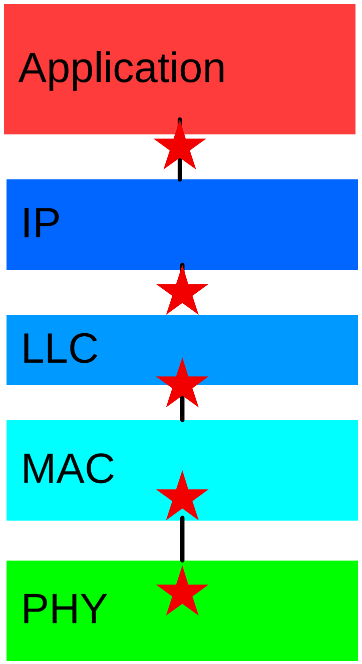
# Link-Layer Error Control Schemes

Wireless Networking
Sunghyun Choi, Associate Professor
Multimedia & Wireless Networking Lab. (MWNL)
School of Electrical Engineering
Seoul National University

# Error Control

- One of the functions of link layer is reliable transmission
- Various FEC approaches




- Which layer will we implement the error control techniques at?

  → That depends on the given design constraints.

  ex) If we must use the exsiting PHY chip-sets, there is no other choice to choose above-MAC error control approach.

# Error Control Schemes

- Forward Error Correction (FEC)
  - Using error correction codes
- Automatic Repeat reQuest (ARQ)
  - Using error detection codes
  - Retransmission if not successful
- Hybrid ARQ (HARQ)
  - Typically classified into two types, i.e., Type I, Type II
  - Chase combining, Incremental redundancy

# Forward Error Correction (FEC)

- Typically done at PHY layer along with modulation/demodulation
  - Convolutional code, convolutional/RS concatenated code, Turbo code, LDPC code
- Can be additionally implemented at MAC
  - To make a link more reliable for a specific type of traffic on top of the same PHY
  - Typically, block coding is used
  - E.g., RS coding in HIPERLAN/2 MAC (European 5GHz WLAN)

# FEC Decoder Outcomes

- Error corrected
  - No errors present
  - Decoder detects and corrects bit errors
- Decoding failure
  - Decoder detects but cannot correct bit errors; reports uncorrectable error
- Decoding error
  - Decoder detects errors, and believes that the errors were corrected while it is not true

# Reed-Solomon (RS) Code

- (N,K) non-binary block code
  - K information symbols & (N-K) redundant symbols
    - Redundant symbols used to detect and correct symbol errors
  - One symbol = $l$ bits, where q=2^$l$
    - Often $l$ =8 (to make one byte a symbol)
  - N=q-1
    - (255,K) RS code
  - K can be any value

# RS Error Correction Capability

- Error correction capability of (N,K) RS code
  - $t = \lfloor N - K \rfloor / 2$
  - Correction of up to t symbol errors is guaranteed
- Powerful to correct bursty errors
  - Note that one bit error has the same effect as 8 bit errors (in case of q=256) as long as the 8 errors are in the same symbol boundary

# Error Performance

- Block error probability for symbol error probability $P_s$
  - With RS code

$$P_B \le \sum_{i=t+1}^{N} \binom{N}{i} P_s^i (1-P_s)^{N-i}$$

  - Without RS code

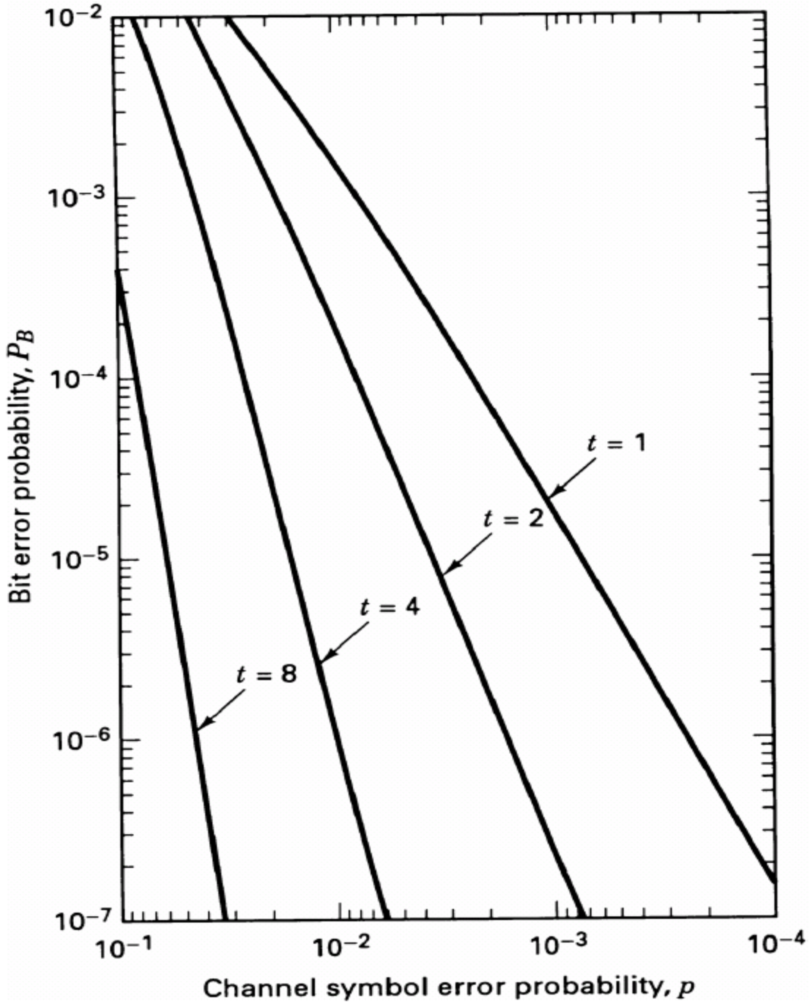
$$P_B = 1-(1-P_s)^N$$

  - Error and throughput curves ?

# Shortening and Puncturing

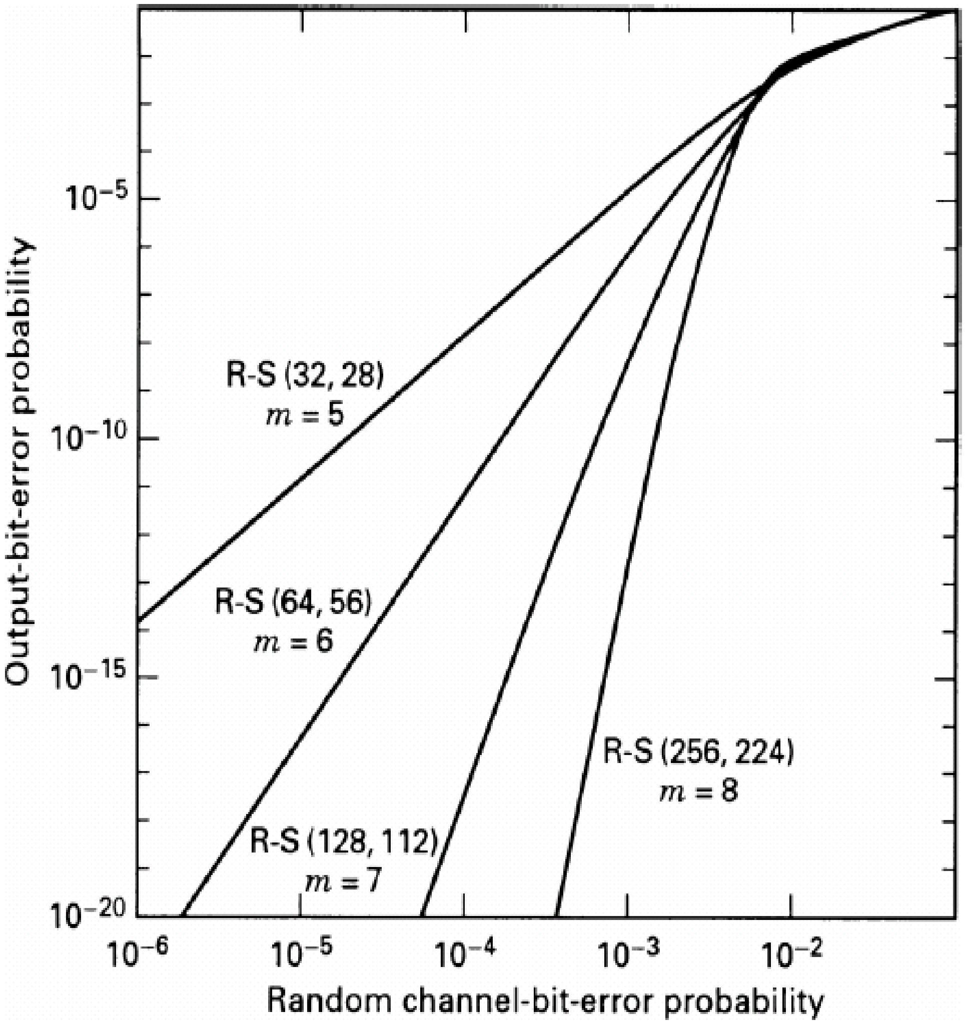
- Different codes from same original code
  - The same codec can be used
- (N,K) original code
  - $R = K/N,\ t = \lfloor N-K \rfloor/2$
- (N−s,K−s) shortened code
  - $R_1 = (K-s)/(N-s) < R,\ t_1 = \lfloor N-K \rfloor/2 = t$
- (N−p,K) punctured code
  - $R_2 = K/(N-p) > R,\ t_2 = \lfloor N-K-p \rfloor/2 < t$
  - Puncturing is very popular with conv. code!

# Performance: Error Correction Capability



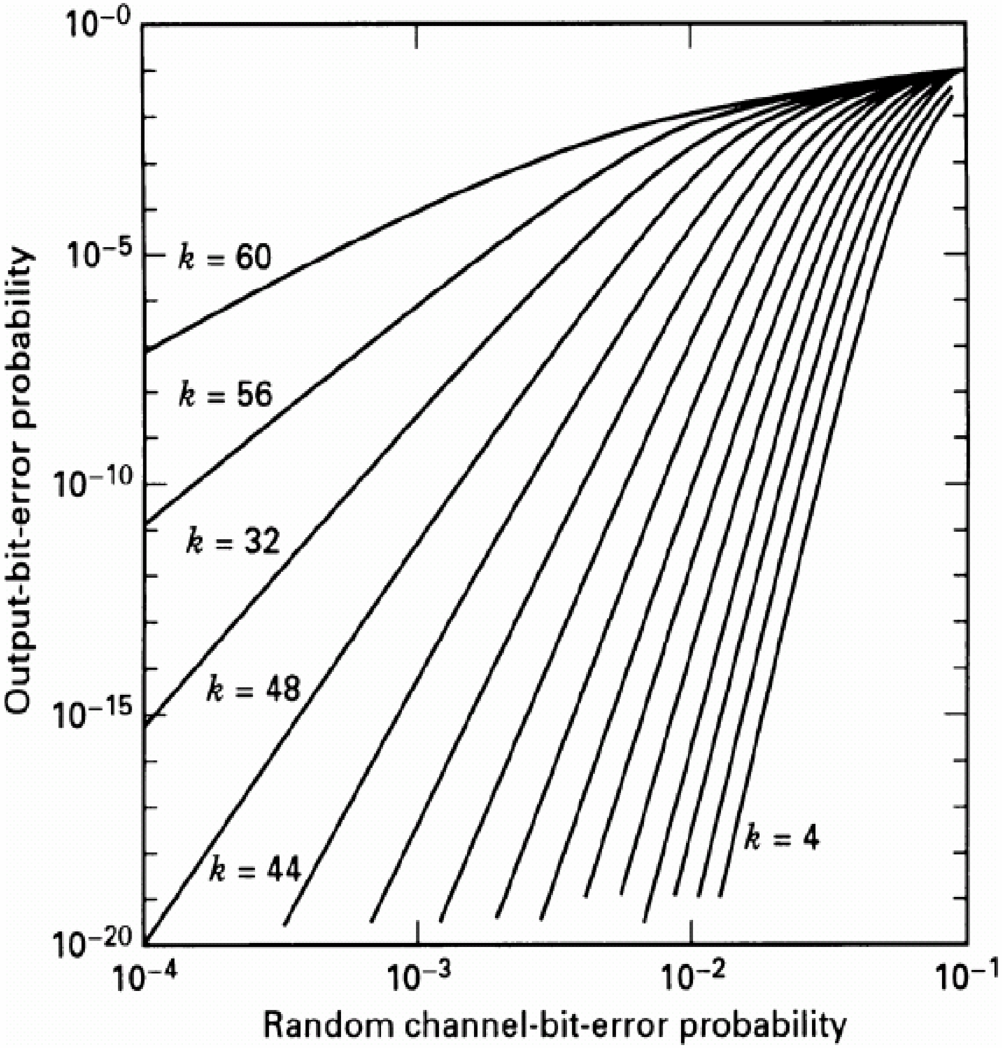

- Figure 1
  - Bit error probability versus symbol error for 32-ary orthogonal signaling and probability and $n$=31, $t$-error-correcting Reed-Solomon code

# Performance: Block Size




- Figure 2
  - Reed–Solomon, rate 7/8, decoder performance as a function of symbol size

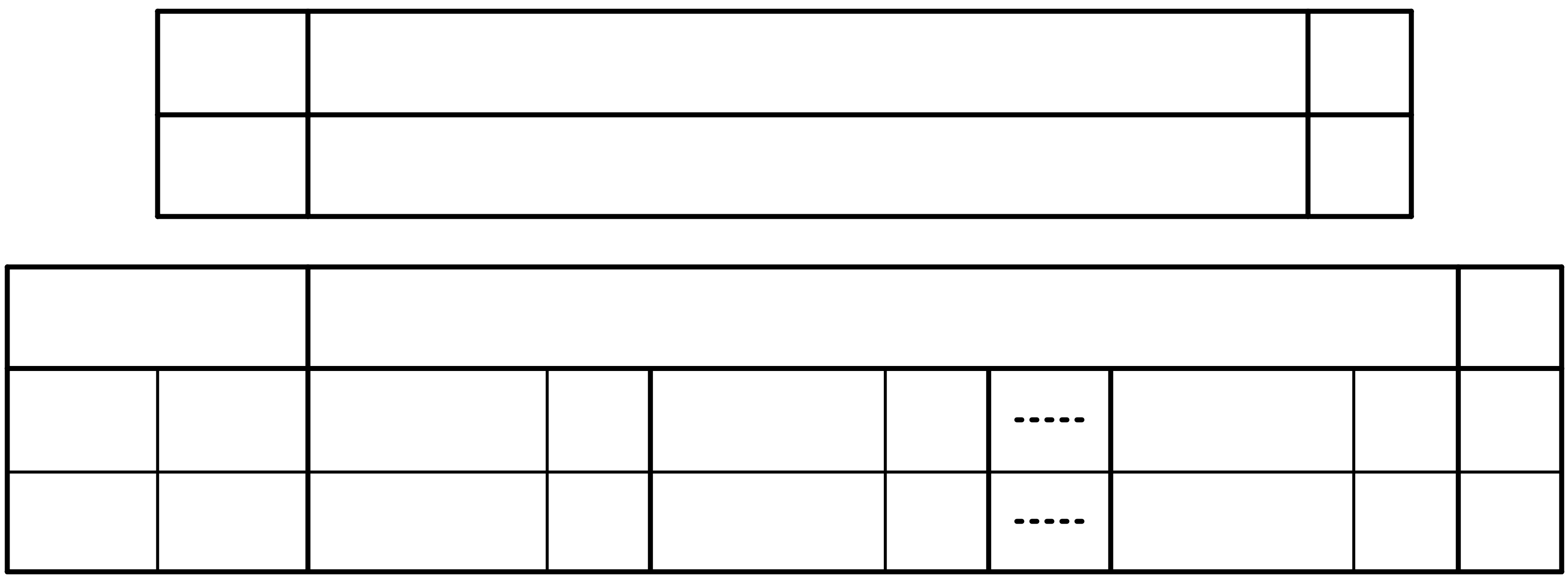
# Performance: Redundancy




- Figure 3
  - Reed–Solomon (64, $k$) decoder performance as a function of redundancy

# RS Coding for 802.11 MAC – Example (Not in standard) Ref: [Choi05]

- (224,208) shortened* RS over GF(256)
  - correcting 8 octet errors
- Up to 12 RS blocks in the frame boy



*(255, 239) original code

# Comments on FEC

- Good to reduce the latency compared to the ARQ
  - Retransmission = more delay
- But, FEC adds a fixed overhead to the link layer
  - When PHY delivers error-free frames to MAC, FEC implies the waste of the bandwidth
- Adaptive FEC? → AMC
  - Depending on the link condition
  - Depending on the traffic type

# Automatic Repeat reQuest (ARQ)

- Error detection and retransmission if transmission has failed
  - Provides in-order delivery of frames
- May be good if the transmission success ratio is highly time-varying
  - Ex 1–when FEC does not help when the channel is really bad while the channel can become good later
  - Ex 2–in Random Access channel, if the frames collide the FEC will not help; RA inherently assumes ARQ

# Error Detection

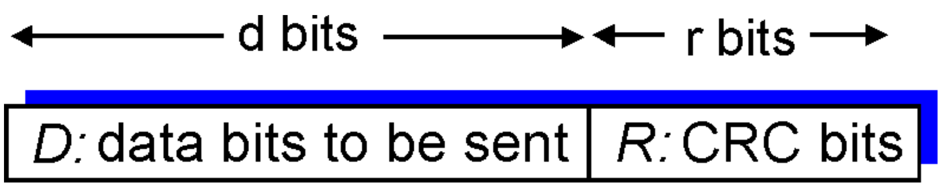
- Block code can be used with n−k parity bits
  - Note that one needs to detect errors first in order to invoke retransmission!
- Error detection may fail due to the incurred error pattern
  - The more parity bits, the less likely the error detection failure!
  - Parity bits are fixed overheads, so tradeoff

# Parity Check

- Parity bit appended to a block of data
- Even parity
  - Added bit ensures an even number of 1s
- Odd parity
  - Added bit ensures an odd number of 1s
- Example, 7-bit character [1110001]
  - Even parity [11100010]
  - Odd parity [11100011]

# Cyclic Redundancy Check

- view data bits, D, as a binary number
- choose r+1 bit pattern (generator), G
- goal: choose r CRC bits, R, such that
  - <D,R> exactly divisible by G (modulo 2)
  - receiver knows G, divides <D,R> by G. If non-zero remainder: error detected!
  - can detect all burst errors less than r+1 bits
- widely used in practice




*bit pattern*

$D * 2^r$ XOR R

*mathematical formula*

# CRC Example

Want:

$D \cdot 2^r$ XOR $R = nG$

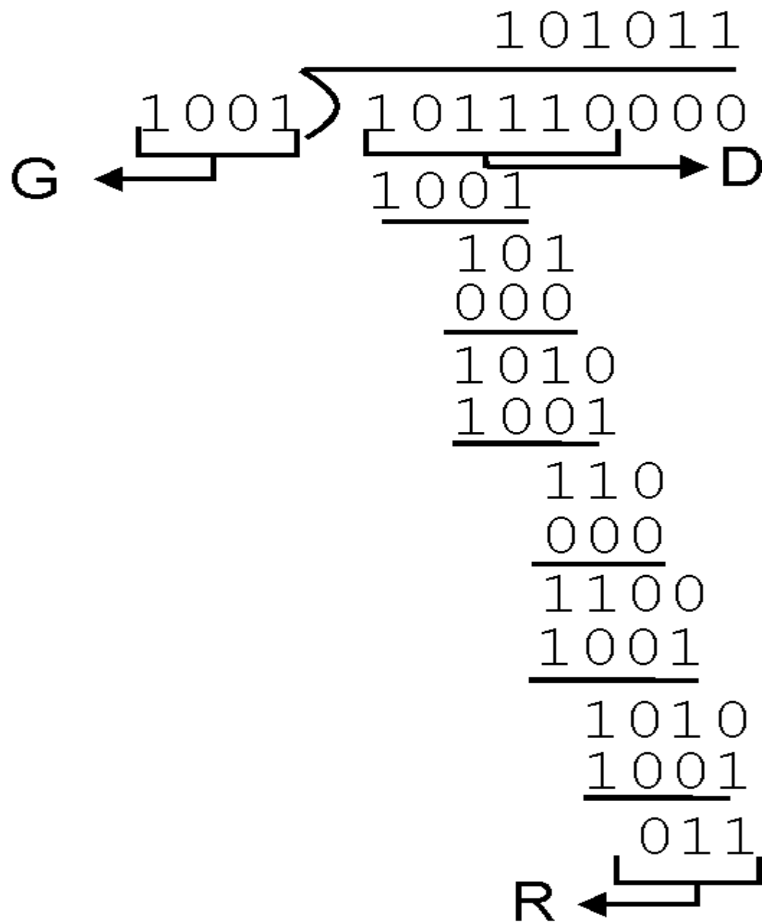
*equivalently:*

$D \cdot 2^r = nG$ XOR $R$

*equivalently:*

if we divide $D \cdot 2^r$ by G, want remainder R

$$R = \text{remainder}\left[\frac{D \cdot 2^r}{G}\right]$$



# CRC using Polynomials

- Widely used versions of *G(X)*
  - CRC–12
    - $X^{12} + X^{11} + X^{3} + X^{2} + X + 1$
  - CRC–16
    - $X^{16} + X^{15} + X^{2} + 1$
  - CRC – CCITT
    - $X^{16} + X^{12} + X^{5} + 1$
  - CRC – 32 (used by IEEE 802.11 MAC FCS)
    - $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X + 1$

# Digital Logic CRC

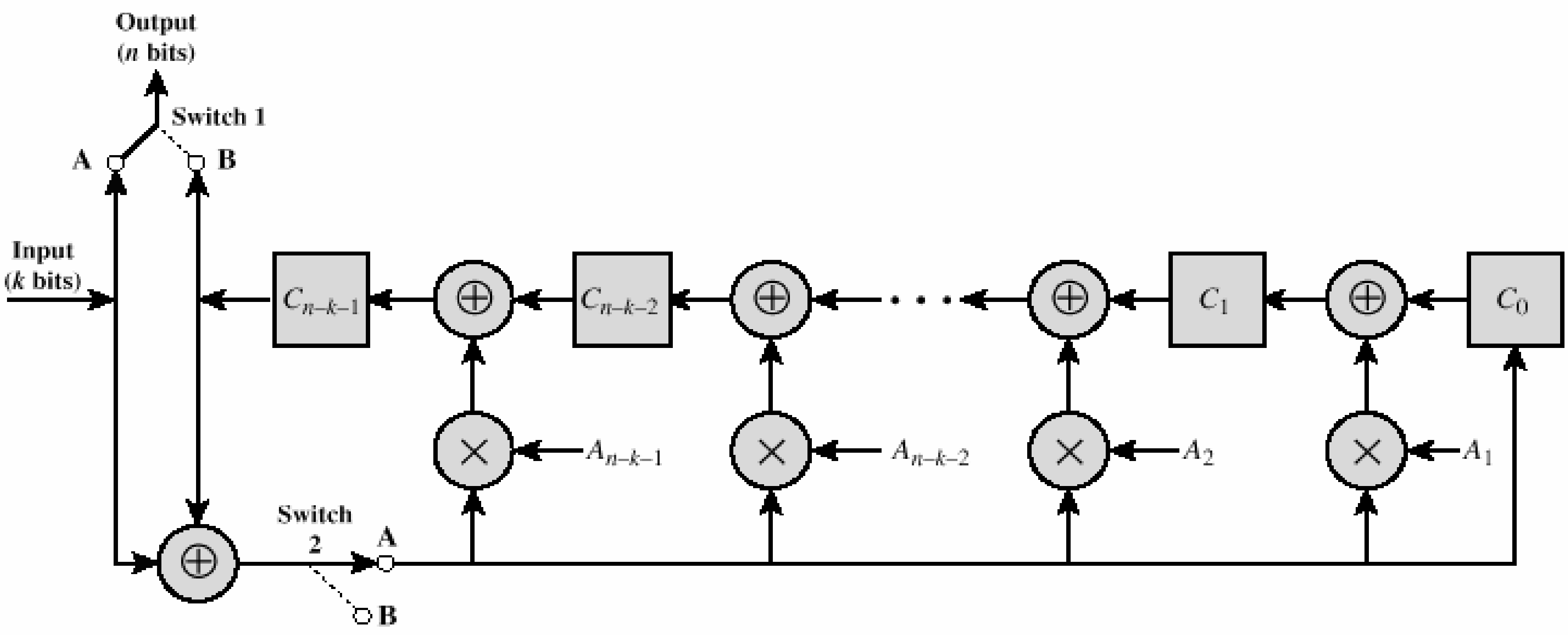



**Figure 8.4 General CRC Architecture to Implement Divisor**
$1 + A_1X + A_2X^2 + \ldots + A_{n-1}X^{n-k-1} + X^{n-k}$

# Types of ARQ Schemes

- Stop-and-Wait
- Go-Back-N
- Selective Repeat (SR)

# Stop-and-Wait ARQ

- Transmits a frame and waits for the corresponding ACK
  - Retransmit if ACK timeout
- Simplest with no buffering requirement
  - At the cost of low efficiency due to round-trip time and ACK overhead
- Used in IEEE 802.11 MAC
  - Shouldn't be too bad for one-hop link since the round-trip time shouldn't be a big deal!

# Go-Back-N ARQ

- Sliding window ARQ
  - ACK with Request Number RN acknowledges all frames prior to RN
  - N frames with Sequence Number SN (RN<=SN<=RN+N-1)
  - In the absence of ACK with Request Number > RN, the whole N frames are retransmitted
- Buffering requirement at sender side only
  - If no out-of-order reception possible
- Used in some L2 protocols, e.g., HDLC, and L4 TCP

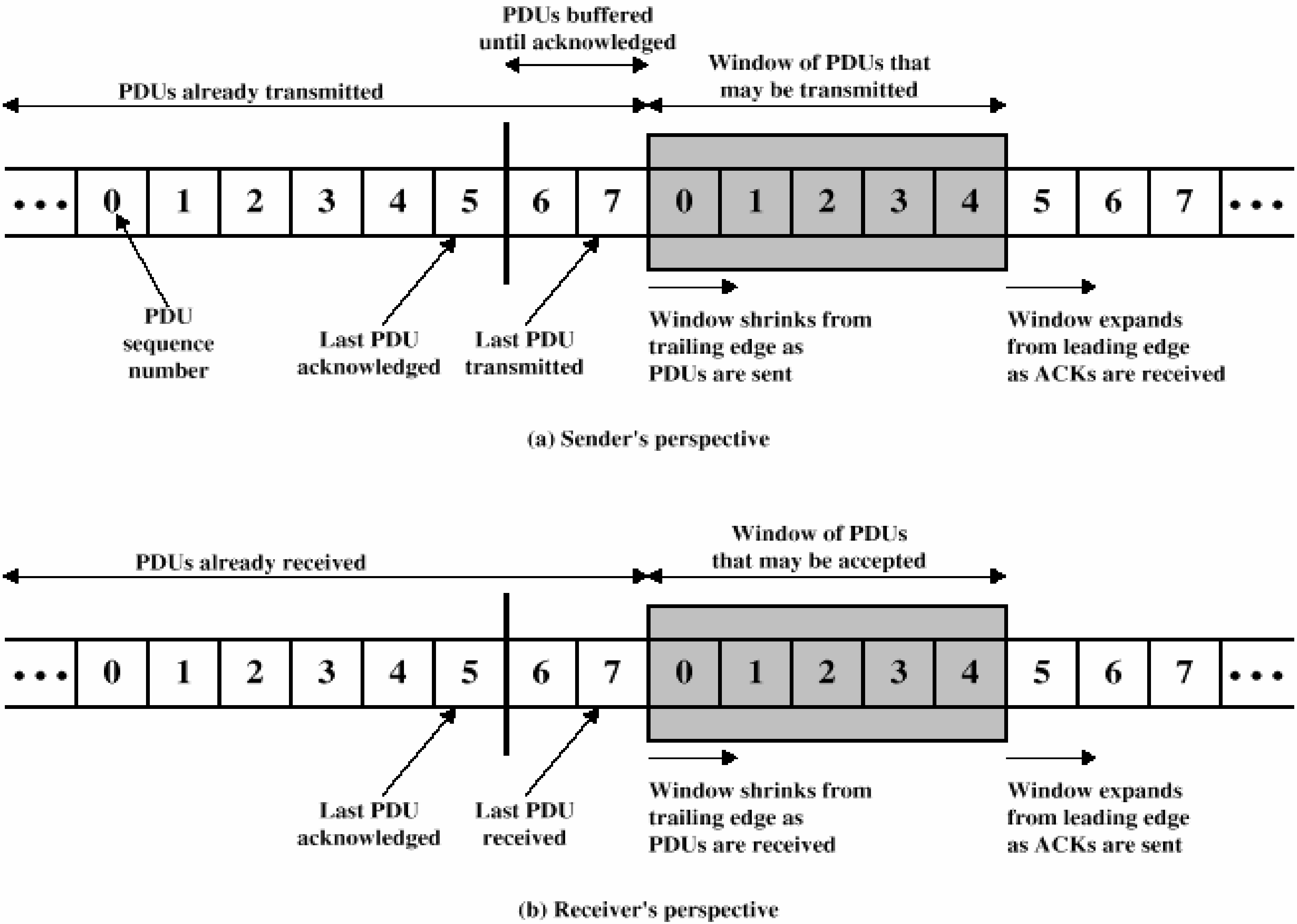
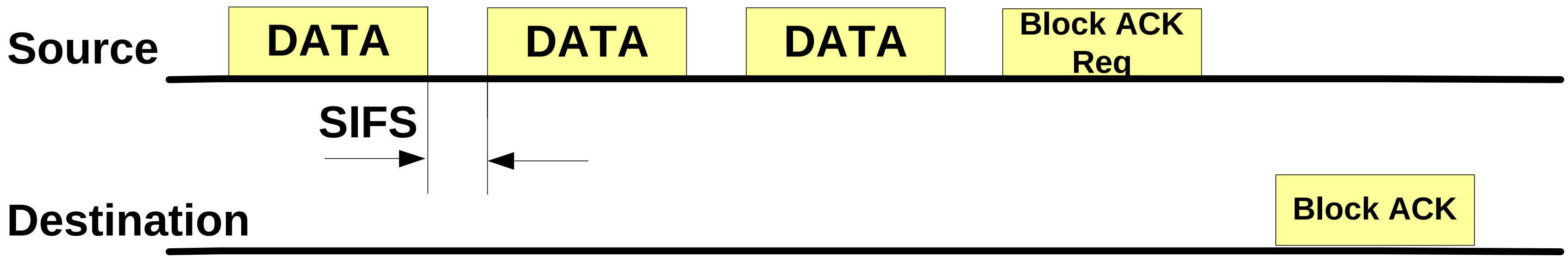



**Figure 8.17 Sliding-Window Depiction**

# Selective Repeat (SR) ARQ

- Again based on sliding window
- Receiver specifies which frames were received and which were missing
- Sender retransmits missing frames only
- Buffering requirement at both sender and receiver sides
- Most efficient ARQ scheme

# SR ACK: IEEE 802.11e Block ACK (based on IEEE 802.11e)




- Data frame header specifies whether the normal ACK or block ACK is used
- Block ACK can acknowledge up to 64 MSDUs, each may be up to 11 fragmented MPDUs

# Comments on ARQ

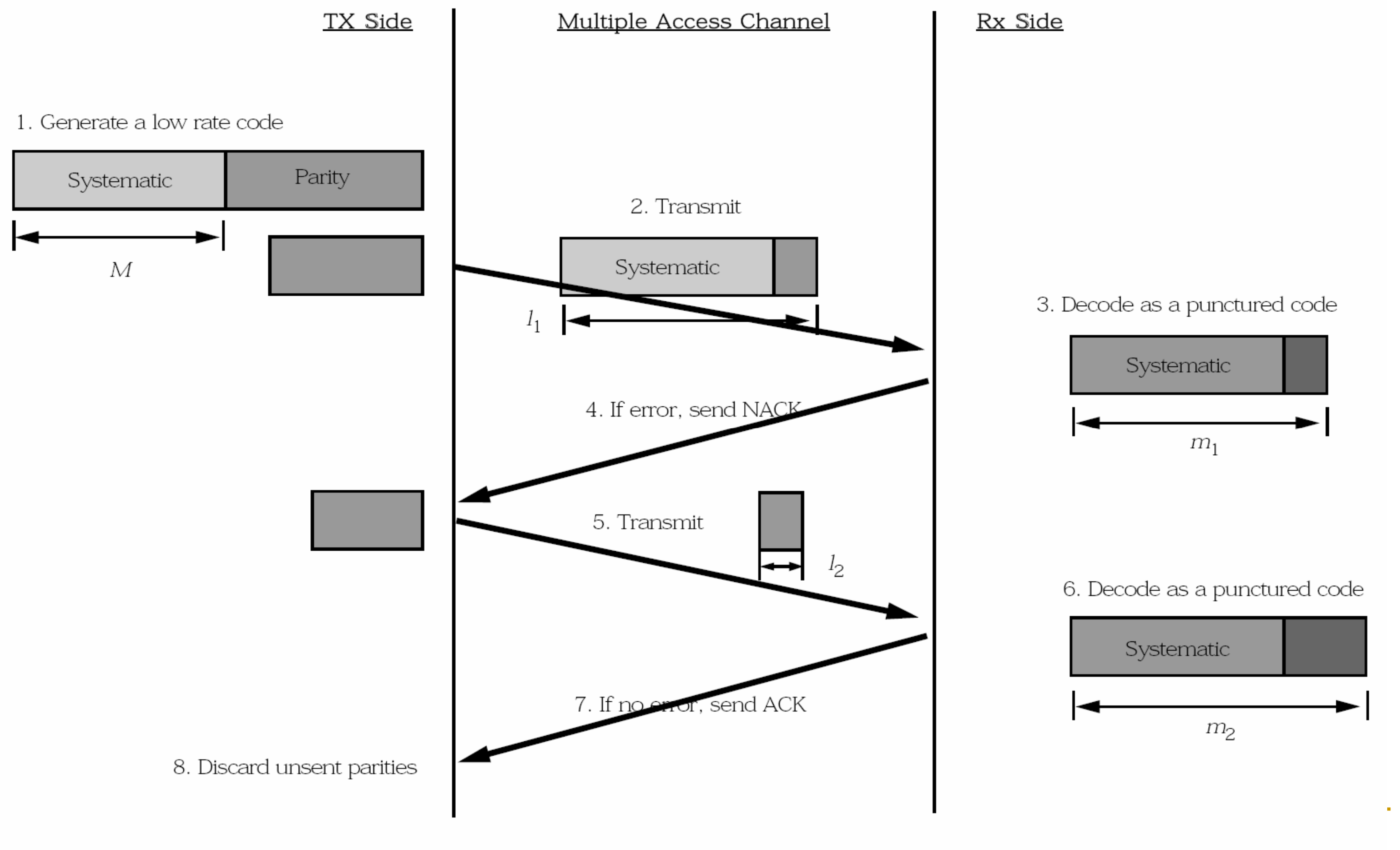
- Different ARQs for different environments
  - Error probability
  - Round-trip time
  - Complexity vs. performance tradeoff
- Feedback (i.e., ACK) transmission is not free
  - The ACK transmission mechanism is incorporated into the MAC protocol

# Hybrid ARQ (HARQ)

- Type I (Packet combining or Chase combining)
  - Retransmit the same information with original transmission
  - Combine the original and retransmitted version in soft state (i.e., before decoding)
  - Obtain SINR gain via MRC (Maximum Ratio Combining)

- Type II (Code combining or Incremental redundancy)
  - Using rate compatible punctured code, e.g., RCPC, RCPT
  - Retransmit only a part of redundancy bits, not information bits
  - Obtain time diversity similar to chase combining, and additional coding gain
  - Bandwidth efficient over chase combining

# Example of IR



# Discussion

- HARQ vs. AMC?
- Is HARQ part of PHY or MAC?
- How can we differentiate between PHY and MAC (H)ARQ schemes?

# Acknowledgement


- This material was partly adapted from
  - Computer Networking: A Top-Down Approach Featuring the Internet, 2nd Ed. by J. Kurose & K. Ross
  - Digital Communications: Fundamental and Applications, 2nd ed. by Bernard Sklar
  - Wireless Communications and Networks by W. Stallings
  - [Choi05] Sunghyun Choi, Youngkyu Choi, and Inkyu Lee, "IEEE 802.11 MAC-Level FEC with Retransmission Combining," *IEEE Trans. on Wireless Communications*, January 2006.