Datalink layer: Flow and Error Control

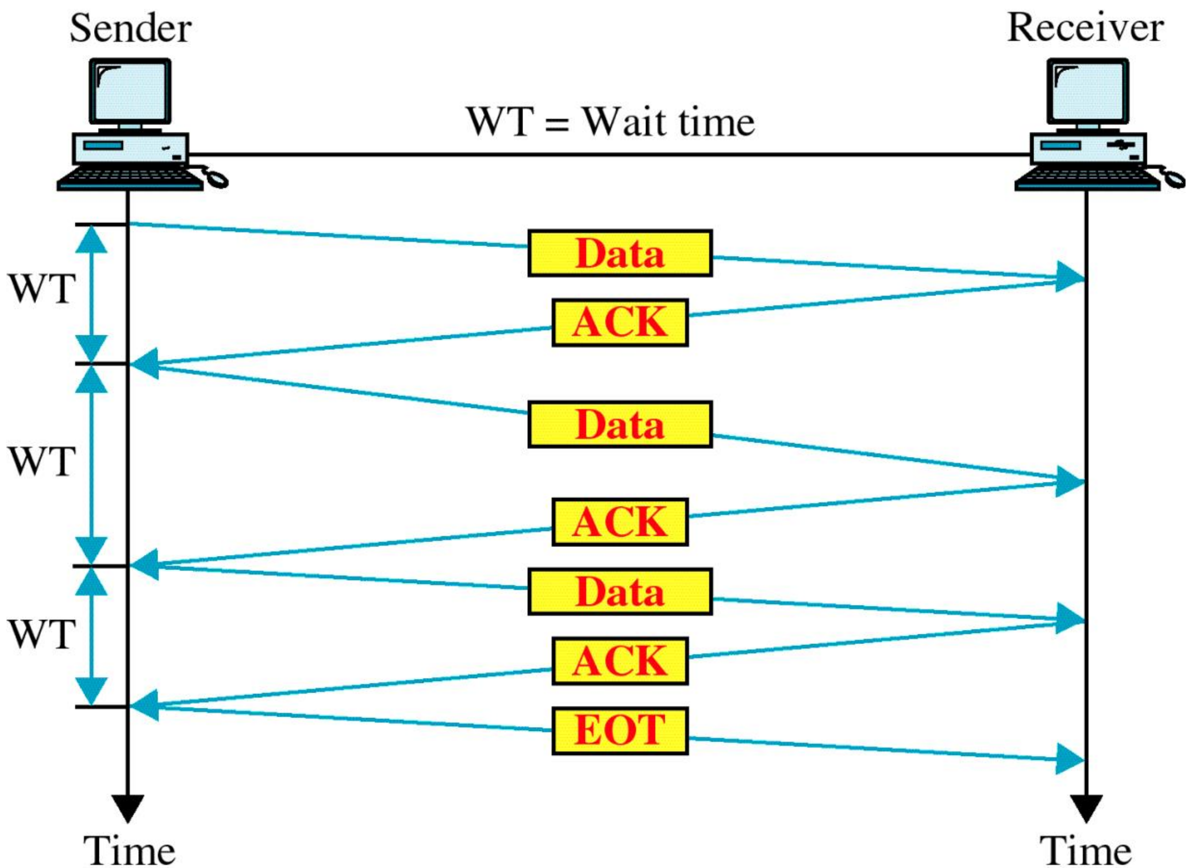
- **flow control** specifies how much data the Sender can transmit before receiving *permission to continue* from the Receiver
- **error control** allows the Receiver to tell the Sender about frames damaged or lost during transmission, and coordinates the *re-transmission* of those frames by the Sender
  - since flow control provides the Receiver's acknowledgement (ACK) of correctly-received frames, it is closely linked to error control

- ***basic idea of flow control:*** even if frames are received error-free, the Receiver will be forced to drop some of them if the Sender transmits faster than the Receiver can process them $\Rightarrow$ **signal** the Sender to slow down to a rate acceptable to the Receiver
  - this signal can be **explicit** or **implicit** (e.g. delay sending ACK to Sender)

- ***basic idea of error control:*** ACK every correctly-received frame and negatively acknowledge (NAK) each incorrectly-received frame
  - Sender keeps copies of un-ACKed Frames to re-transmit if required
  - **want:** packets (inside frames) passed to Receiver's Network layer **in order**

Stop-and-wait flow control

Sender waits for ACK after each frame transmission:



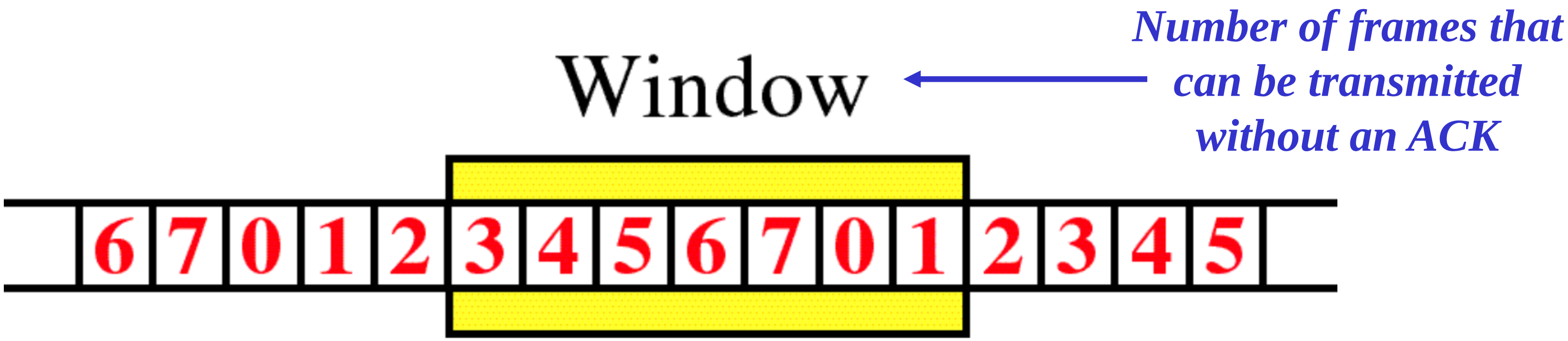
ACK can be a frame by itself, or a control field in data frames going from Receiver to Sender ("piggybacking")

Note: wait times may vary for different frame transmissions, as is the case here ⇒ can talk about <u>average</u> wait time

Advantage: simplicity. Disadvantage: inefficiency (wait times).

Sliding window flow control

- Sender can transmit several frames **continuously** before needing an ACK
- if ACK received by Sender before continuous transmission is finished, Sender can continue transmitting
- an ACK can acknowledge the correct receipt of **multiple** frames at the Receiver
- Frames and ACKs must be numbered:
    - each Frame's number is 1 greater than the previous Frame
    - each ACK's number is the number of the *next Frame expected* by the Receiver

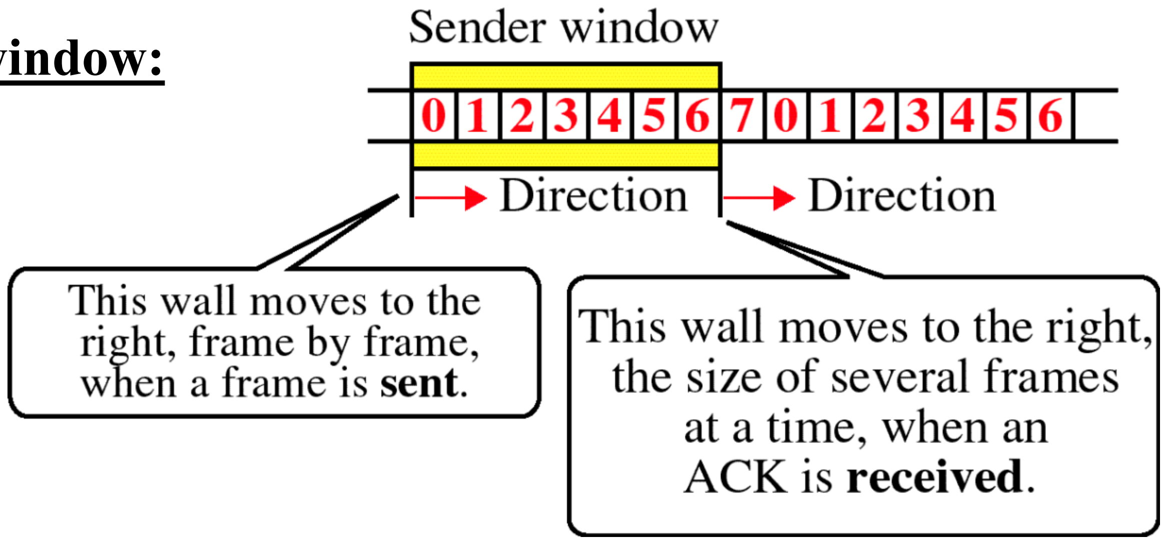


- Frames may be acknowledged by the Receiver at any time, and may be transmitted by the Sender as long as the Window hasn't filled up
- Frames are numbered modulo-n, from 0 to n-1: **0, 1,…, n-1, 0, 1,…,n-1, 0, 1,…**
- Size of the Window is n-1: 1 less than the number of different Frame numbers

Sliding window flow control (cont.)

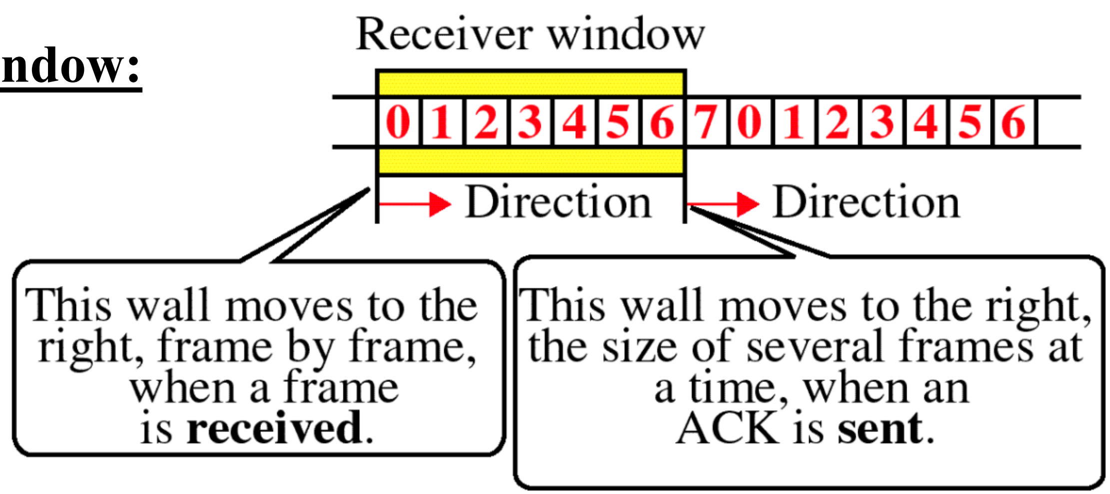
<u>Sender's sliding window:</u>

if Sender receives e.g. ACK 4, then it knows Frames up to and including Frame 3 were correctly received

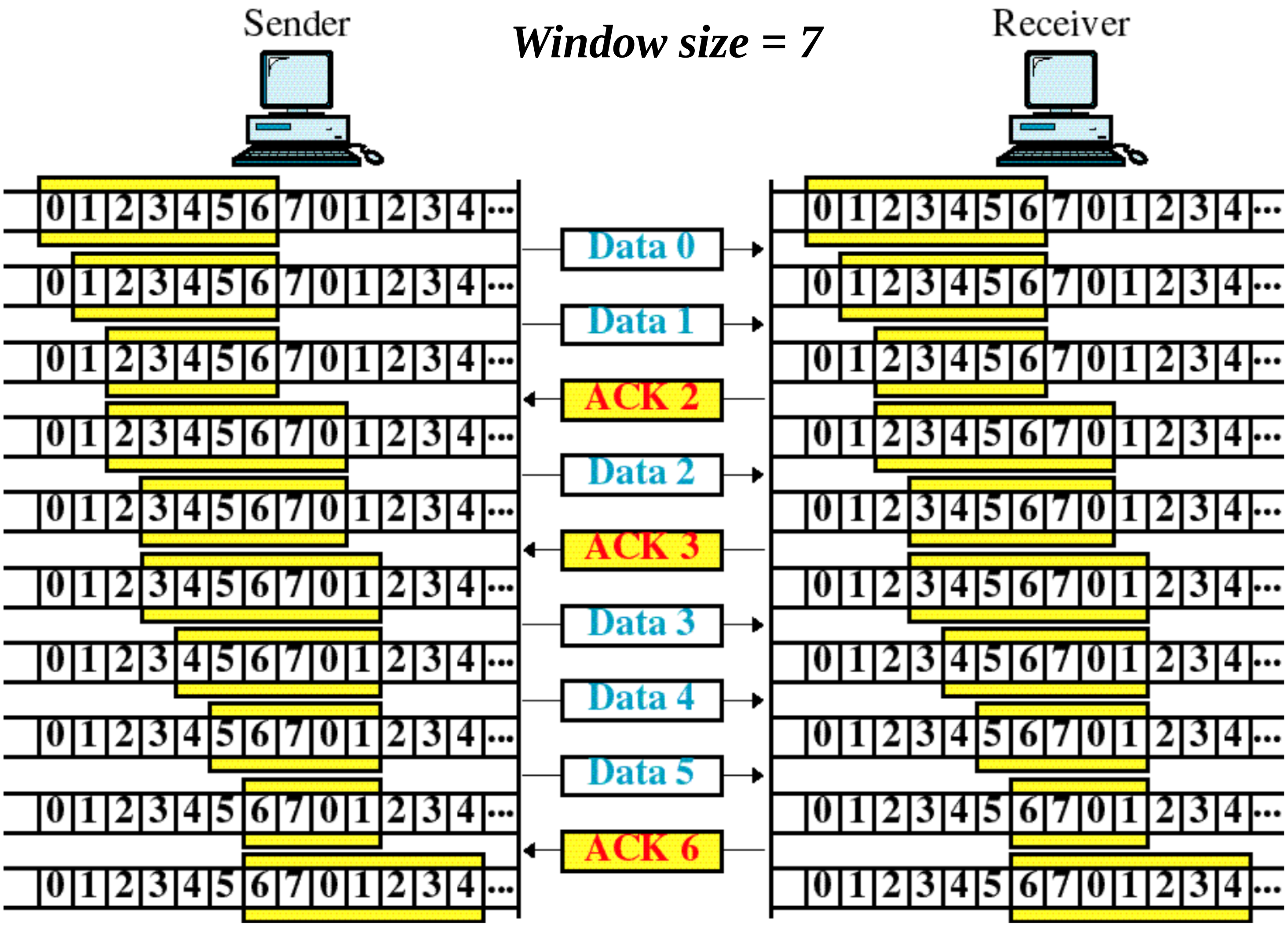


<u>Receiver's sliding window:</u>

Receiver's window represents the number of un-ACKed Frames



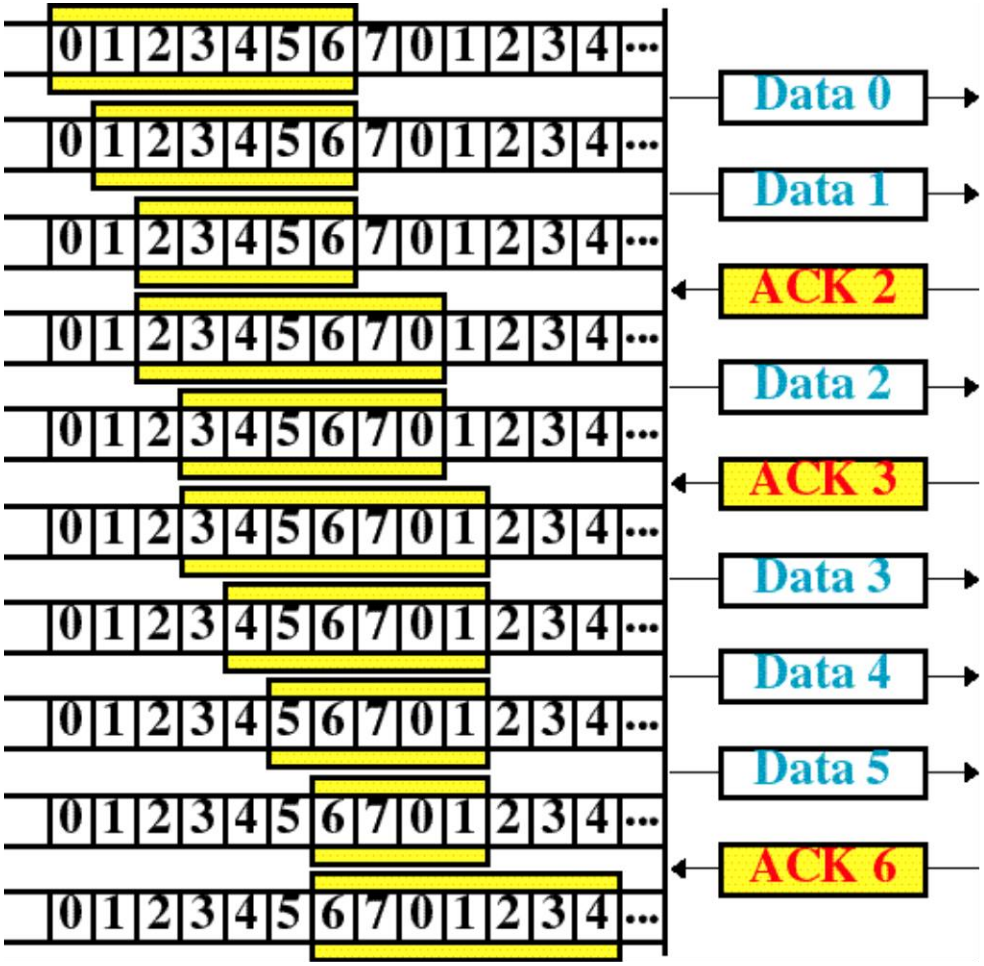
Sliding window flow control: Example *(assume no errors)*



Note: neither Sender nor Receiver has this kind of "global picture"…

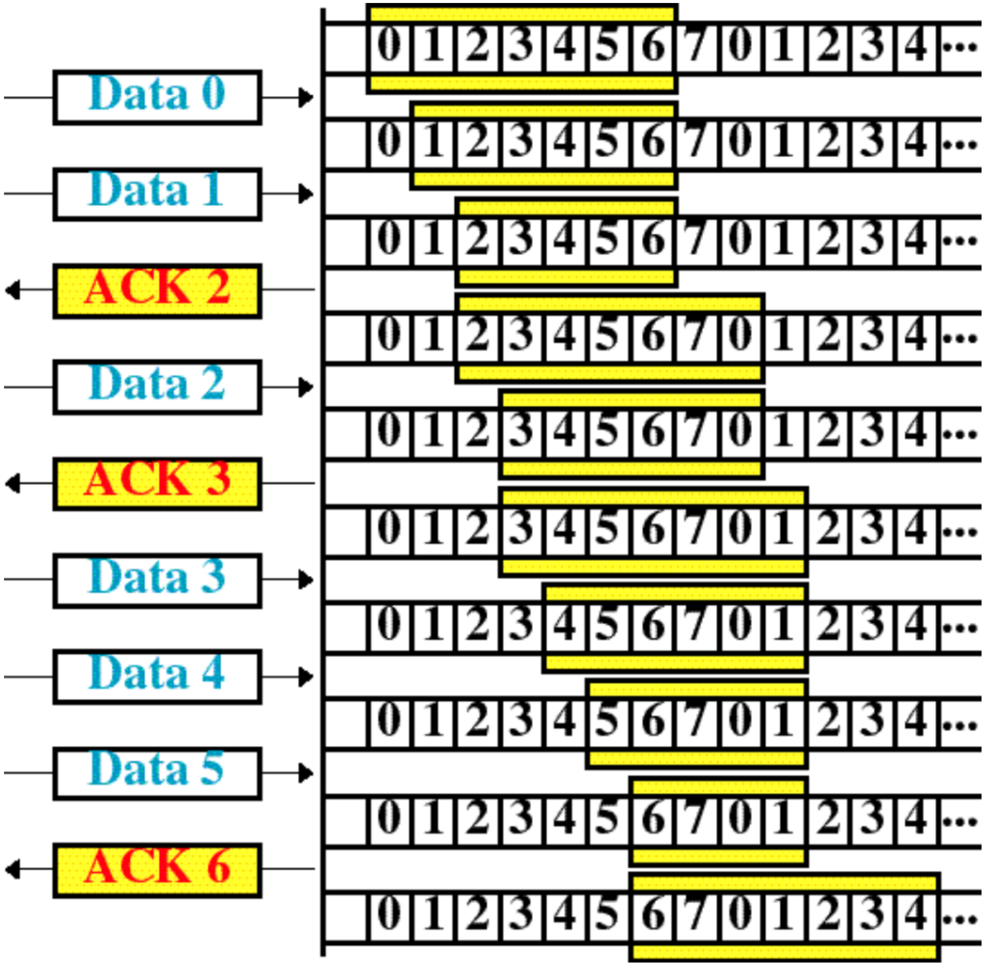
Sliding window flow control: Example (cont.)

Sender behaviour



Sliding window flow control: Example (cont.)

Receiver behaviour



Frame Correct First Time

- Could try and get everything through correctly
- If we knew pb then we could work out a parity correcting scheme to do this.
- Generally have to add substantial overhead even for fairly low Bit Error Rates (BER)
- However for most networks we do not know in advance the properties of the links
- We would also like our scheme to deal well with all types of links as best as is possible.
- Use a different philosophy to do this.

Automatic Repeat Request (ARQ) Schemes

- Send frames and find out is they are in error by using error detection schemes
- Then if correct release to higher layers, if in error then get the transmitter to retransmit it again.
- The most common error detection schemes are based on the use of Polynomial Codes, which involves the computation of Cyclic Redundancy Check, CRC.
- This will tell is the frame is good or not, at least to many more orders of reliability than the frame error probability.

Number Everything

- Have to number the frame that is sent.
- If the frame CRC is received okay then send an Acknowledgement (ACK) back to the transmitter.
- If the CRC has a remainder then we can send back to the transmitter a Negative ACK or NACK.
- If we miss a frame then the timer kicks in.
- The sequence numbers are put in the control field which also tells us which type of frame we have.
- N(S) sent frame number
- N(R) next requested frame

Mathematical Definitions

Time to transmit a frame is TRANSF ,where

TRANSF = length of the frame in bits / bit rate of the link.

An ACK is expected later, where

Ts = TRANSF+ time for propagation + receiver processing + reverse transmit + propagation + transmitter processing

The hope is that before we run out of numbers to put in the frames we will have an ACK.

The size of the link can be given by the ratio of round trip time to TRANSF which is denoted by a

a = (Ts)/ TRANSF

Examples of delays

Terrestrial low speed link

- If the frame size is 1200 bits, link speed is 9.6 kb/s, frame length TRANSF = 125 ms. If link length is 100 km, speed of light is 3E+8 m/s, propagation time is 0.33 ms, if processing time is small, but the ACK comes back in another frame, a = 2

Terrestrial high speed link

- A high speed ATM link with link speed of 620 Mb/s, the frame length, TRANSF = 0.7 us. If same length of cable, a > > 1

Satellite low speed link

- Satellites are generally about 36,000 km's above earth. Propagation delay is 120 ms up and the same down, a = 6

Error Control: ARQ (Automatic Repeat Request) schemes

- if error(s) detected in received Frame, return NAK to Sender
  - NAK can be **explicit** or **implicit** (Sender's Timeout timer expires)

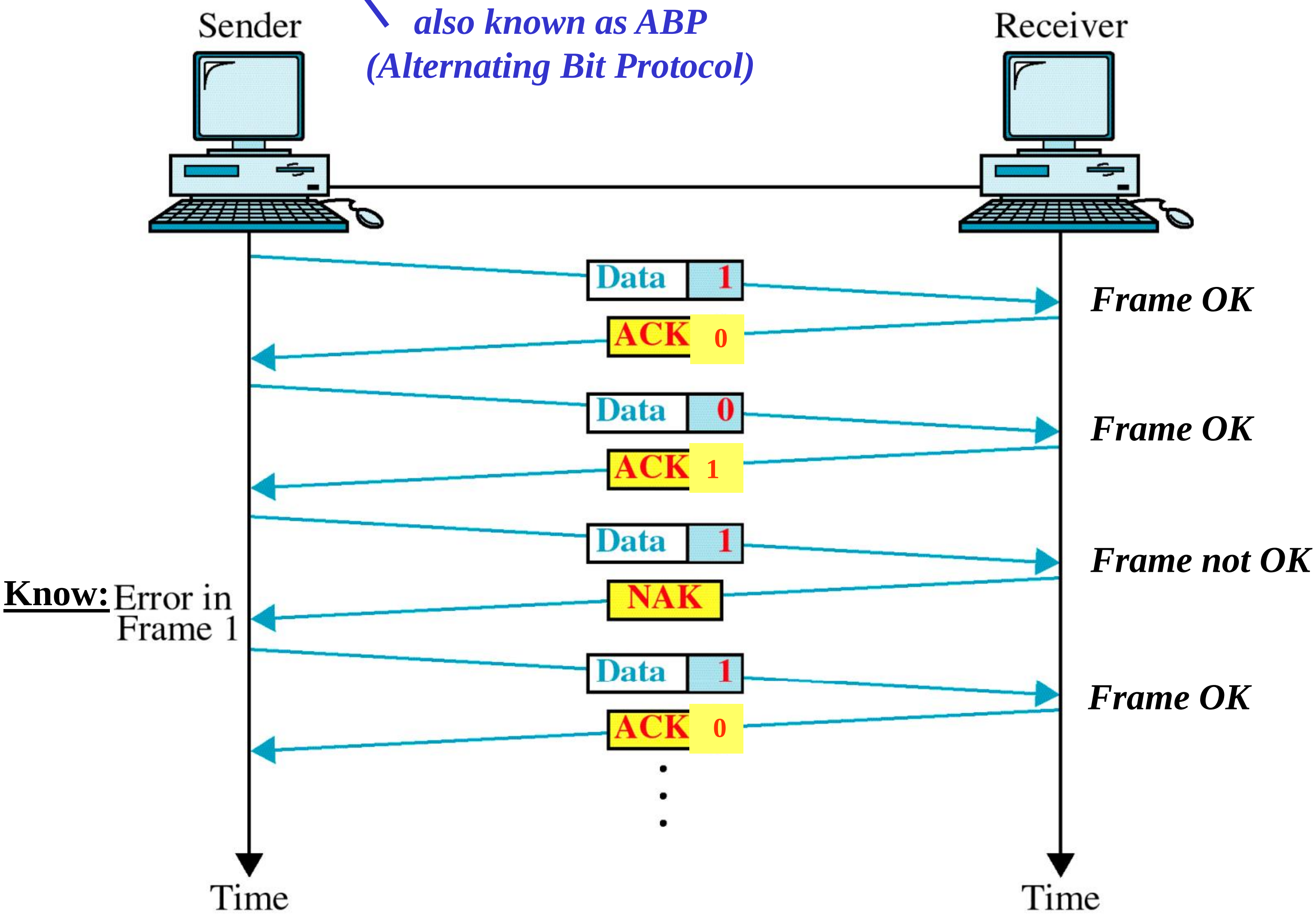
- Sender keeps a copy of each un-ACKed Frame to re-transmit if required
  - ACK received by Sender for Frame $\Rightarrow$ discard copy
  - NAK received by Sender for Frame $\Rightarrow$ decide how to re-transmit Frame

- Sender starts Timeout timer for each Frame when it is transmitted
  - appropriate Timeout value = the expected delay for Sender to receive ACK for the Frame (in practice, set Timeout slightly larger than this…)
  - packet is not considered to be delivered successfully to the Receiver's Network layer until the Sender knows this (by getting ACK for it)

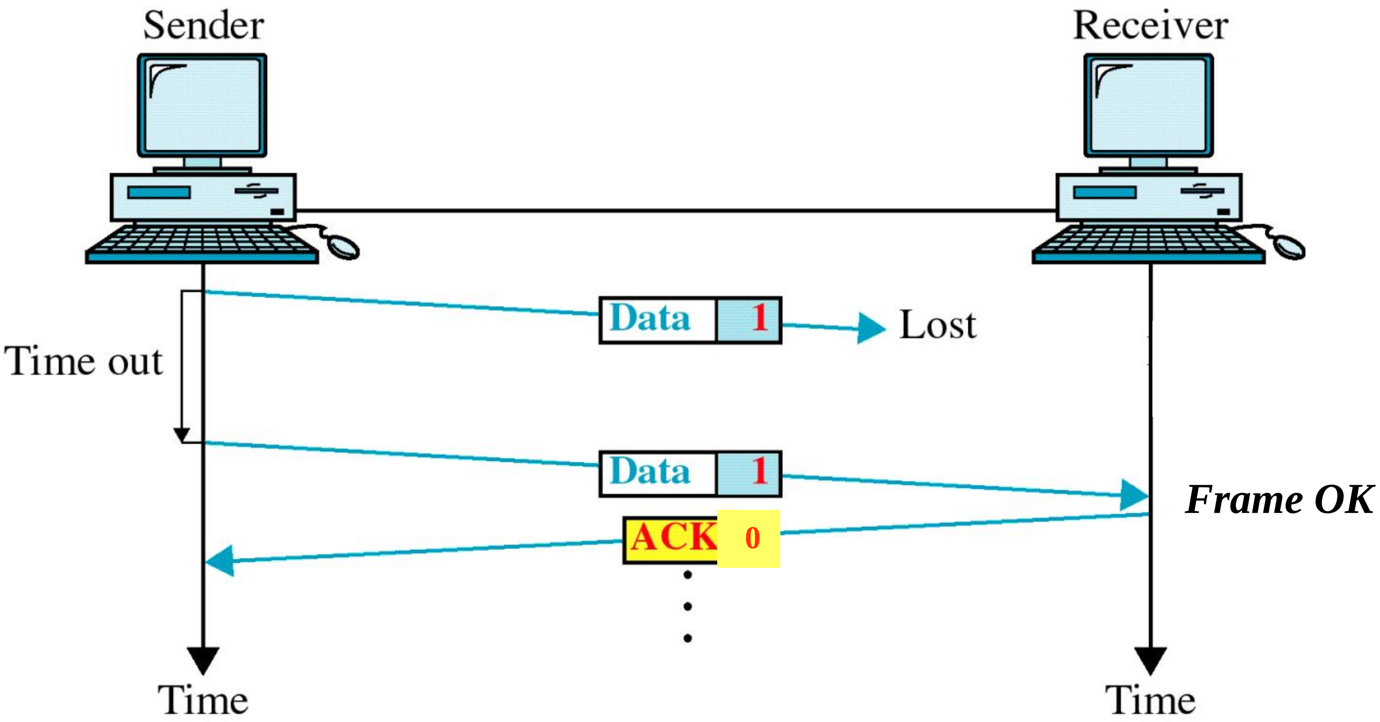
- 3 types of ARQ scheme:
  - ***Stop-and-wait ARQ*** – extension of Stop-and-wait flow control
  - Sliding window ARQ – extension of sliding window flow control:
    - ***Go-back-n ARQ*** – Receiver must get Frames in correct order
    - ***Selective repeat ARQ*** – correctly-received out-of-order Frames are stored at Receiver until they can be re-assembled into correct order

Stop-and-wait ARQ, damaged data frame

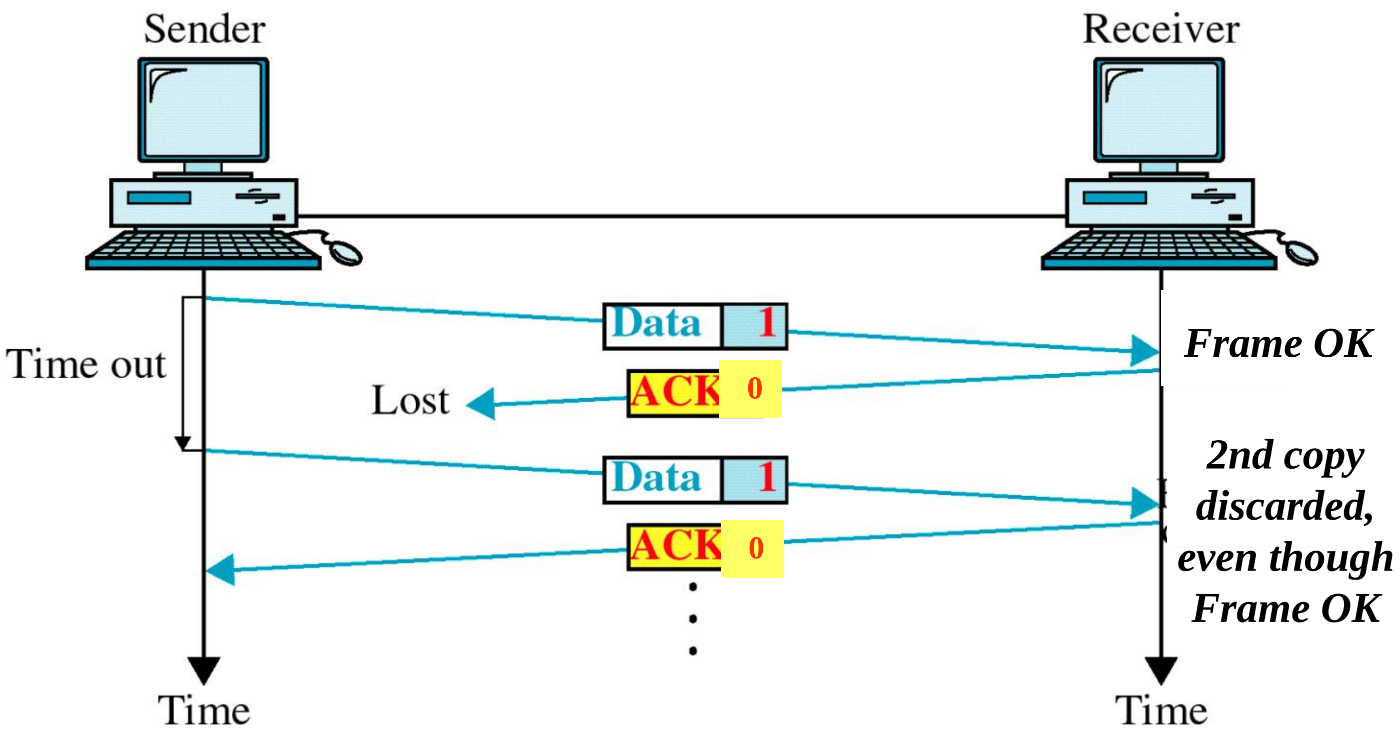
also known as ABP (Alternating Bit Protocol)



Stop-and-wait ARQ, lost data frame



Stop-and-wait ARQ, lost ACK



Q: why does Receiver send ACK 0 for copy of data Frame 1 ?

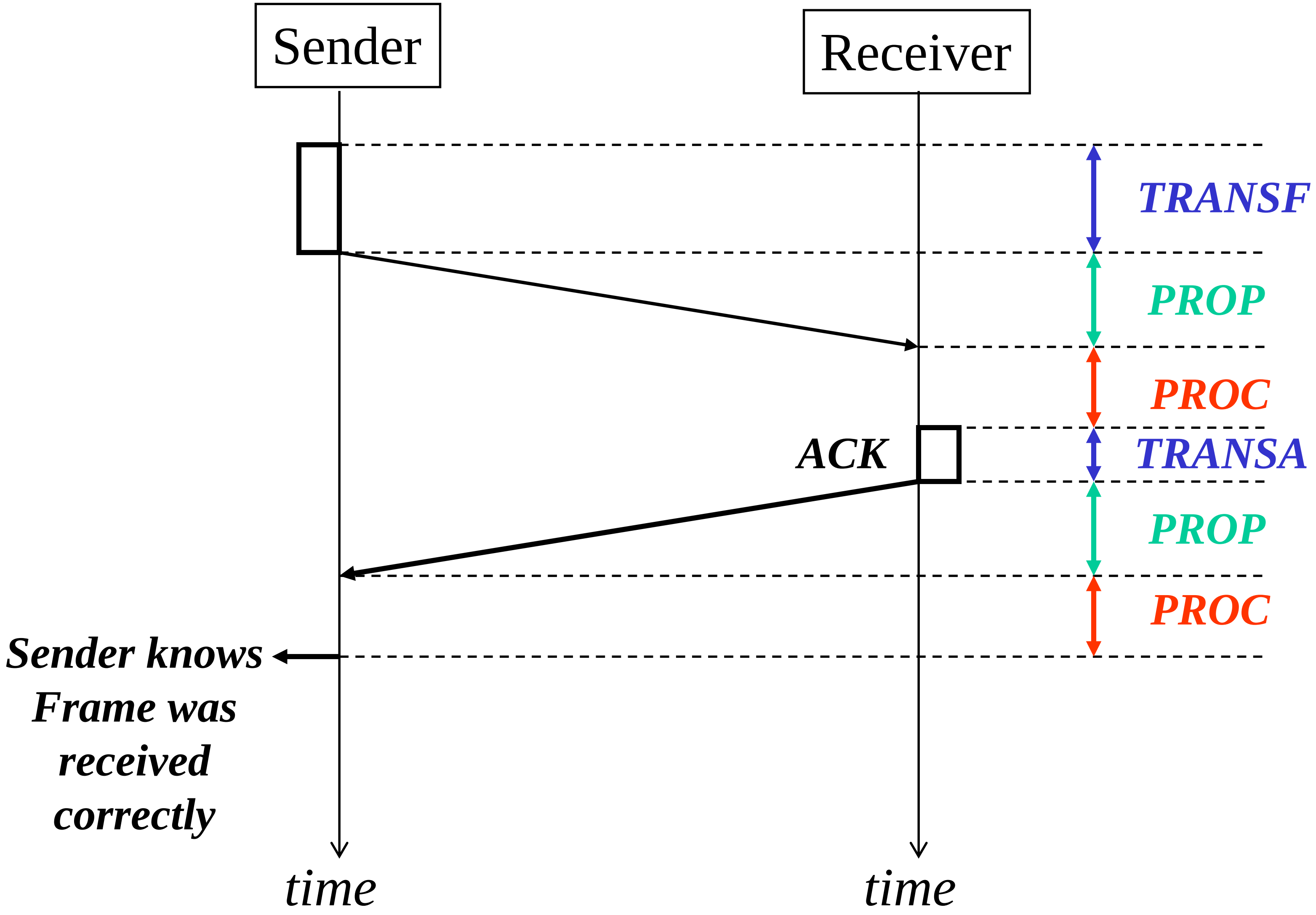
Stop-and-wait ARQ: Performance Analysis

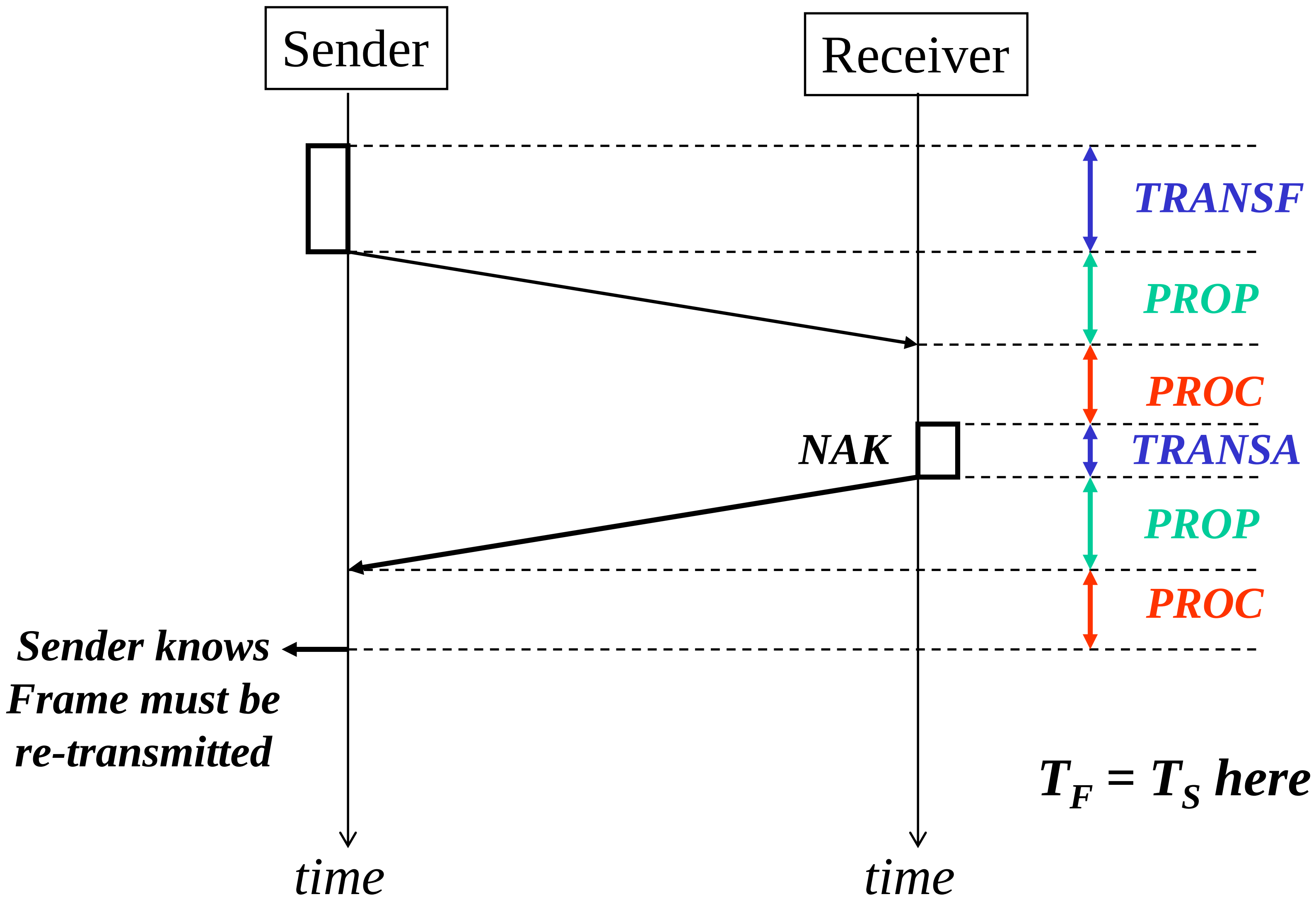
- parameters:
  - Frame transmission time at Sender is ***TRANSF***
  - ACK/NAK transmission time at Receiver is ***TRANSA***
    - if ACKs are piggybacked ⇒ ***TRANSF = TRANSA***, assuming Frames have equal (average) lengths in both directions
  - link propagation delay is ***PROP***
  - Frame processing time at Sender or Receiver is ***PROC***
  - probability of Frame error in Sender-Receiver direction is ***p***
  - probability of Frame error in Receiver-Sender direction is ***q***
  - Sender times out after ***TIMEOUT***
- assume Sender has an endless supply of Network layer packets to transmit, so idle time at Sender is 0

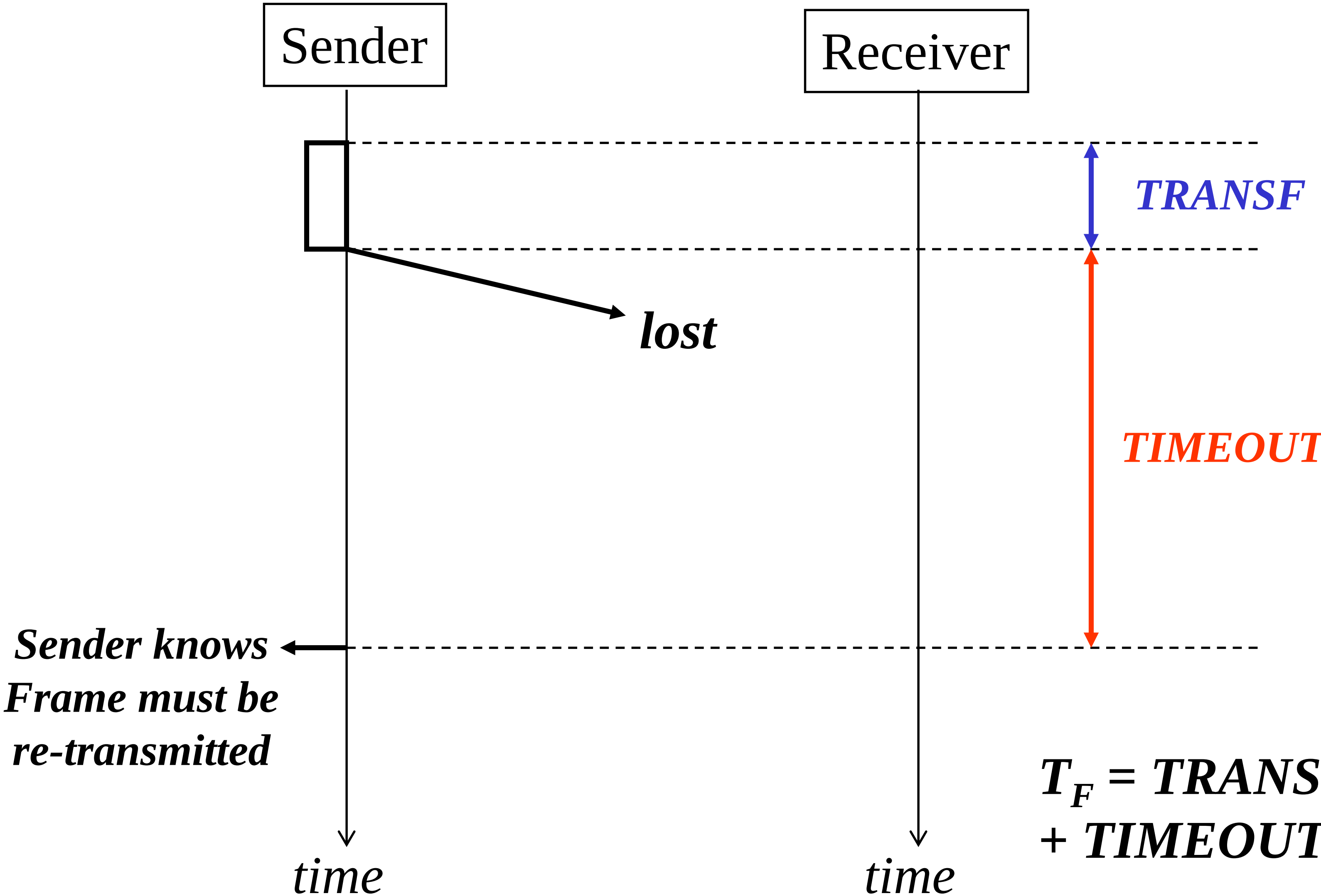
- error-free packet delivery takes T_S = ***TRANSF + TRANSA + 2*(PROP + PROC)*** and occurs with probability ***(1-p)*(1-q)***

- errored delivery takes T_F and occurs with probability ***r = p+(1-p)*q***

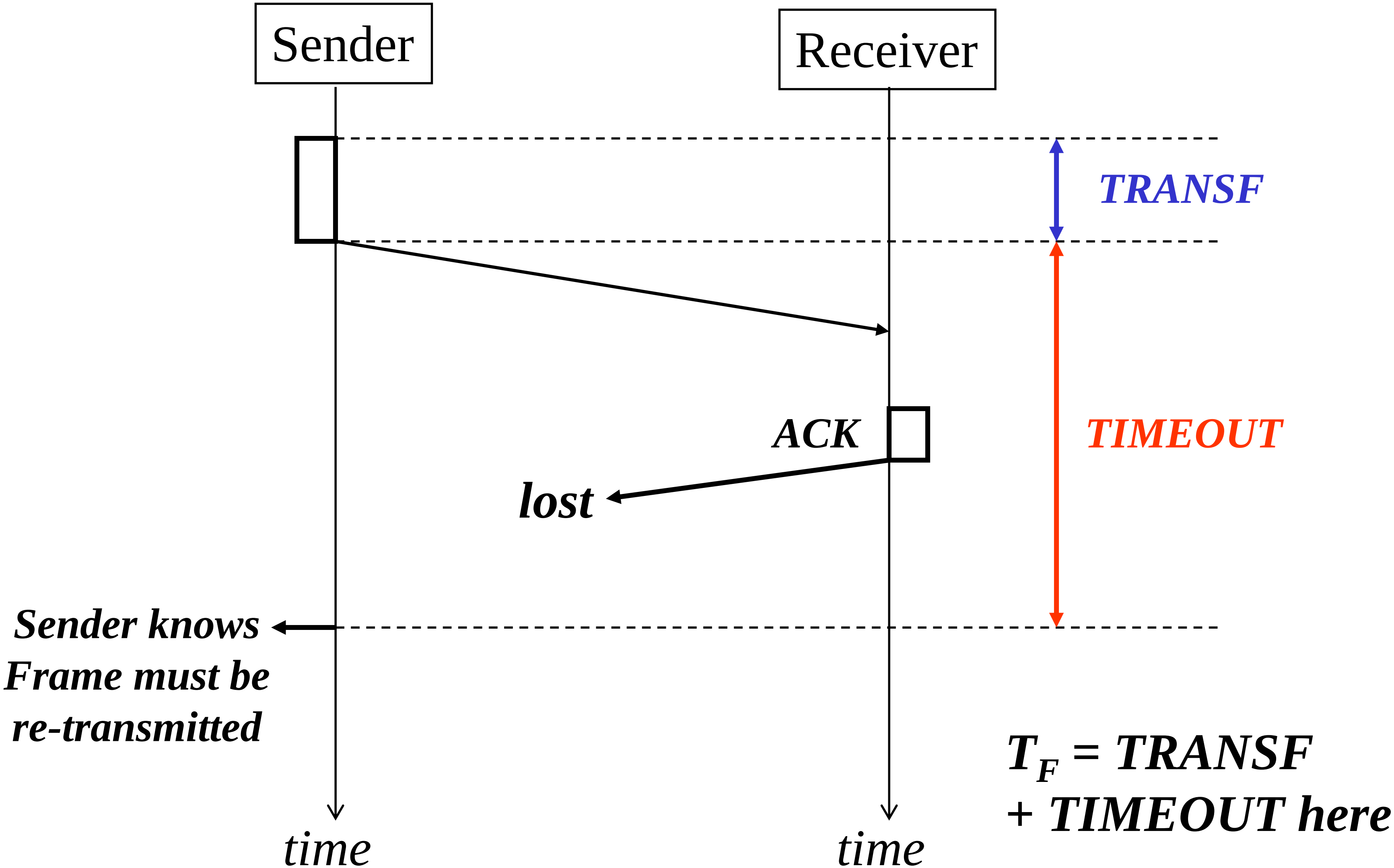
r = 1-(probability of error-free delivery), as expected


Sender
Receiver
TRANSF
PROP
PROC
ACK
TRANSA
PROP
PROC
Sender knows Frame was received correctly
time
time


Sender
Receiver
TRANSF
PROP
PROC
NAK
TRANSA
PROP
PROC
Sender knows
Frame must be
re-transmitted
$T_F = T_S$ here
time
time



T_F = TRANSF
+ TIMEOUT here


Sender
Receiver
TRANSF
TIMEOUT
ACK
lost
Sender knows
Frame must be
re-transmitted
time
time
T_F = TRANSF
+ TIMEOUT here

Stop-and-wait ARQ: Performance Analysis (cont.)

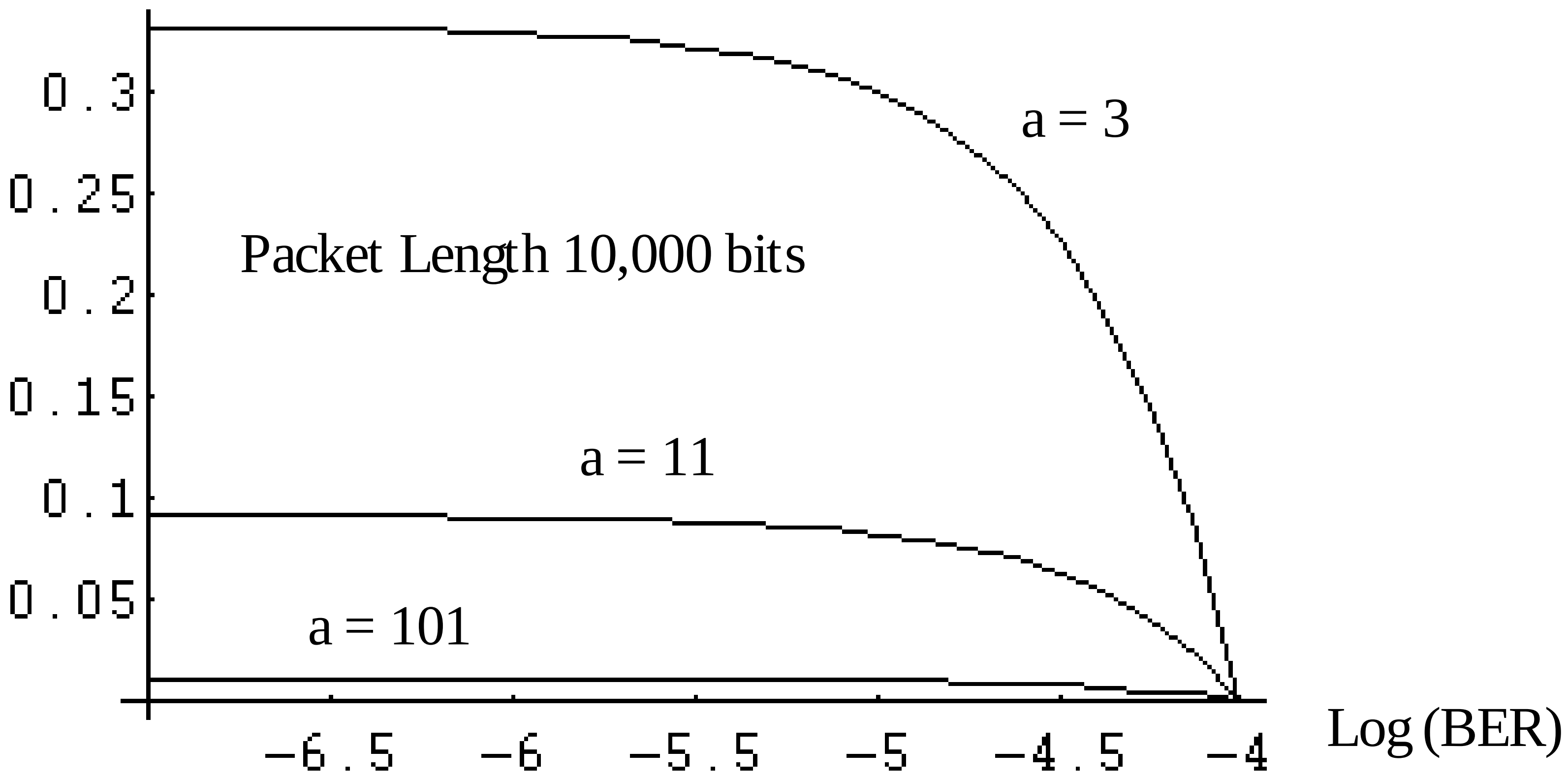
- average number of Frame transmissions to successfully deliver 1 packet to Receiver is ***E = 1/(probability of error-free delivery) = 1/[(1-p)*(1-q)] = 1/[1-r]***
- therefore average packet delay in ARQ scheme is $\boldsymbol{D = (E-1)*T_F + T_S}$
- therefore ***average packet throughput = 1/D*** and ***efficiency = TRANSF/D***
  - remember: efficiency is the fraction of the time <u>new</u> packets are delivered

- **<u>special cases:</u>**
  - symmetrical case: $\boldsymbol{p = q} \Rightarrow$ in this case, $\boldsymbol{E = 1/(1-p)^2}$
  - no errors in ACKs/NAKs: $\boldsymbol{q = 0} \Rightarrow$ in this case, $\boldsymbol{r = p}$ and $\boldsymbol{E = 1/(1-p)}$
  - optimal choice of ***TIMEOUT = TRANSA + 2*(PROP + PROC)*** $\Rightarrow$ in this case, $\boldsymbol{T_S = T_F}$ and therefore $\boldsymbol{D = E*T_S}$
  - Sender and Receiver processing time negligible: ***PROC = 0***

Throughput versus BER for Stop and Wait

- no errors in ACKs/NAKs: $q = 0 \Rightarrow$ in this case, $r = p$ and $E = 1/(1-p)$
- and optimal choice of ***TIMEOUT = TRANSA + 2*(PROP + PROC)*** $\Rightarrow$ in this case, $T_S = T_F$ and therefore $D = E^*T_S$

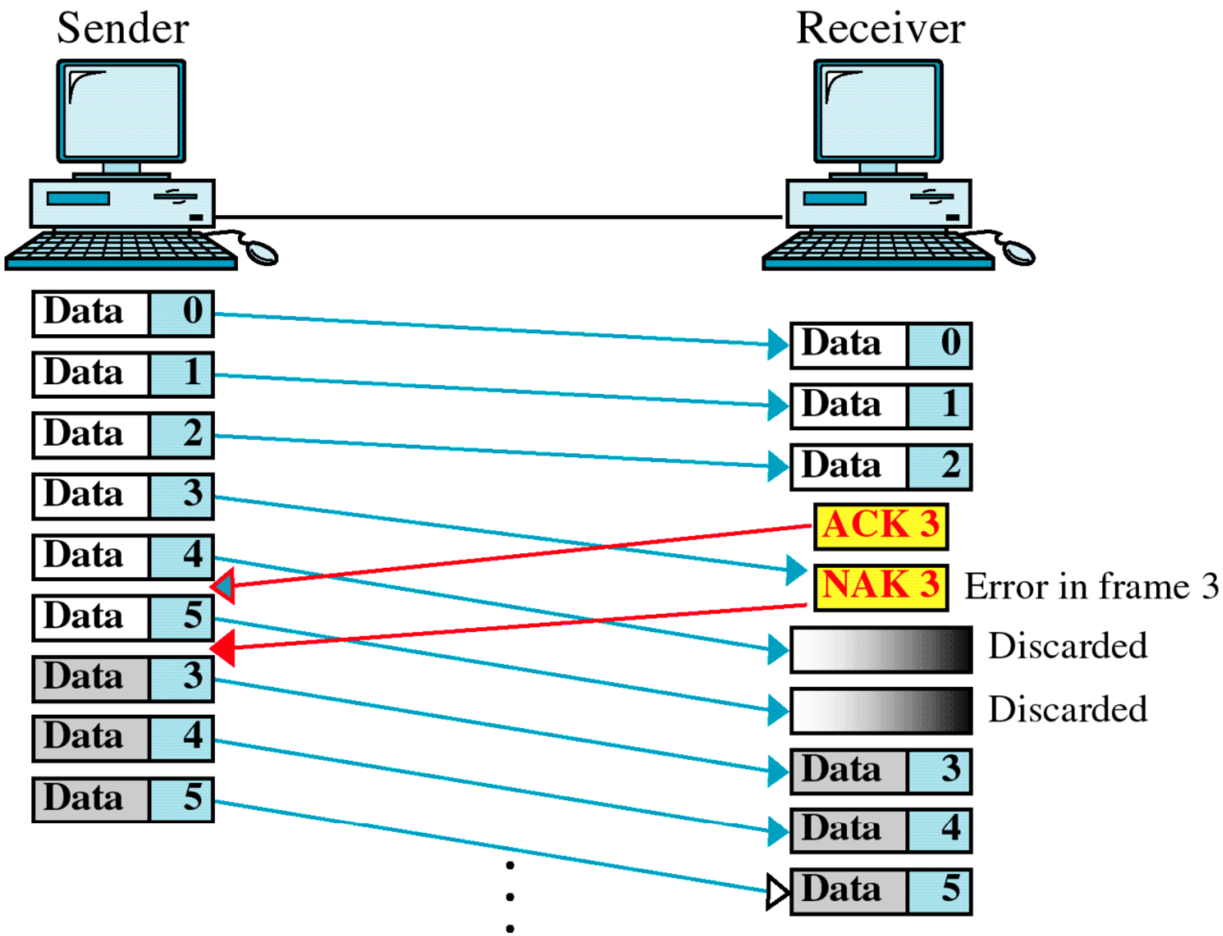
$$D = T_S/(1-p)$$



Go-Back-N ARQ Scheme

- Transmit frames continuously if possible.
- N outstanding frames at most on the link.
- As we are using a Modulo-N numbering scheme we need a big enough N(S) and N(R) to achieve good performance.
- It is possible to have sequence number starvation where you run out of numbers.
- If a frame is received with a remainder, or if N(S) skips a number, then we have an error, so roll back the clock and retransmit.

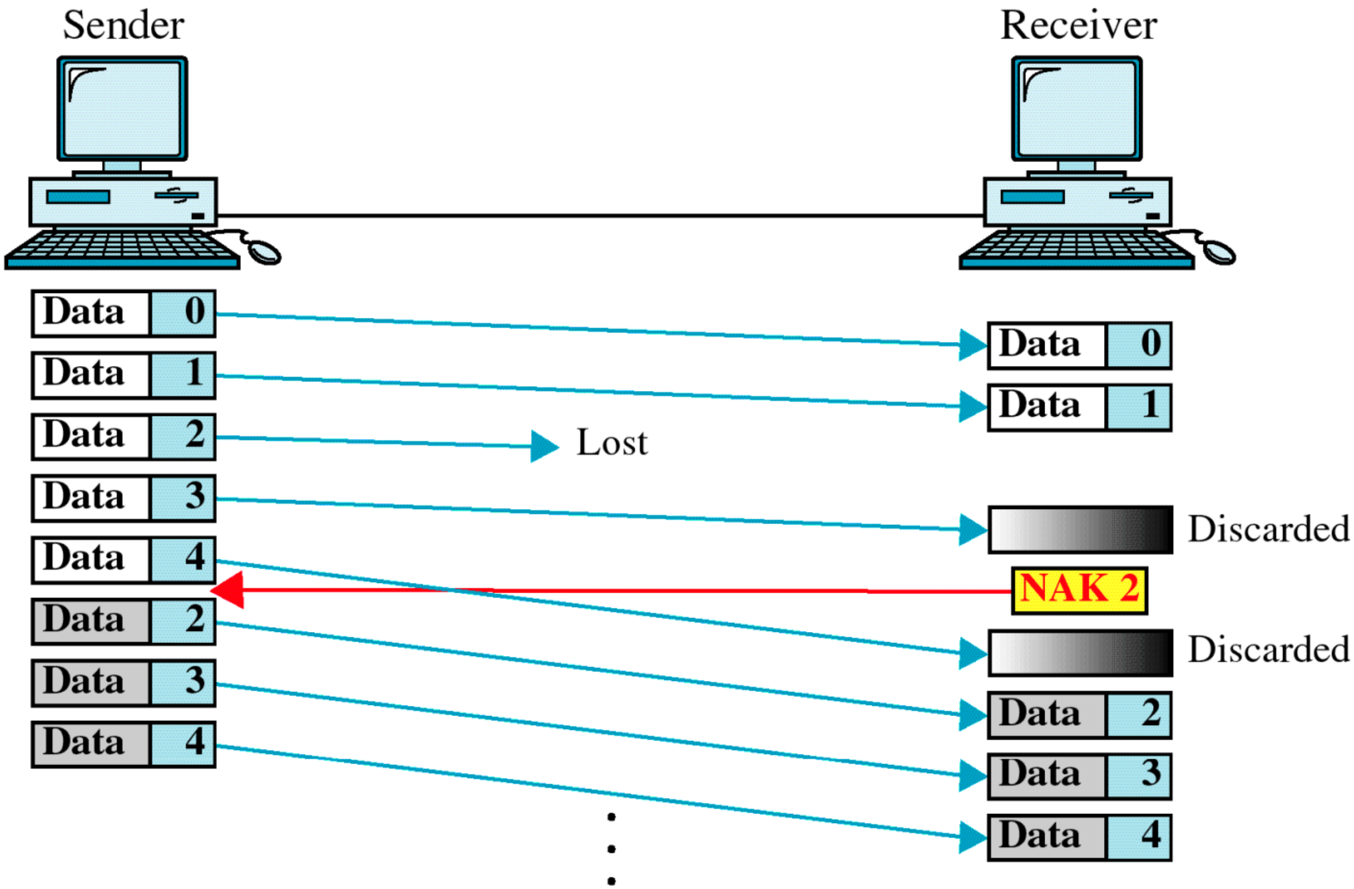
Go-back-n ARQ, damaged data frame



NAK 3 means re-transmit Frame 3 (also tells Sender that all Frames before Frame 3 were received correctly)

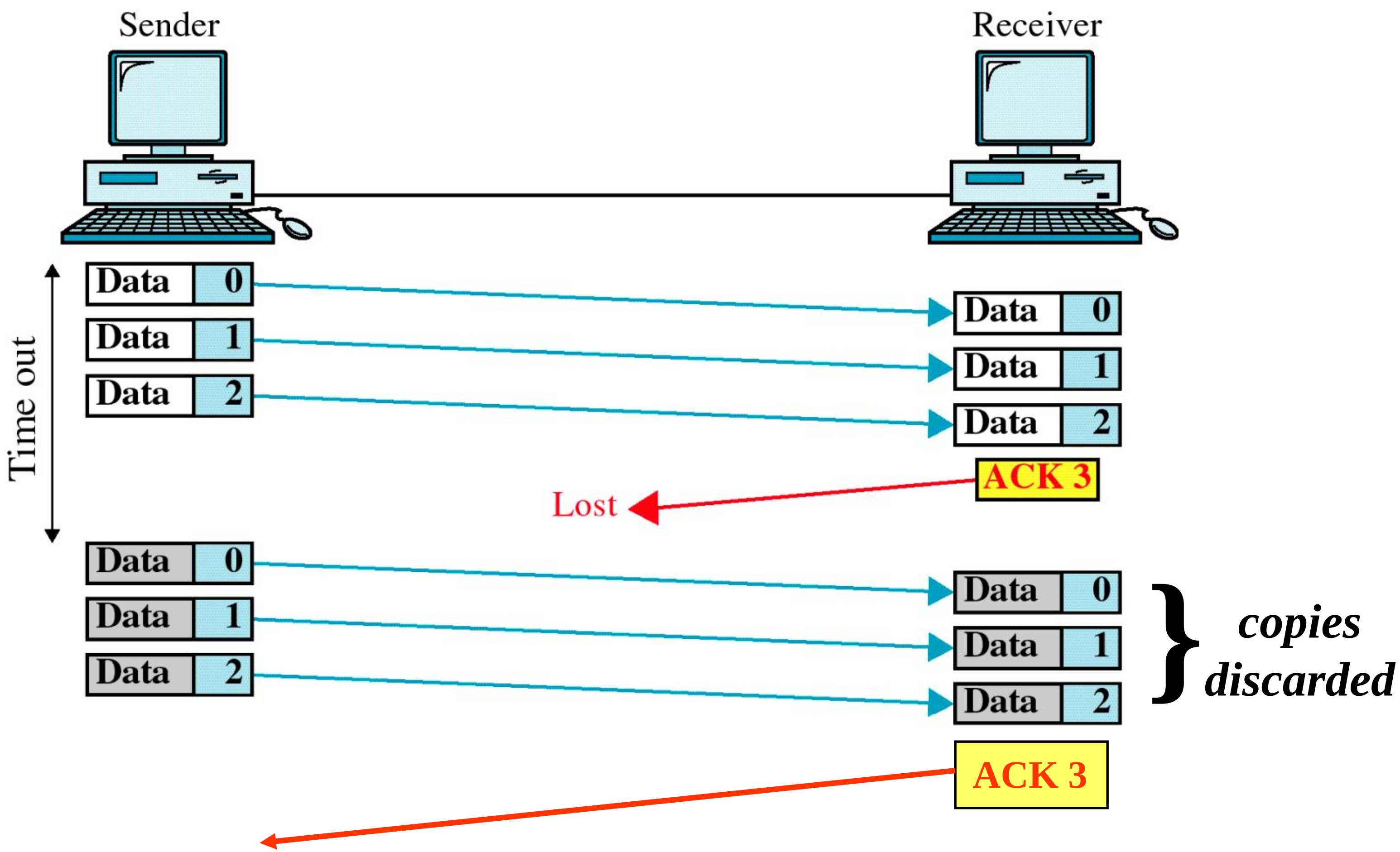
Receiver only accepts correctly-received Frames in the correct order (so Receiver doesn't have to buffer any Frames and re-order them…)

Go-back-n ARQ, lost data frame



Frame 3 discarded even though it was correctly received BECAUSE Receiver was expecting Frame 2 (same for Frame 4)

Go-back-n ARQ, lost ACK



Go-back-n ARQ: Performance Analysis

• parameters: same as before
• assume optimum choice of ***TIMEOUT = TRANSA + 2*(PROP + PROC)***
• assume Window is large enough that Sender can transmit continuously if there are no transmission errors

• can show average packet delay is ***D = [TRANSF + r*TIMEOUT] / (1-r)*** where ***r = p+(1-p)*q*** as before

• therefore ***average packet throughput = (1-r) / [TRANSF + r*TIMEOUT]*** and ***efficiency = [(1-r)*TRANSF] / [TRANSF + r*TIMEOUT]***

• note that as r $\rightarrow$ 0, efficiency of Go-back-n $\rightarrow$ 1 (or 100%): this shows that Go-back-n is capable of continuously delivering packets in the absence of errors

  • under these conditions, you can show that the efficiency of Stop-and-wait ARQ is ***[(1-r)*TRANSF] / [TRANSF + TIMEOUT]***

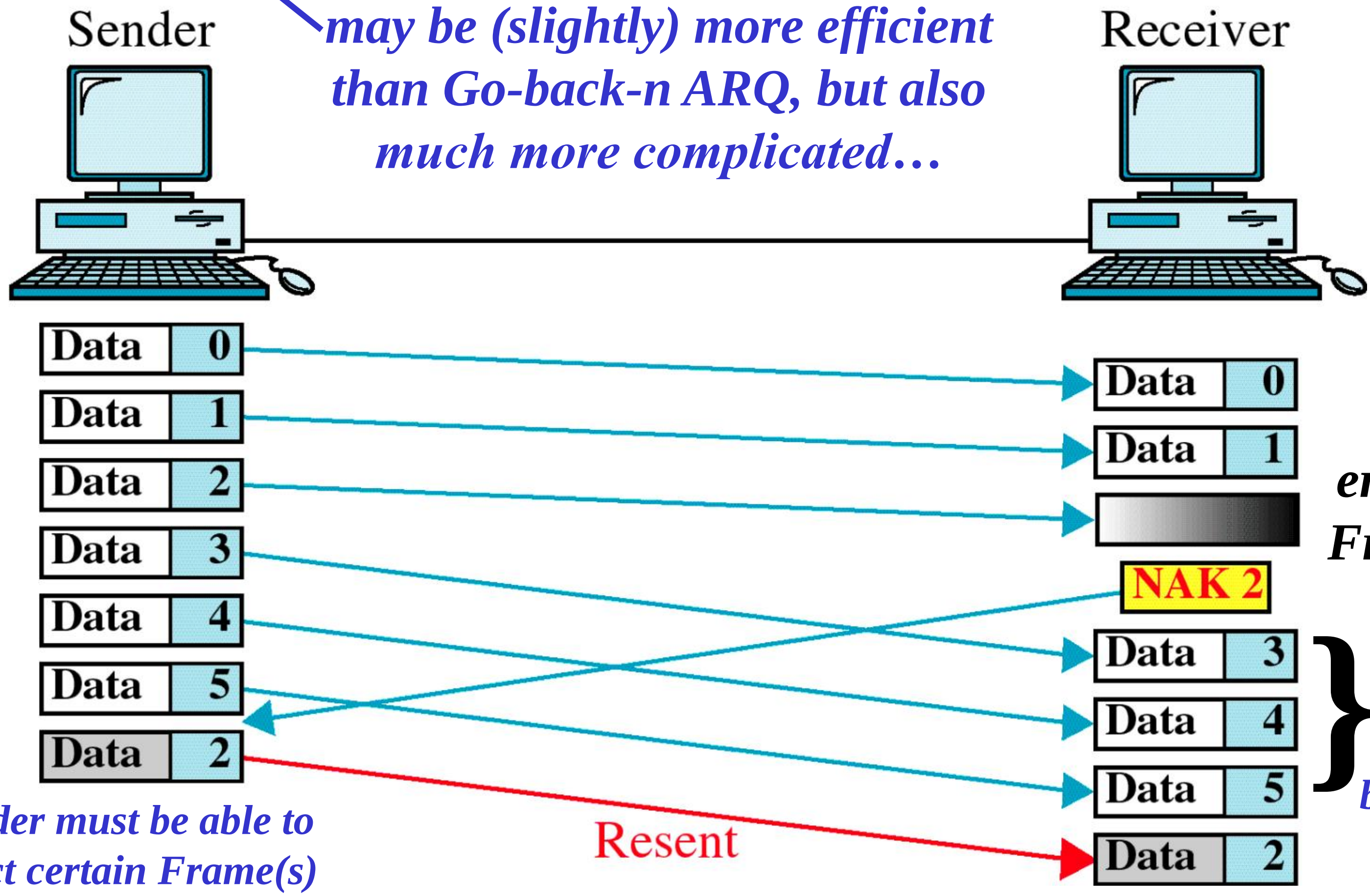
  • Stop-and-wait ARQ efficiency $\rightarrow$ ***TRANSF / [TRANSF + TIMEOUT]*** as r $\rightarrow$ 0: this shows the *built-in inefficiency* of the Stop-and-wait approach

Selective Repeat

- Why go back and retransmit all the frames? Some might be good
- Only retransmit the bad frames!
- Improvement over Go Back N
- Attains the theoretical maximum throughput
- Out of order frames so must reorder them
- More complex transmitter and receiver so only use when needed

Selective repeat ARQ, damaged data frame

may be (slightly) more efficient than Go-back-n ARQ, but also much more complicated…



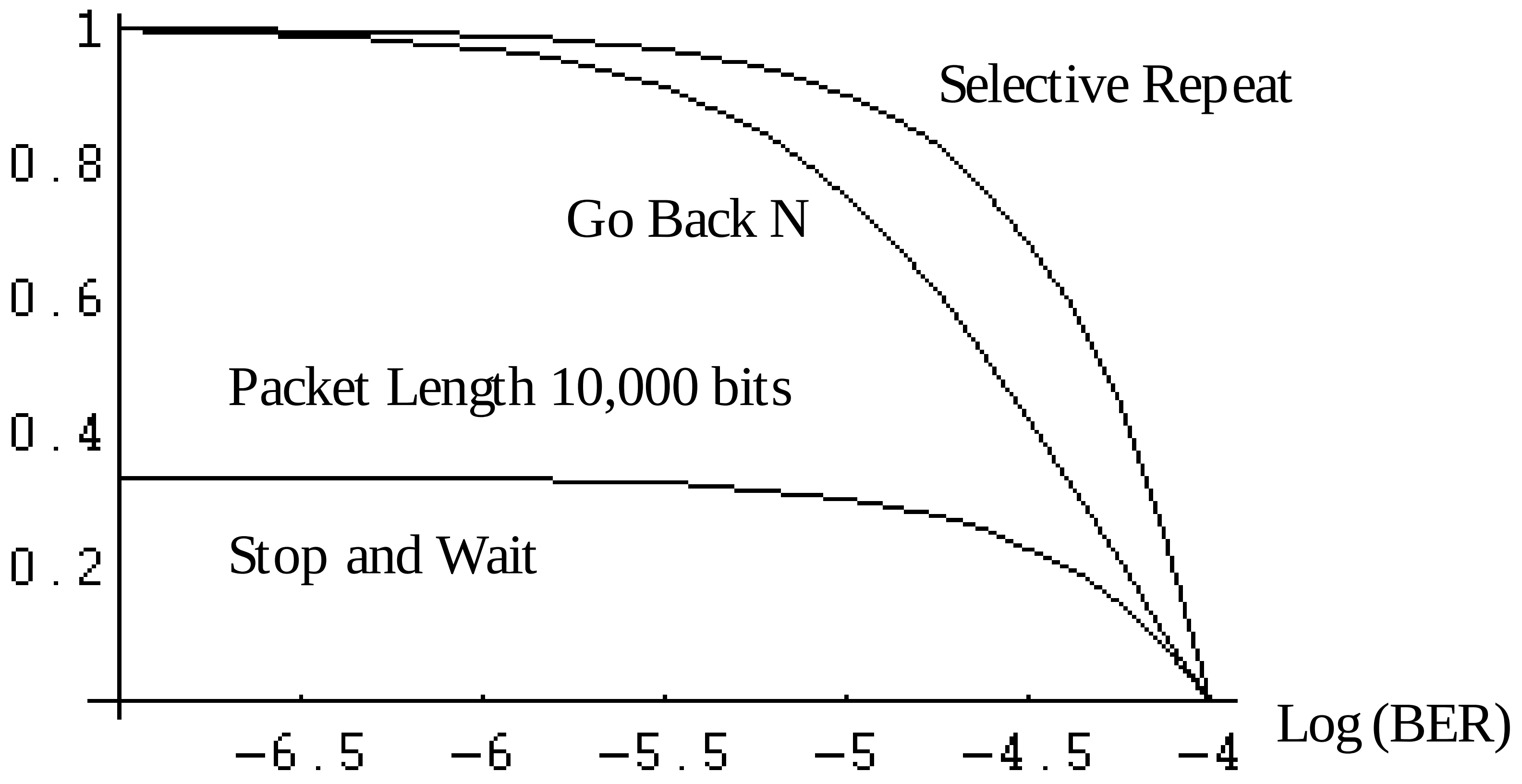
error in Frame 2

stored at Receiver but cannot be ACKed (yet)

Sender must be able to select certain Frame(s) for re-transmission

When frame 2 received correctly, Receiver can deliver the packets in Frames 2-5 to its Network layer and send ACK 6 back to Sender. Lost or damaged ACK/NAK handled similarly to Go-back-n ARQ.

Comparison of Throughput: Stop & Wait, Go Back N, Selective Repeat



Optimum Frame Size

- Classic tradeoff for frame size
- Small frames give low frame error probability, but give high overhead
- Large frames give high frame error probability, but give low overhead
- In between have an optimum frame size for each application.

Mathematical Analysis

- If there are l bits of information in the frame and l' bits of overhead.

$$p = 1 - (1 - pb)^{(l + l')}$$

- The maximum throughput for the Go-Back-N is:

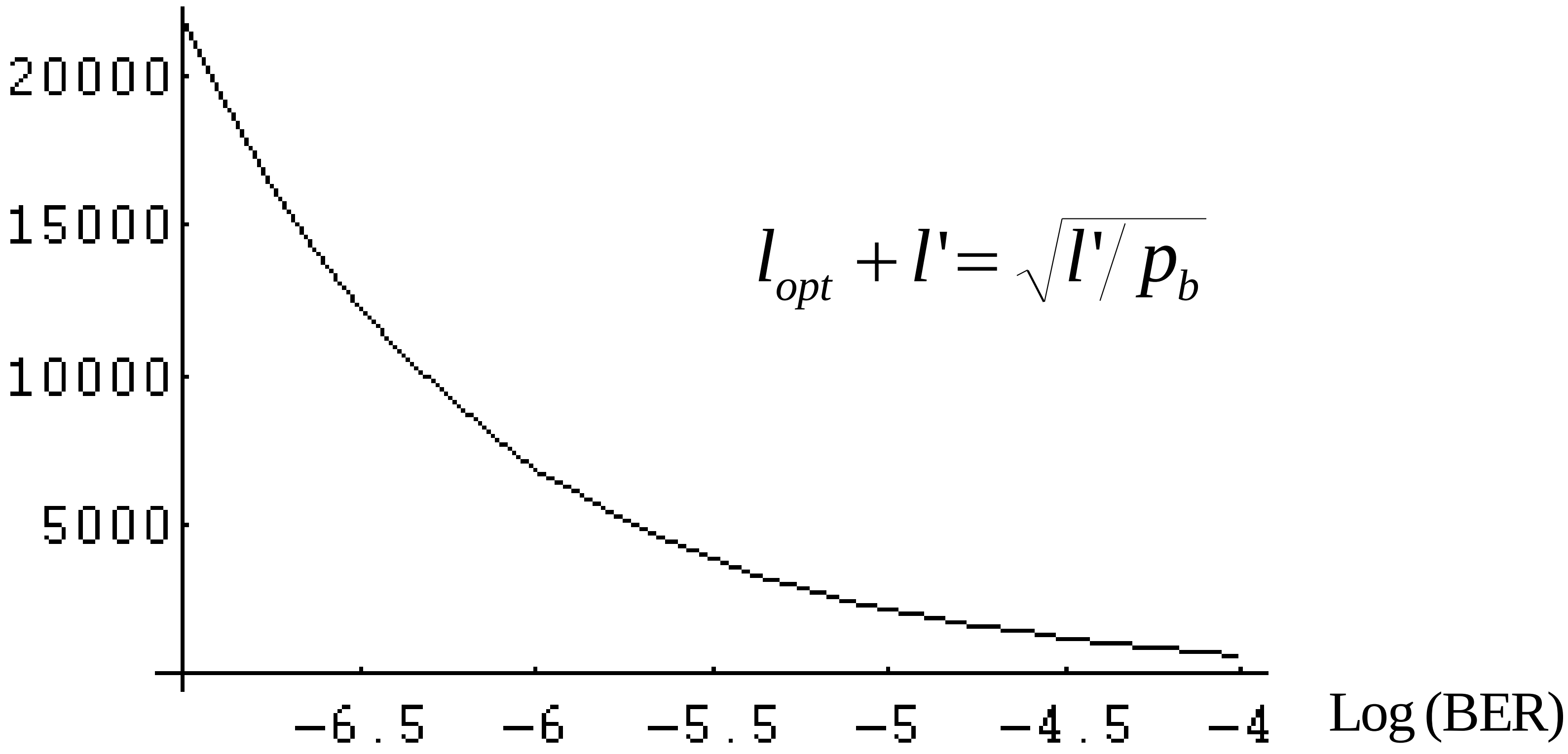
$$= (1 - p) / [1 + (a - 1) p]$$

- Of this only l / (l + l') bits are actually information:

$$= (1 - p) / [1 + (a - 1) p] . l / (l + l')$$

- Find the optimum frame size, l opt, by differentiating this w.r.t. l and solve. Will depend on header length, l', & bit error rate, pb .

Optimum Frame Length v's BER for 6 Byte header

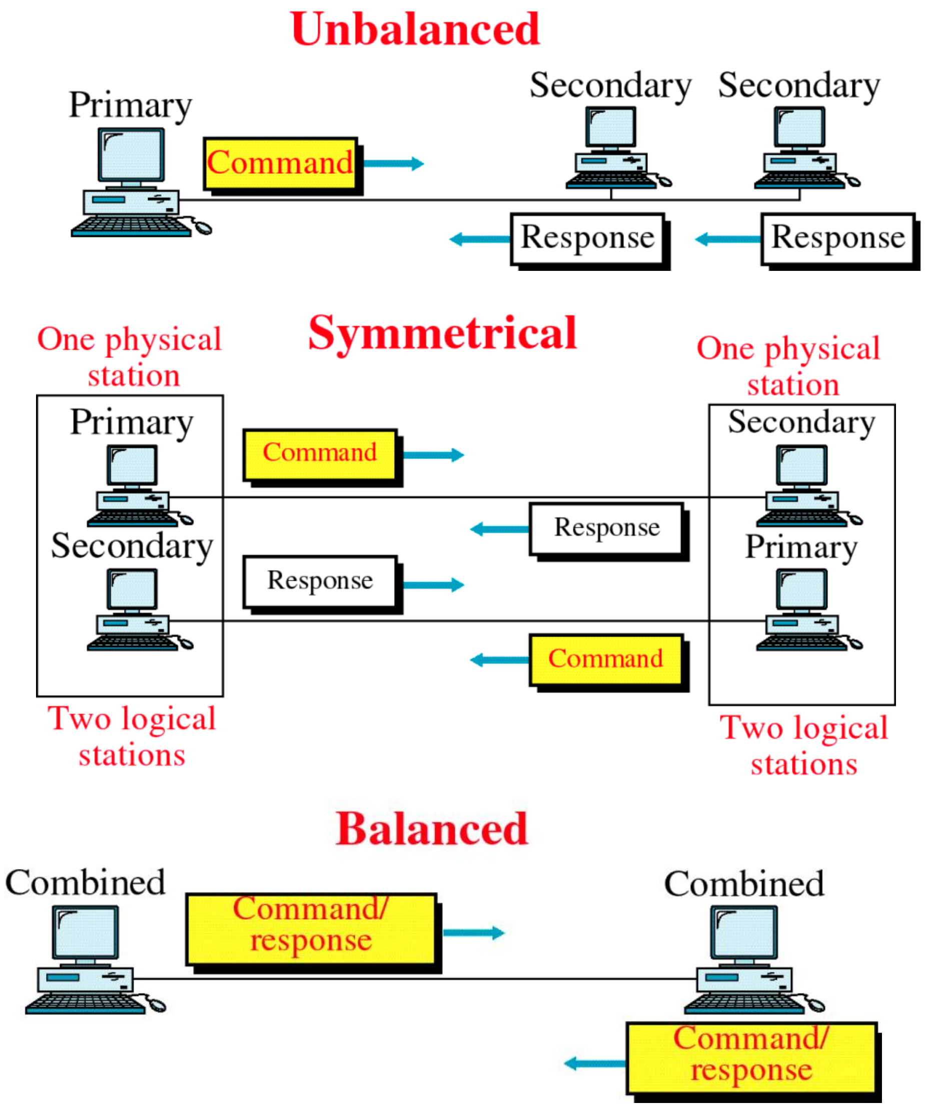


High-level Data Link Control (HDLC) protocol

- HDLC standardised by ISO in 1979
- now accepted by most other standards bodies (ITU-T, ANSI, …)
- X.25 packet-switching networks use a subset of HDLC called LAPB (Link Access Procedure, Balanced) – e.g. in ISDN

- 3 types of end-stations:
  - *Primary* – sends commands
  - *Secondary* – can only respond to Primary's commands
  - *Combined* – can both command and respond

- 3 types of configuration (Note: no balanced multipoint)



High-level Data Link Control (HDLC) protocol: Modes

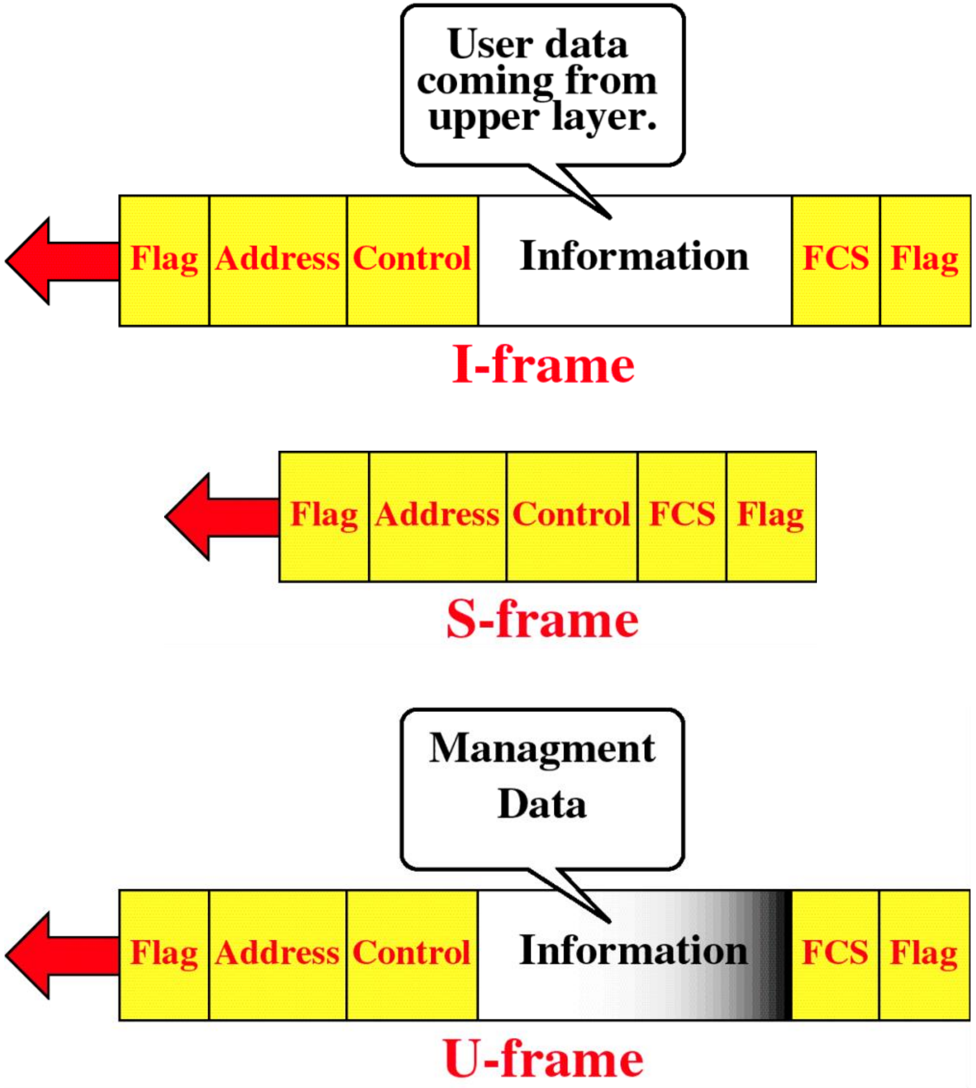
- **mode = relationship between 2 communicating devices; describes who controls the link**
  - **NRM = Normal Response Mode**
  - **ARM = Asynchronous Response Mode**
  - **ABM = Asynchronous Balanced Mode**

	NRM	ARM	ABM
Station type	Primary & secondary	Primary & secondary	Combined
Initiator	Primary	**Any**	**Either**

- ***in ARM, a secondary may initiate a transmission if the link is idle, but the transmission must still be sent to the Primary for relaying to another secondary as in NRM: only difference is that secondary needs permission from the Primary in NRM, but doesn't need permission from the Primary in ARM***

High-level Data Link Control (HDLC) protocol: Frames

- 3 types of Frames are defined (what is it about the number 3?!?!):
  - *I-Frame* – transports user data and control info. about user data (e.g. ACK)
  - *S-Frame* – supervisory Frame, only used for transporting control info.
  - *U-Frame* – unnumbered Frame, reserved for system management (managing the link itself)

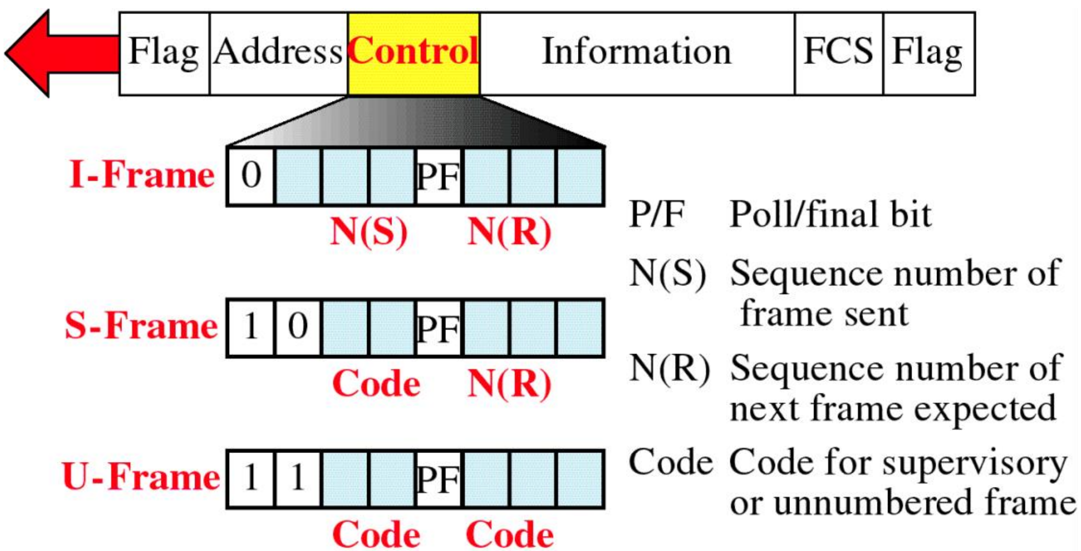


High-level Data Link Control (HDLC) protocol: Frames

Frame format: ***in back-to-back Frame transmissions, the end-flag of one Frame can be used as the start-flag of the next Frame***

Bits	8	8	8	≥ 0	16	8
	0 1 1 1 1 1 1 0	Address	Control	Data	Checksum	0 1 1 1 1 1 1 0
	flag					*flag*

Control field: **ARQ is Go-back-7 (or Go-back-127 in "extended mode")**



N(S), N(R) 3 bits long ⇒ window size = 7; in "extended mode", N(S), N(R) 7 bits long ⇒window size = 127

FCS = Frame Check Sequence (or Checksum): 2-byte or 4-byte CRC