Redes de Computadores

The Data Link Layer

Manuel P. Ricardo

Faculdade de Engenharia da Universidade do Porto

» *What are the main functions of the Data Link layer?*
» *What is the difference between a packet and a frame?*
» ***What techniques can be used to frame data? Why is stuffing important?***
» ***What is the relationship between BER and FER?***
» ***What techniques can be used to detect errors?***
» ***How does the Internet checksum work?***
» ***How does a CRC work and what are its error detection capabilities?***
» *What is the purpose of ARQ and what are the common ARQ mechanisms?*
» ***Why are sequence numbers required in the STOP&Wait ARQ? What is its efficiency?***
» ***What is a sliding window?***
» *What are the main differences between the Go Back N ARQ and STOP&WAIT ARQ? What is the efficiency of the former?*
» ***What are the differences between the Selective Reject ARQ and the Go Back N ARQ mechanisms? How does this efficiency evolve with FER?***
» ***What framing, stuffing, error detection and ARQ mechanisms are used in Ethernet, PPP and WLAN?***
» ***What are the differences between End-to-End ARQ and Link-by-Link ARQ?***
» ***Where are the ARQ mechanisms in the TCP/IP reference model ?***

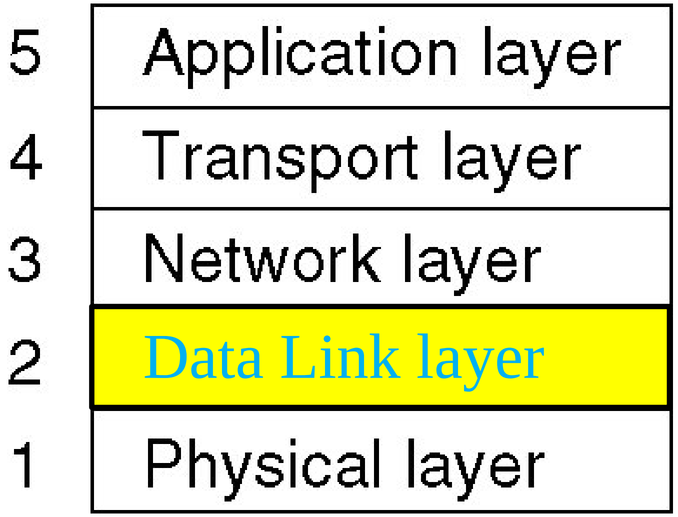
Data Link layer functions and services

Data Link Layer – Functions and Services

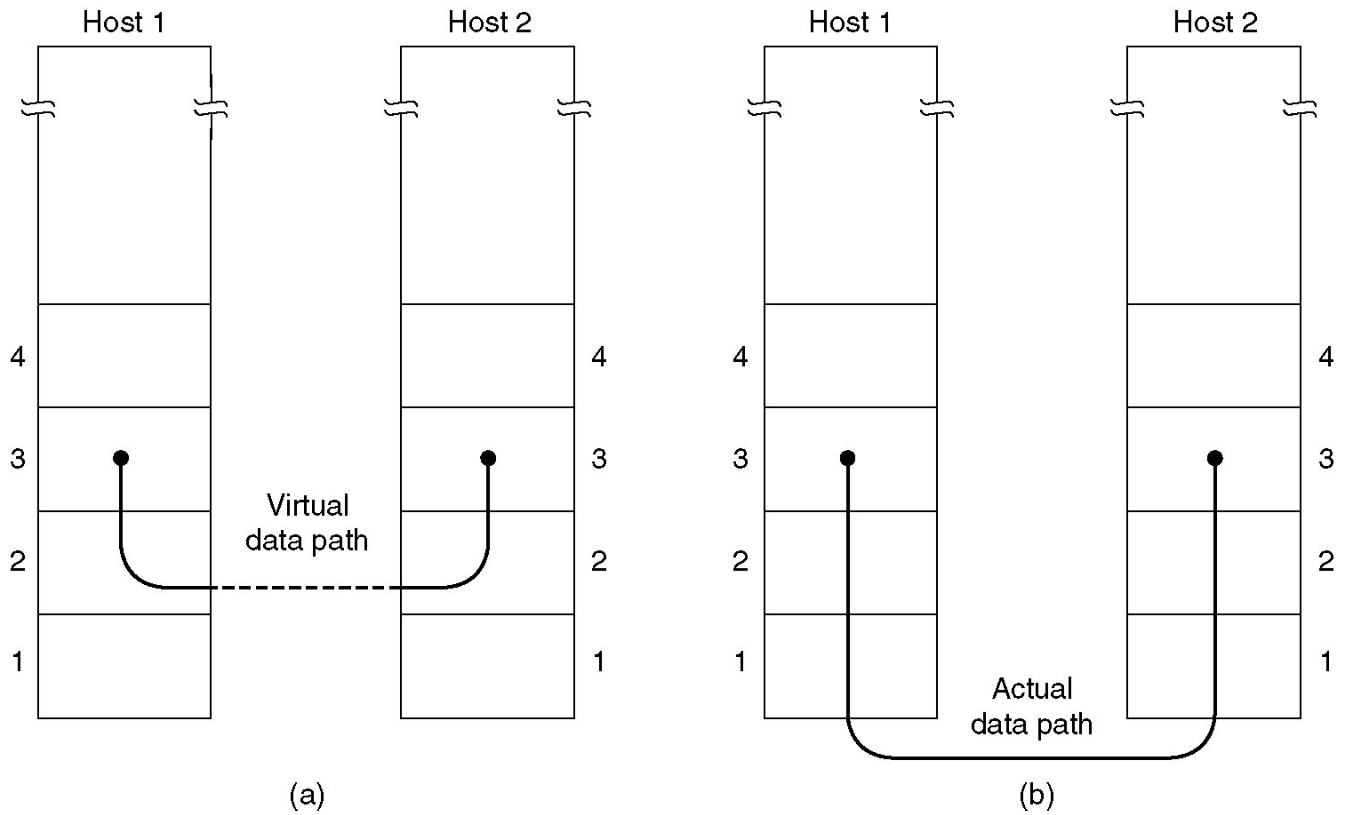
- Main functions
  - » Provide service interface to the network layer
  - » Eliminate/reduce transmission errors
  - » Regulate data flow

    Slow receivers not swamped by fast senders

- Services provided
  - » Unacknowledged connectionless service
  - » Acknowledged connectionless service
  - » Acknowledged connection-oriented service

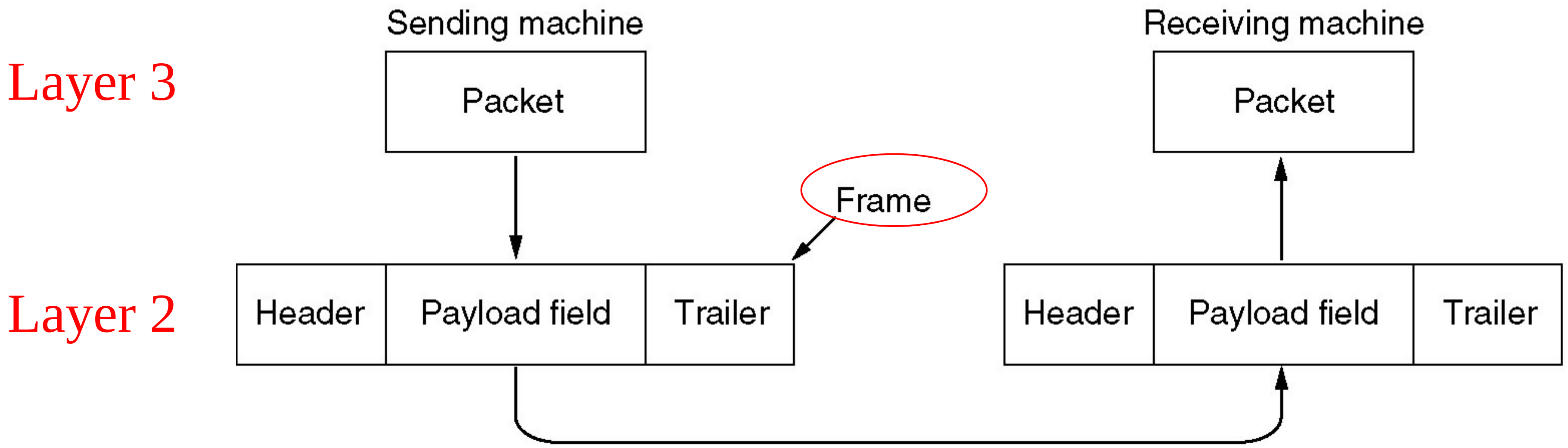


Services Provided to Network Layer

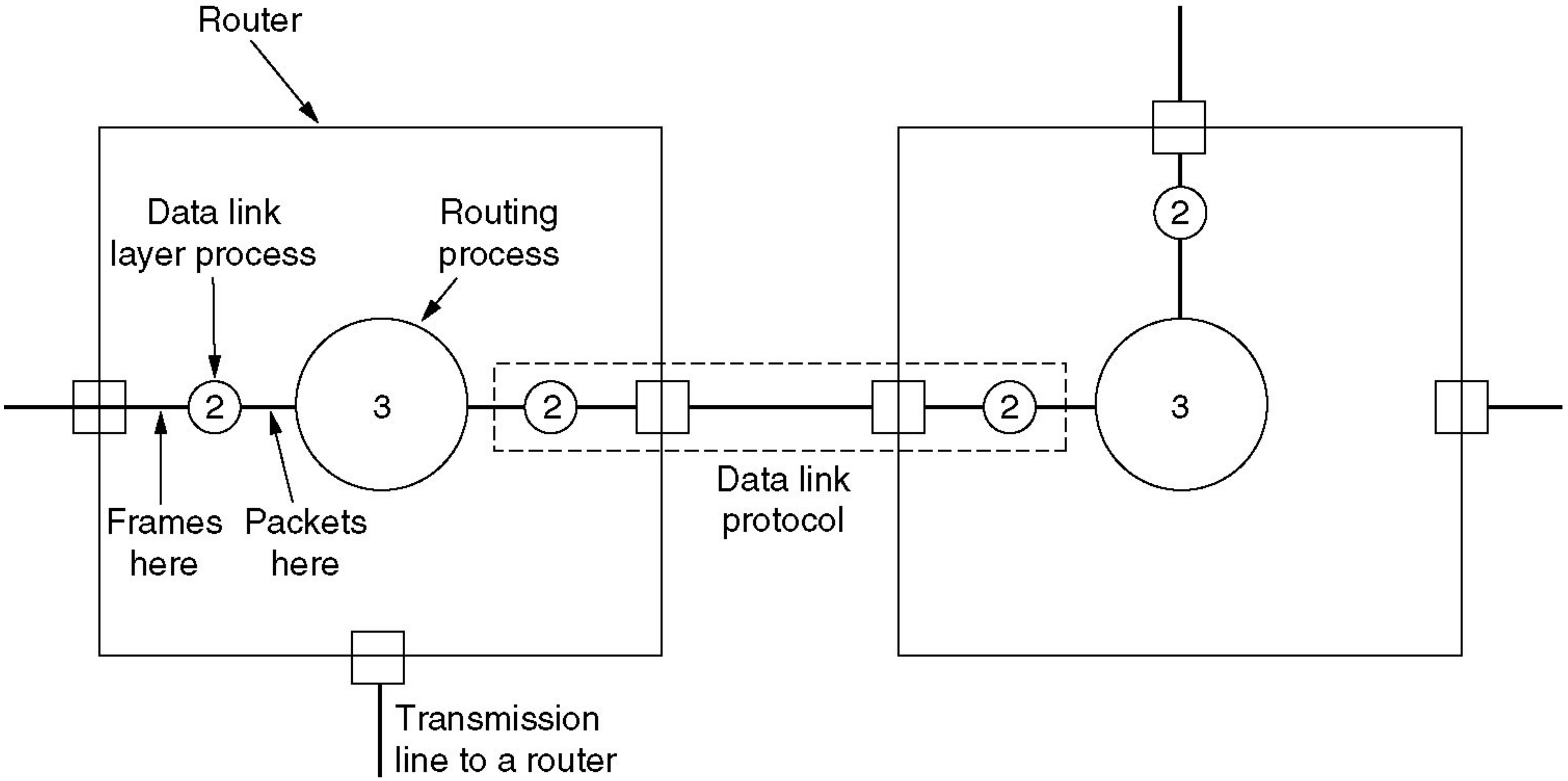


(a) Virtual communication (b) Actual communication

Layer 3 Packets and Layer 2 Frames



Placement of the Data Link Protocol



Framing

To Think

- [Sender] → ..1001101101101101101101110101.. → [receiver]
  - » Where is the data? Where does the frame start and stop?

How to split this bit stream into frames (sets of bits)?

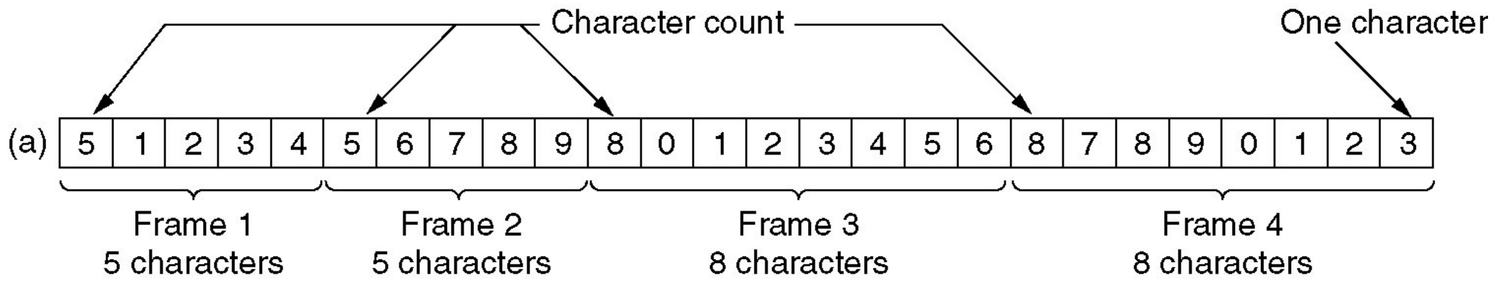
Framing

- [Sender] → ..1001101101101101101101110101.. → [receiver]
  - Where is the data?
  - Where does the frame start and stop?
- Three methods
  - **Character count**
  - **Flag bytes with byte stuffing**
  - **Start and ending flags, with bit stuffing**

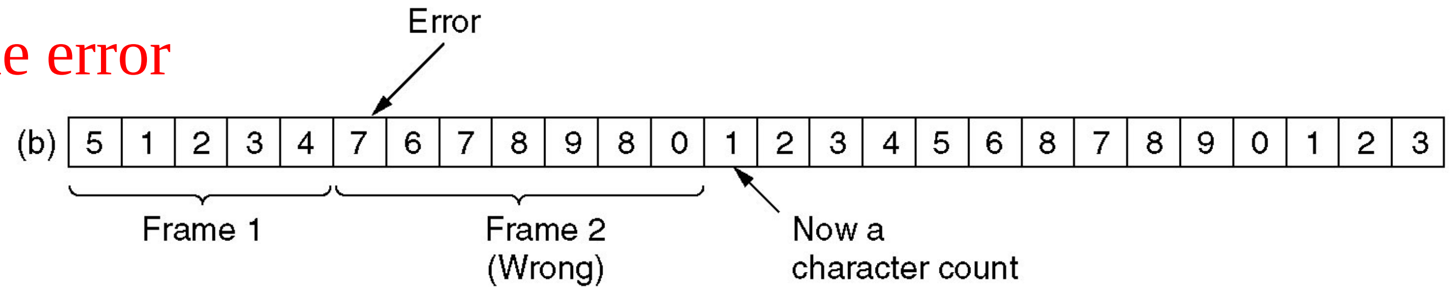
Framing – Character count

A stream of characters

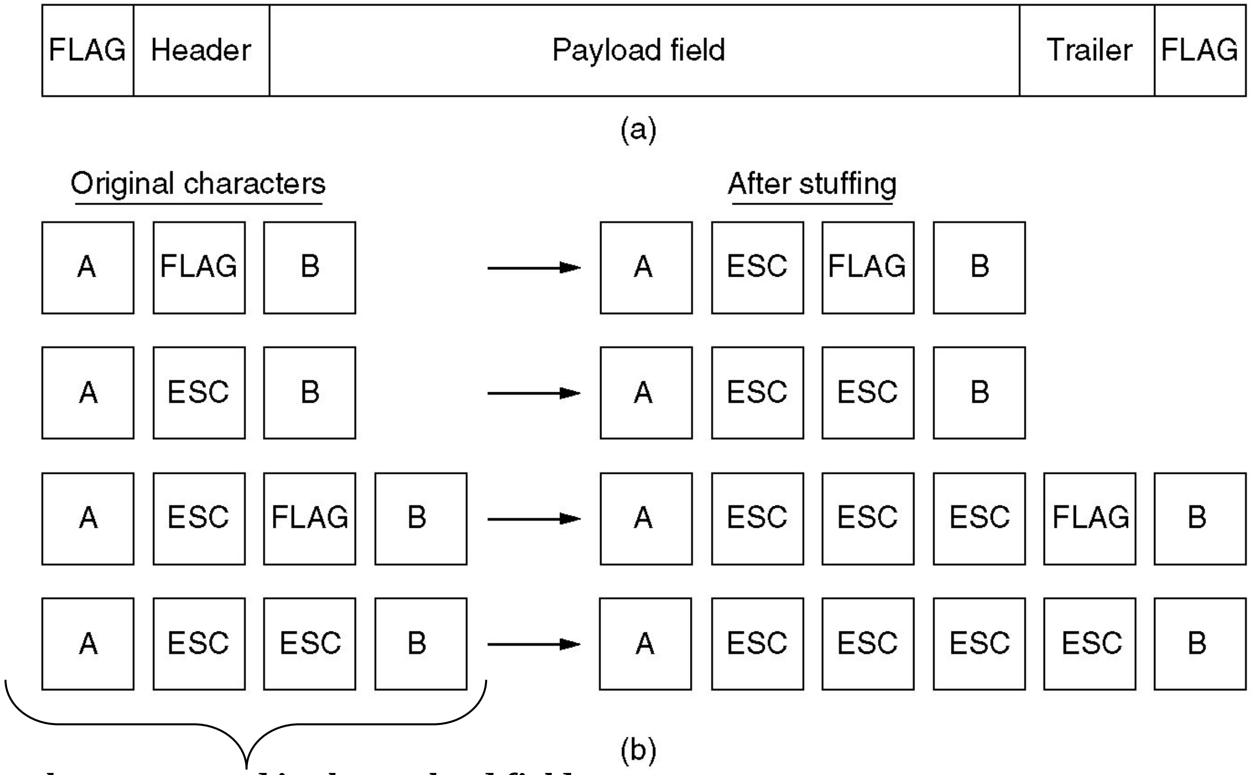
(a) Without errors



(b) With one error



Framing - Flag bytes with byte stuffing

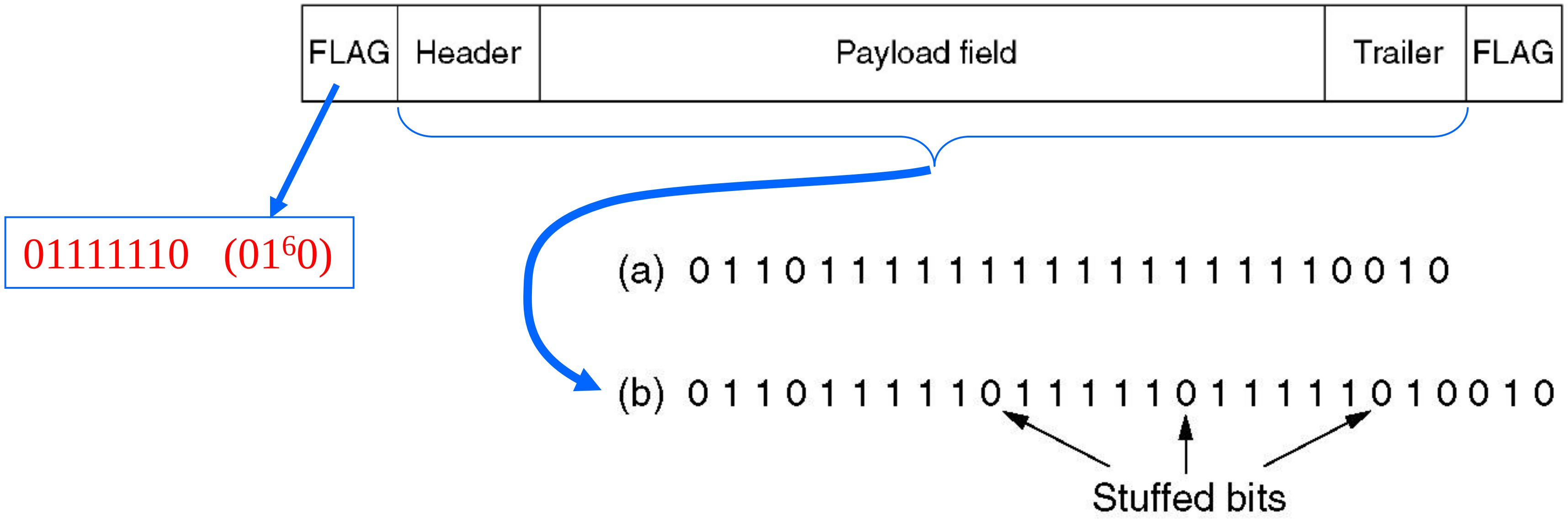


Data to be transported in the payload field

(a) A frame delimited by flag bytes

(b) Four examples of byte sequences before and after stuffing

Framing - Start and ending flags, with bit stuffing


FLAG
Header
Payload field
Trailer
FLAG
01111110 (01⁶0)
(a) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0
(b) 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0
Stuffed bits

(c) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

Bit stuffing

(a) The original data

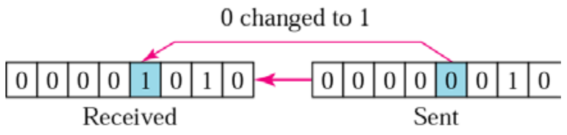
(b) The data as it appears on the line: $1^5 \rightarrow 1^5\underline{0}$

(c) The data as stored in receiver's memory **after destuffing:** $\mathbf{01^5\underline{0}} \rightarrow 01^5$

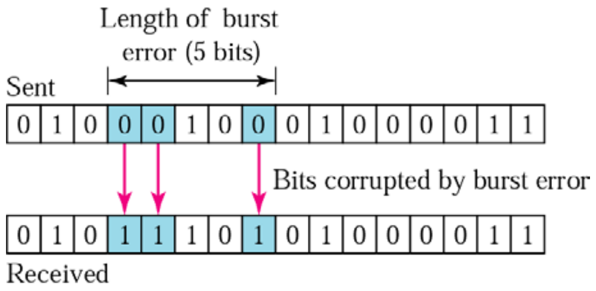
Error detection

Types of Errors

- Simple Error
  - » Random and independent from previous error



- Errors in burst
  - » Not independent; affect neighbour bits
  - » Burst lenght defined by the first and last bits in error



To Think

- Assume
  - p – bit error probability (or Bit Error Ratio – BER)
  - n – frame length
  - Independent errors
  - FER: Frame Error Ratio

- Student A explains to Student B

  why $\boxed{\text{P[frame has no errors]} = (1-p)^n}$

- Student B explains to Student A

  why $\boxed{\text{P[frame has errors]} = 1-(1-p)^n \Leftrightarrow FER = 1-(1-BER)^n}$

Counting Errors

- Assume
  - p – bit error probability (or Bit Error Ratio – BER)
  - n – frame length
  - Independent errors

- P[frame has no errors]= $(1-p)^n$

  the n bits are good!

- P[frame has errors]= $1-(1-p)^n$

  P[frame has errors]= Frame Error Ratio – FER

- P[1 bit received in error]= $\binom{n}{1} p(1-p)^{n-1}$

- P[i bits received in error]= $\binom{n}{i} p^i (1-p)^{n-i}$

$p=10^{-7}$ (**good wired channel**)
$n=10^4$ (~ Ethernet frame length)

P[FER]= $1-(1-10^{-7})^{10^4} \approx 10^{-3}$

$p=10^{-3}$ (**wireless channel**)
$n=10^4$ (~ Ethernet frame length)

P[FER]= $1-(1-10^{-3})^{10^4} \approx 1$

$$FER = 1-(1-BER)^n$$

Error Techniques

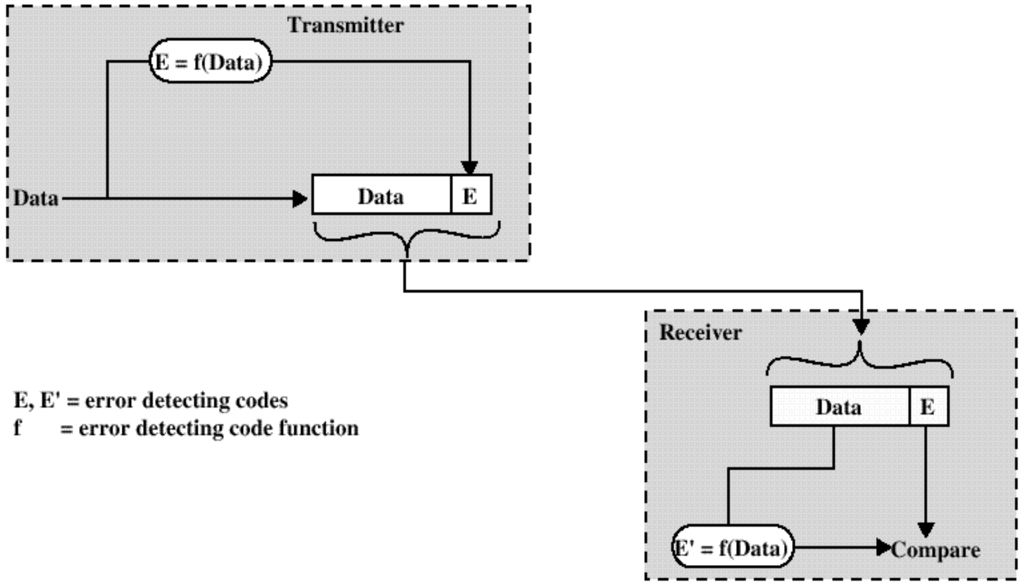
- Error techniques required!
  - Detection (and correction)

- Effectiveness of error detection technique (code) characterized by
  - Minimum distance of code: d
    - min number of bit errors undetected in a block of n bits
    - if fewer than d errors occur, errors are detected
  - Burst detecting ability: B
    - max burst length of errors detected

Error Detection Techniques

- Used by the receiver to determine if a packet contains errors
  - » If a packet is found to contain errors,
    the receiver may request the transmitter to re-send the packet

- Introducing redundancy ➔
  - » k ➔ k+r
    k: data bits; r: redundancy bits



- Error detection techniques
  - » Parity check
  - » Cyclic Redundancy Check (CRC)
  - » …

Simple Parity Check

- One parity bit added to every k information bits so that
  - » The total number of bits 1 even ➔ even parity

    1110111 1101110 1010110 1101100 1100100

    11101110 11011101 10101100 11011000 11001001
  - » The total number of bit 1 is odd ➔ odd parity

- Detection of
  - » simple errors
  - » any number of odd errors in a block of k+1 bits

- Undetected
  - » Even number of errors in a block

    n=k+1, block size

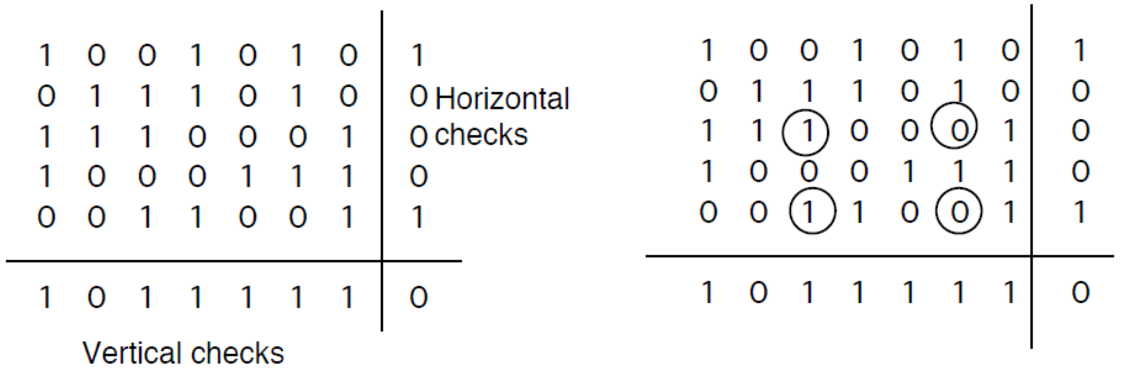
    p: bit error probability

$$P(undetected) = \sum_{i \text{ even}} \binom{n}{i} p^{i}(1-p)^{n-i}$$

- Used in Character Oriented protocols

Bi-dimensional Parity

- Blocks represented in rows
  - » Parity bit per row; parity bit per column



- Minimum code distance d=4

  Any four errors in a rectangular configuration becomes undetectable

Internet Checksum

- The Internet (not layer 2) uses a checksum
  - easily implementable in software
  - 1’s complement sum of 16 bit words
  - Performance: d=2

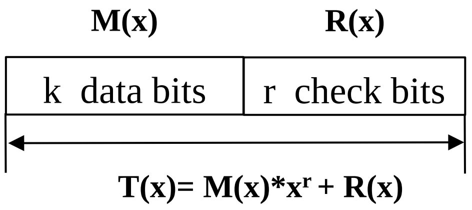
```
u_short
cksum(u_short *buf, int count)
{
    register u_long sum = 0;

    while (count--)
    {
        sum += *buf++;
        if (sum & 0xFFFF0000)
        {
            /* carry occurred,
               so wrap around */
            sum &= 0xFFFF;
            sum++;
        }
    }
    return ~(sum & 0xFFFF);
}
```

- One’s complement sum
  - Mod-2 addition with carry-out
  - Carry-out in the most-significant-bit is added to the least-significant bit
  - Get one’s complement of “one’s complement sum”

```
                         1010011
                         0110110
            carry-out  1 0001001
Carry wrap-around        0000001
                         0001010
One's complement =       1110101
```

Cyclic Redundancy Check

- Bit string represented as a polynomium
  - » 110011 → x^5+x^4+x+1
- Module 2 operations
  - » Additions and subtractions identical to exclusive OR
  - » no carry, no borrow
- $M(x)$; $R(x)$; $T(x)=M(x) * x^r + R(x)$



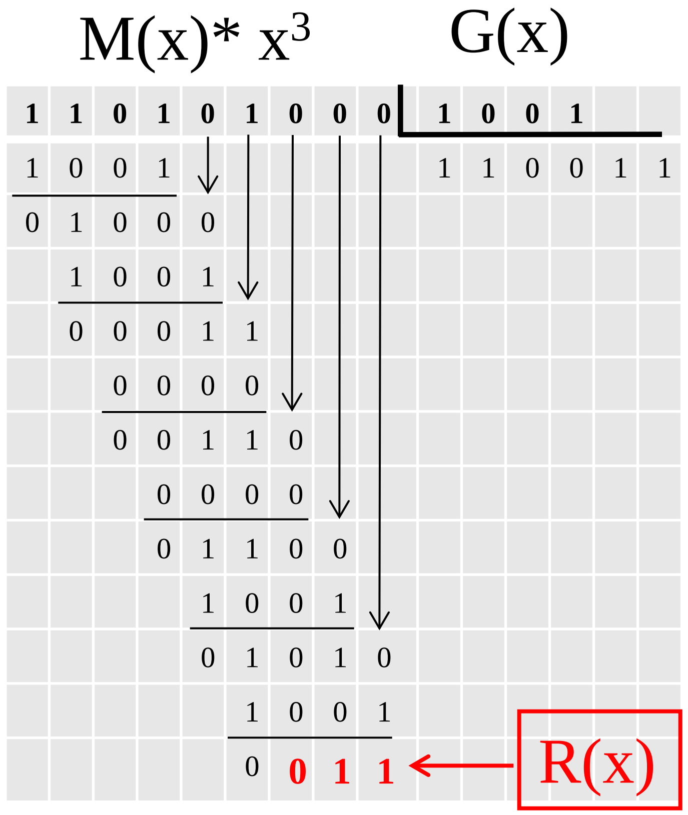
- How to compute the check bits: R(x)?
  - » Choose a generator string G(x) of length r+1 bits
  - » Choose R(x) such that T(x) is a multiple of G(x): $T(x)=A*G(x)$; $R(x)=0$
- $T(x)=M(x)x^r+R(x) = A*G(x) \Leftrightarrow$

  $M(x)x^r = A*G(x) + R(x)$ (mod 2 arithmetic)

  $\Rightarrow R(x)$ = remainder of $M(x)x^r / G(x)$
- Choice of G(x) is very important! ($G(x)=x^r+\ldots+1$)

CRC - Generating R(x)

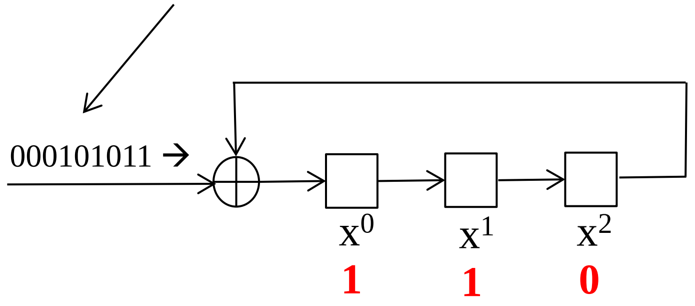
- $r=3$, $x^r=x^3$; $G(x)=x^3+1$ (1001)
- $M(x)=x^5+x^4+x^2+1$ (110101)
- $M(x)*x^3 = x^8+x^7+x^5+x^3$ (110101000)
- $R(x)=$ remainder of $M(x)x^r / G(x)$
- $R(x)=x+1$ (011)
- Sent word
  - » $T(x)=M(x)*x^r+R(x) = x^8+x^7+x^5+x^3+x+1 =$ 110101 011



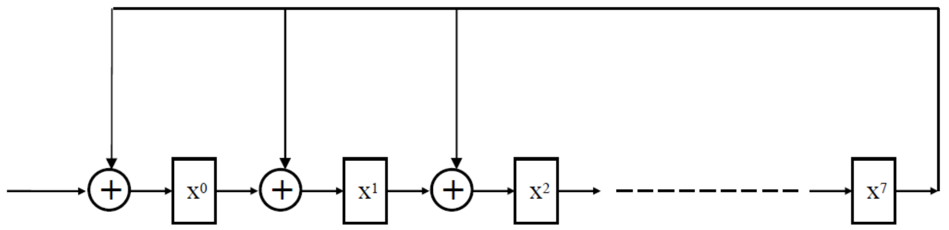
CRC - Generating R(x) with a Shift Register

- R(x) easily generated in hardware

- $G(x)=x^3+1$
  - » $M(x)^* x^3 = x^8+x^7+x^5+x^3$ (110101000)
  - » $R(x)=x+1$ (011)



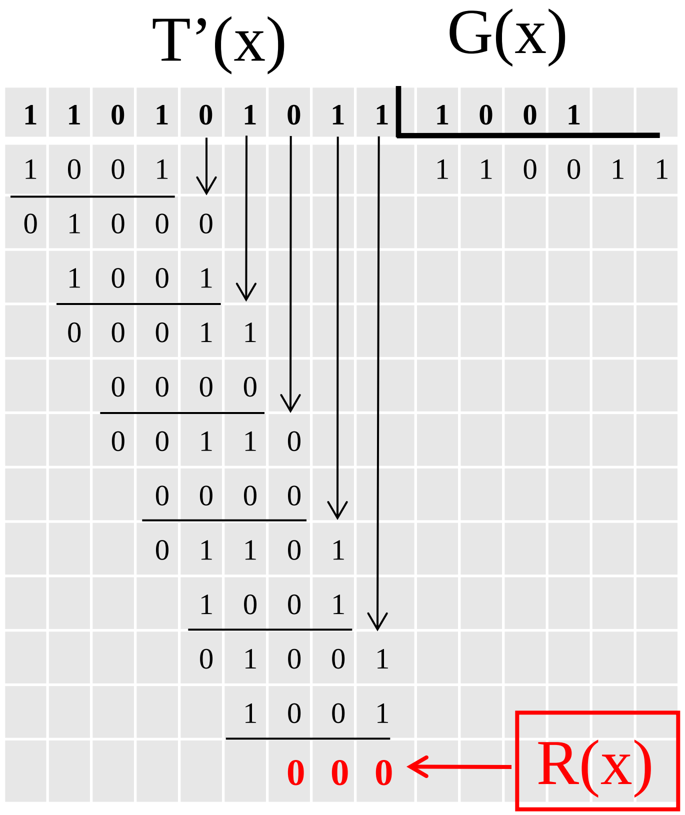
- $G(x)=x^8+x^2+x+1$



CRC – Checking at the Receiver

- Let T’(x) be the received word
  - » T’(x)= $x^8+x^7+x^5+x^3+x+1$ (110101 011)
- Divide T’(x) by G(x)
  - » If remainder R(x)=0 ➔ no errors
  - » If remainder R(x) !=0

    ➔ errors have occurred



```
1 1 0 1 0 1 0 1 1 | 1 0 0 1
1 0 0 1              1 1 0 0 1 1
0 1 0 0 0
  1 0 0 1
  0 0 0 1 1
    0 0 0 0
    0 0 1 1 0
      0 0 0 0
      0 1 1 0 1
        1 0 0 1
        0 1 0 0 1
          1 0 0 1
            0 0 0
```

CRC - Performance

- For r check bits per frame the following can be detected
  - All patterns of 1, 2, or 3 errors ($d > 3$)
  - All bursts of errors of r or fewer bits
  - All errors consisting of an odd number of inverted bits
- Common polynomials
  - ITU-16: r=16, $G(x) = x^{16} + x^{12} + x^5 + 1$ (10001000000100001)
  - ITU-32: r=32,

    $G(x) = x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x^1+1$

Automatic Repeat reQuest (ARQ)

Automatic Repeat ReQuest (ARQ)

- When the receiver detects errors in a frame

  how to ask the sender to retransmit the frame?

- ARQ systems

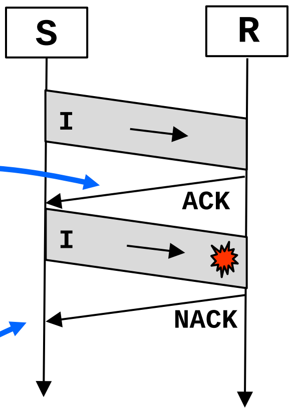
  Those which automatically request the retransmission of

  - missing packets
  - packets with errors

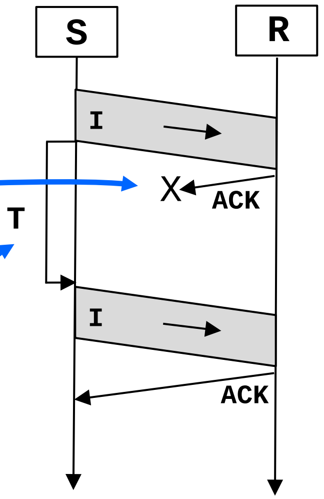
- Three common ARQ schemes
  - » **Stop and Wait**
  - » **Go Back N**
  - » **Selective Repeat**

Stop and Wait ARQ

- Sender
  - » transmits Information frame **I**
    waits for positive confirmation **ACK** from receiver
- Receiver: receives **I** frame
  - » If **I** frame has no error ➔ confirms with **ACK**
  - » If **I** frame has error ➔ sends **NACK**

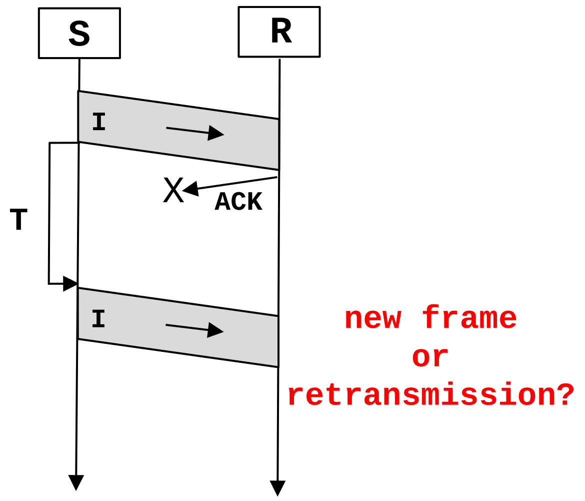


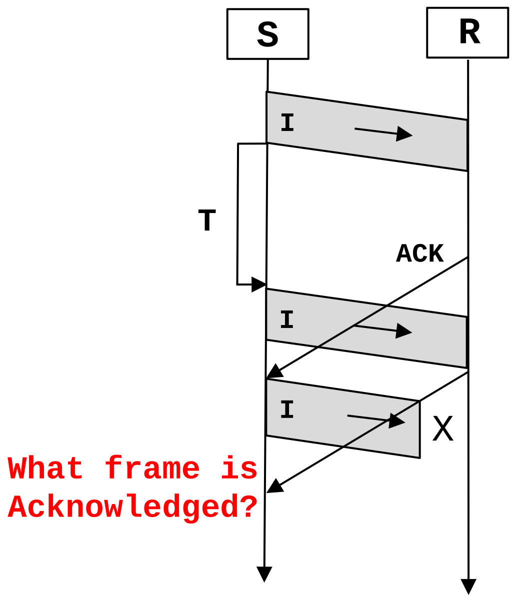
- Sender
  - » Receives **ACK** ➔ proceeds and transmits new frame
  - » Receives **NACK** ➔ retransmits frame **I**



- Problem
  - » What happens if **I, ACK** or **NACK** is lost?
    ➔ Timeout required!

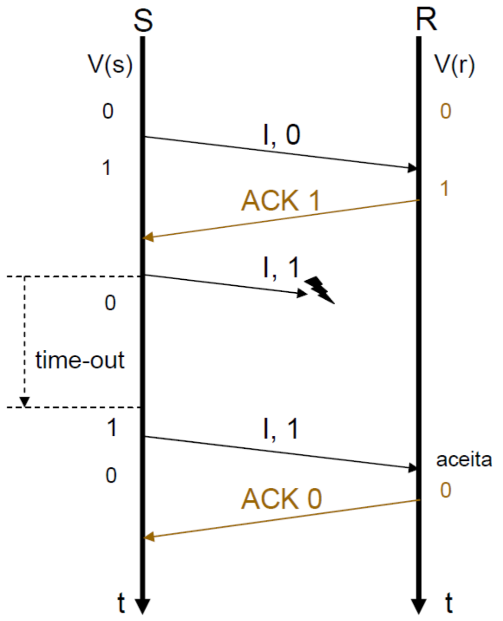
Stop and Wait ARQ – Sequence Numbers Required

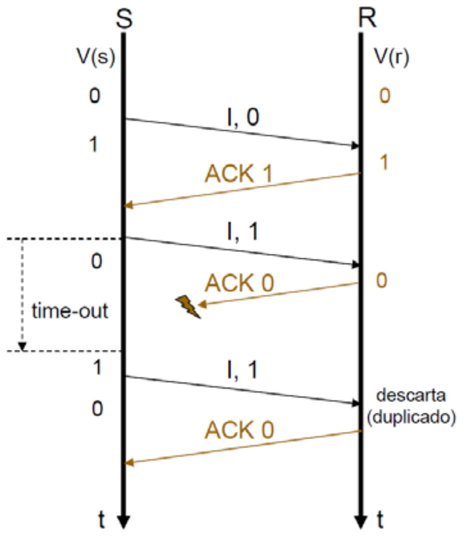


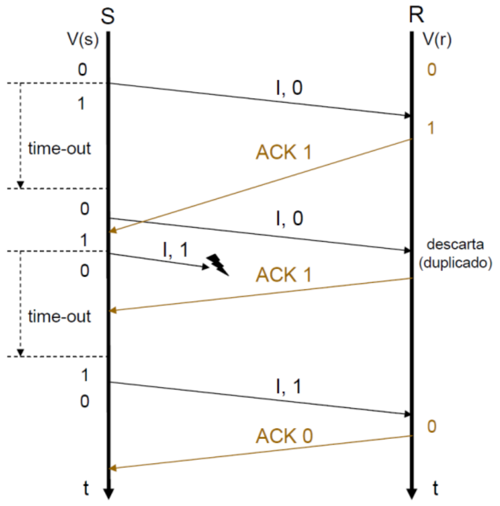


- Solution
  - **I** frames numbered: **I(0)**, **I(1)**
  - **ACK** frames numbered: **ACK(0), ACK(1)**
  - **ACK(i)** indicates that receiver is waiting for frame **I(i)**
  - No **NACK** required
  - Module 2 numbers

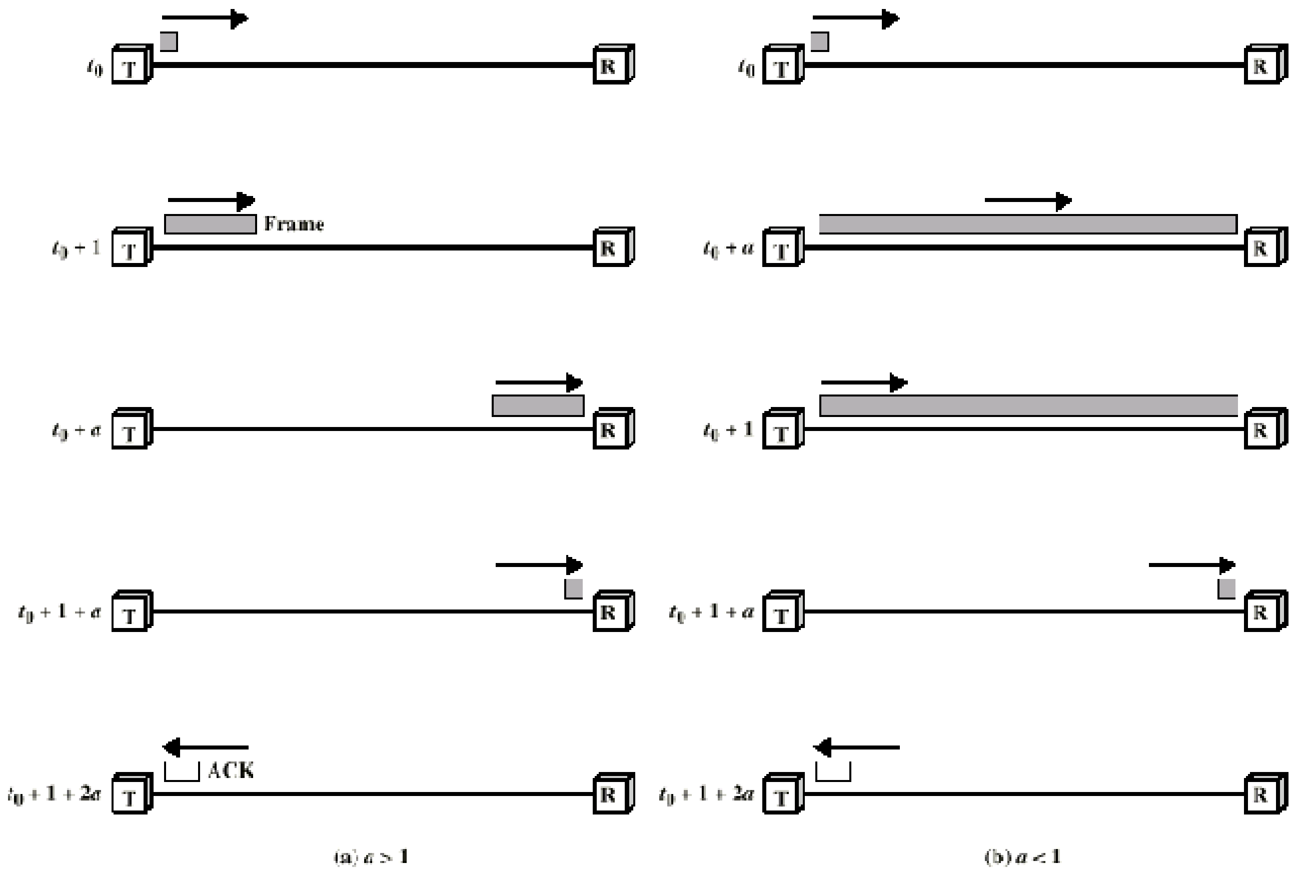
Stop and Wait ARQ – Examples







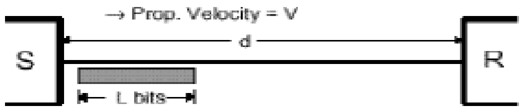
Stop and Wait - Efficiency



Stop-and-Wait Link Utilization (transmission time = 1; propagation time = *a*)

Stop and Wait – Efficiency Example

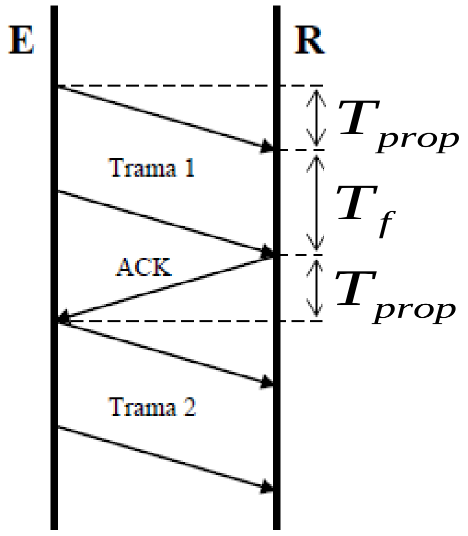
- WAN ATM
  - d = 1000 km
  - L = 424 bit, R = 155.52 Mbit/s
  - T_t = 2.7 µs
  - Fibra óptica → 5 µs/km → τ = 5 ms
  - a = 1852
  - **S = 1 / 3705 = 0.0003**
- LAN
  - d = 0.1 ~ 10 km
  - L = 1000 bit, R = 10 Mbit/s
  - T_t = 100 µs
  - Cabo coaxial → 4 µs/km → τ = 0.4 ~ 40 µs
  - a = 0.004 ~ 0.4
  - **S = 0.55 ~ 0.99** (e se R = 100 Mbit/s?)
- Modem sobre linha telefónica
  - d = 1000 m
  - L = 1000 bit, R = 28.8 kbit/s
  - T_t = 34.7 ms
  - UTP → 5 µs/km → τ = 5 µs
  - a = $1.44 \cdot 10^{-4}$
  - **S ≃ 1.0**



→ R – *data rate* (bit/s)

$$T_{prop} = \tau = \frac{d}{V}$$

$$T_f = \frac{L}{R}$$



$$a = \frac{T_{prop}}{T_f}$$

$$S = \frac{T_f}{T_{prop} + T_f + T_{prop}} = \frac{1}{1+2a}$$

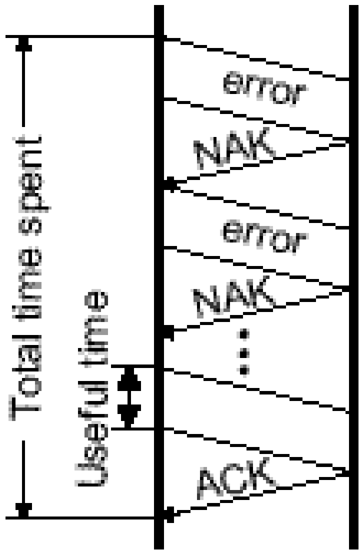
Stop and Wait ARQ – Efficiency with Errors

» p_e – frame error probability

» P[A=k]

– probability of **k Attempts** required to transmit a frame with success

$$\mathrm{P}\left[A=k\right]= {p_e}^{k-1}(1-p_e)$$



» E[A]

expected number of Attempts to transmit a frame with success

$$E[A]=\sum_{k=1}^{+\infty} k*\mathrm{P}[A=k]=\frac{1}{1-p_e}$$

» Efficiency

$$S=\frac{T_f}{E[A](T_f+2T_{prop})}=\frac{1}{E[A](1+2a)}=\frac{1-p_e}{1+2a}$$

To Think

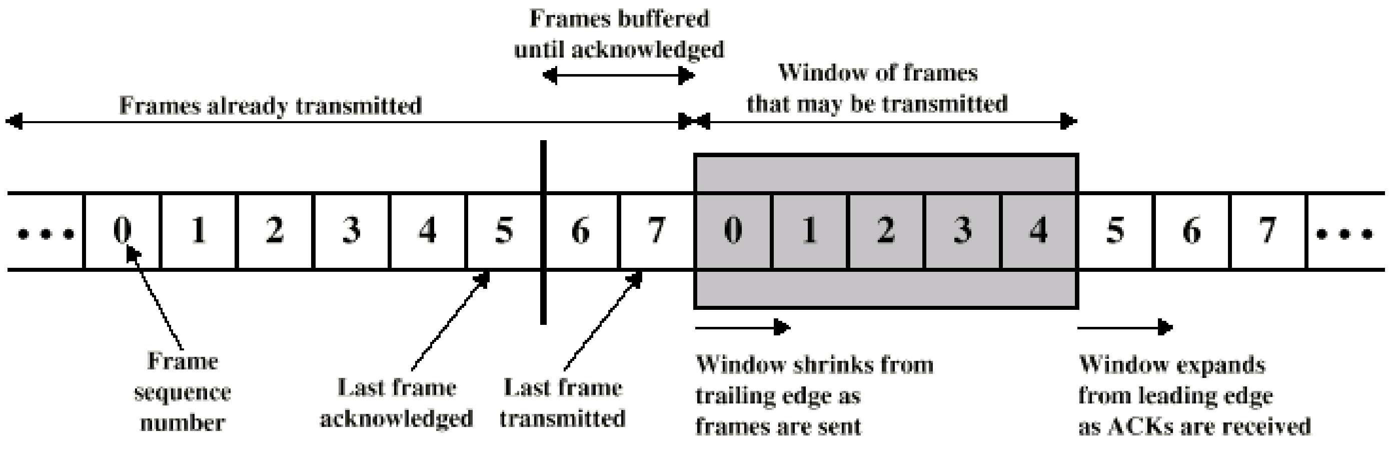
- Assume Sender and Receiver are separated by a large distance?

  How to improve the Efficiency of the STOP&Wait ARQ?

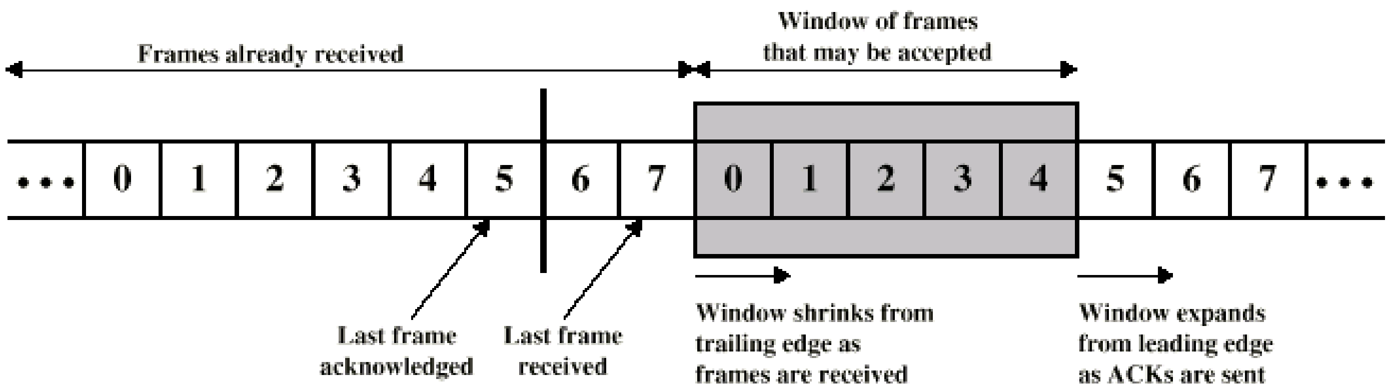
Go Back N ARQ (Sliding Window)

- Stop and Wait
  - » inefficient when $T_{prop} > T_f$ (a>1)
  - » sends only one frame per Round-Trip Time ($RTT=2*T_{prop} + T_f$)
- Go Back N
  - » allows transmission of new packets before earlier ones are acknowledged
  - » uses a Sliding Window mechanism
    - – sender can send packets that are within a “window” (range) of packets
    - – window advances as acknowledgements for earlier packets are received

Sliding Window - Model

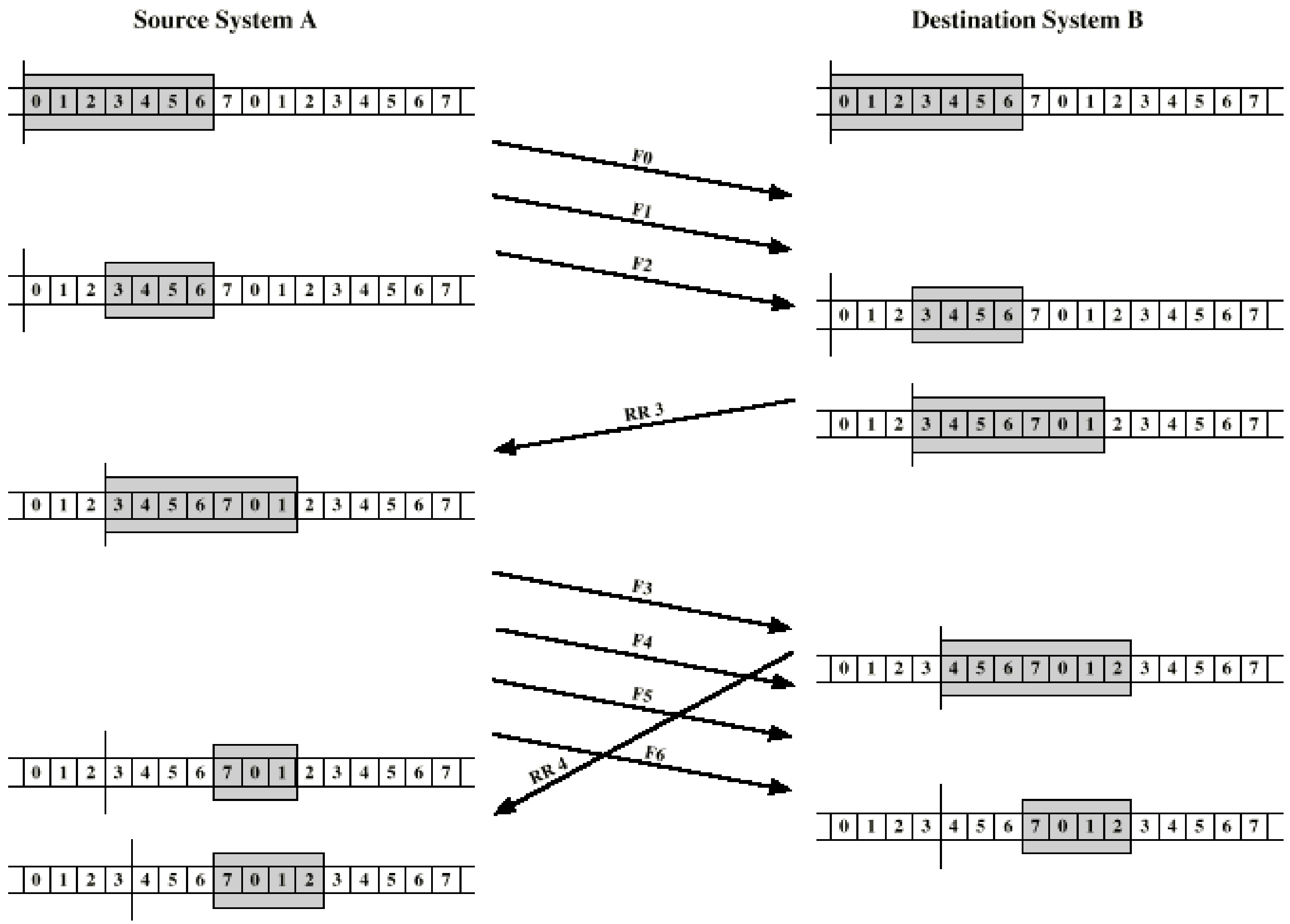

Frames buffered until acknowledged
Window of frames that may be transmitted
Frames already transmitted
• • • 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 • • •
Frame sequence number
Last frame acknowledged
Last frame transmitted
Window shrinks from trailing edge as frames are sent
Window expands from leading edge as ACKs are received

(a) Sender's perspective


Window of frames that may be accepted
Frames already received
• • • 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 • • •
Last frame acknowledged
Last frame received
Window shrinks from trailing edge as frames are received
Window expands from leading edge as ACKs are sent

(b) Receiver's perspective

Sliding Window - Example



Go Back N ARQ – Basic Behaviour

- Sender
  - » may transmit up to W frames without receiving RR

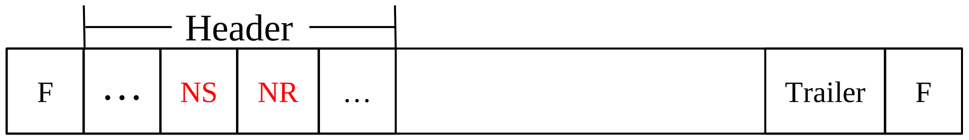
    RR - Receiver Ready = ACK
  - » I frames are numbered sequentially I(NS): I(0), I(1), I(2), …
  - » Cannot send I(NS=i+W) until it has received the RR(NR=i)

- Receiver
  - » does not accept frames out of sequence
  - » sends RR(NR) to sender indicating
    - – that all the packets up to NR-1 have been received in sequence
    - – the sequence number, NR, of the next expected frame

Go Back N ARQ – Maximum Window, Extensions

- Sequence numbers are represented module M
  - » NS, NR in {0, 1, …, M-1}

- Maximum Window
  - » W = M-1= 2^k -1
  - » k is number of bits used to code sequence numbers



- Extensions to basic behaviour
  - » Piggybacking can be used for bidirectional flows
  - » RR information can be sent in the data packets of opposite direction

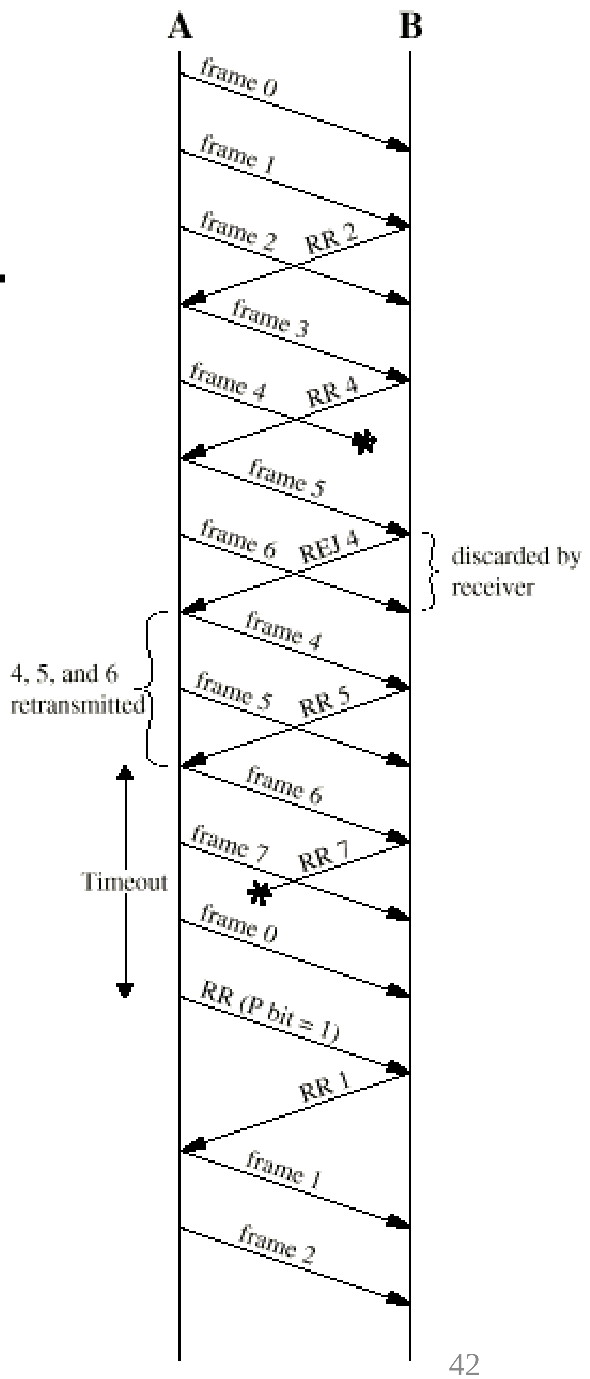
Go Back N ARQ – Behaviour under Errors

- Frame with errors
  is silently discard by the Receiver

- If Receiver receives Data frame out of sequence
  - » First out-of-sequence-frame?
    - – Receiver sends REJ(NR)
    - – NR indicates the next in-sequence frame expected
  - » Following out-of sequence-frames
    - – Receiver discards them; no REJ sent

- When Sender receives REJ(NR=x), the Sender
  - » Goes-Back and retransmits I(x), I(x+1), …
  - » Continues using Sliding Window mechanism

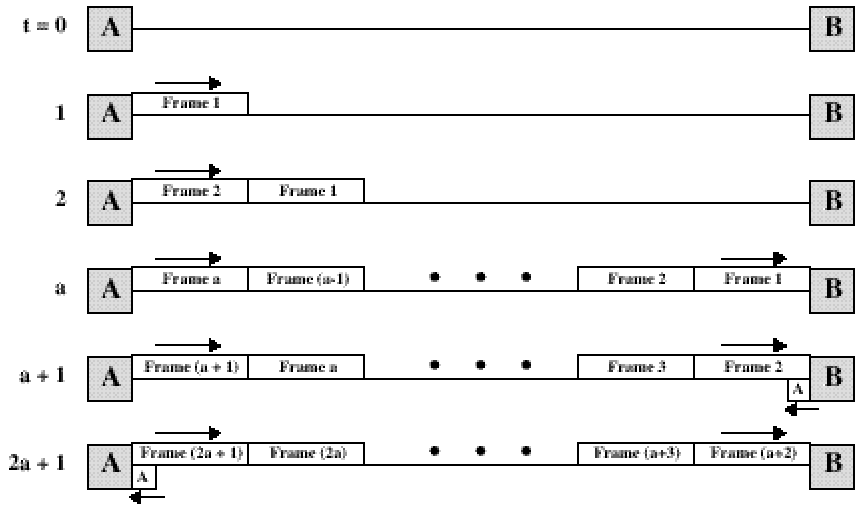
- If timeout occurs, the Sender
  - » requests the Receiver to send a RR message
  - » by sending a special message (RR command message)



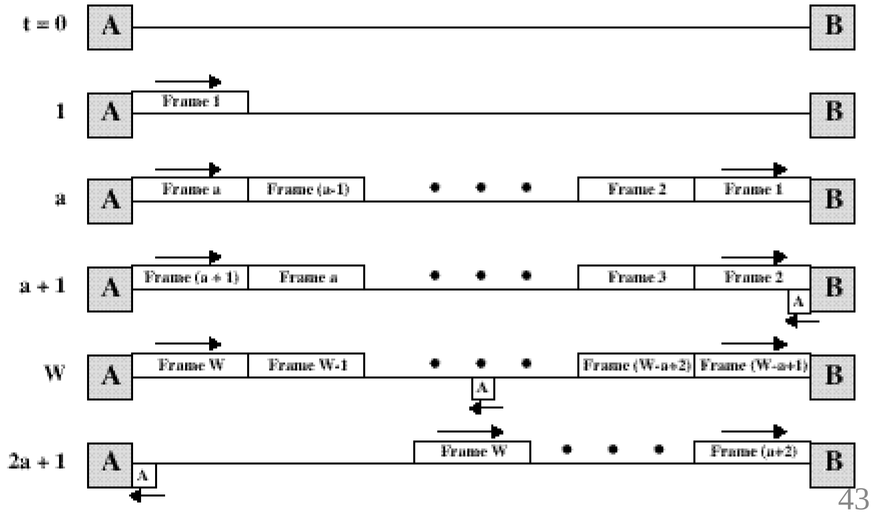
Go Back N – Efficiency

- If W ≥ 1+2a ➔ S = 1

- If W < 1+2a ➔ S = W/(1+2a)



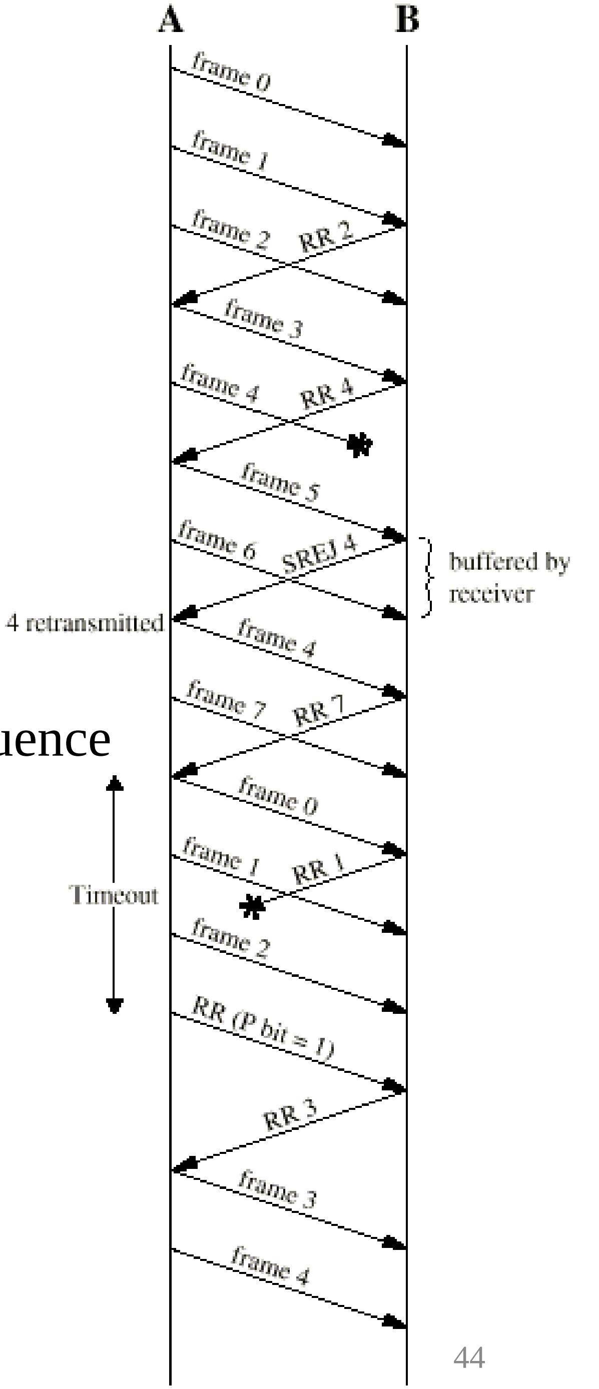
(a) W 2a + 1



(b) W < 2a + 1

Selective Repeat ARQ

- Uses Sliding Window, but …
- Receiver
  - » accepts out-of-sequence-frames
  - » confirms negatively, SREJ, a frame not arrived
  - » uses RR to confirm blocks of frames arrived in sequence
- Sender
  - » retransmits only the frames signaled by SREJ
- Adequate if W (a) is very large
- Maximum window size, $W = \dfrac{M}{2} = 2^{k-1}$



Go-Back-N and Selective Repeat ARQ – Efficiency under Errors

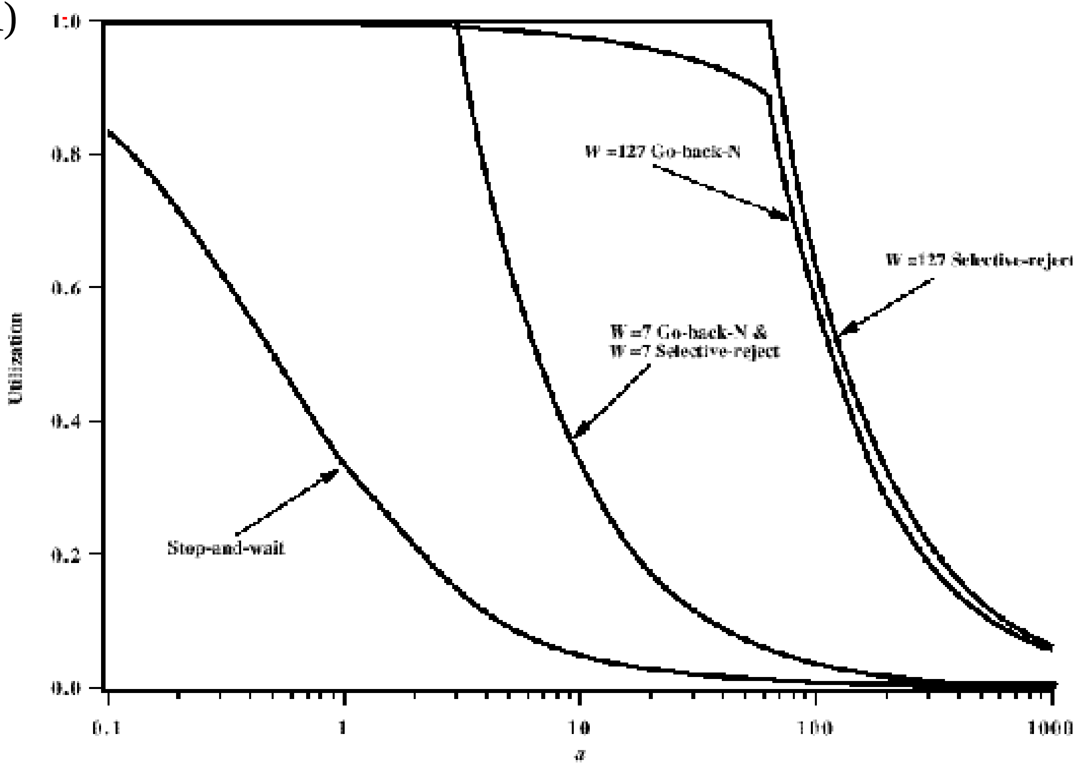
- *Go-Back-N ARQ*

  p_e – frame error probability (ratio, FER)

$$S = \begin{cases} \dfrac{1-p_e}{1+2ap_e} & ,W \geq 1+2a \\ \dfrac{W(1-p_e)}{(1+2a)(1-p_e+Wp_e)} & ,W < 1+2a \end{cases}$$

- *Selective Repeat ARQ*

$$S = \begin{cases} 1-p_e & ,W \geq 1+2a \\ \dfrac{W(1-p_e)}{1+2a} & ,W < 1+2a \end{cases}$$



ARQ Utilization as a Function of a ($P = 10^{-3}$)

Framing, Error detection and ARQ in common networks

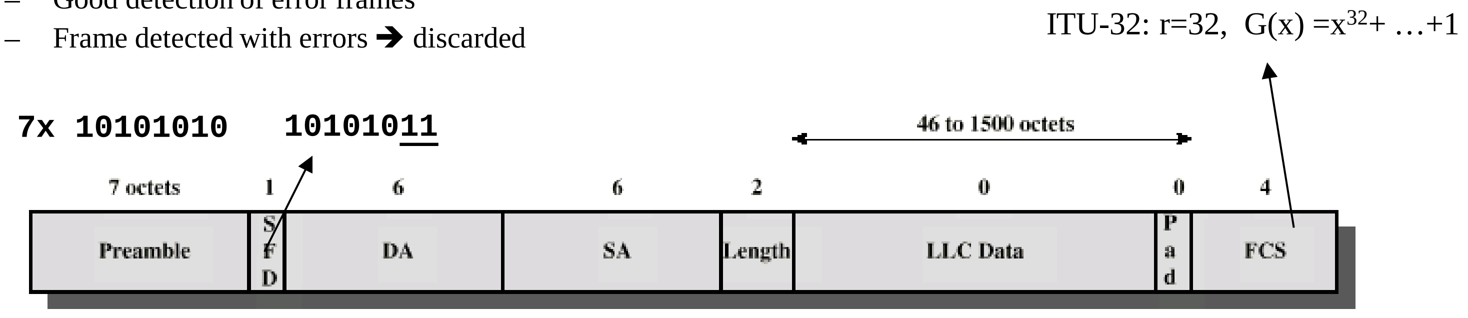
Ethernet

- Framing
  - » Start of frame: preamble + SFD
  - » End of frame: end of signal transitions (Manchester code), length
- Error detection: FCS ➔ ITU-32, $G(x) = x^{32} + \ldots + 1$
- No ARQ
  - » Bit Error ratio (BER) very low
    ➔ Frame Error Ratio (FER) low
  - » CRC/FCS strong
    - – Good detection of error frames
    - – Frame detected with errors ➔ discarded

$p=10^{-7}$ (**good wired channel**)

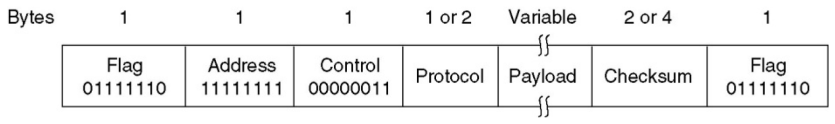
$n=10^{4}$ (~ Ethernet frame length)

P[frame has errors]= $1-(1-10^{-7})^{10^4} \approx 10^{-3}$



SFD = Start of frame delimiter
DA = Destination address
SA = Source address
FCS = Frame check sequence

Point to Point Protocol

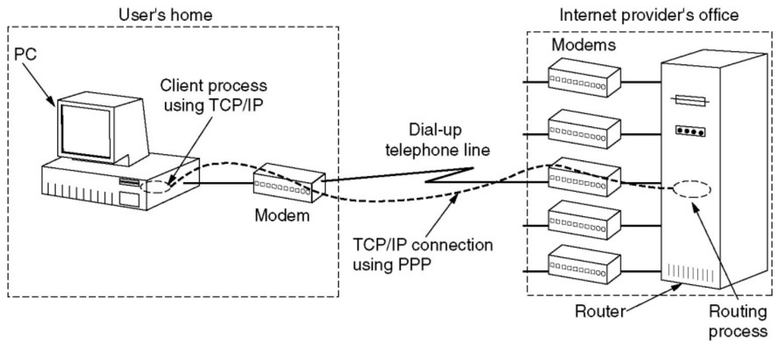


Bytes	1	1	1	1 or 2	Variable	2 or 4	1
	Flag 01111110	Address 11111111	Control 00000011	Protocol	Payload	Checksum	Flag 01111110

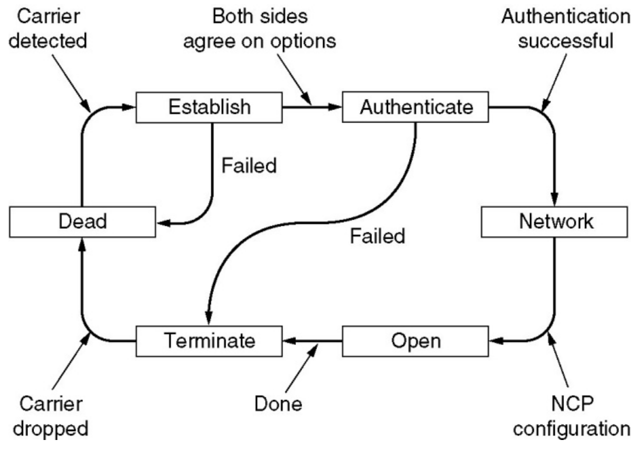
`Byte stuffing`

- Framing: Flags - 0x7E
- Byte stuffing: ESC – 0x7D
- Error detection – can be negotiated
- No ARQ

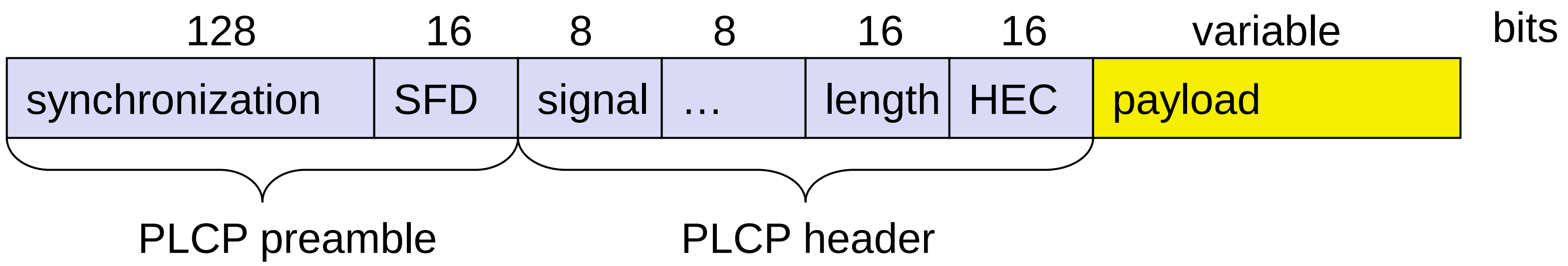
- Two additional protocols
  - » LCP – Link Control Protocol
  - » NCP – Network Control Protocol



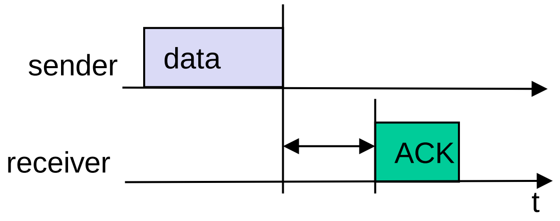
User's home
PC
Client process using TCP/IP
Modem
Dial-up telephone line
TCP/IP connection using PPP
Internet provider's office
Modems
Router
Routing process



Wireless LAN

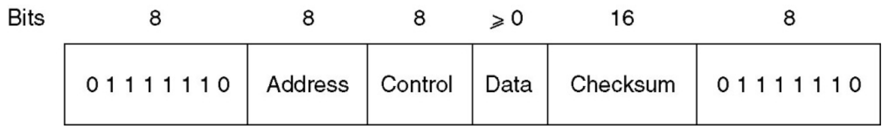


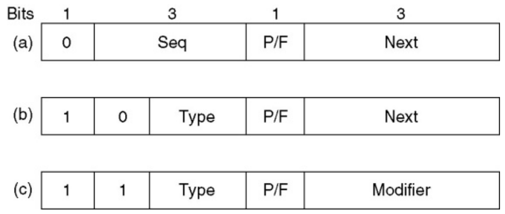
- Framing
  - Synchronization: 0101010 …
  - SFD (Start Frame Delimiter → 1111001110100000
  - Length → Payload length **<u>in us</u>**
- HEC (Header Error Check)
  - ITU-16, G(x)= $x^{16}+x^{12}+x^5+1$
- Payload (data)
  - Protected by strong codes
- ARQ
  - modified version of Stop and Wait
- Signal: Payload bitrate (0A: 1 Mbit/s DBPSK; 14: 2 Mbit/s DQPSK)



High-Level Data Link Control

- HDLC, Data Link Control, bit oriented





Control field of :

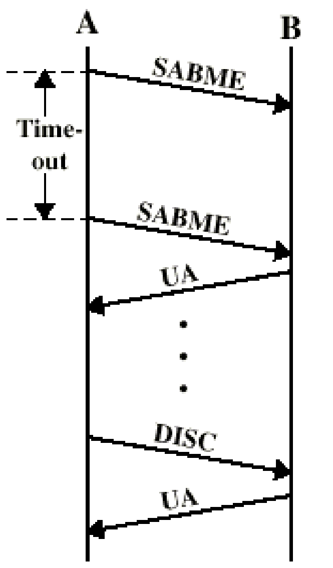
a) An information frame.

b) A supervisory frame.

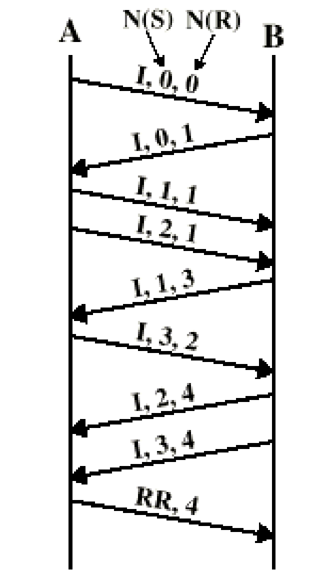
c) An unnumbered frame.

- Framing – FLAGS
- Bit stuffing
- Error detection – ITU-16
- Bit stuffing
- ARQ – Selective Repeat ARQ
- Used as basis for many telecom networks
  - » GSM/GPRS/UMTS, Frame Relay
  - » LAP-x protocols

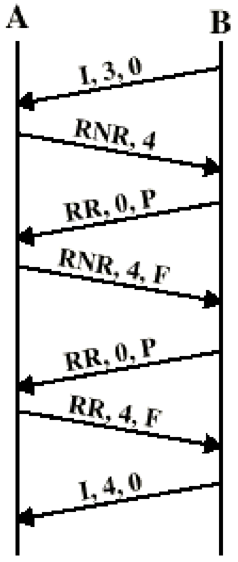
HDLC - Examples



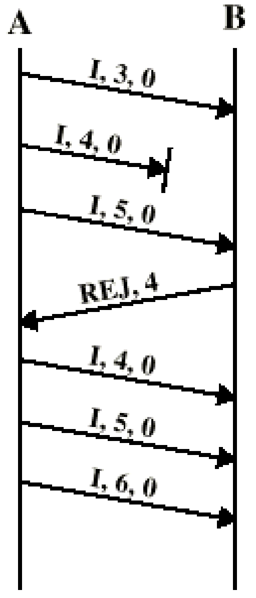
(a) Link setup and disconnect



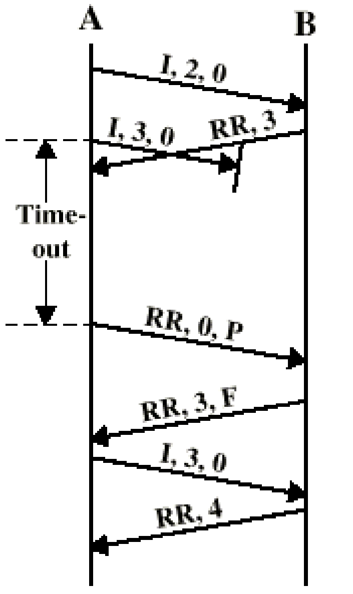
(b) Two-way data exchange



(c) Busy condition



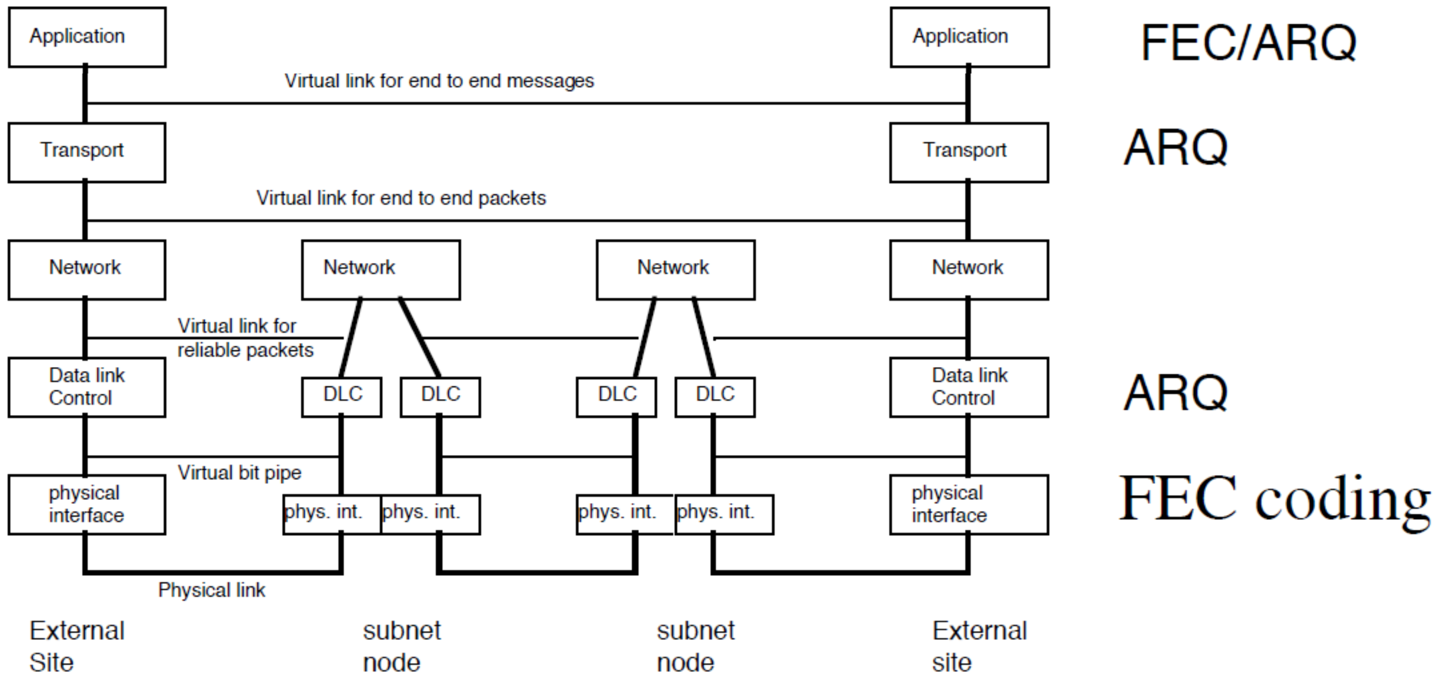
(d) Reject recovery



(e) Timeout recovery

Reliability in the Protocol Stack

Reliability in the Protocol Stack



Reliability in the TCP/IP Reference Model

- The TCP/IP reference model assumes
  - » Every layer 2 offers an error free service to the upper layer
  - » Service Data Units are
    - – delivered to upper layer without error,
    - – or discarded

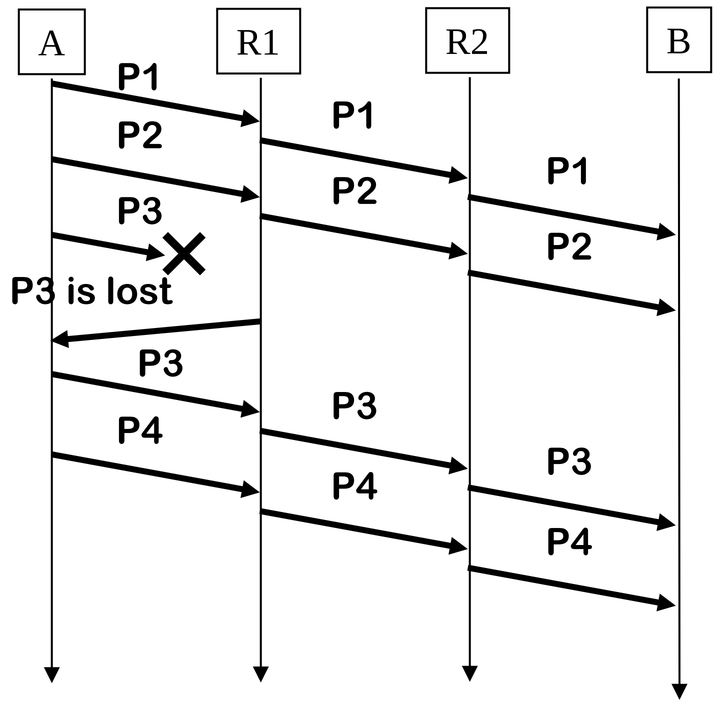
$$\boxed{FER = 1-(1-BER)^n}$$

- The layered model transforms bit error in packet losses

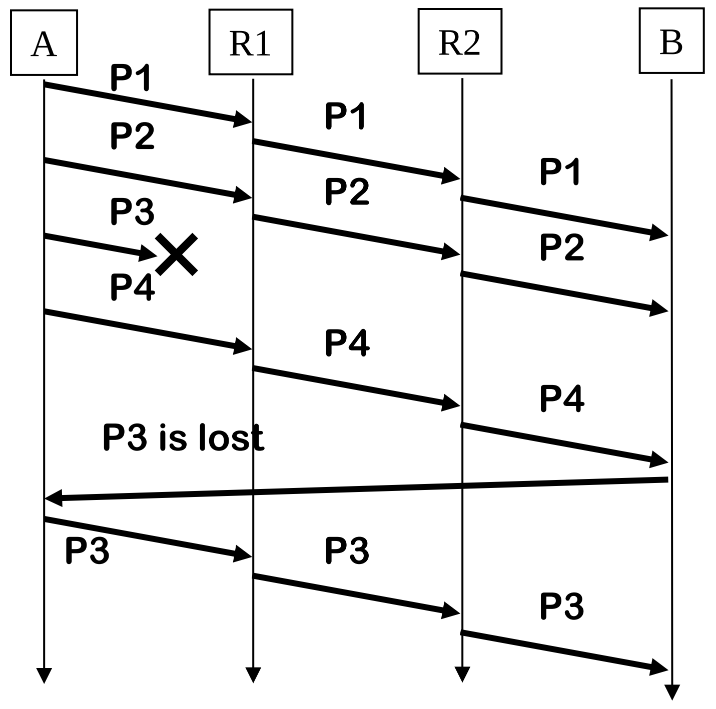
  Therefore, packet losses must be repaired ➔ ARQ solutions

- Two strategies can be used
  - » Link-by-Link ARQ
  - » End-to-end ARQ

Link-by-Link ARQ versus End-to-End ARQ



Link-by-Link ARQ
(data link layer)

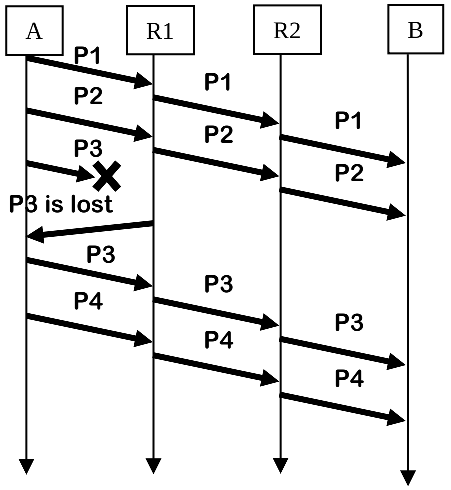


End-to-End ARQ
(transport or Application layers)

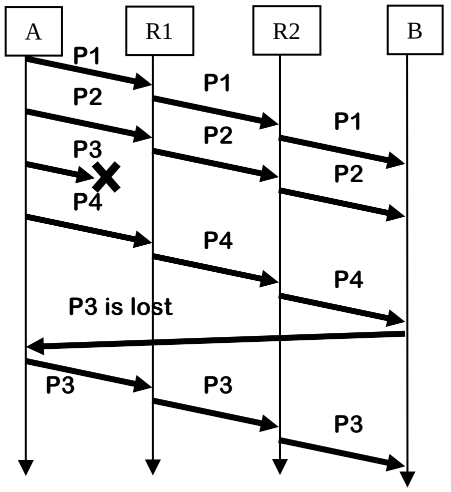
To Think

- Link-by-Link ARQ versus End-to-end ARQ

  Compare these approaches from the points of view of **delays and bitrates**



Link-by-Link ARQ
(data link layer)



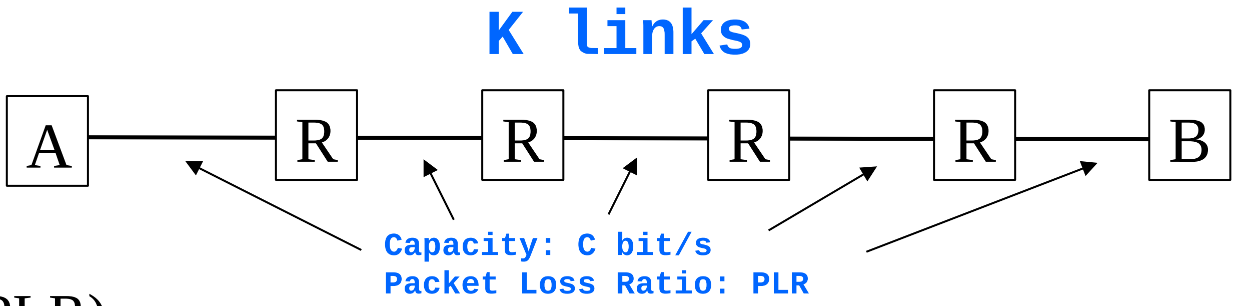
End-to-End ARQ
(transport or Application layers)

Link-by-Link ARQ versus End-to-End ARQ

- Link-by-Link ARQ
  - » Repairs losses link by link
  - » Requires network elements to
    - – remember information about packet flows ➔ high processing per frame/packet
    - – store packets in case they have to be retransmitted ➔ memory required

- End-to-end ARQ
  - » Low complexity
    - – Intermediate network elements (switches, routers) become simple
  - » Packets may follow different end to end paths
  - » But, not acceptable when Packet Loss Ratio is high
  - » Let’s see why …

End to End Capacity



- Packet Loss Ratio (PLR)
  - Packet ➔ layer 3; Frame ➔ layer 2
  - Let's assume PLR=FER

    (not considering losses in queues)

k	PLR	C_{EE}	C_{LL}
10	0.05	**0.6*C**	0.95 *C
10	0.0001	0.9990 *C	0.9999 *C

- Capacity of one link $C_l=C*(1-PLR)$
- End to End capacity
  - using Link-by-Link ARQ: $C_{LL}=C_l=C*(1-PLR)$
  - Using End-to-End ARQ: $C_{EE}=C*(1-PLR)^K$
- End-to-end ARQ ➔ **Inefficient when PLR is High**

ARQ in the TCP/IP Reference Model

- The TCP/IP architecture assumes that
  - The Data Link layer provides error-free packets to the network layer
  - Data link layer provides a service with very low FER
  - End-to-End ARQ is used, implemented at

    Transport or Application layers

- In the TCP/IP reference model, packet losses are repaired
  - At the Data Link layer on lossy channels (e.g. wireless data links)
  - At the end systems (transport layer or application layer)

Homework

1. Review slides

2. Read from Tanenbaum
   » Chapter 3 (5th edition)

3. Read from Bertsekas&Gallager
   » Sections 2.3, 2.4

4. Answer questions at moodle