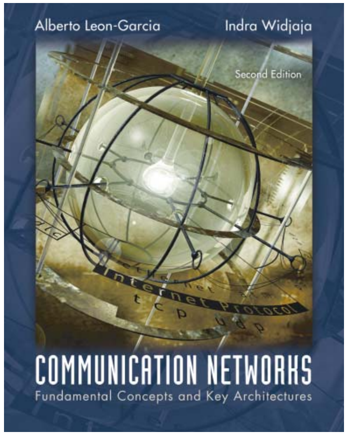
Chapter 5
Peer-to-Peer Protocols and Data Link Layer



Error and Flow Control

CSE 3213, Winter 2010

Instructor: Foroohar Foroozan

Error Control

Error Control Approaches

(1) **Forward Error Correction (FEC)**

(2) Error Detection + Automatic Retransmission Req. (ARQ)

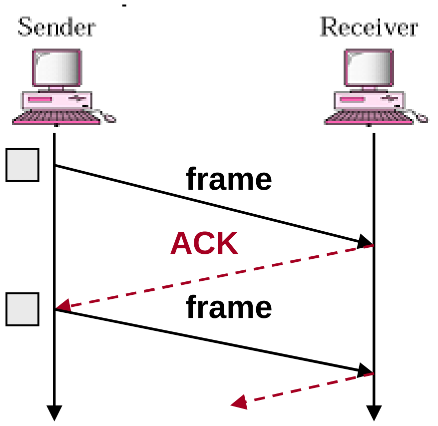
- not enough redundant info to enable error correction

case (a) receiver detects no errors

- an ACK packet is sent back to sender

case (b) receiver detects errors

- no ACK sent back to sender
- sender retransmits frame after a 'time-out'

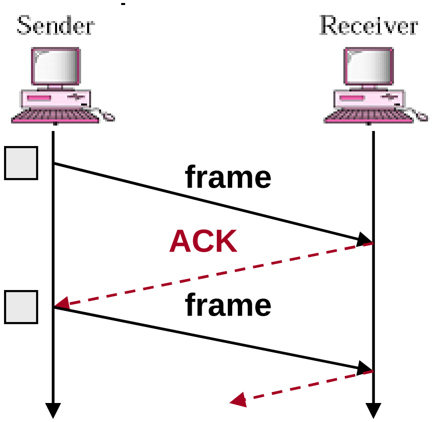


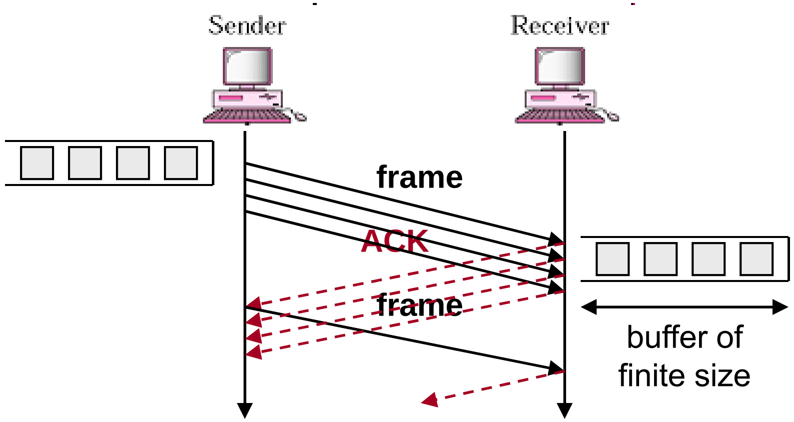
Error Control (Cont.)

Challenges of ARQ-based Error Control

- send one frame at the time, wait for ACK
  - easy to implement, but inefficient in terms of channel usage

- send multiple frames at once
  - better channel usage, but more complex to implement - sender must keep (all) sent but unACKed frame(s) in a buffer, as such frame(s) may have to be retransmitted





How many frames should be sent at any point in time?

How should frames be released from the sending buffer?

Flow and Error Control

Flow Control

– set of procedures used to restrict the amount of data that sender can send while waiting for acknowledgment

- two main strategies

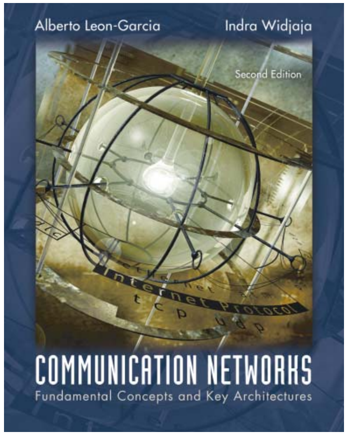
  (1) Stop-and-Wait: sender waits until it receives ACK before sending next frame

  (2) Sliding Window: sender can send W frames before waiting for ACKs

Error Detection + ARQ (error detection with retransmissions) must be combined with methods that intelligently limit the number of 'outstanding' (unACKed) frames.

Fewer unACKed frames ⇒ fewer packets buffered at sender and receiver.

Chapter 5
Peer-to-Peer Protocols and Data Link Layer



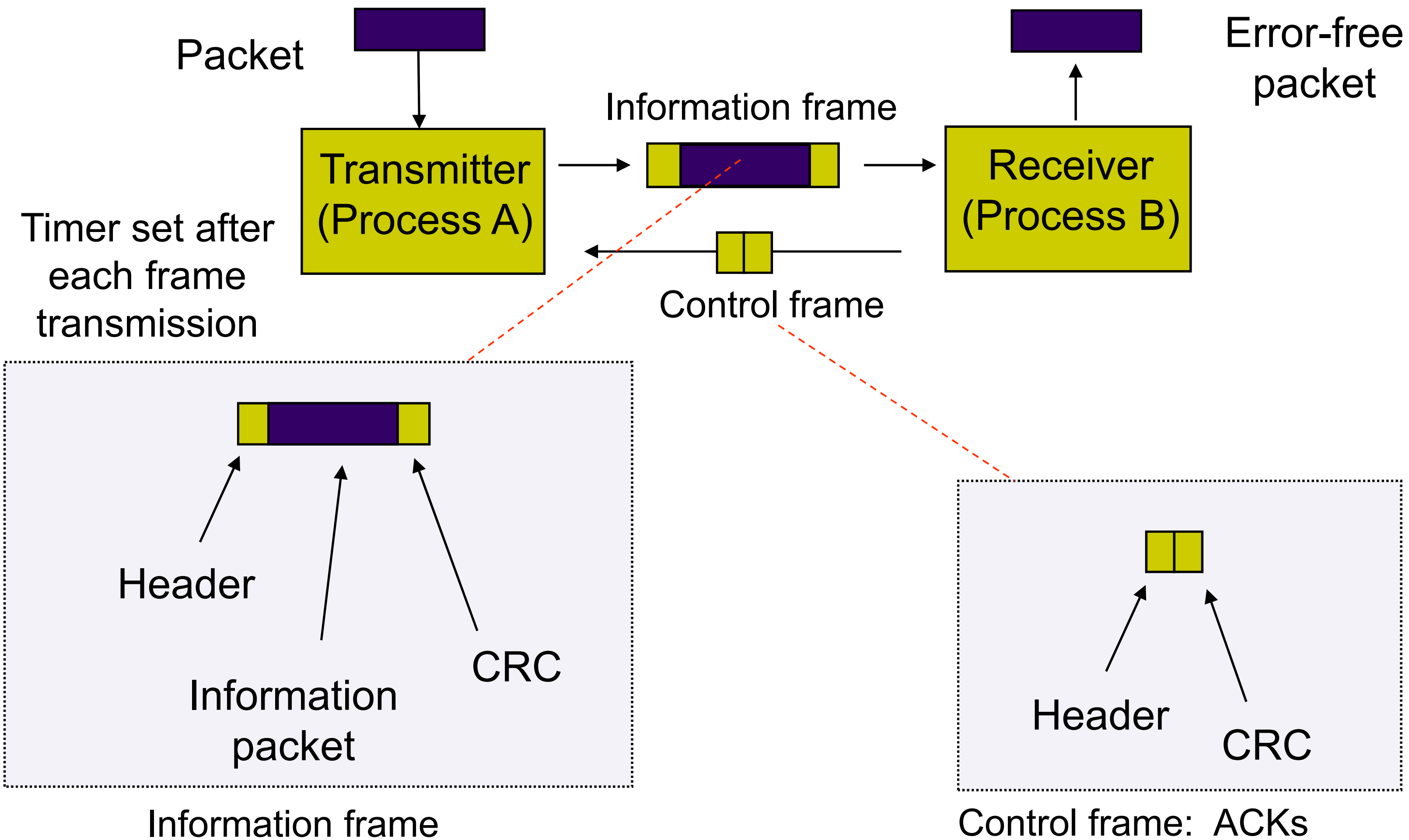
ARQ Protocols and Reliable Data Transfer

Automatic Repeat Request (ARQ)

- *Purpose*: to ensure a sequence of information packets is delivered in order and without errors or duplications despite transmission errors & losses
- We will look at:
  - Stop-and-Wait ARQ
  - Go-Back N ARQ
  - Selective Repeat ARQ
- Basic elements of ARQ:
  - *Error-detecting code* with high error coverage
  - *ACKs* (positive acknowledgments
  - *NAKs* (negative acknowlegments)
  - *Timeout mechanism*

Stop-and-Wait ARQ

- sender sends an information frame to receiver
- sender, then, stops and waits for an ACK
- if no ACK arrives within time-out, sender resends the frame, and again stops and waits time-out period > roundtrip time

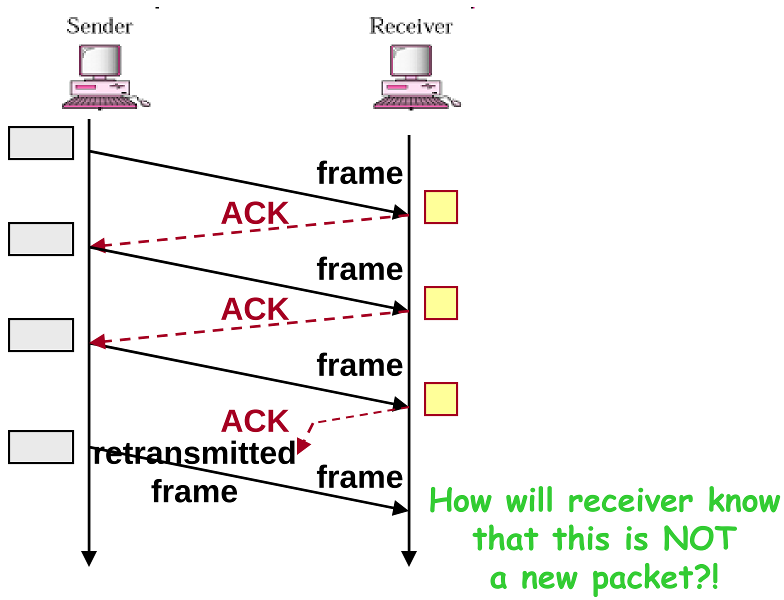


Stop-and-Wait ARQ (Cont.)

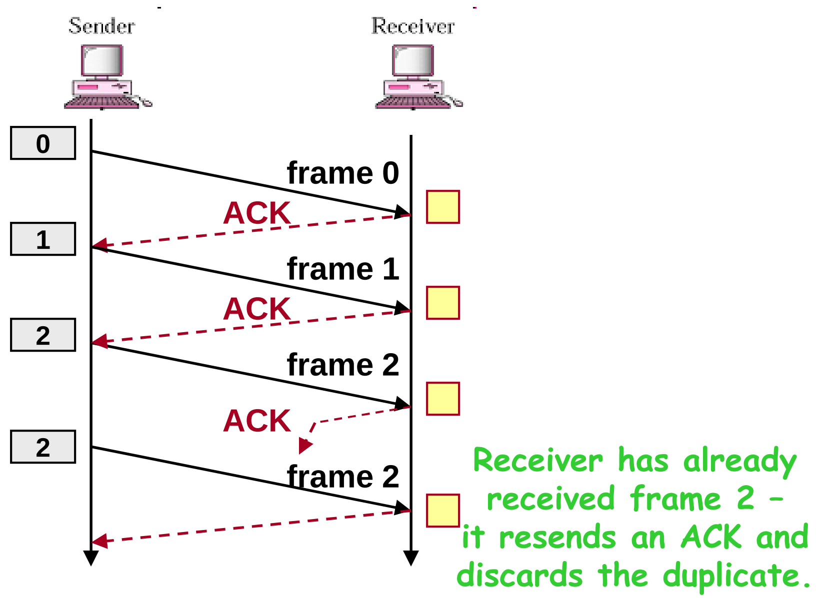
- abnormalities:
  - (1) lost acknowledgment
  - (2) delayed acknowledgment

Lost Acknowledgment

- frame is received correctly, but ACK undergoes errors / loss
  - after time-out period, sender resends frame
  - receiver receives the same frame twice
- frames must be numbered so that receiver can recognize and discard duplicate frames
  - sequence number are included in packet header



without packet numbering

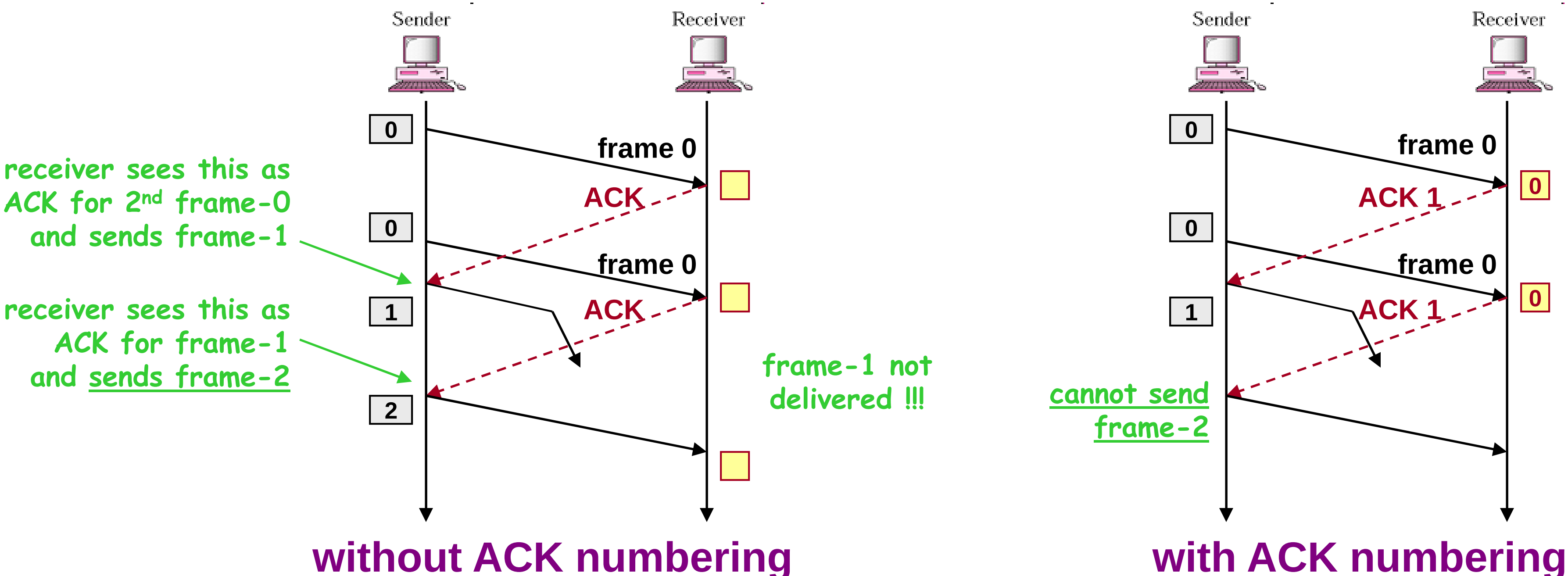


with packet numbering

Stop-and-Wait ARQ (Cont.)

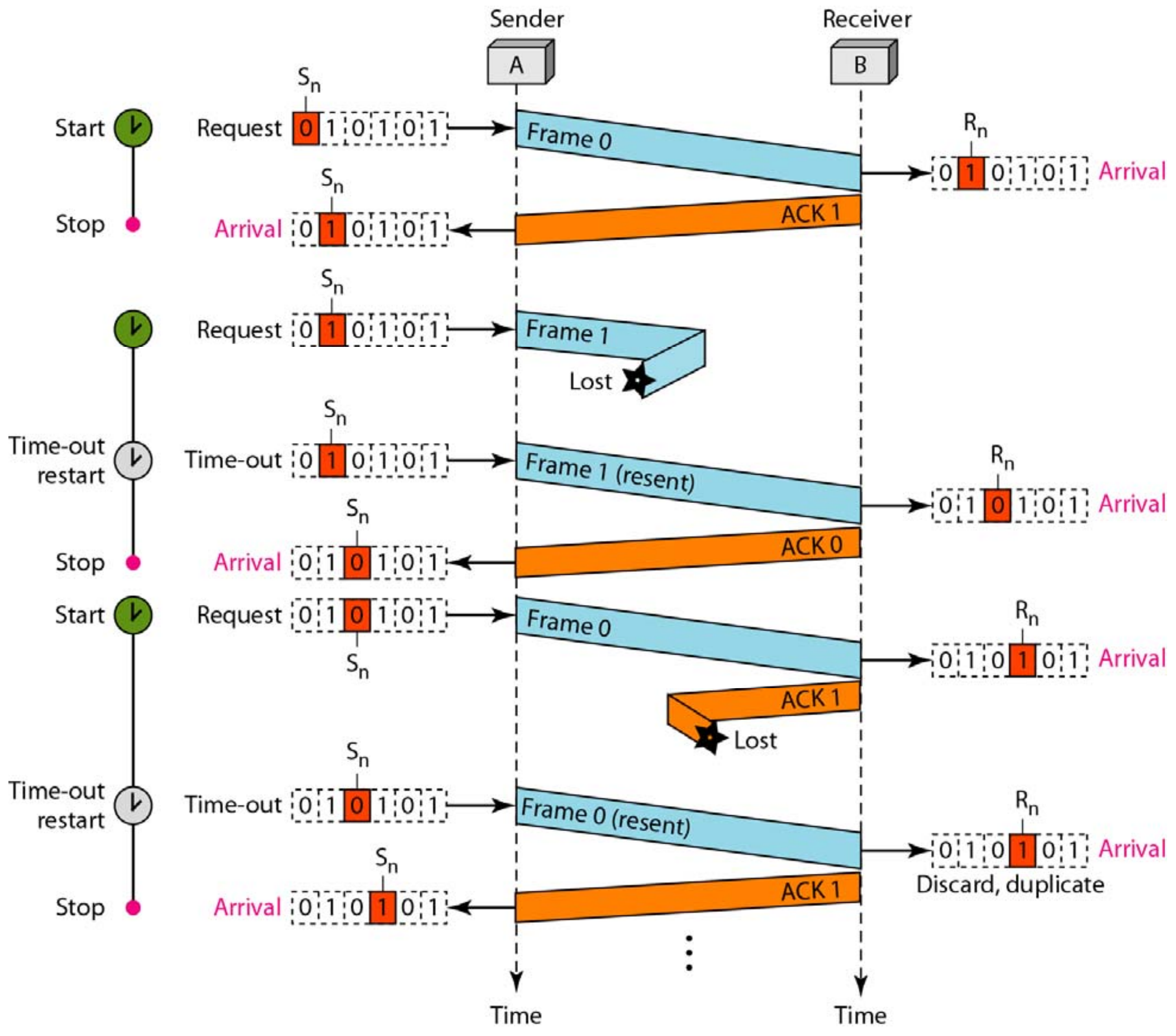
Delayed Acknowledgment (Premature Timeout)

- ACKs can be delayed due to problems with links or network congestion
  - time-out expires early, sender resends frame
  - when delayed ACK arrives, sender assumes that given ACK corresponds to last frame sent
- ACKs must be numbered to prevent gaps in delivered packet sequence

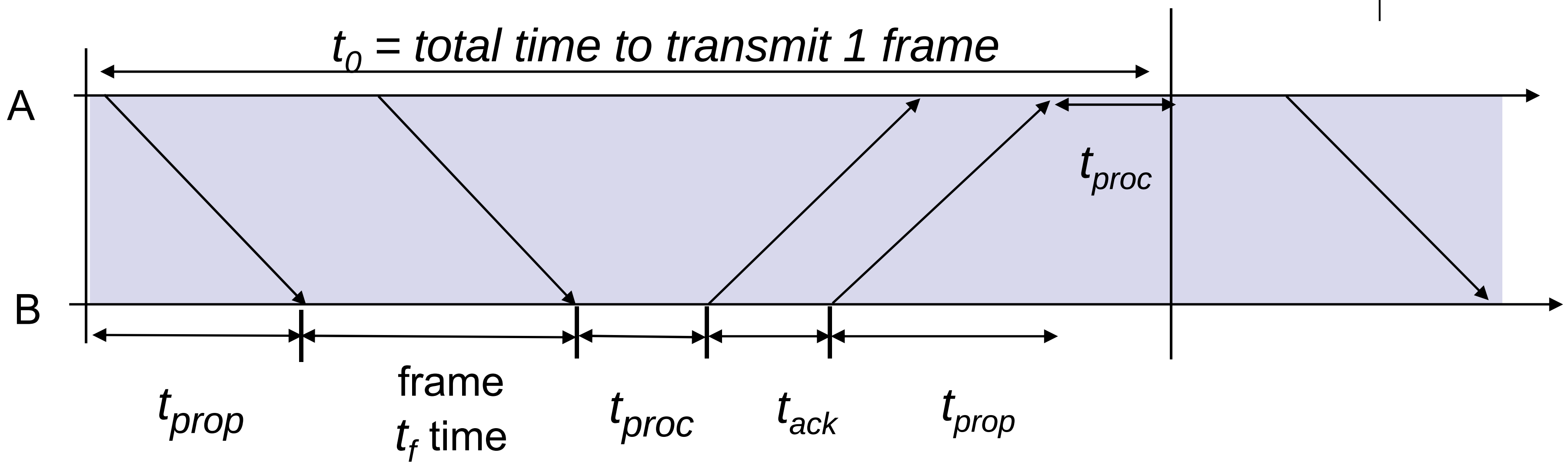


How large should the packet / ACK sequence be? Only 1-bit long !!!

Stop-and-Wait ARQ (Cont.)



Stop-and-Wait Model



$$t_0 = 2t_{prop} + 2t_{proc} + t_f + t_{ack}$$

$$= 2t_{prop} + 2t_{proc} + \frac{n_f}{R} + \frac{n_a}{R}$$

bits/info frame

bits/ACK frame

channel transmission rate

S&W Efficiency on Error-free channel

Effective transmission rate:

bits for header & CRC

$$R_{eff}^{0} = \frac{\text{number of information bits delivered to destination}}{\text{total time required to deliver the information bits}} = \frac{n_f - n_o}{t_0},$$

Transmission efficiency:

$$\eta_0 = \frac{R_{eff}}{R} = \frac{\frac{n_f - n_o}{t_0}}{R} = \frac{1 - \frac{n_o}{n_f}}{1 + \frac{n_a}{n_f} + \frac{2(t_{prop} + t_{proc})R}{n_f}}.$$

Effect of frame overhead

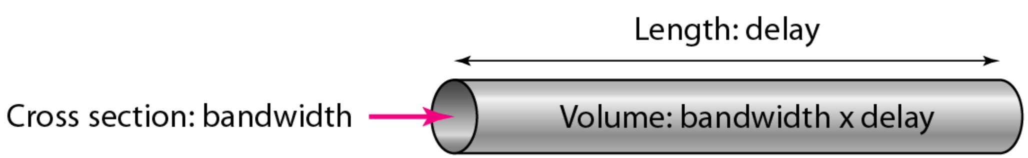
Effect of ACK frame

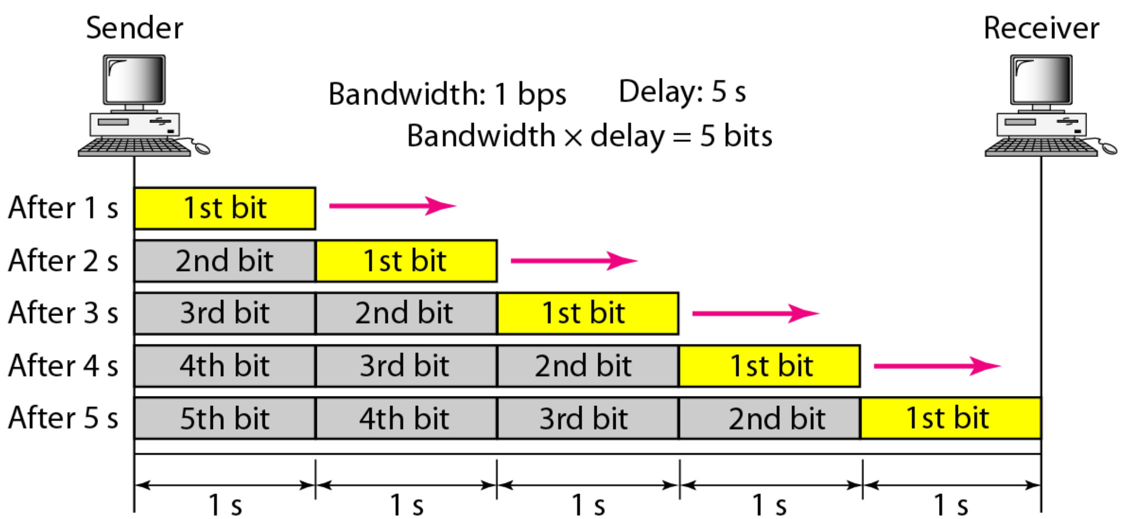
Effect of ***Delay-Bandwidth Product***

Stop-and-Wait ARQ (Cont.)

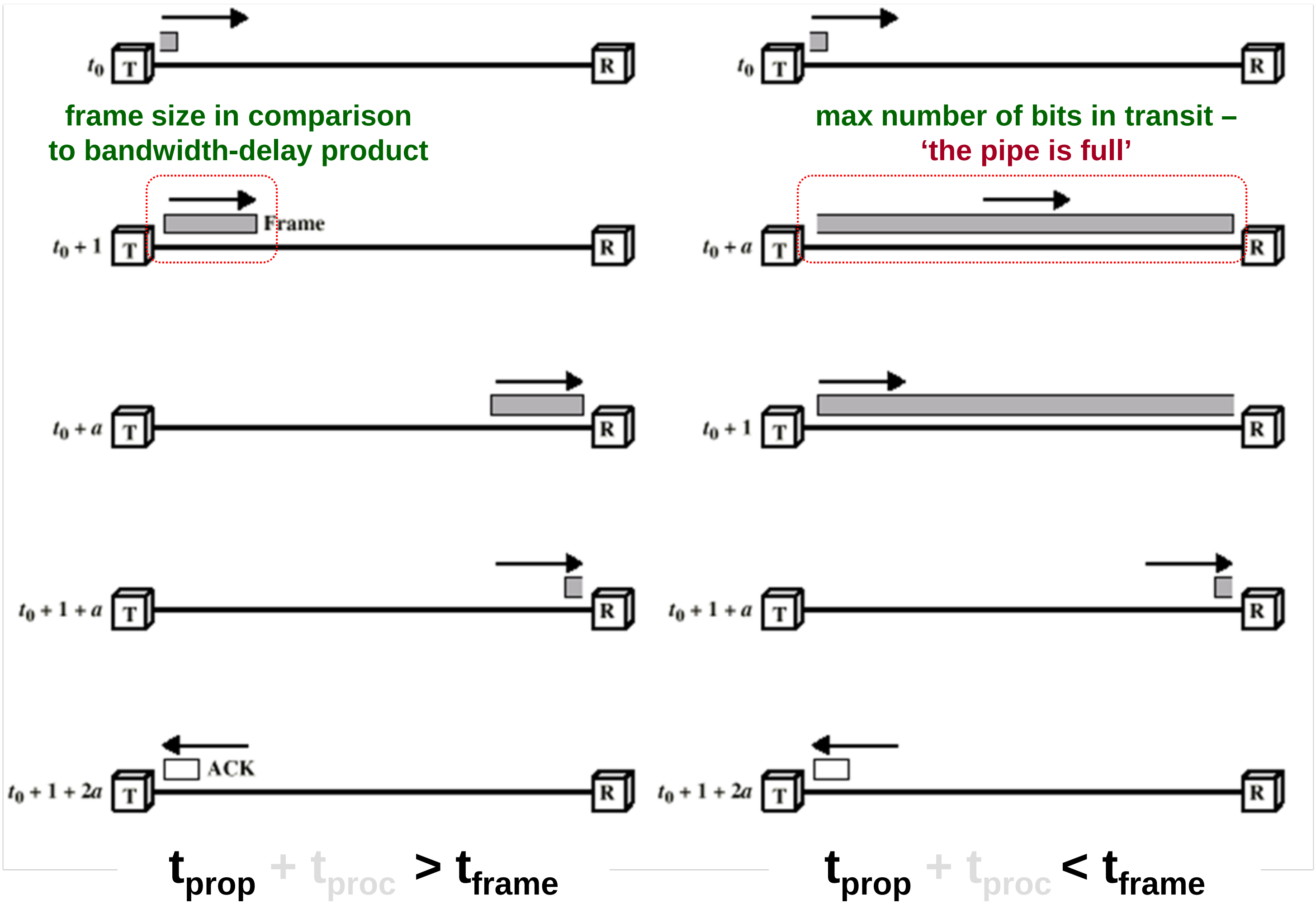
Bandwidth-delay product = $2*(t_{prop} + t_{proc})*R$ =

= capacity of the transmission pipe from the sender to the receiver and back.





Stop-and-Wait ARQ (Cont.)



Stop-and-Wait ARQ becomes inadequate when data is fragmented into small frames, such that $n_f / R = t_{frame}$ is small relative to t_{prop}.

Stop-and-Wait ARQ (Cont.)

Example [impact of delay-bandwidth product]

$$\left.\begin{array}{l} n_f = 1250 \text{ bytes} = 10000 \text{ bits} \\ n_{ACK} = n_{header} = 25 \text{ bytes} = 200 \text{ bits} \end{array}\right\} \Rightarrow \frac{n_{ACK}}{n_f} = \frac{n_{header}}{n_f} = 0.02$$

$$\eta_{SW} = \frac{R_{eff}}{R} = \frac{1 - \frac{n_{header}}{n_f}}{1 + \frac{n_{ACK}}{n_f} + \frac{2\cdot(t_{prop} + t_{proc})R}{n_f}} = \frac{0.98}{1.02 + \frac{2\cdot(t_{prop} + t_{proc})R}{n_f}}$$

Efficiency	200 km (t_{prop} = 1 ms)	2000 km (t_{prop} = 10 ms)	20000 km (t_{prop} = 100 ms)	200000 km (t_{prop} = 1 sec)
1 Mbps	10^3 88%	10^4 49%	10^5 9%	10^6 1%
1 Gbps	10^6 1%	10^7 0.1%	10^8 0.01%	10^9 0.001%

Stop-and-Wait does NOT work well for very high speeds or long propagation delays.

Stop-and-Wait ARQ (Cont.)

Stop-and-Wait Efficiency in Channel with Errors

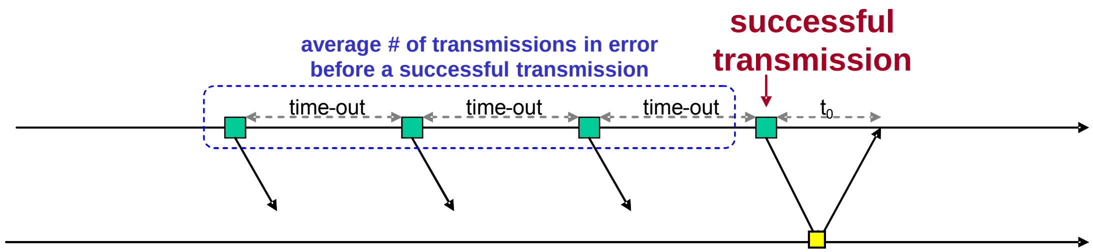
- P_f = probability that transmitted frame has errors and need to be retransmitted
  - $(1-P_f)$ – probability of successful transmission
  - $\frac{1}{1-P_f}$ – average # of (re)transmission [and including] until first correct arrival
  - total delay per frame: $t_0 \cdot (\text{average \# of retrans.}) = t_0 \cdot \frac{1}{1-P_f}$

$$\eta_{SW\_error} = \frac{R_{eff\_error}}{R} = \frac{\dfrac{n_f - n_{header}}{\dfrac{t_0}{(1-P_f)}}}{R} = (1-P_f) \cdot \frac{1 - \dfrac{n_{header}}{n_f}}{1 + \dfrac{n_{ACK}}{n_f} + \dfrac{2(t_{prop} + t_{proc})R}{n_f}} \quad (*)$$

$$\eta_{SW\_error} = (1-P_f) \cdot \eta_0$$

P_f increases $\Rightarrow$ η_{SW} decreases

Stop-and-Wait ARQ (Cont.)



Probability that i transmission are needed to deliver frame successfully

(i-1 transmission in error and the ith transmission is error free):

$$\mathbf{P[\ \#\ of\ trans.\ in\ error = i\text{-}1\ ] = (1\text{-}P_f)\ P_f^{\,i-1}}$$

$$E[\#\text{ of transmissions in error}] = \sum_{i=1}^{\infty}(i-1)\cdot P[n_{\text{trans in error}} = i-1] = \sum_{i=1}^{\infty}(i-1)\cdot(1-P_f)P_f^{i-1} =$$

$$= (1-P_f)\cdot\sum_{i=1}^{\infty}(i-1)\cdot P_f^{i-1} = (1-P_f)\cdot\sum_{n=1}^{\infty} n\cdot P_f^{n} =$$

$$= (1-P_f)\cdot P_f\cdot\sum_{n=1}^{\infty} n\cdot P_f^{n-1} = (1-P_f)\cdot P_f\cdot\frac{1}{(1-P_f)^2} =$$

$$= \frac{P_f}{1-P_f}$$

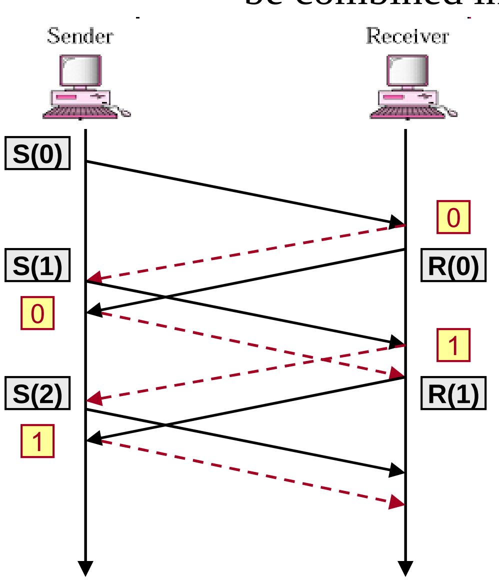
Total average delay per frame:

$$t_0 + \text{time-out}\cdot E[\#\text{ of transmiss in error}] = t_0 + \text{time-out}\cdot\frac{P_f}{1\text{-}P_f} \approx \frac{1}{1\text{-}P_f}t_0$$

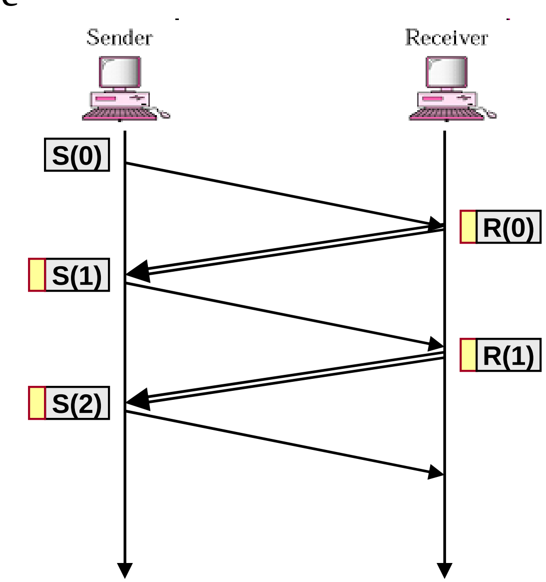
Stop-and-Wait ARQ (Cont.)

Piggybacking

- Stop-and-Wait discussed so far was ‘unidirectional’
- in ‘bidirectional’ communications, both parties send and acknowledge data, i.e. both parties implement flow control
- piggybacking method: outstanding ACKs are placed in the header of information frames
- piggybacking can save bandwidth since the overhead from a data frame and an ACK frame (addresses, CRC, etc) can be combined into just one frame



without piggybacking



with piggybacking

Applications of Stop-and-Wait ARQ

- IBM *Binary Synchronous Communications protocol* (Bisync): character-oriented data link control
- *Xmodem*: modem file transfer protocol
- *Trivial File Transfer Protocol* (RFC 1350): simple protocol for file transfer over UDP

Go-Back-N ARQ

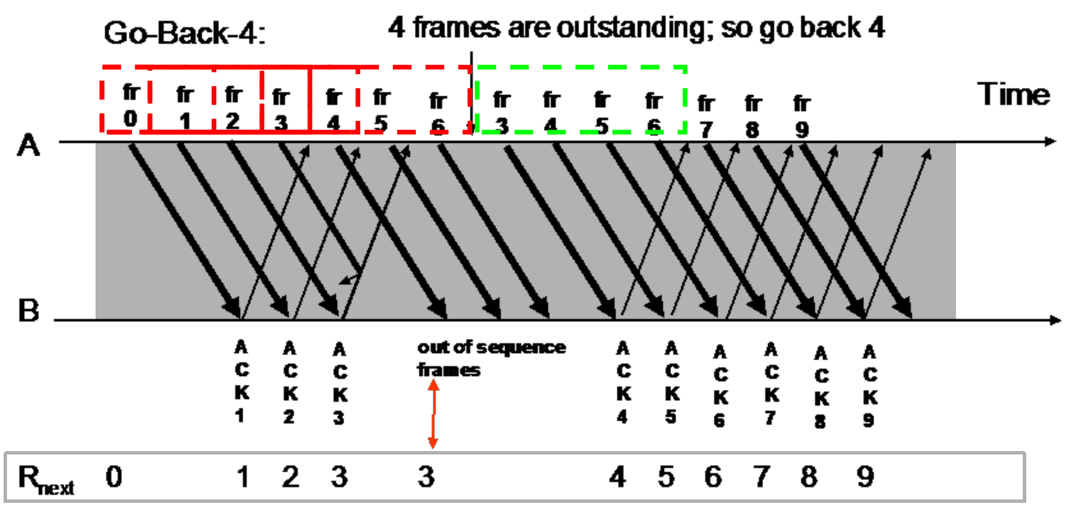
- Improve Stop-and-Wait by not waiting!
- Keep channel busy by continuing to send frames
- Allow a window of up to W_s outstanding frames
- Use *m*-bit sequence numbering
- If ACK for oldest frame arrives before window is exhausted, we can continue transmitting
- If window is exhausted, pull back and retransmit all outstanding frames
- Alternative: Use timeout

Go-Back-N ARQ

Go-Back-N ARQ

– overcomes inefficiency of Stop-and-Wait ARQ – sender continues sending enough frames to keep channel busy while waiting for ACKs

- a window of W_s outstanding frames is allowed
- m-bit sequence numbers are used for both - frames and ACKs, and $W_s = 2^m$-1

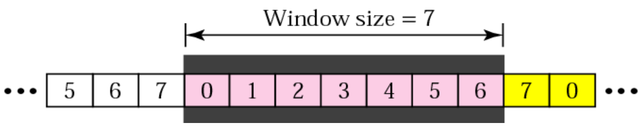


Assume: W_s= 4

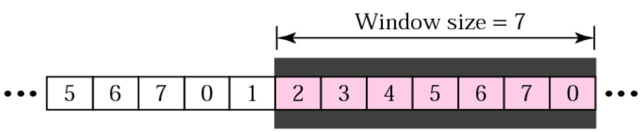
1) sender sends frames one by one
2) frame 3 undergoes transmission error – receiver ignores frame 3 and all subsequent frames
3) sender eventually reaches max number of outstanding frames, and takes following action:
   - go back N=W_s frames and retransmit all frames from 3 onwards

Go-Back-N ARQ (Cont.)

Sender Sliding Window



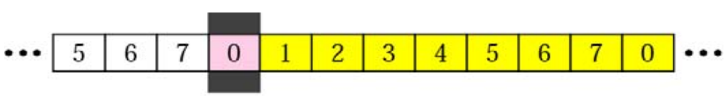
before ACKs for frames 0 and 1 arrive



after ACKs for frames 0 and 1 arrive and window slides

- all frames are stored in a buffer, outstanding frames are enclosed in a window
  - frames to the left of the window are already ACKed and can be purged
  - frames to the right of the window cannot be sent until the window slides over them
  - whenever a new ACK arrives, the window slides to include new unsent frames
  - once the window gets full (max # of outstanding frames is reached), entire window gets resent

Receiver Sliding Window



a. Before sliding

b. After sliding

- the size of receiver window is always 1
  - receiver is always looking for a specific frame to arrive in a specific order
  - any frame arriving out of order is discarded and needs to be resent

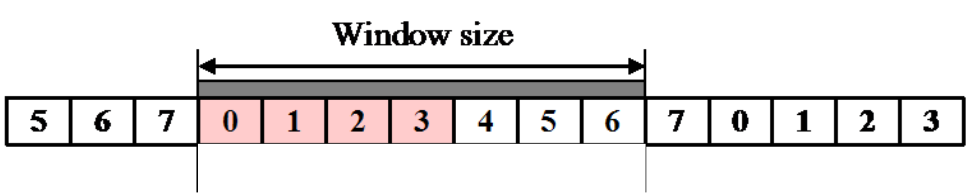
The complexity of the receiver in Go-Back-N is the same as that of Stop-and-Wait!!!

Only the complexity of the transmitter increases.

Go-Back-N ARQ (Cont.)

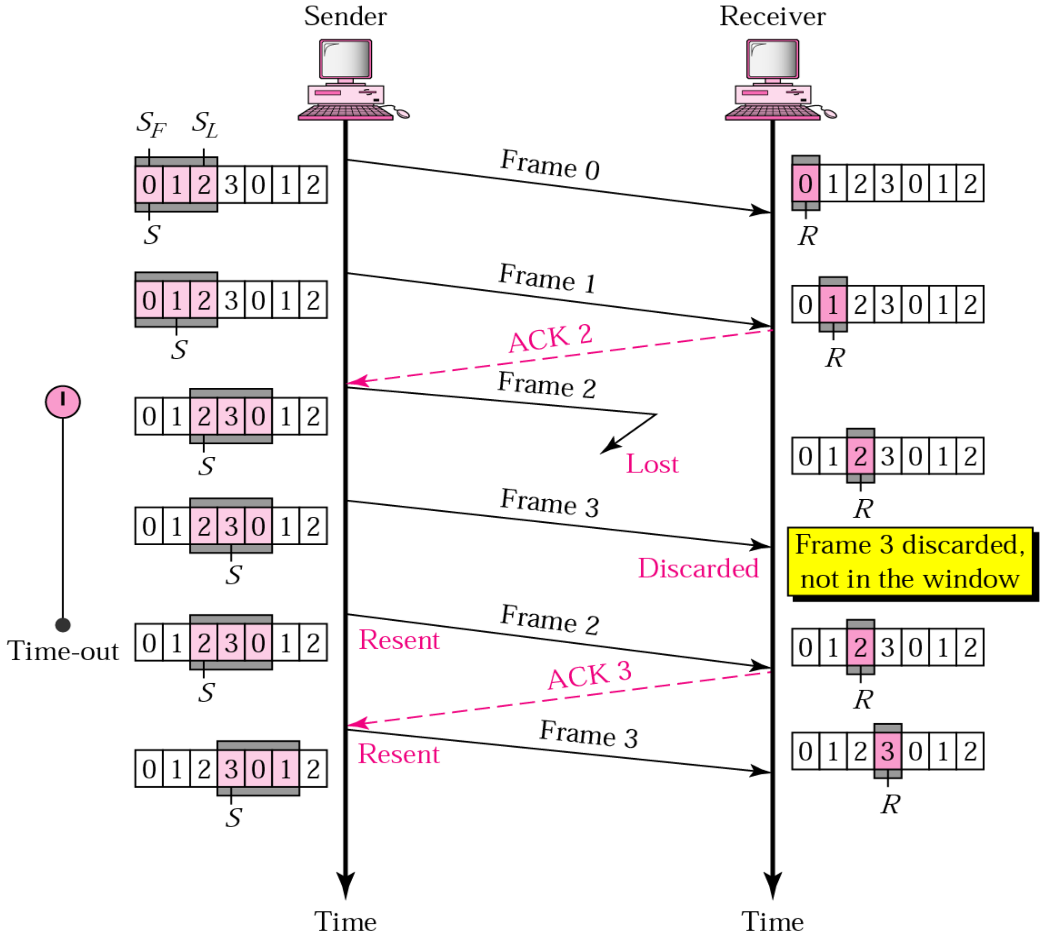
Problems with Go-Back-N (Go-Back-N with Timeout)

- Go-Back-N works correctly (retransmission of damaged frames gets triggered) as long as the sender has an unlimited supply of packets that need to be transmitted
  - but, in case when packets arrive sporadically, there may not be W_s-1 subsequent transmissions $\Rightarrow$ window will not be exhausted, retransmissions will not be triggered
  - this problem can be resolved by modifying Go-Back-N such that:
    1) set a timer for each sent frame
    2) resend all outstanding frames either when the window gets full or when the timer of first frame expires



Go-Back-N ARQ (Cont.)

Example [lost frame in Go-Back-N with time-out]



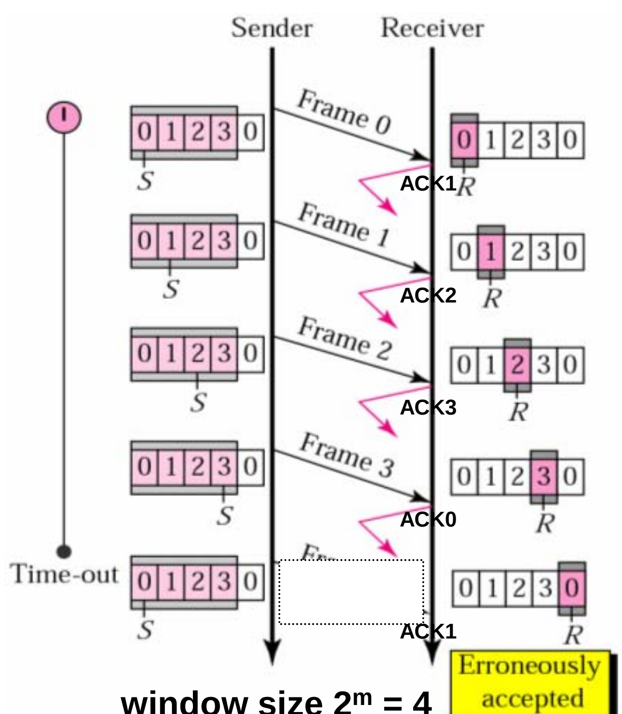
Note:

- ACKs number always defines the number of the next expected frame !!!
- in Go-Back-N, receiver does not have to acknowledge each frame received – it can send one cumulative ACK for several frames

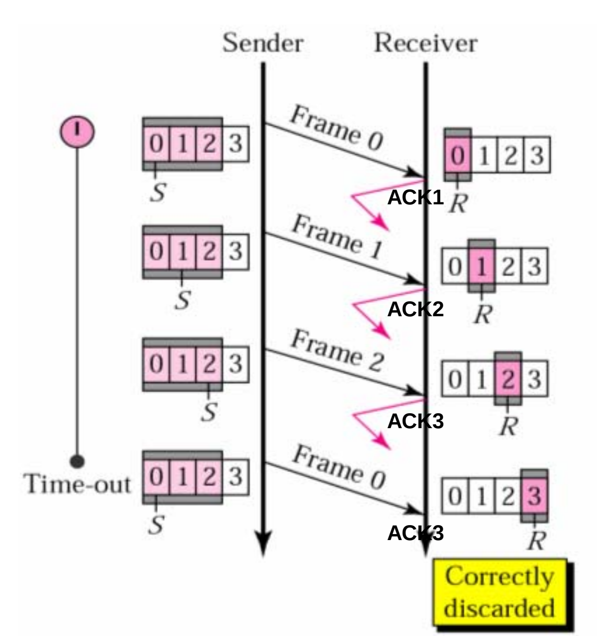
Go-Back-N ARQ (Cont.)

Sequence Numbers and Window Size

- m bits allotted within a header for sequence numbers $\Rightarrow$ 2^m possible sequence numbers
  - how big should the sender window be!?
  - W > 2^m cannot be accepted – multiple frames with same seq. number in the window $\Rightarrow$ ambiguous ACKs
  - W = 2^m can still cause some ambiguity – see below
  - W = 2^m – 1 acceptable !!!



window size $2^m = 4$

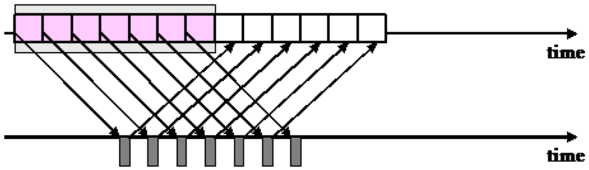


window size $2^m - 1 = 3$

Go-Back-N ARQ (Cont.)

Go-Back-N Efficiency

- completely efficient if W_s is large enough to keep channel busy, and if channel is error free



- in case of error-prone channel, with P_f frame loss probability, time to deliver a frame is:

  - t_{frame} - if 1st transmission succeeds – prob. $(1-P_f)$
  - $t_{frame} + \frac{1}{1-P_f} \cdot W_s \cdot t_{frame}$ - if 1st transmission does NOT succeeds – prob. P_f

average # of frame/window (re)transmission until a successful transmission

- total average time required to transmit a frame:

$$t_{GBN} = (1-P_f)\cdot t_{frame} + P_f \cdot \left(t_{frame} + \frac{1}{1-P_f}\cdot W_s\right) = t_{frame} + \frac{P_f}{1-P_f}\cdot W_s \cdot t_{frame}$$

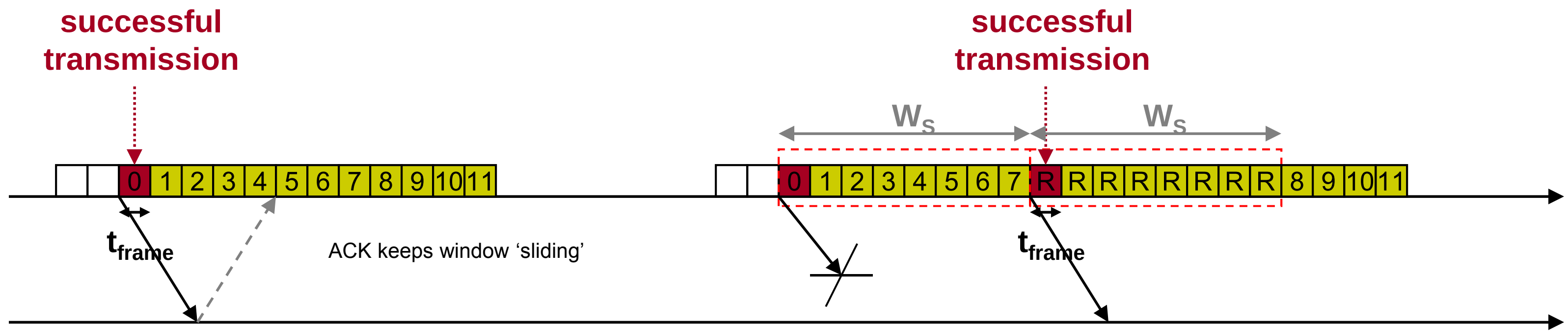
- transmission efficiency

$$\eta_{GBN} = \frac{\frac{n_f - n_{header}}{t_{GBN}}}{R} = \frac{1-\frac{n_{header}}{n_f}}{1+(W_s-1)P_f}(1-P_f)$$

(**)

Go-Back-N ARQ (Cont.)

What is total <u>average time</u> required to transmit a frame, assuming P_f?



1st attempt successful: $$t_{GBN} = t_{frame}$$

2nd attempt successful: $$t_{GBN} = t_{frame} + W_S \cdot t_{frame}$$

average case: $$t_{GBN} = t_{frame} + E[\#\text{ of transmissions in error}] \cdot W_S \cdot t_{frame}$$

$$E[\#\text{ of transmissions in error}] = \frac{P_f}{1-P_f}$$

$$t_{GBN} = t_{frame} + \frac{P_f}{1-P_f} W_S \cdot t_{frame}$$

$\Rightarrow$

$$\eta_{GBN} = \frac{\dfrac{n_f - n_{header}}{t_{GBN}}}{R} = \frac{1-\dfrac{n_{header}}{n_f}}{1+(W_s-1)P_f}(1-P_f)$$

Go-Back-N ARQ (Cont.)

Example [Stop-and-Wait vs. Go-Back-N]

n_f = 1250 bytes = 10000 bits

$n_{ACK} = n_{header}$ = 25 bytes = 200 bits

Compare S&W with GBN efficiency for random bit errors with $p_b = 0, 10^{-6}, 10^{-5}, 10^{-4}$ and bandwidth-delay product $R*2*(t_{prop}+t_{proc})$ = 1 Mbps * 100 ms = 100000 bits = 10 frames → use $W_s = 11$.

Efficiency	$p_b=0$	$p_b=10^{-6}$	$p_b=10^{-5}$	$p_b=10^{-4}$
S&W	8.9%	8.8%	8.0%	3.3%
GBN	98%	88.2%	45.4%	4.9%

- Go-Back-N provides significant improvement over Stop-and-Wait for large delay-bandwidth product
- Go-Back-N becomes inefficient as error rate increases

Applications of Go-Back-N ARQ

- *HDLC* (High-Level Data Link Control): bit-oriented data link control
- *V.42 modem*: error control over telephone modem links

Selective Repeat ARQ

- Go-Back-N ARQ inefficient because *multiple* frames are resent when errors or losses occur
- Selective Repeat retransmits *only an individual frame*
  - Timeout causes individual corresponding frame to be resent
  - NAK causes retransmission of oldest un-acked frame
- Receiver maintains a *receive window* of sequence numbers that can be accepted
  - Error-free, but out-of-sequence frames with sequence numbers within the receive window are buffered
  - Arrival of frame with R_{next} causes window to slide forward by 1 or more

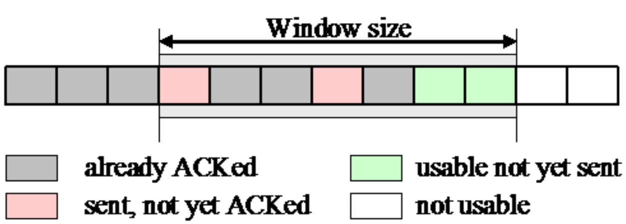
Selective Repeat ARQ

Selective Repeat ARQ

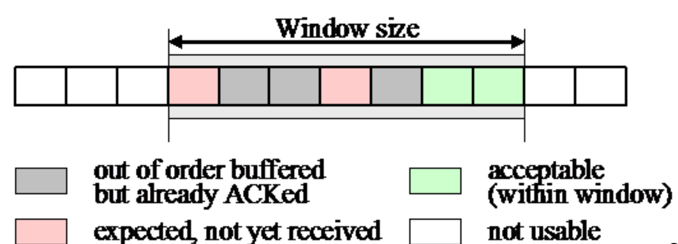
- Go-Back-N is NOT suitable for 'noisy links' – in case of a lost/damaged frame a whole window of frames need to be resent
  - excessive retransmissions use up the bandwidth and slow down transmission
- Selective Repeat ARQ overcomes the limitations of Go-Back-N by adding 2 new features

  (1) receiver window > 1 frame, so that out-of-order but error-free frames can be accepted

  (2) retransmission mechanism is modified – only individual frames are retransmitted
- Selective Repeat ARQ is used in TCP !!!



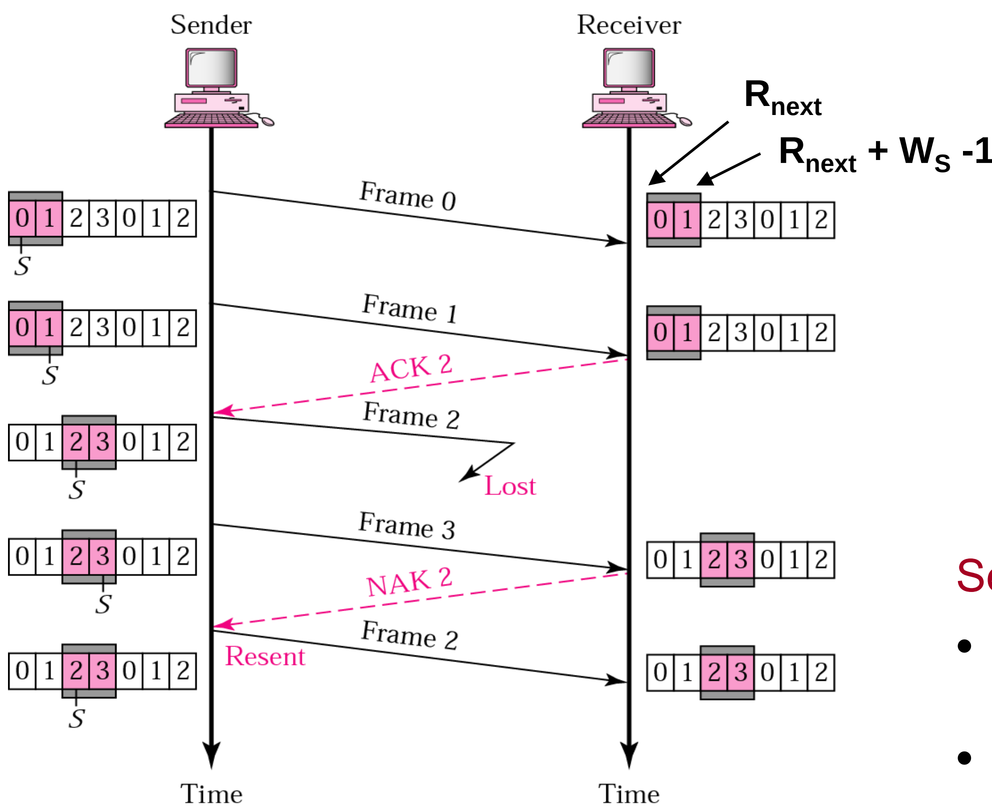
sender window of size W_S



receiver window of size W_R

Selective Repeat ARQ

Selective Repeat ARQ Operation



Receiver:

- window advances whenever next in-order frame arrives
- out-of-order frames are accepted only if their sequence numbers satisfy

$$R_{next} < R_{frame} < R_{next} + W_s$$

- a negative ACK (NAK) with sequence number R_{next} is sent whenever an out-of-sequence frame is observed

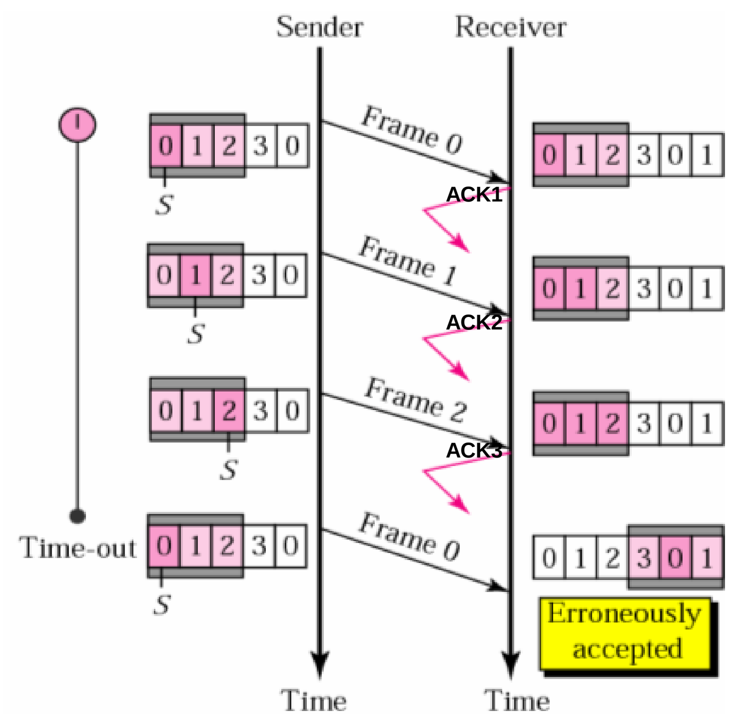
Sender:

- window advances whenever an ACK arrives
- if a timer expires, the corresponding frame is resent, and the timer is reset
- whenever a NAK arrives, R_{next} frame is resent

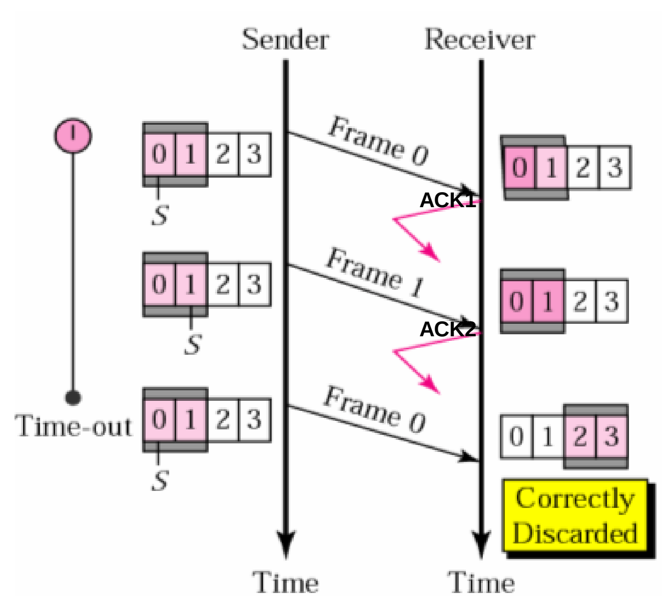
Selective Repeat ARQ

Window Sizes – W_S and W_R

- m bits allotted within a header for sequence numbers ⇒ 2^m possible sequence numbers
  - how big should the windows be!?
  - W_S and $W_R = 2^m - 1$ cannot be accepted due to possible ambiguity as shown below
  - $W = 2^m/2 = 2^{m-1}$ acceptable !!!



window size 2^m-1 = 3



window size 2^{m-1} = 2

Selective Repeat ARQ

Selective Repeat Efficiency

- completely efficient if W_s is large enough to keep channel busy, and if channel is error free
  - of course, sequence number space must be 2X sequence sequence number space of Go-Back-N

- in case of error-prone channel, total average time required to transmit a frame:

$$t_{SR} = \frac{t_{frame}}{1-P_f} = \frac{n_f}{R \cdot (1-P_f)}$$

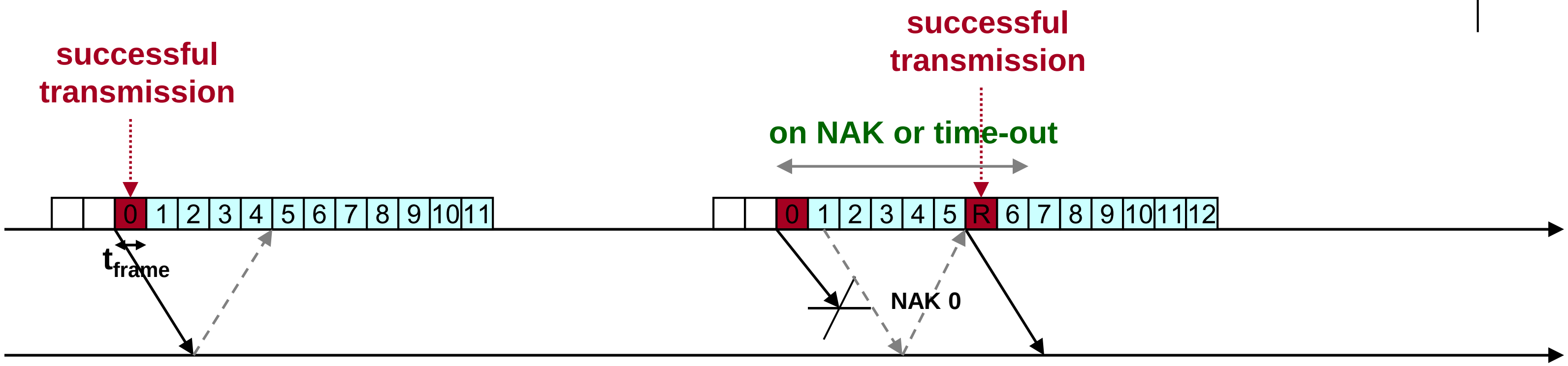
- transmission efficiency

$$\eta_{SR} = \frac{R_{eff}}{R} = \frac{\frac{n_f - n_{header}}{t_{SR}}}{R} = \left(1 - \frac{n_{header}}{n_f}\right) \cdot (1-P_f)$$

(***)

Selective Repeat ARQ

What is total average time required to transmit a frame, assuming P_f?



1st attempt successful: $\mathbf{t_{SR} = t_{frame}}$

2nd attempt successful: $\mathbf{t_{SR} = t_{frame} + t_{frame}}$

average case: $\mathbf{t_{SR} = t_{frame} + E[\#\ of\ transmissions\ in\ error] \cdot t_{frame}}$

$$\mathbf{\frac{P_f}{1-P_f}}$$

$$t_{SR} = t_{frame} + \frac{P_f}{1-P_f} \cdot t_{frame} = \frac{1}{1-P_f} \cdot \frac{n_f}{R}$$

$$\Rightarrow$$

$$\eta_{SR} = \frac{\dfrac{n_f - n_{header}}{t_{SR}}}{R} = \left(1 - \frac{n_{header}}{n_f}\right)(1-P_f)$$

Stop-&-Wait vs. Go-Back-N vs. Selective Repeat

Performance Comparison

- assume n_{ACK} and n_{header} are negligible relative to n_f, and

$$\frac{2(t_{prop}+t_{proc})R}{n_f} = L = W_s - 1$$

size of the "pipe" in multiples of frames

W_S is for 1 less than the number of frames currently in transit

- efficiencies of three ARQ techniques are

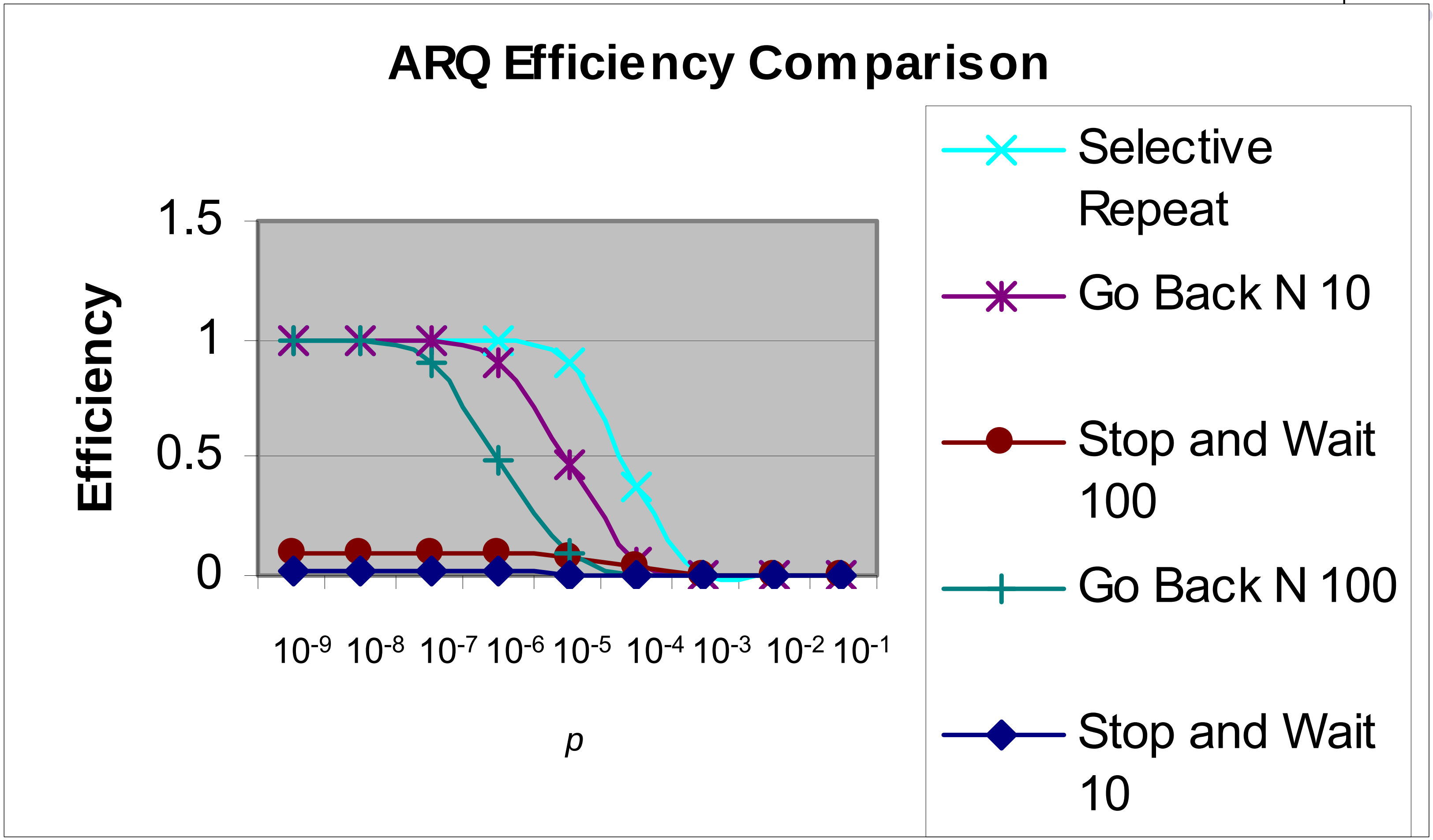
$$\eta_{SW} = \frac{1}{1+L}\cdot(1-P_f)$$

$$\eta_{GBN} = \frac{1}{1+LP_f}(1-P_f)$$

$$\eta_{SR} = (1-P_f)$$

$\eta_{SW} < \eta_{GBN} < \eta_{SR}$

- for $0 < P_f < 1$, Selective Repeat provides best performance
- for $P_f \to 0$ Go-Back-N as good as Selective Repeat



Delay-Bandwidth product = 10, 100