Data Link Control

Contents

Flow Control

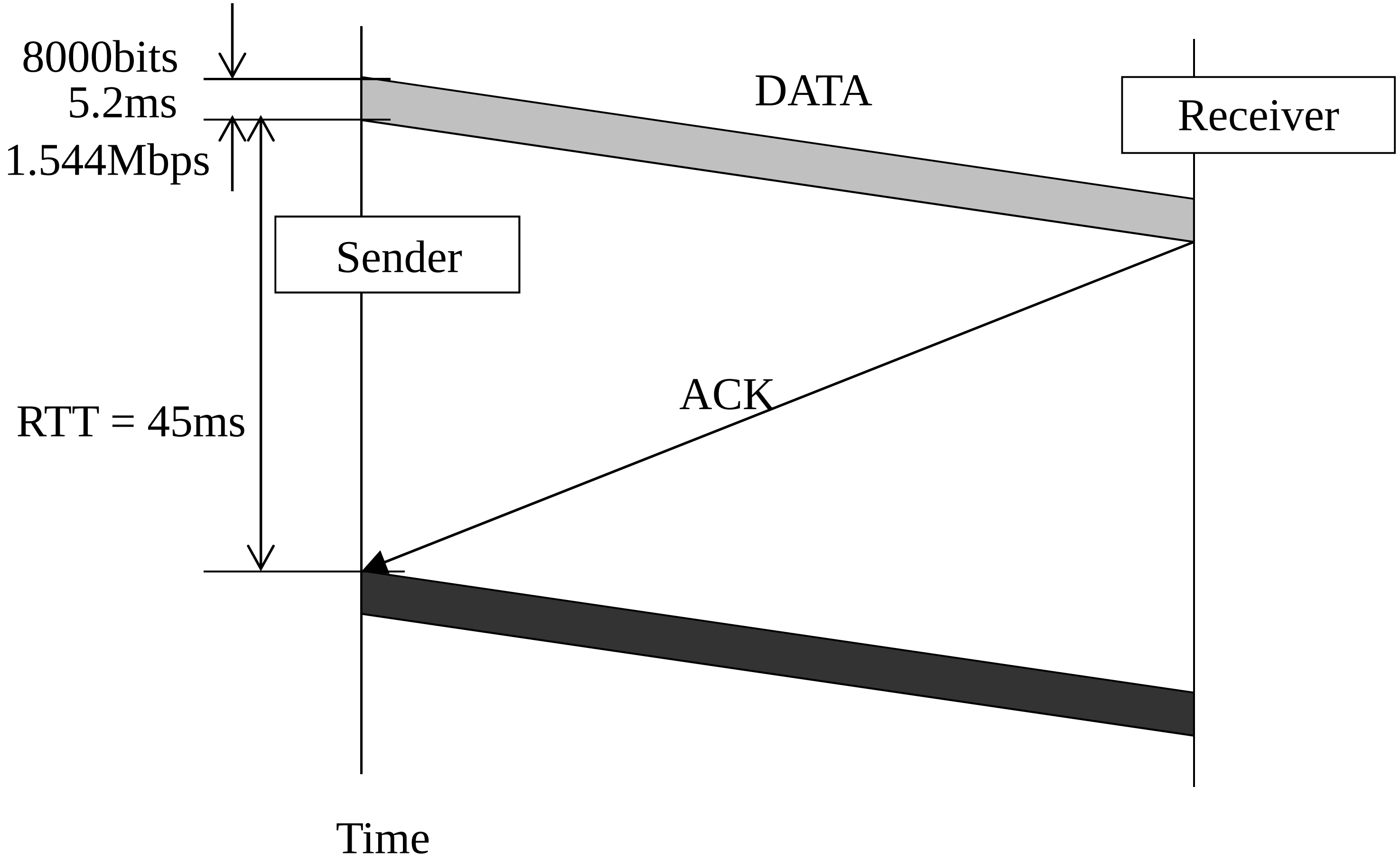
- Flow control is a technique for assuring that a transmitting entity does not overwhelm a receiving entity with data
- A fixed buffer is provided by the receiver
- Frame transmission
- Error-free flow control, arrive in the same order they are sent, each frame suffers delay

Flow Control: Stop-and-wait flow control

- A source must wait until it receives the acknowledgment (ACK) before sending the next frame
- A source breaks up a large block of data into smaller blocks and tx the data in many frames
  - Limited buffer size
  - Errors are detected sooner with smaller frames
  - On a shared medium (LAN), it's desirable not to permit one station to occupy the medium for an extended period

Stop-and-wait flow control

1.544Mb/s, RTT about 45ms; S&W; 1Kbyte frames



Flow Control: Stop-and-wait flow control

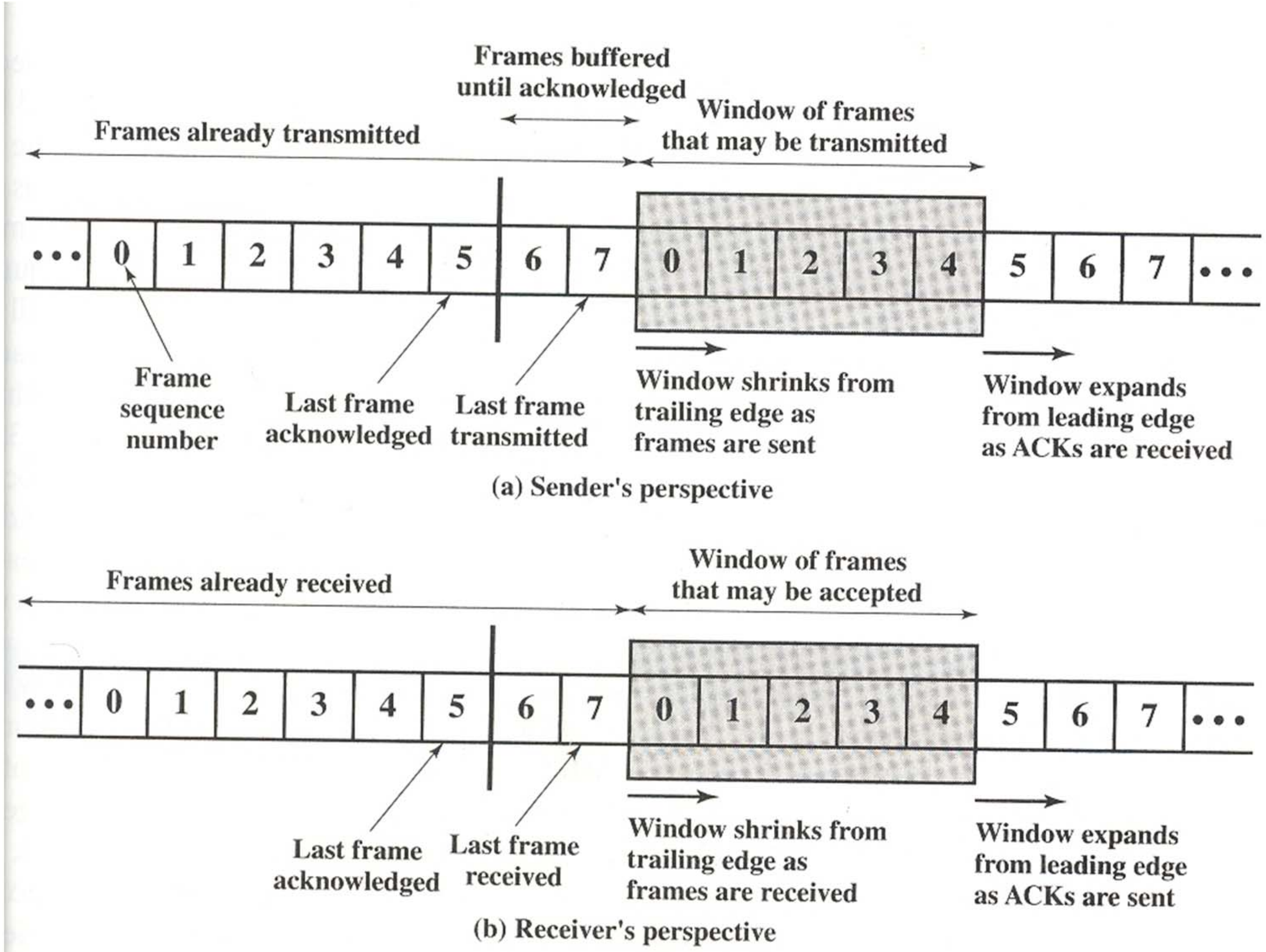
Inefficiency results from that only one frame at a time can be in tx

For very high data rates or for very long distances between sender and receiver, stop- and-wait provides inefficient line utilization

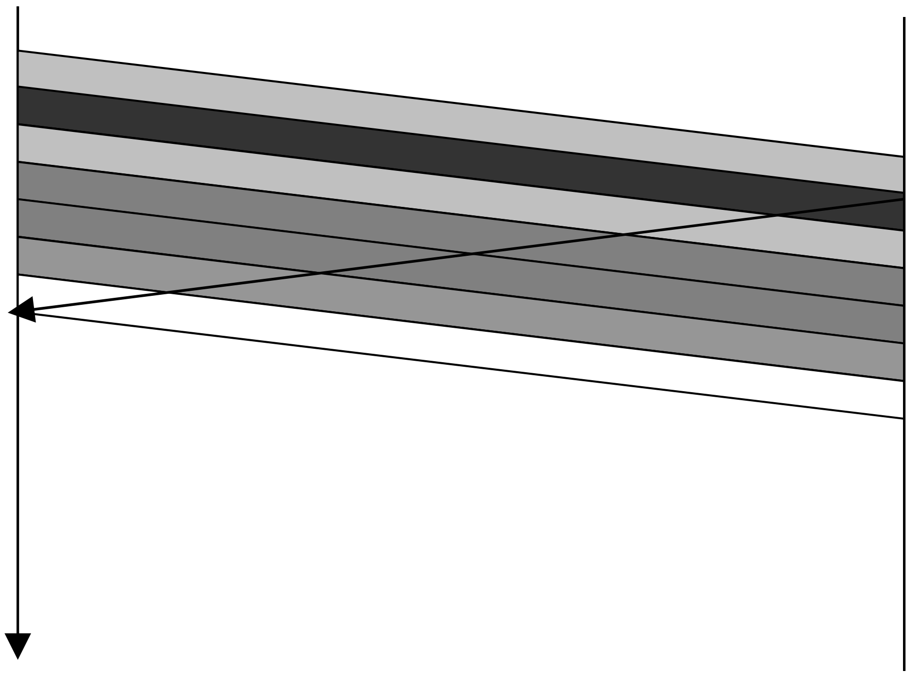
Flow Control: Sliding-window flow control

- 3-bit **sequence number** to tx 7 frames without an Ack: max window size = 7
- Each time a frame is sent the window shrinks; each time a new acknowledgment is received the window grows
- A station is allowed to completely cut off the flow of frames from the other side by sending a Receive Not Ready (RNR) message, which acknowledges former frames but forbids transfer of future frames
- On a satellite link a 7-bit sequence number, which allows up to 127 frames is used

Flow Control: Sliding-window flow control



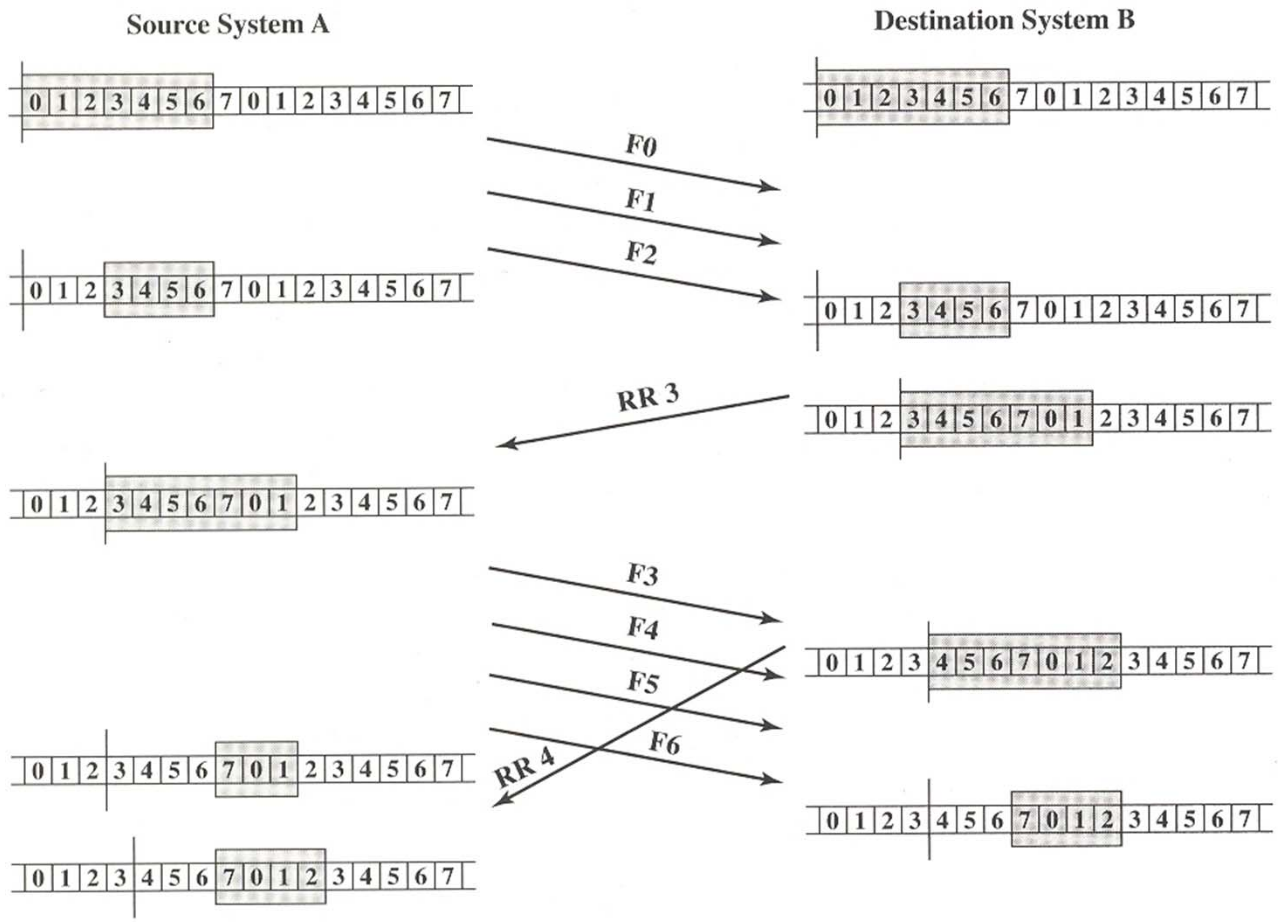
Sliding-window flow control:



Sends 7 frames in one RTT instead of 1

Time

Flow Control: Sliding-window flow control



Flow Control: Sliding-window flow control

If two stations exchange data, each needs to maintain two windows, one for transmit and one for receive

Piggybacking: each data frame includes a sequence number of that frame plus a field that holds the sequence number used for acknowledgment

Error Detection/Correction

The Need for Error Control

- The ability to control errors is an increasingly important task of a data communication system
- Uncorrected and undetected errors can degrade performance, response time, and possibly increase the need for intervention by human operator
- Powerful and efficient error-control technique enables us to discard or downgrade the most expensive and troublesome element in a tx system

  ex) In satellite communications

Parity Checks

The simplest error-detection scheme is to append a parity bit to the end of block of data

even parity – synchronous transmission

odd parity – asynchronous transmission

In ASCII transmission, a parity bit is attached to each 7-bit ASCII character

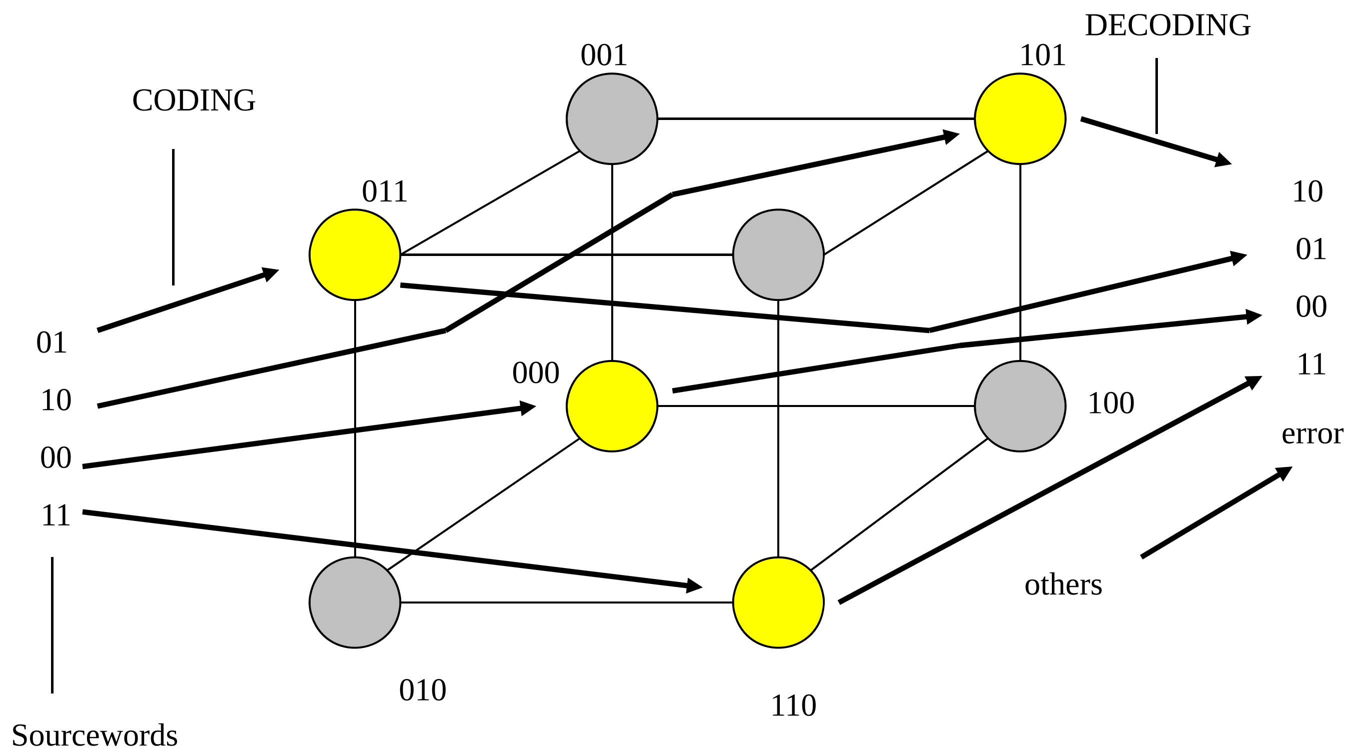
G: 1110001 → 11100011 (odd parity)

Errors in any odd number of bits – detectable

Errors in any even number of bits – undetectable

Examples of Codes - Parity Bit

Code separates the transmitted words



Cyclic Redundancy Check

More powerful and efficient than the simple parity bit

Used For synchronous tx

The message is treated as one long binary number

On tx: This number is divided by a unique prime binary number, and the remainder is attached to the frame as a FCS to be tx

On reception: The receiver performs the same division, using the same divisor

If there is no remainder, the receiver assumes there was no error

Cyclic Redundancy Check

- No errors if the frame including the FCS is divisible by the prime divisor
- The most used divisor: 17-bit divisor (a 16-bit remainder) and 33-bit divisor (a 32-bit remainder)
- This is a very powerful means of error detection and requires very little overhead

Examples of Codes - CRC

M = [10010001…10110] Message (sourceword)

C = [10010001…10110<u>010..110011</u>] Codeword

Extra bits = CRC bits

CRC bits (r bits) calculated from M so that

C = A*G where

G = given r+1 bit word

= generator of code (standardized)

* = operations modulo 2 without carry

CRC: Calculation

7 6 5 4 3 2 1 0

$M = 10011010 = x^7 + x^4 + x^3 + x^1$

2 1 0

$\underline{R} = \underline{101}$

... 7 6 5 4 3 2 1 0

$[MR] = 10011010\underline{101} = M*x^3 + R$

$G = \underline{1101} = x^3 + x^2 + 1$

$M*x^3 = 10011010000 = x^{10} + x^7 + x^6 + x^4$

Find A and R (3bits) so that $M*x^3 = A*G + R$

Then $[MR] = M*x^3 + R = A*G + R + R = A*G$

CRC: Calculation

M = 10011010

G = <u>1101</u>

Find A and R (3bits)

so that $M*x^3 = A*G + R$

Long division of $M*x^3$ by G

[Operations mod2, no carry]

```
G
↓        11111001
1101 | 10011010000        M*x^3
       1101
       ----
        1001
        1101
        ----
         1000
         1101
         ----
          1011
          1101
          ----
           1100
           1101
           ----
              1000
              1101
              ----
               101
```

CRC: G(x)

G(x) is standardized to be small but typically produce remainders. Detects:

- all single bit errors
- all double-bit errors if G(x) has a factor with at least 3 terms
- any odd number of errors, if (x+1) divides G(x)
- any burst error of length < length of FCS
- most large burst errors

CRC: Standard G

CRC-8: 100000111

CRC-10: 11000110011

CRC-12: 1100000001111

CRC-16: 11000000000000101

CRC-CCITT: 10001000000100001

CRC-32: 100000100110000010001110110110111

Internet Checksum Algorithm

Used in IP, ICMP, TCP, UDP, …

Algorithm: 1’s complement of the 1’s complement sum of data interpreted 16 bits at a time

1’s complement addition is “end-round-carry” addition. Why?

2’s complement carry is a zero-crossing; account for -0 by adding one

Ones Complement Arithmetic

Example with 4-bit integers

1's comp of 0101 (5) = 1010 (-5)

1's comp of 0011 (3) = 1100 (-3)

1's comp of 1010 (-5) + 1100 (-3) = ?

First obtain 2's comp sum (carry is not ignored) of
0101 + 0011 = 1000

Then add the carry from the most significant bit (no carry in this example) = 1's comp sum = 1000

Obtain 1's comp of 1000 = 0111 (-8)

Internet Checksum

Message: e3 4f 23 96 44 27 99 f3

2’s comp sum is: 1e4ff

1’s comp sum is: e4ff + 1 = e500

So, Internet cksum is 1aff

Note that message + cksum = ffff

Thus, cksum (message + cksum) = 0000

Error Control

Two type of errors:

- Lost frame: A frame fails to arrive at the other side
- Damaged frame: A recognizable frame does arrive, but some of the bits are in error

Error control is based on

- Error detection
- Positive acknowledgment (ACK)
- Retransmit after timeout
- Negative acknowledgment (NAK) and retransmit

Error Control

These mechanisms are all refereed to as ARQ (automatic repeat request)

Stop-and-Wait ARQ

Based on the stop-and-wait flow-control

Go-back-N ARQ

Error control based on the sliding window flow control

Error Control

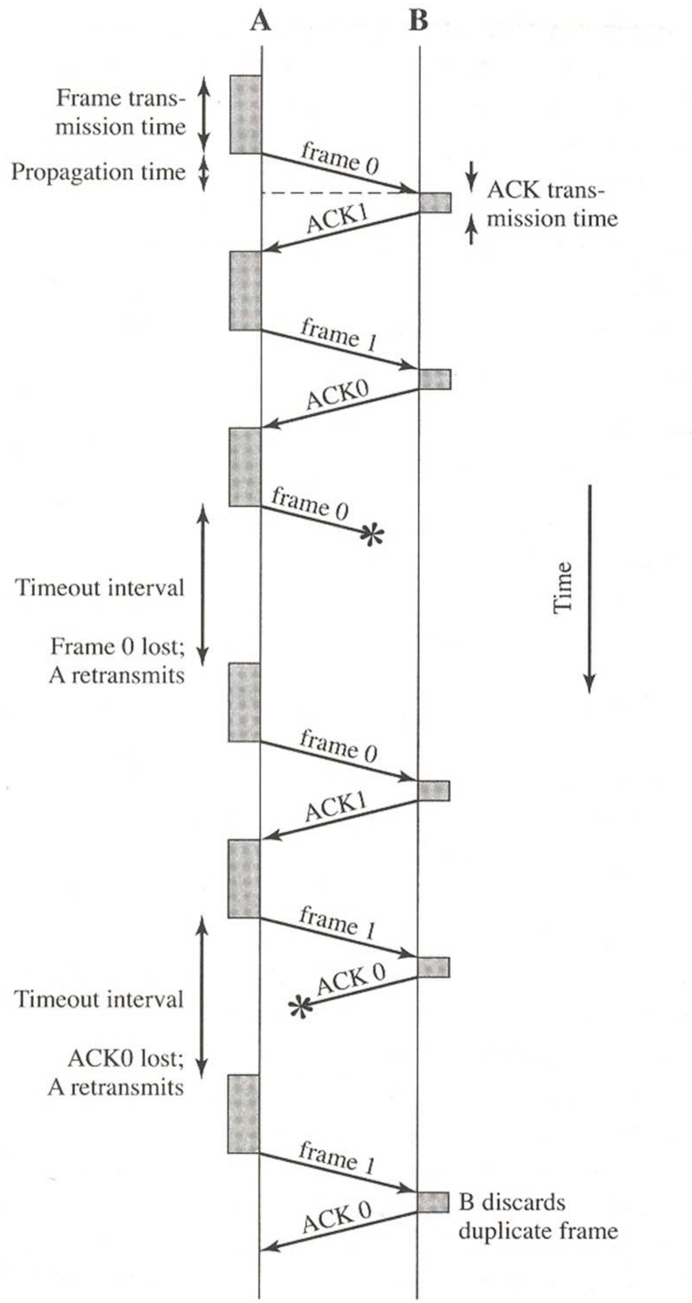


Figure 7.5 Stop-and-Wait ARQ

Error Control

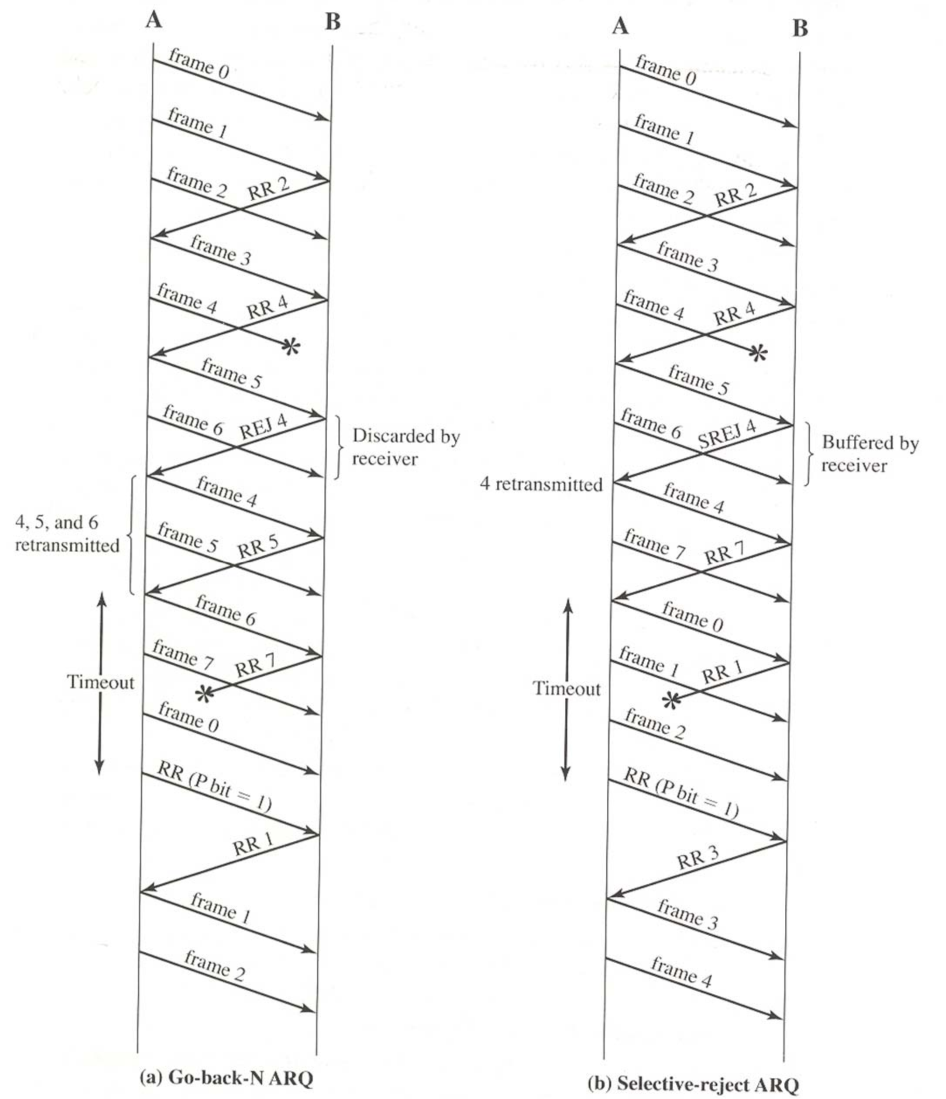


Figure 7.6 Sliding-Window ARQ Protocols

Error Control

Selective Reject ARQ

- The only frames that are retransmitted are those that receive negative acknowledgement or those that time out
- The receiver must maintain a buffer large enough to save post-SREJ frames until the frame in error is retransmitted
- Much less used than go-back-N ARQ

Link Control (Layer 2)

Data link control package is used for flow control, error control, and others

These packages are generally used only for synchronous tx

The data link control module organizes the data into a set of frames

Each frame is supplemented with control bits that allow the two sides to cooperate to deliver the data reliably

Link Utility

U = (time to tx a frame)(# of frames)/total time spent

Links can be 100% efficient if the link is unreliable (no error control required)

Efficiency drops due to propagation times in each direction and channel errors

Link Utility

Stop-and-Wait Flow Control (HDX)

T = Total time to send data

T_F = Time to send one frame and receive an ack

$T = n\,T_F$

$T_F = t_{proc} + t_{frame} + t_{prop} + t_{proc} + t_{ack} + t_{prop}$

$t_{proc} \cong 0,\ t_{ack} \cong 0$

$U = nt_{frame} / n(2t_{prop} + t_{frame})$

$= t_{frame} / (2t_{prop} + t_{frame})$

$= 1/(1+2a)$

where $a = t_{prop} / t_{frame}$

Link Utility

$a = t_{prop} / t_{frame}$

$= (d/v)/(L/R)$

Link Utility

Error-Free Sliding-Window Flow Control

FDX point-to-point line

$U = 1$: $W \geq 2a+1$

$U = W/(2a+1)$: $W < 2a+1$

W: Maximum window size

$\mathbf{W = 2^n - 1}$

W = 1: stop-and-wait

W = 7: adequate for many applications

W = 127: high-speed WANs, satellite links

Link Utility

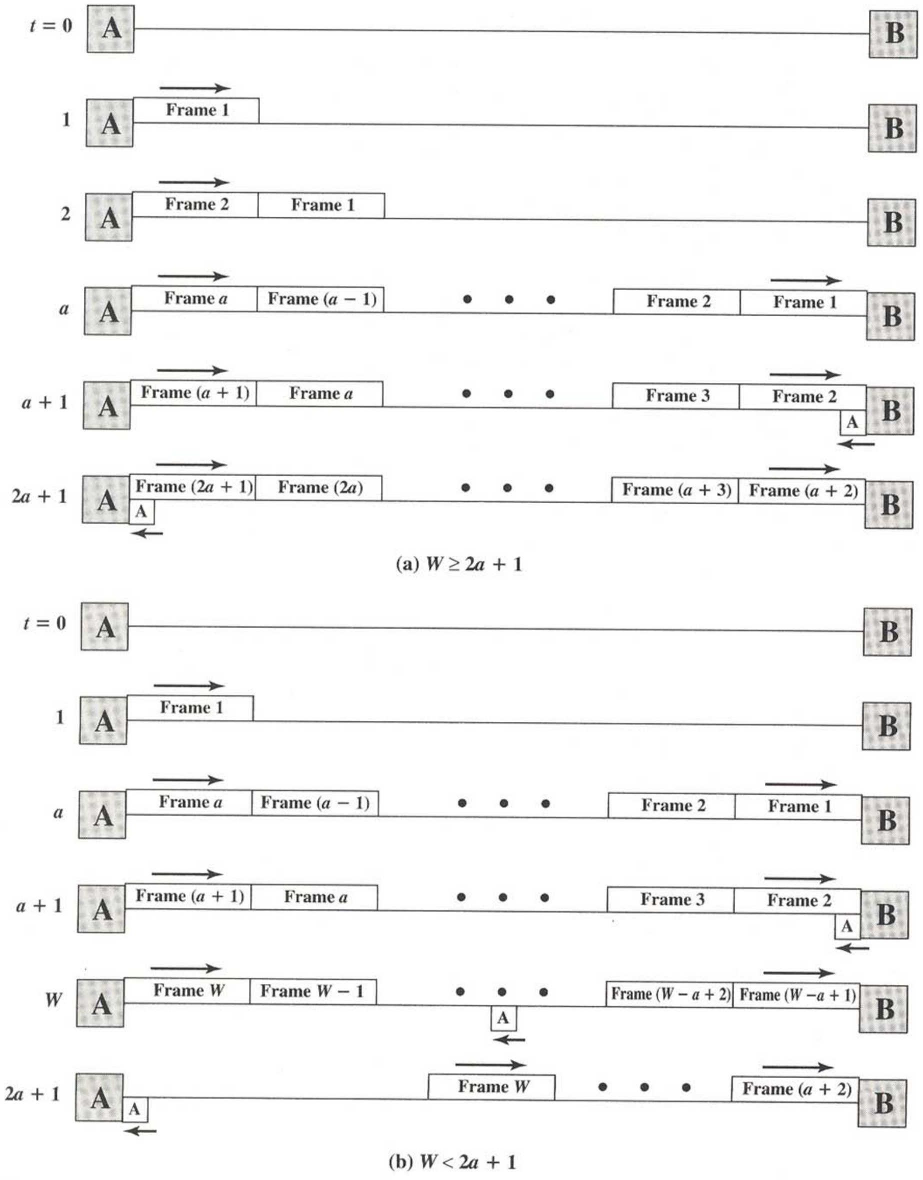


Figure 7.11 Timing of Sliding-Window Protocol

Summary

Flow Control:

- Stop and Wait (HDX)
- Sliding window (FDX)

Error Detection/Correction: Parity bits, CRC, Internet Checksum

Error Control:

- Stop-and-Wait ARQ: timeout interval
- Go-back-N ARQ: RR, REJ
- Selective Reject ARQ: RR, SREJ

Link Performance (Utility)

- **Throughput = (Link capacity) x (Utility)**