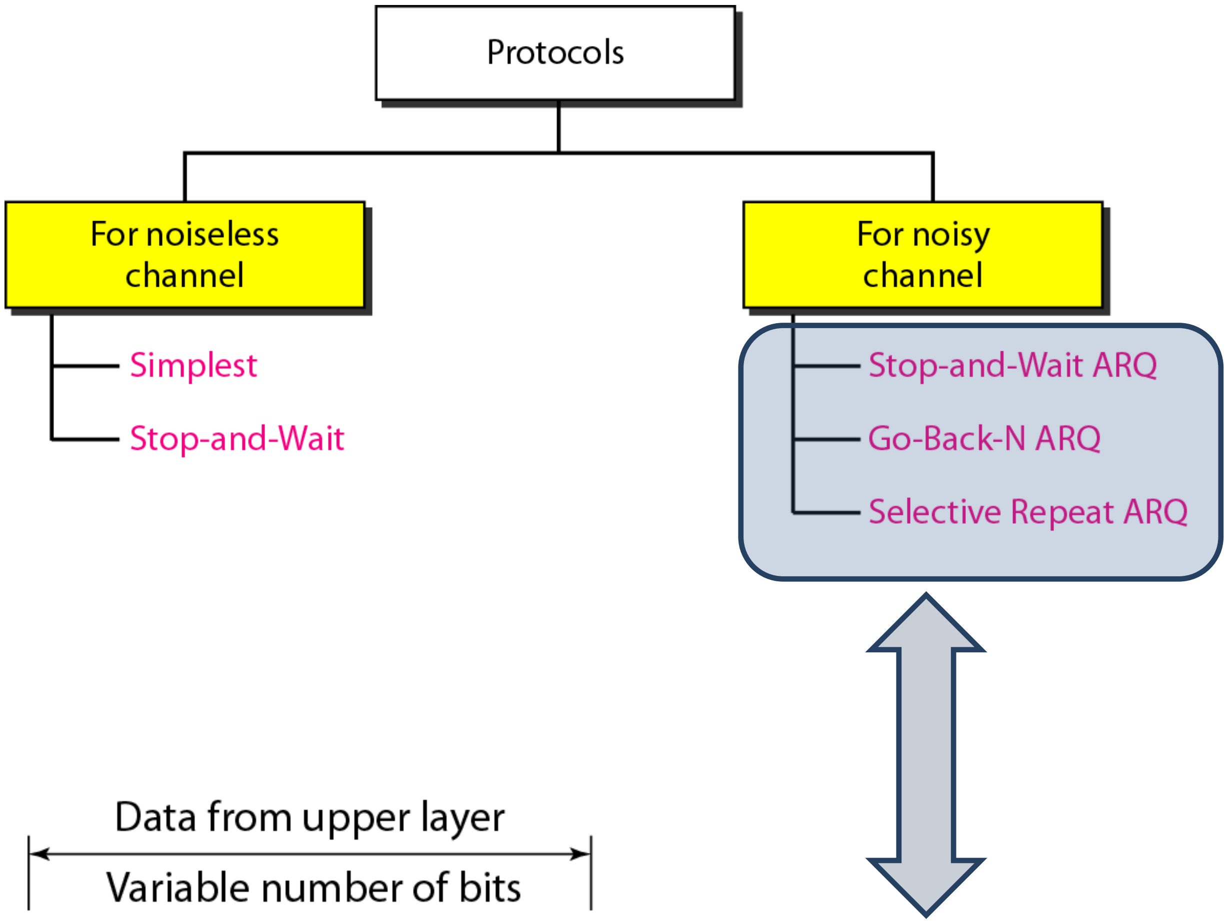
The need for flow control

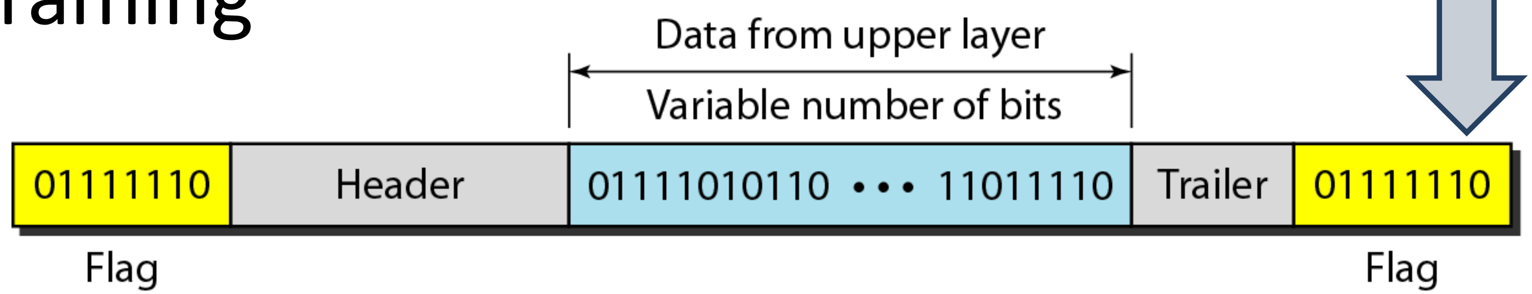
- The receiver must be able to handle all recieved frames. If the transmission rate is too high, the receiver may become overloaded and drop frames due to full buffers.



Data link control protocols

- Flow control
  - Send data
  - Wait for ACK
- Error control
  - Detect error
  - Retransmit
- Framing

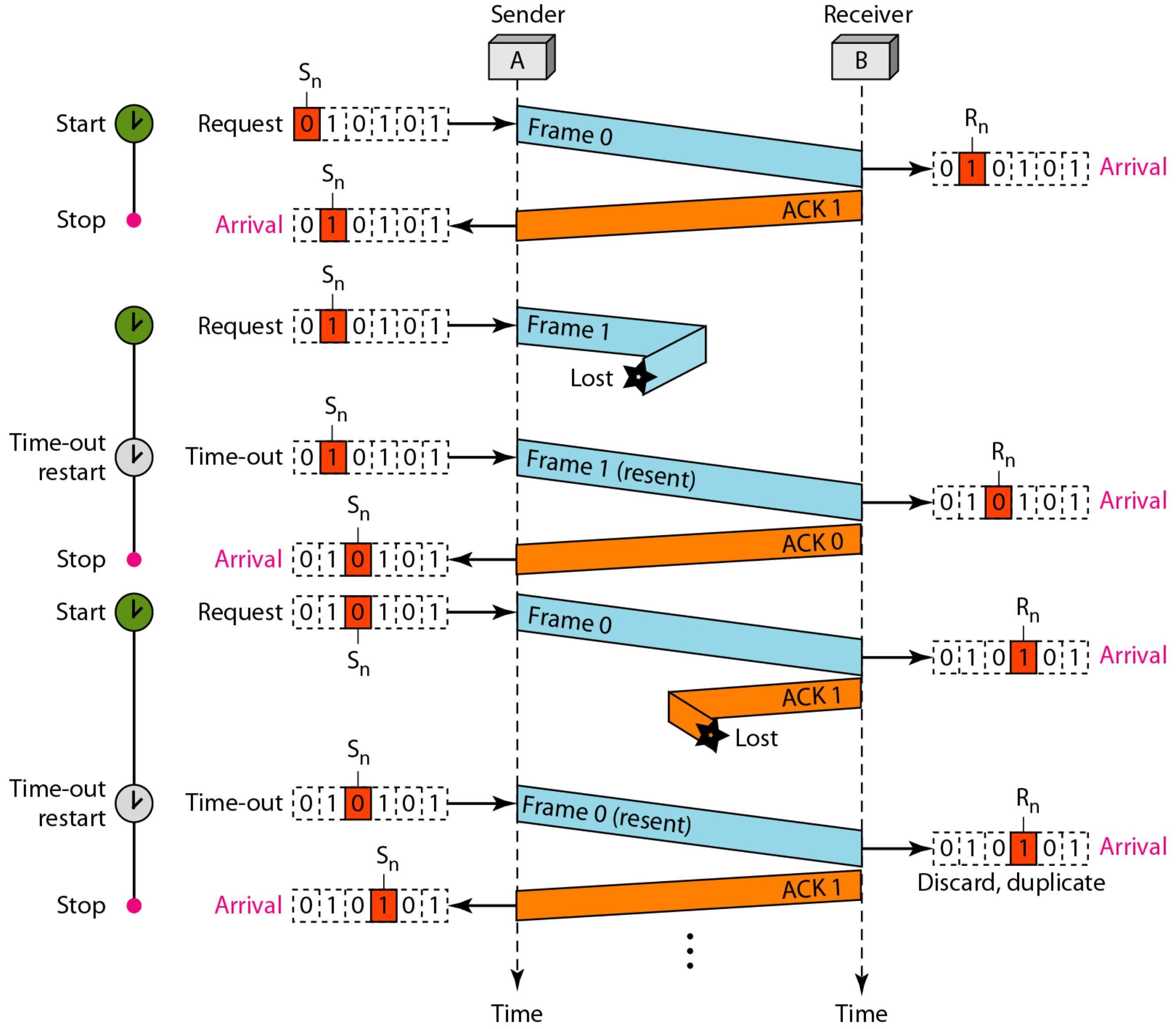




Stop-and-wait ARQ

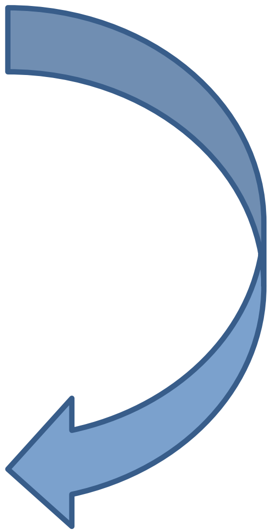
- Send and wait
  - Keep time
  - Wait for ACK
  - Retransmit
- Automatic repeat request
  - Frames (SEQ++)
  - Acknowledgements (SEQ+1)
  - Mismatch = problem!

Stop-and-wait ARQ flow diagram



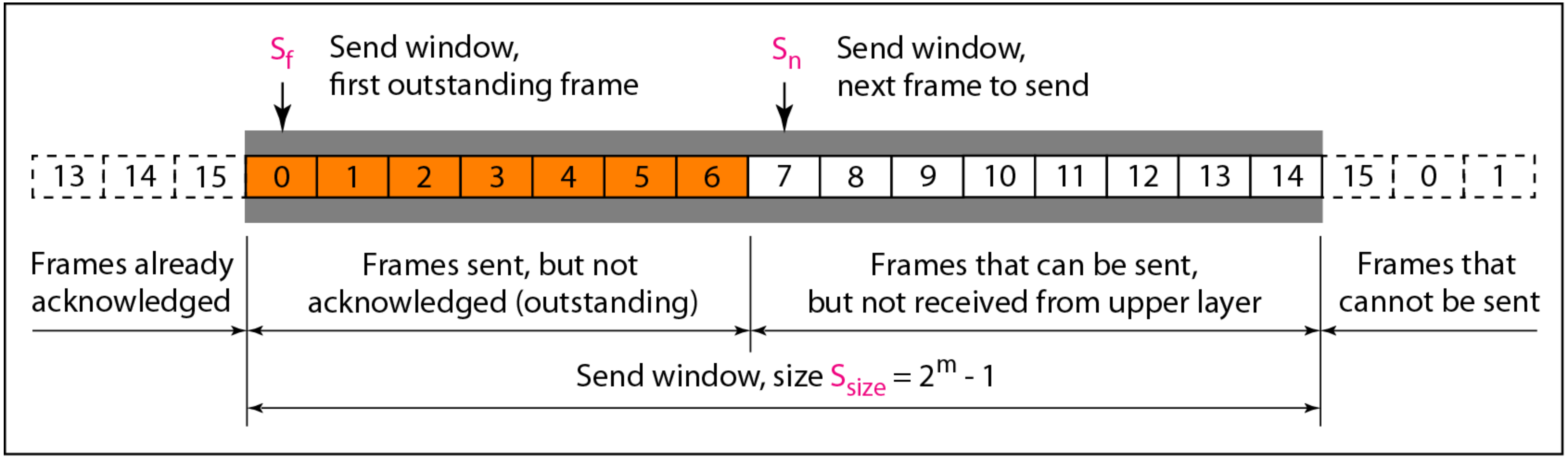
Stop-and-wait ARQ inefficiency

- Too much waiting
- Solution
  - Keep the pipe full
  - But not too full

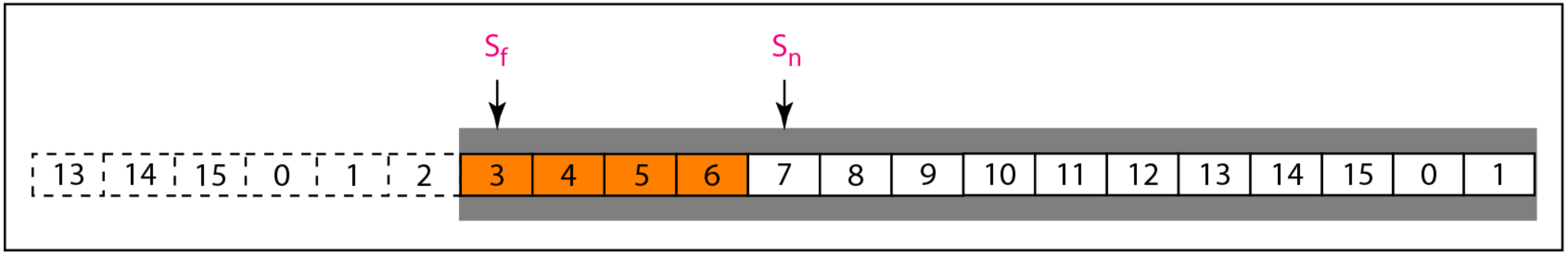


- Sliding window
  - Size matters
  - Window size < 2^m

Sliding window

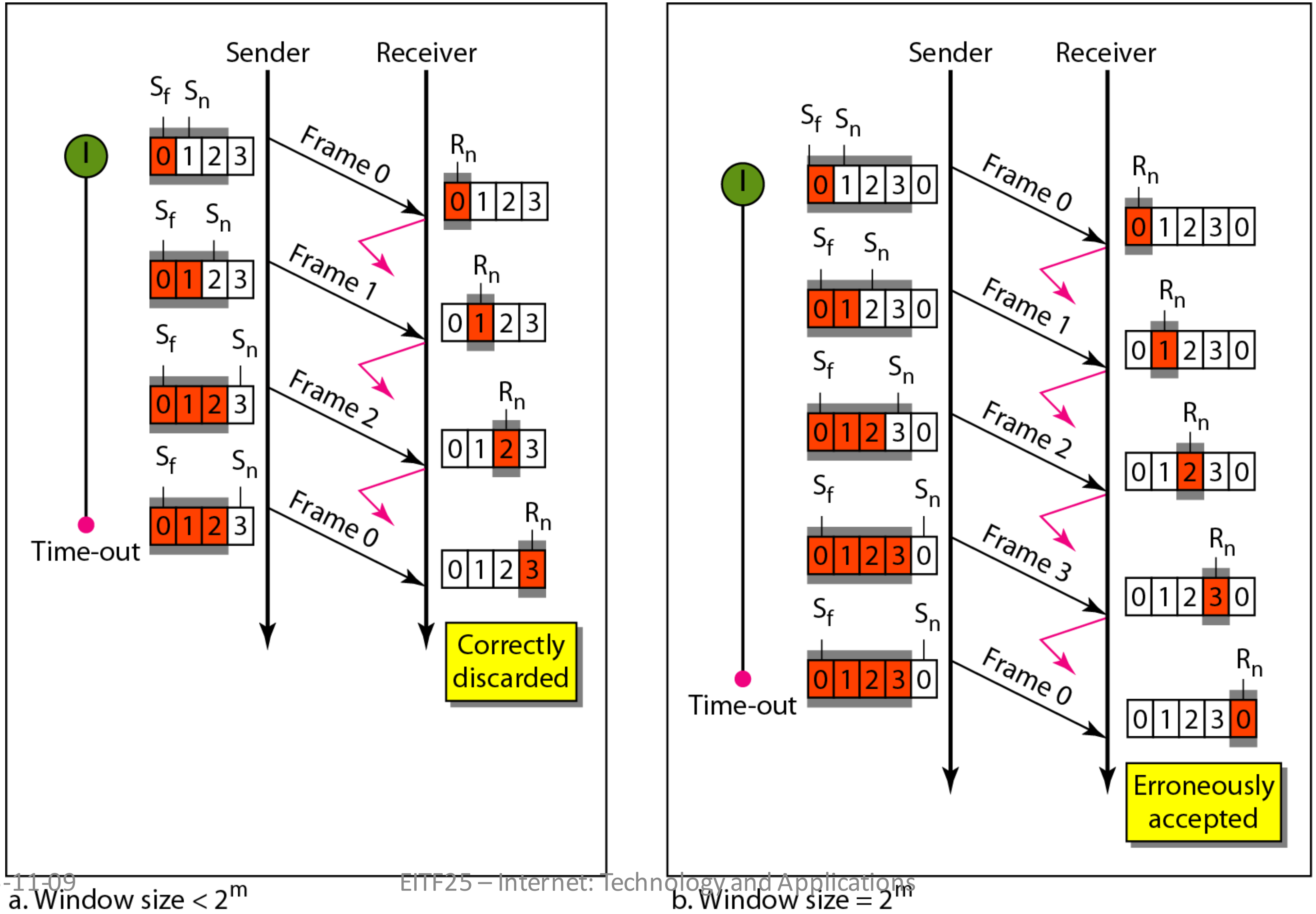


a. Send window before sliding

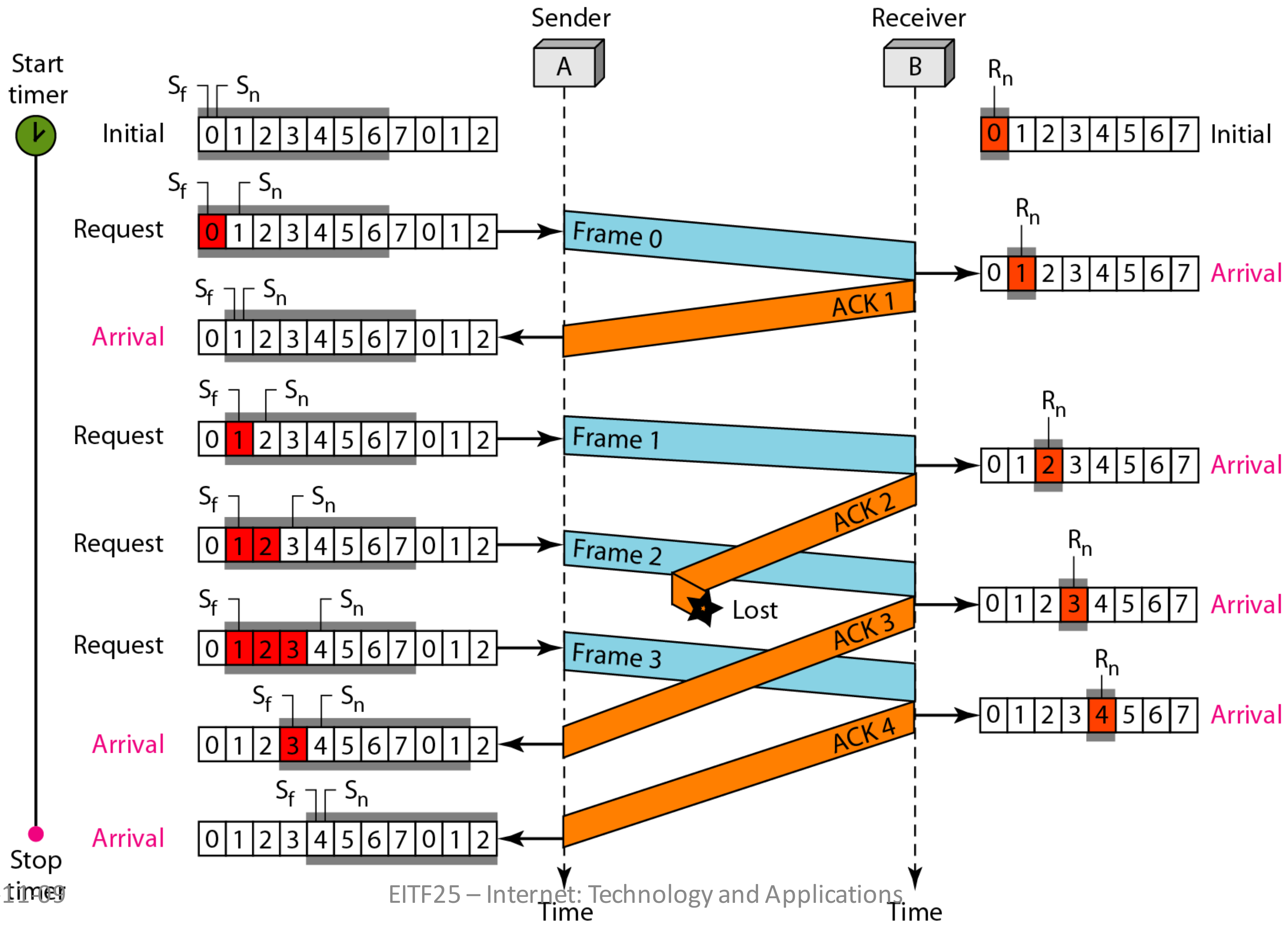


b. Send window after sliding

Go-back-N ARQ window size



Go-back-N ARQ flow diagram

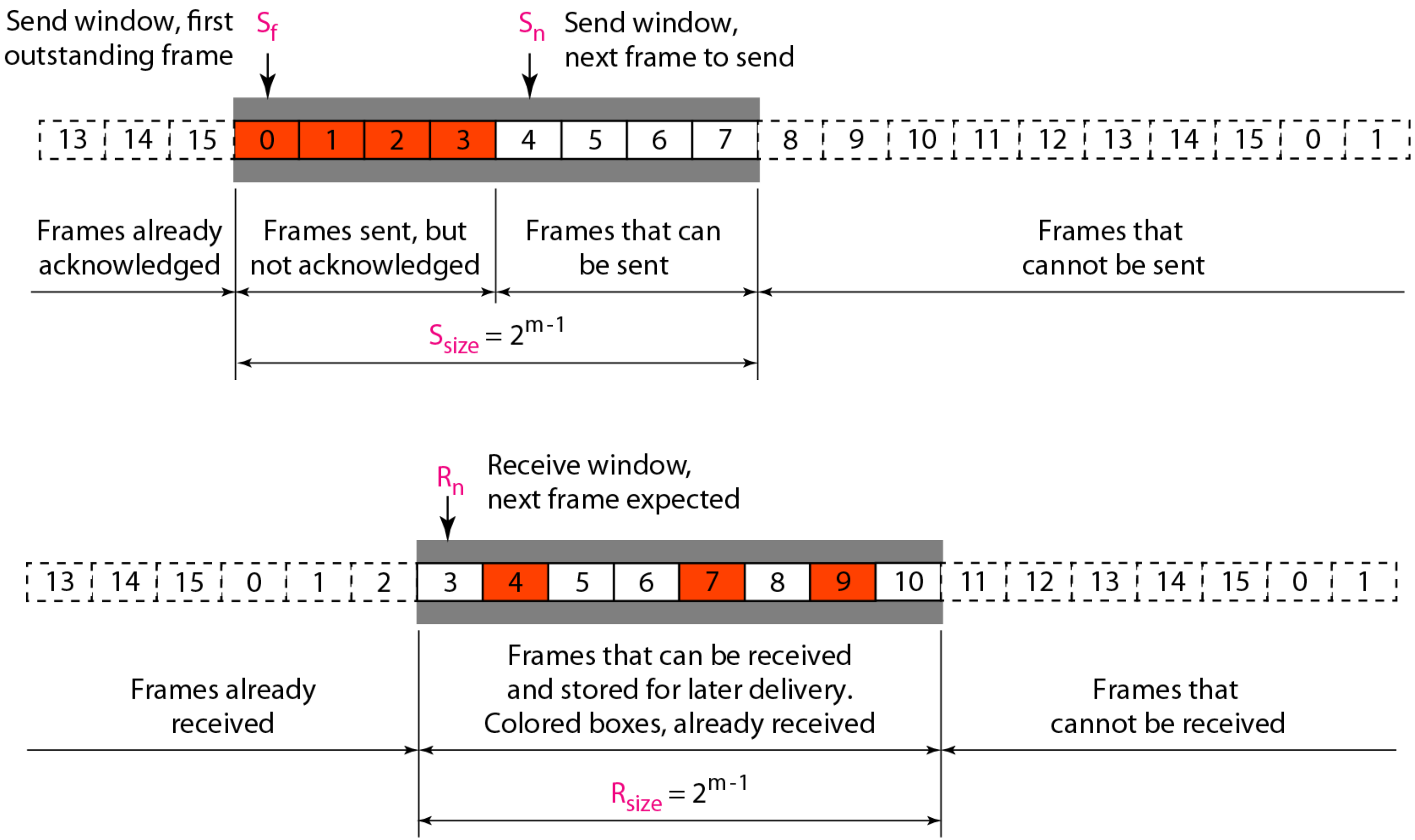


Selective repeat ARQ

- Why?
  - Too many retransmissions
- What if?
  - Just send lost frames
- Higher efficiency
  - Higher receiver complexity

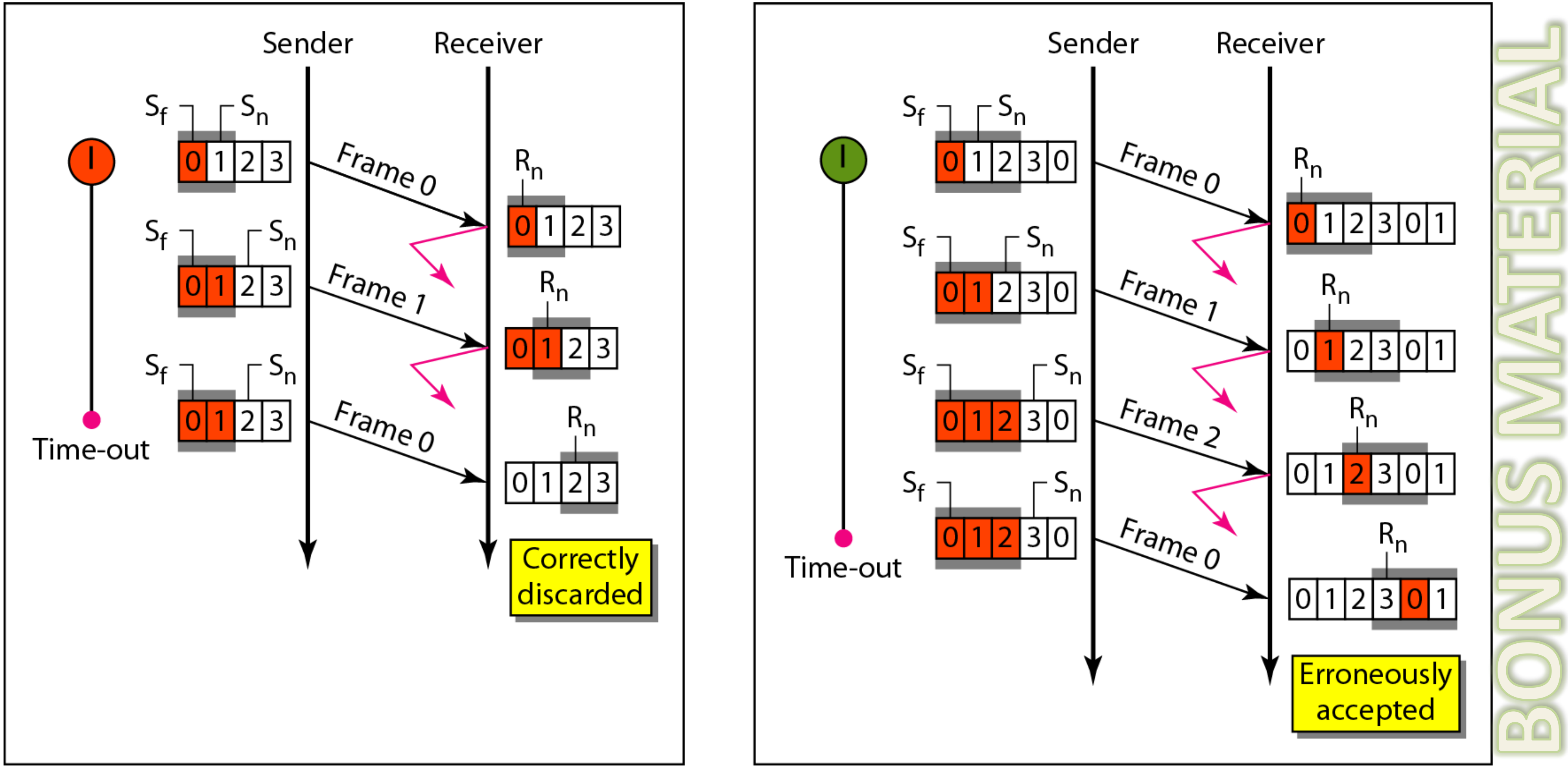
BONUS MATERIAL

Windows again



BONUS MATERIAL

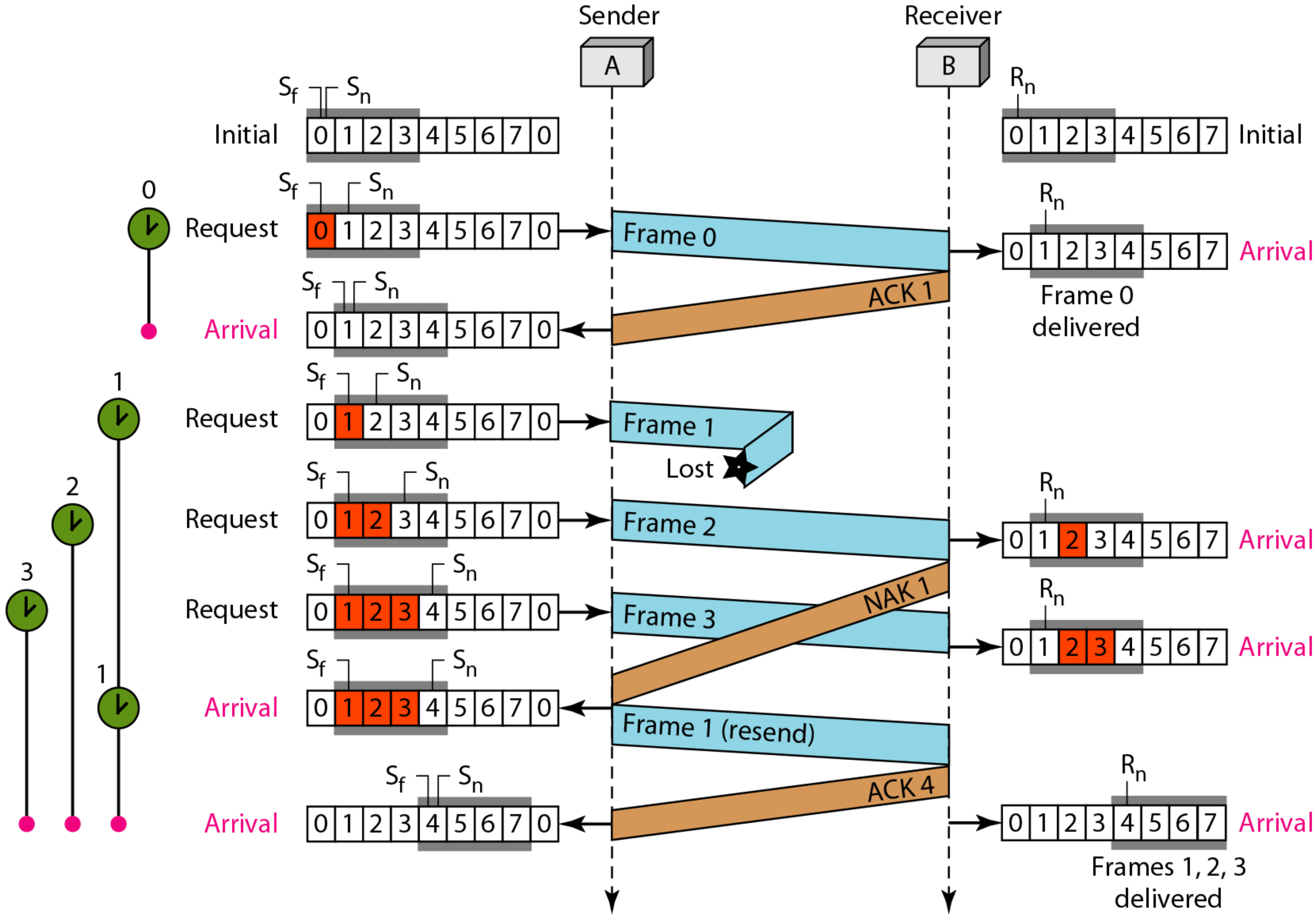
Selective repeat ARQ window size



a. Window size = 2^{m-1}

b. Window size > 2^{m-1}

Selective repeat ARQ flow diagram



Note on ”Selective Repeat ARQ”

Stallings, pp. 248-249

- **ACK_n = RR_n**
  - Acknowledges frame *n-1* and all earlier frames. Receiver says it is expecting frame *n*.
- **NAK_x = $SREJ_n$**
  - Negative acknowledgment for missing frame *x*. Receiver says it has not received frame *x*.

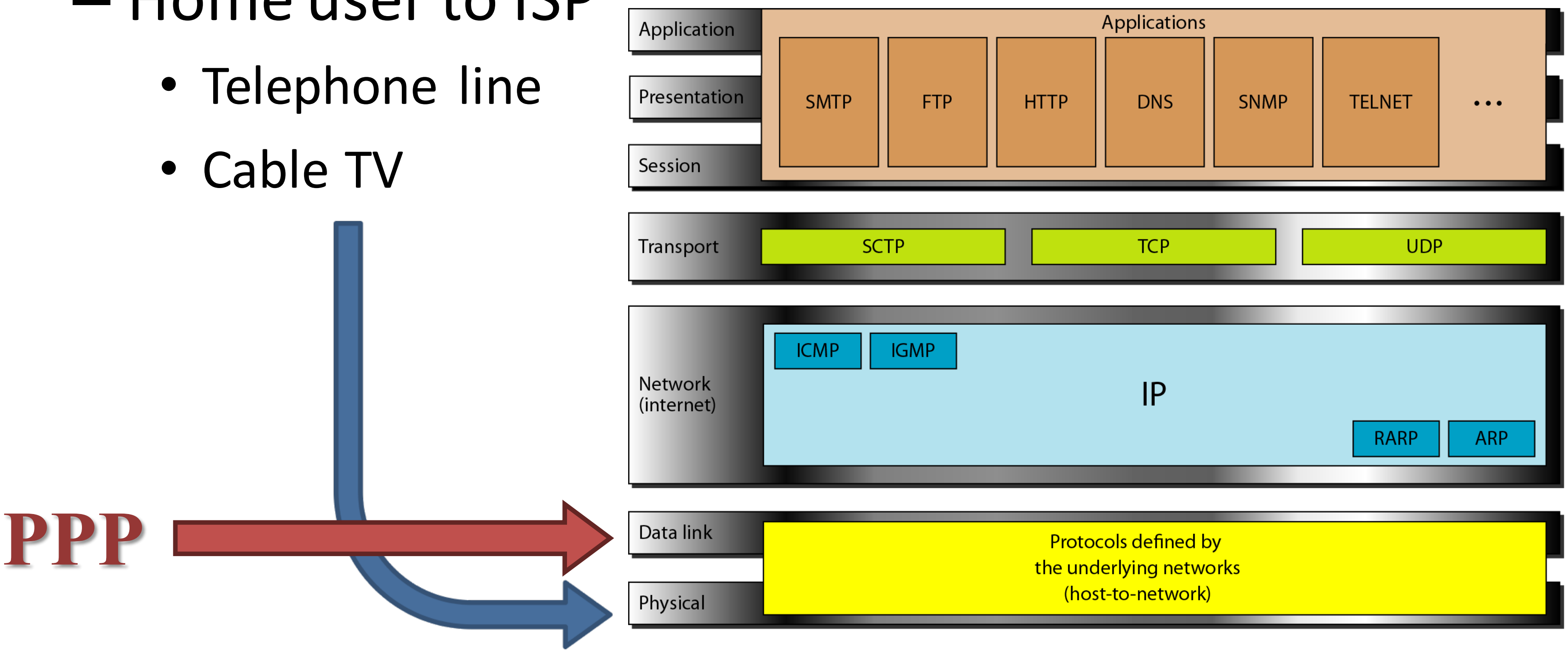
⇩ **= OUR LECTURE SLIDES!**

Forouzan, pp. 720-726

- **ACK_n**
  - Acknowledges frame *n* and frame *n* only. Receiver says it has received frame *n*.
- **NAK_x**
  - There is no such thing as negative acknowledgment. Receiver does not request a missing frame *x* as long as the frames it receives fall inside the receive window.

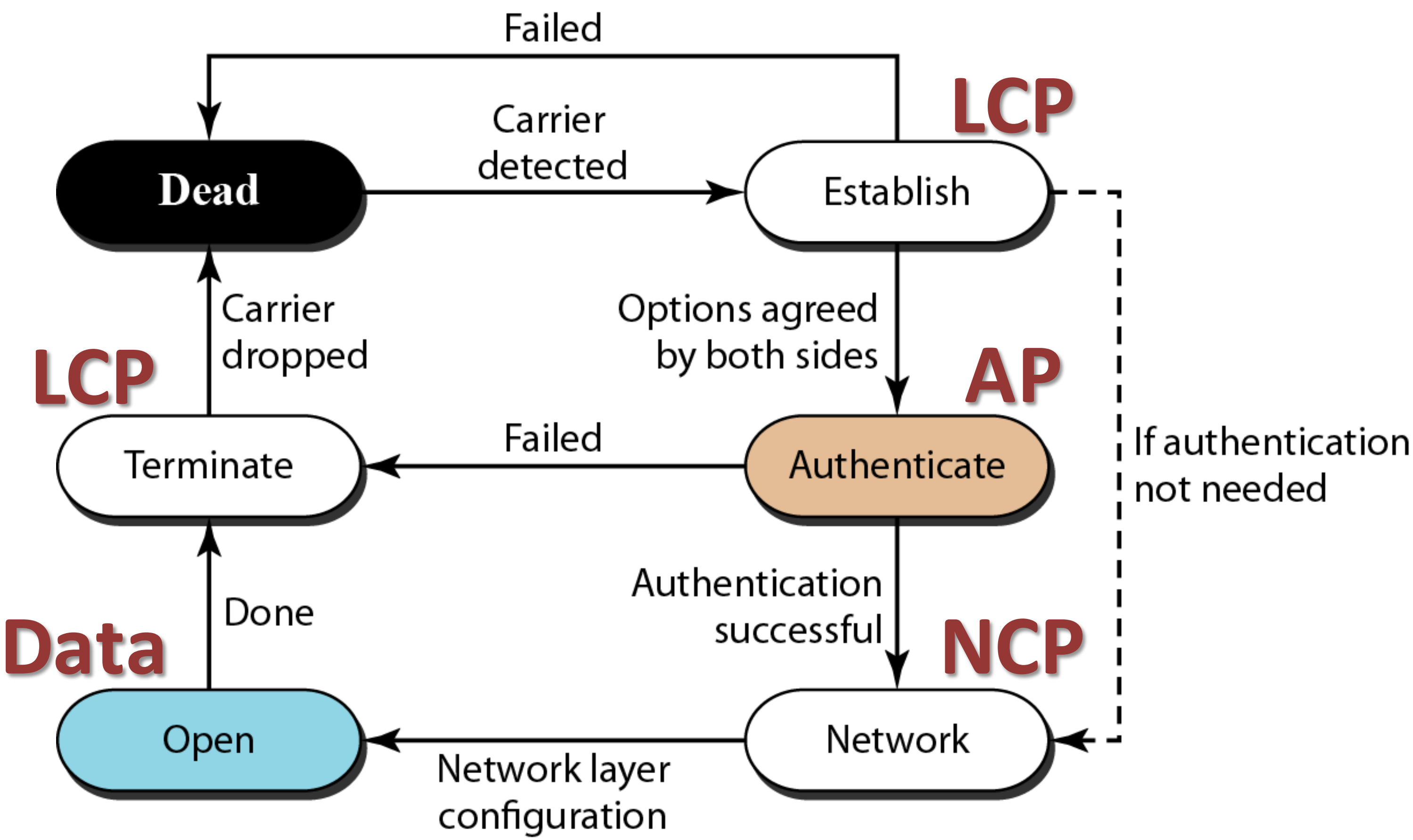
Point-to-point protocol (PPP)

- Direct connection between two nodes
  - Internet access
  - Home user to ISP
    - Telephone line
    - Cable TV



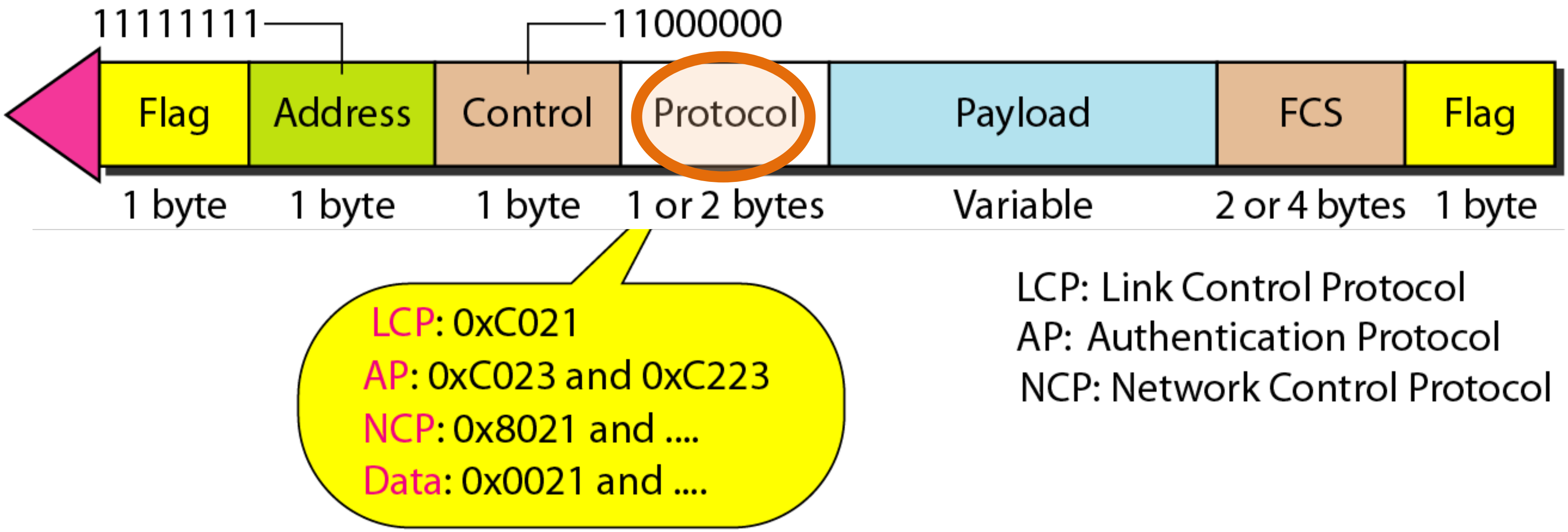
State transitions in PPP

- We need more protocols



PPP frame format

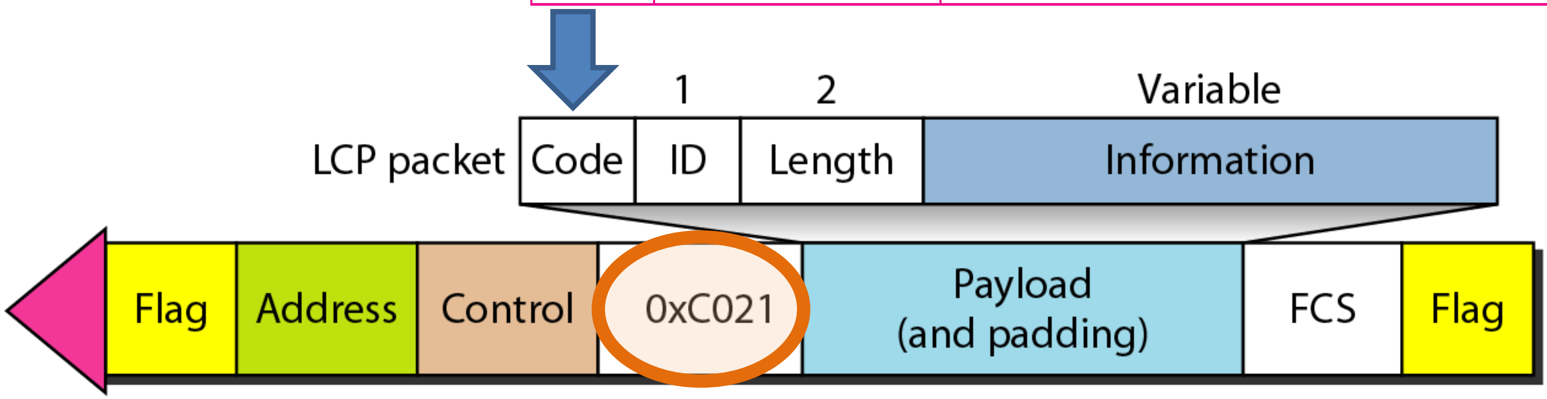
- Support for several (sub)protocols
- Address & control not used
- Maximum payload 1500 bytes



Link control protocol (LCP)

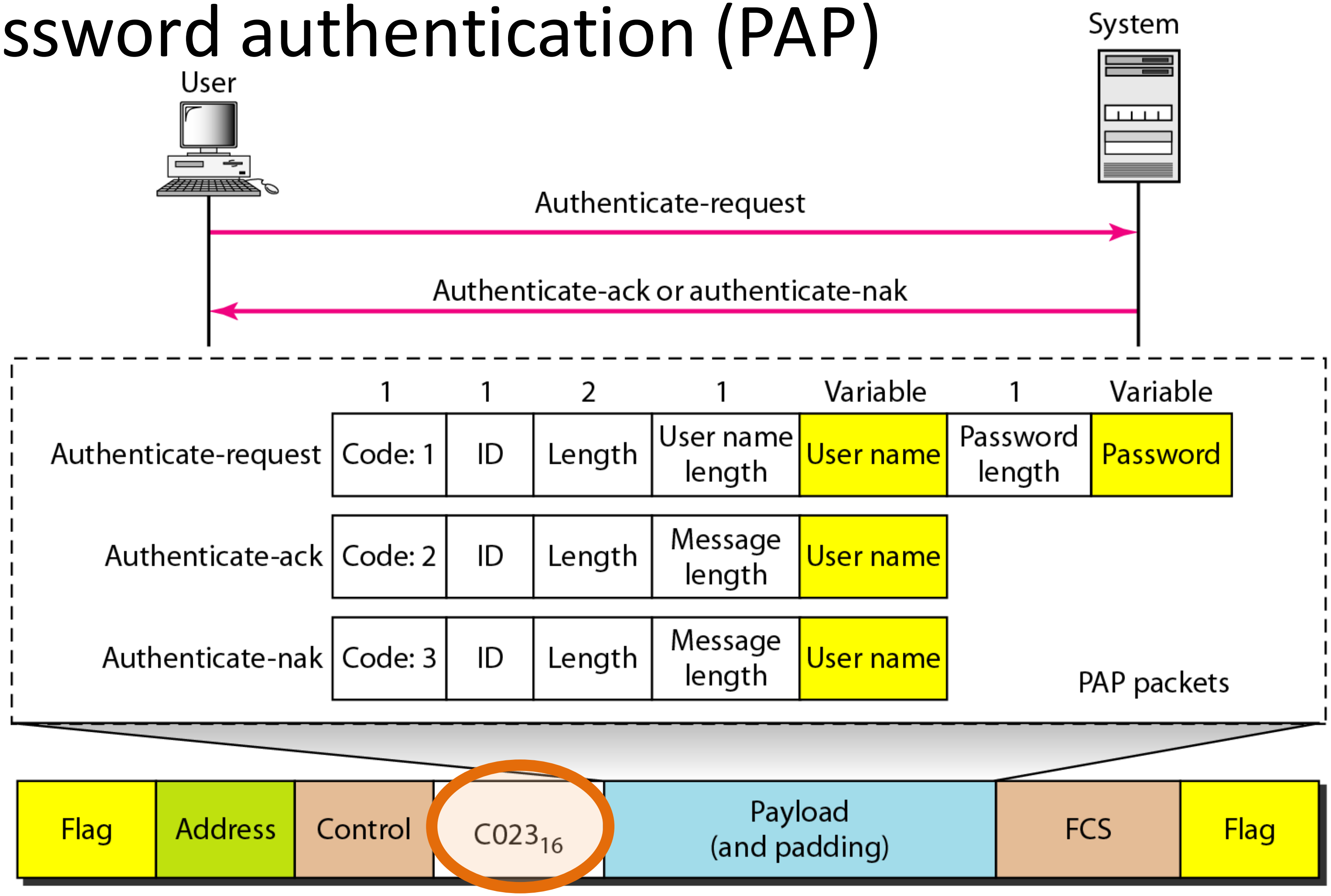
- Establish
- Configure
- Terminate

Code	*Packet Type*	*Description*
0x01	Configure-request	Contains the list of proposed options and their values
0x02	Configure-ack	Accepts all options proposed
0x03	Configure-nak	Announces that some options are not acceptable
0x04	Configure-reject	Announces that some options are not recognized
0x05	Terminate-request	Request to shut down the line
0x06	Terminate-ack	Accept the shutdown request
0x07	Code-reject	Announces an unknown code
0x08	Protocol-reject	Announces an unknown protocol
0x09	Echo-request	A type of hello message to check if the other end is alive
0x0A	Echo-reply	The response to the echo-request message
0x0B	Discard-request	A request to discard the packet



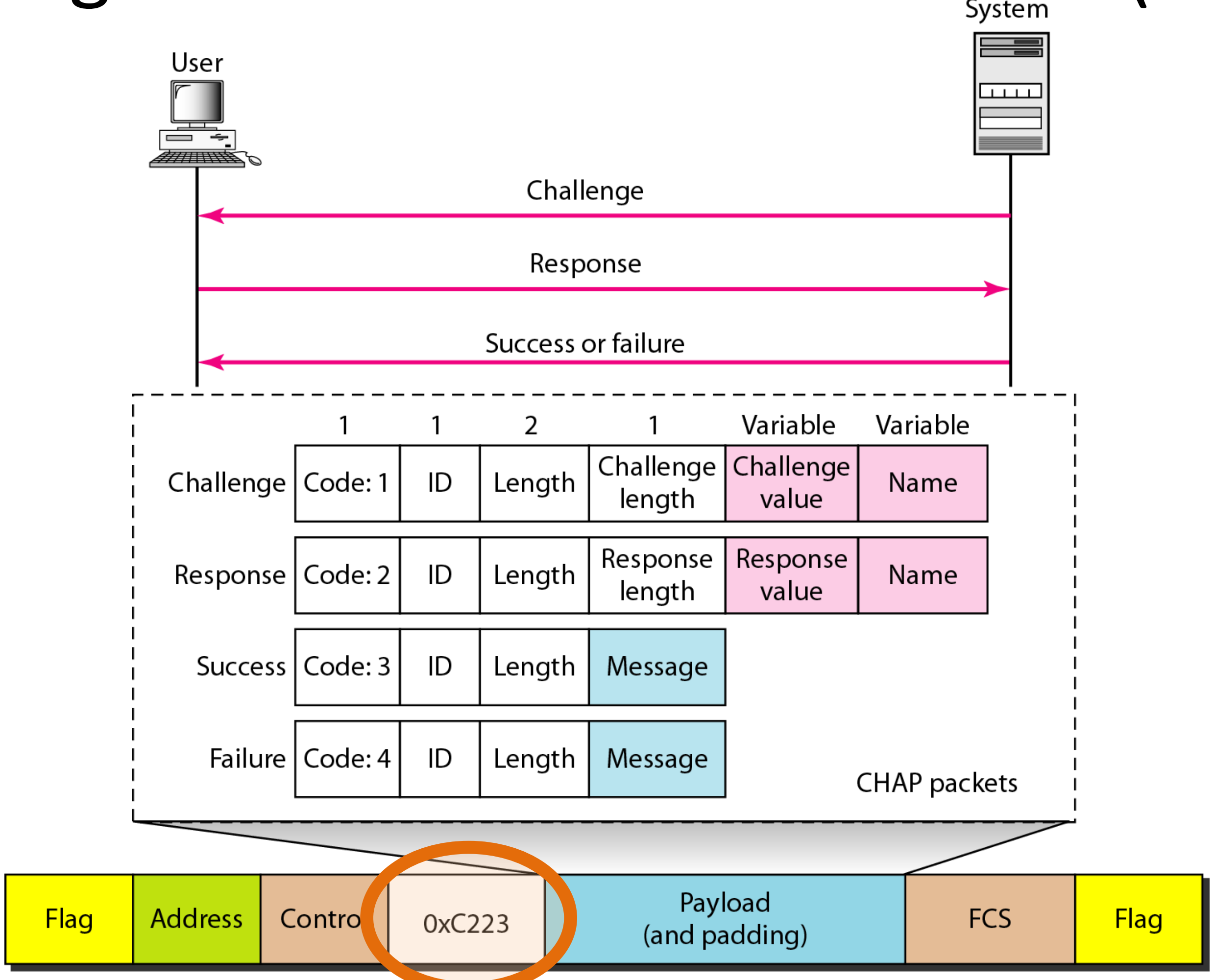
Authentication protocols (AP)

- Password authentication (PAP)



Authentication protocols (AP)

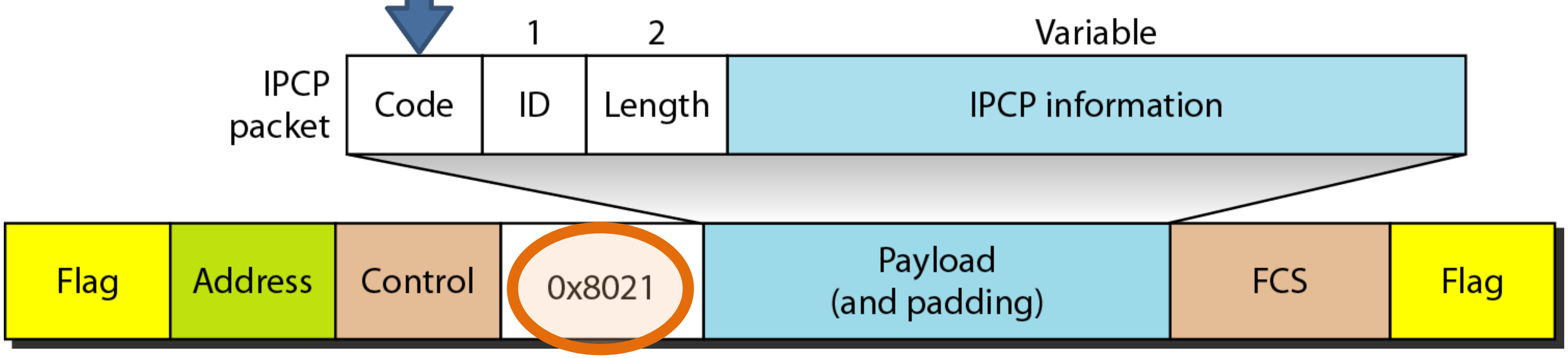
- Challenge handshake authentication (CHAP)



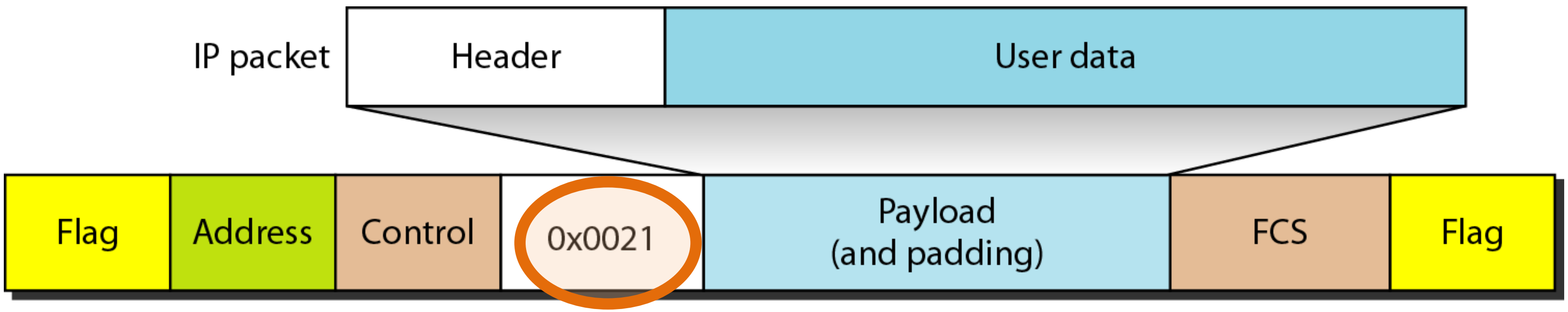
Network control protocols (NCP)

- Preparations for the network layer
  - IPCP for Internet

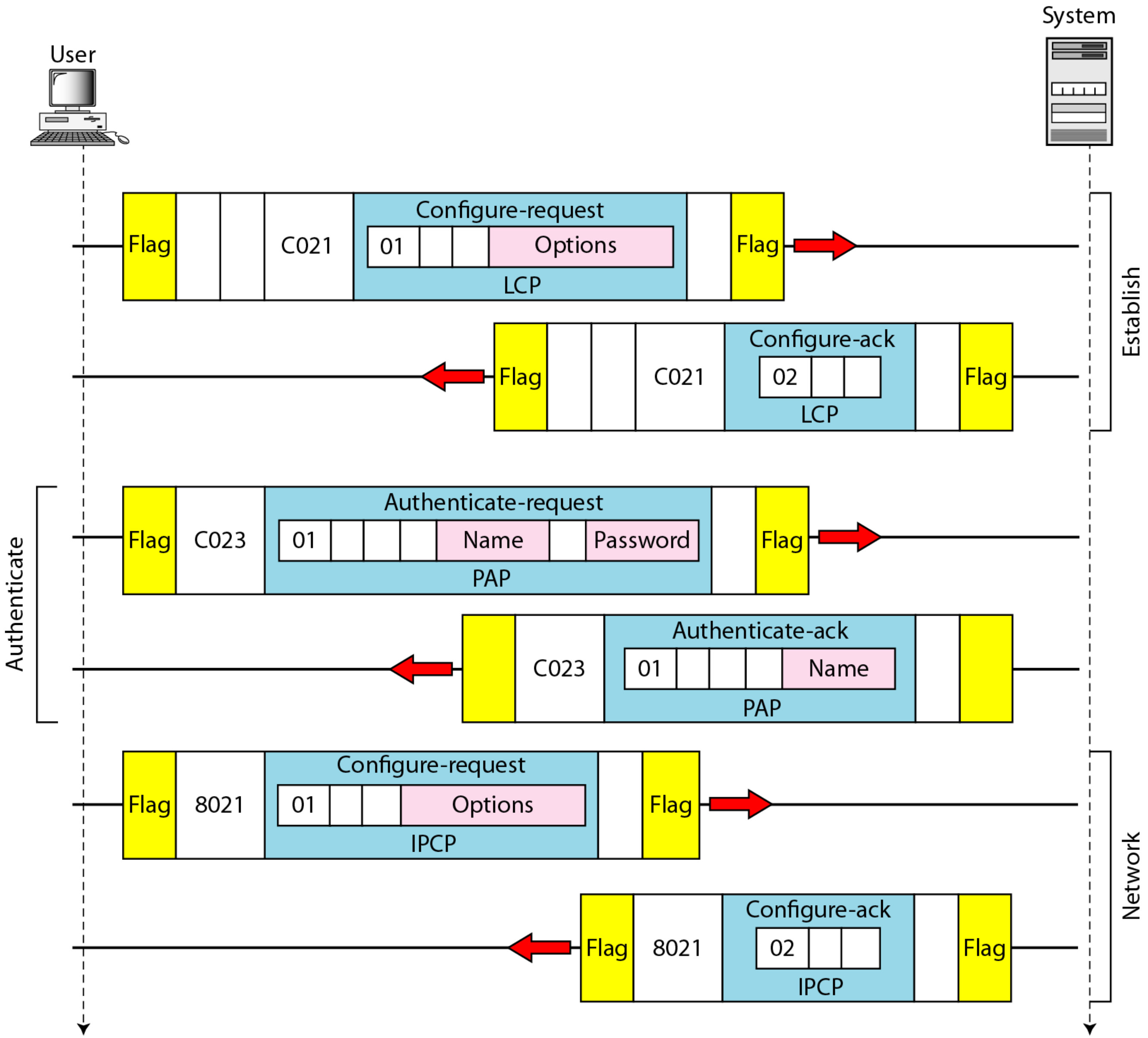
Code	*IPCP Packet*
0x01	Configure-request
0x02	Configure-ack
0x03	Configure-nak
0x04	Configure-reject
0x05	Terminate-request
0x06	Terminate-ack
0x07	Code-reject



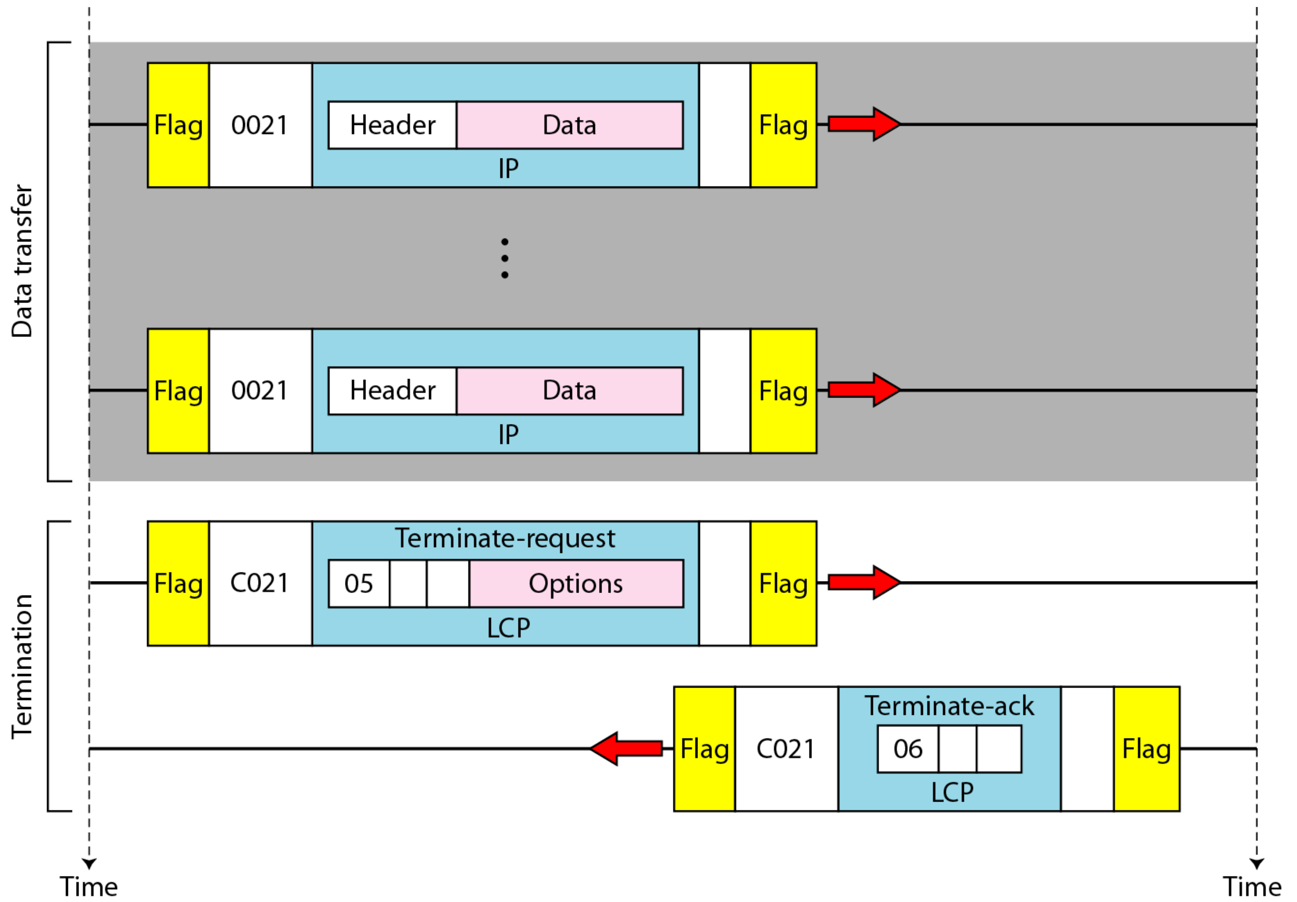
IP datagram encapsulation in PPP



PPP session example



PPP session example (cont.)



Summary: Data Link Layer (1)

Logical Link Control Sublayer

- Frames
- Error control
  - Detection and correction
- Flow control
  - Stop and wait, go back N, selective repeat
- Point-to-point protocol