Flow Control and

Error Control

EP1100 Data Communication and Computer Networks

Illustrations in this material are collected from

Behrouz A Forouzan, *Data Communications and Networking*, 3rd edition, McGraw-Hill.

Outline

- Introduction
- Flow control
  - Stop and wait
  - Sliding window
- Error detection
  - Parity
  - Checksums
  - Cyclic redundancy check (CRC)
- Error handling
  - Error correction
  - Retransmission (*Automatic Repeat Request* ARQ)
    - Stop and wait
    - Go-back-N
    - Selective reject (selective repeat)

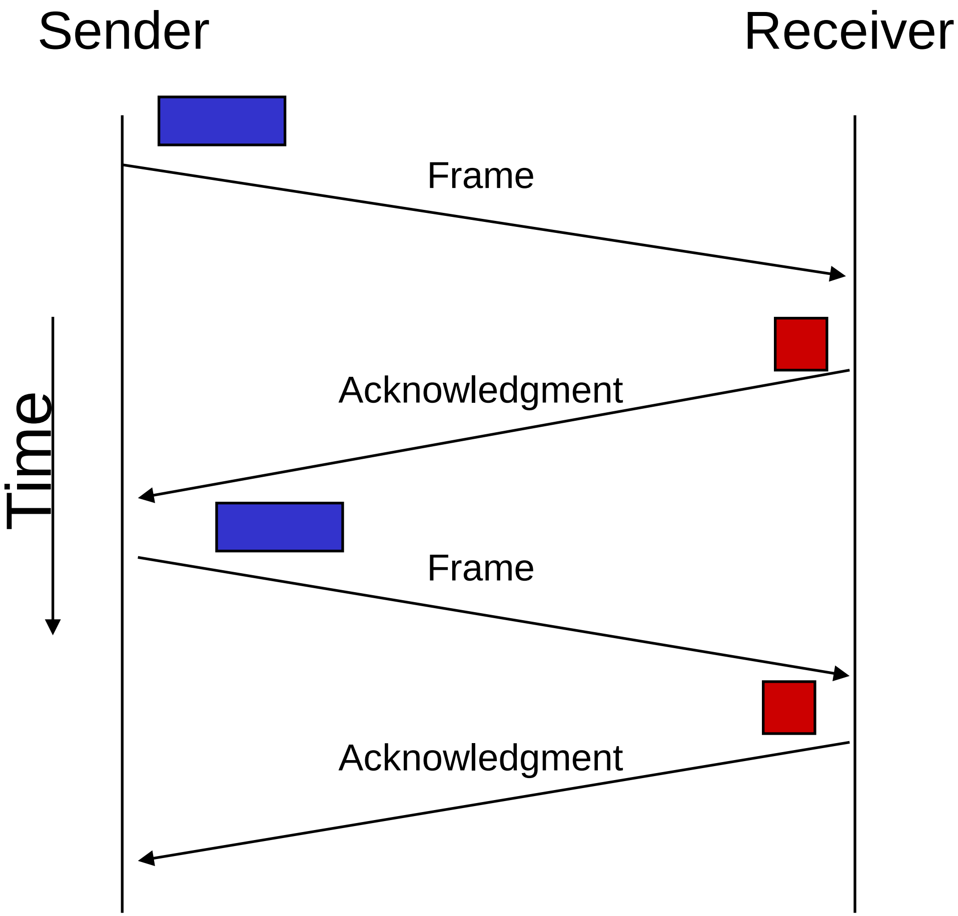
Data Link Layer: Background

- Physical layer provides means to transfer *frames* over a link
- Remaining problems to be solved
  - Adapt sending rate to receiving rate
  - Detect errors in frames and loss of frames
  - Manage errors and losses
  - Multiplexing of sources to shared link

Why Flow Control?

- Problem: Sender can overload receiver
  - Frames arrive too fast
    - In many cases, the receiver is more complicated than the sender
    - Error detection, frame/packet analysis, address lookup
  - Frames are stored in a buffer before they are processed
    - Receiver buffers can overflow and frames be lost
  - Prevent loss of frames
- Control mechanisms
  - Stop and wait
  - Sliding window
- We don’t worry about frame errors and loss for now
  - Will discuss that shortly...

Stop and Wait



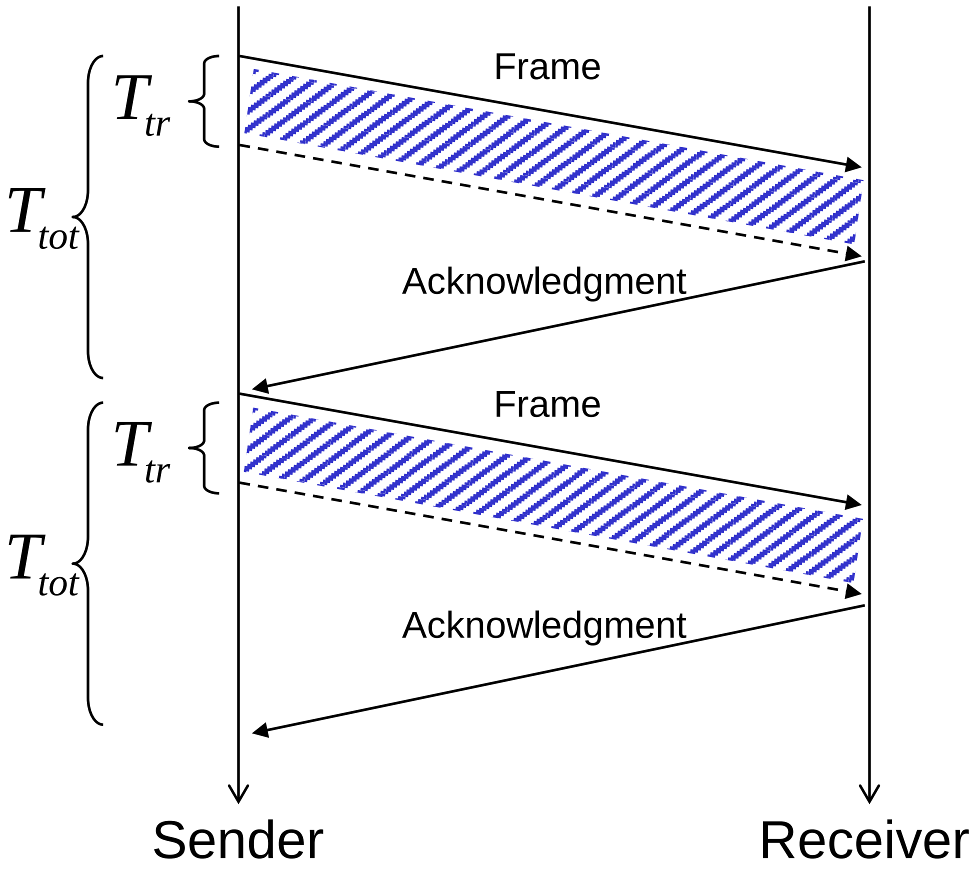
- Sender sends one frame
  - Waits for acknowledgment
- Receiver sends acknowledgment
  - When ready to receive next frame
- Next frame sent after acknowledgment

Link Utilization

T_{tr} Transmission time

T_{tot} Total time

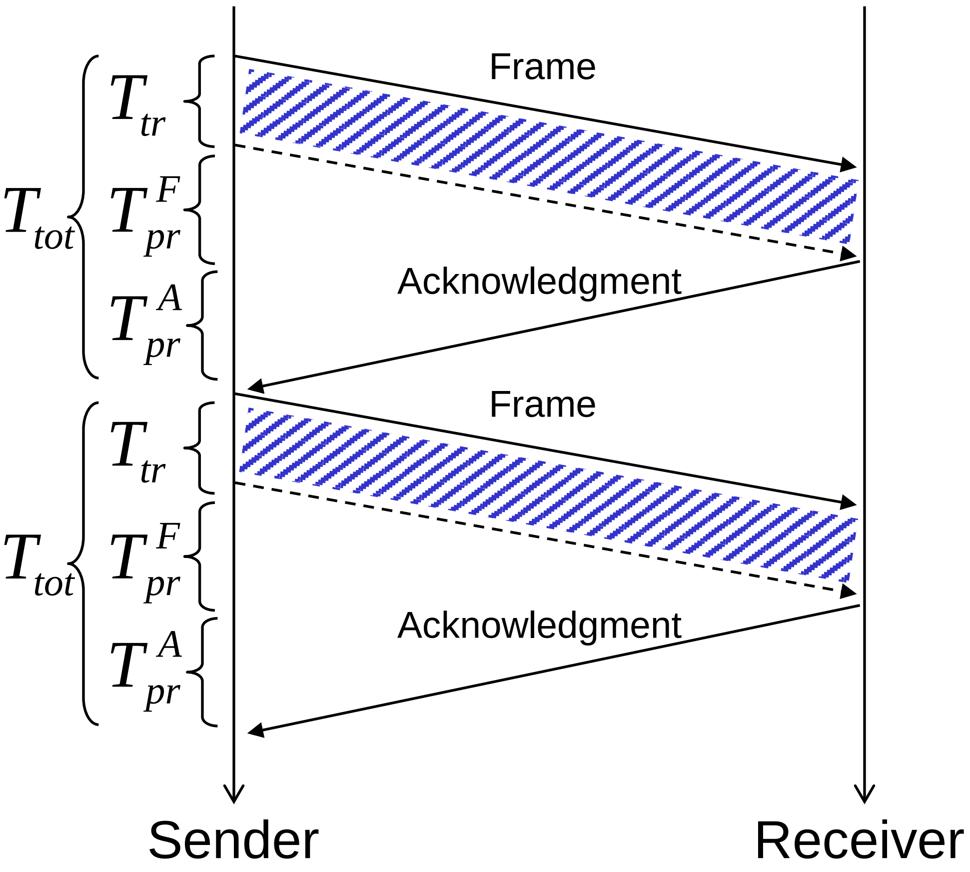
$$U = \frac{T_{tr}}{T_{tot}}$$



- Transmission time
  - Time between first and last bit of a frame
    - Frame length (bits) divided by link capacity
    - [b] / [b/s] = [s]
- Total time
  - Time from first bit is sent until acknowledgment arrives
    - How long before sender can start sending again
- Assumptions
  - Zero transmission time for acknowledgments
  - Zero processing time in sender and receiver

Link Utilization

$$U = \frac{T_{tr}}{T_{tr} + T_{pr}^{F} + T_{pr}^{A}}$$



- Propagation time
  - Time for one bit to propagate over the link
    - Link length divided by propagation speed
    - [m] / [m/s] = [s]

T_{pr}^{F} Propagation time for frame

T_{pr}^{A} Propagation time for acknowledgment

Link Utilization—Symmetrical Links

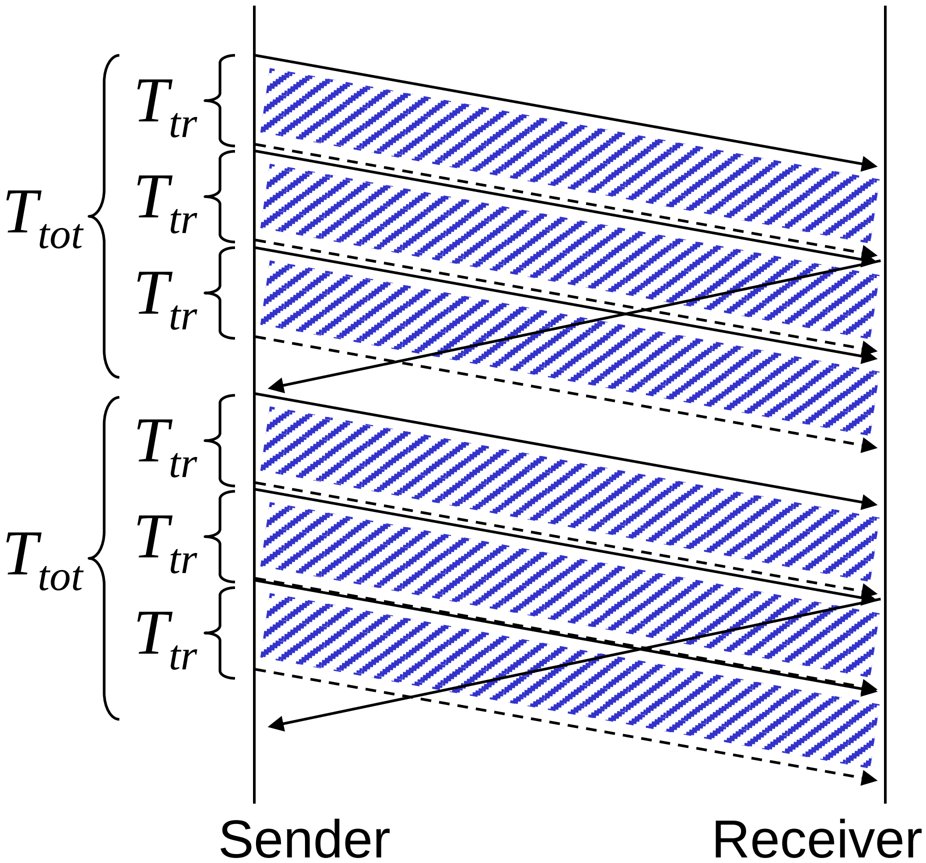
For symmetrical links:

$$T_{pr}^{F} = T_{pr}^{A} = T_{pr}$$

$$U = \frac{T_{tr}}{T_{tr} + 2T_{pr}} = \frac{1}{1+2a}, \text{ where } a = \frac{T_{pr}}{T_{tr}}$$

- The parameter a is the ratio between the length of the link and length of a frame (in time)
  - $T_{pr} < T_{tr}$ $(a < 1)$: less than one frame fits on the link
  - $T_{pr} \geq T_{tr}$ $(a \geq 1)$: one or more frames on the link
- ”Length” of a bit
  - Signal propagation speed divided by link capacity
    - [m/s] / [b/s] = [m/b]
  - Speed of light in optical fiber is about 2×10^8 m/s
    - 1 kb/s: 200 km per bit
    - 1 Mb/s: 200 m per bit
    - 1 Gb/s: 20 cm per bit

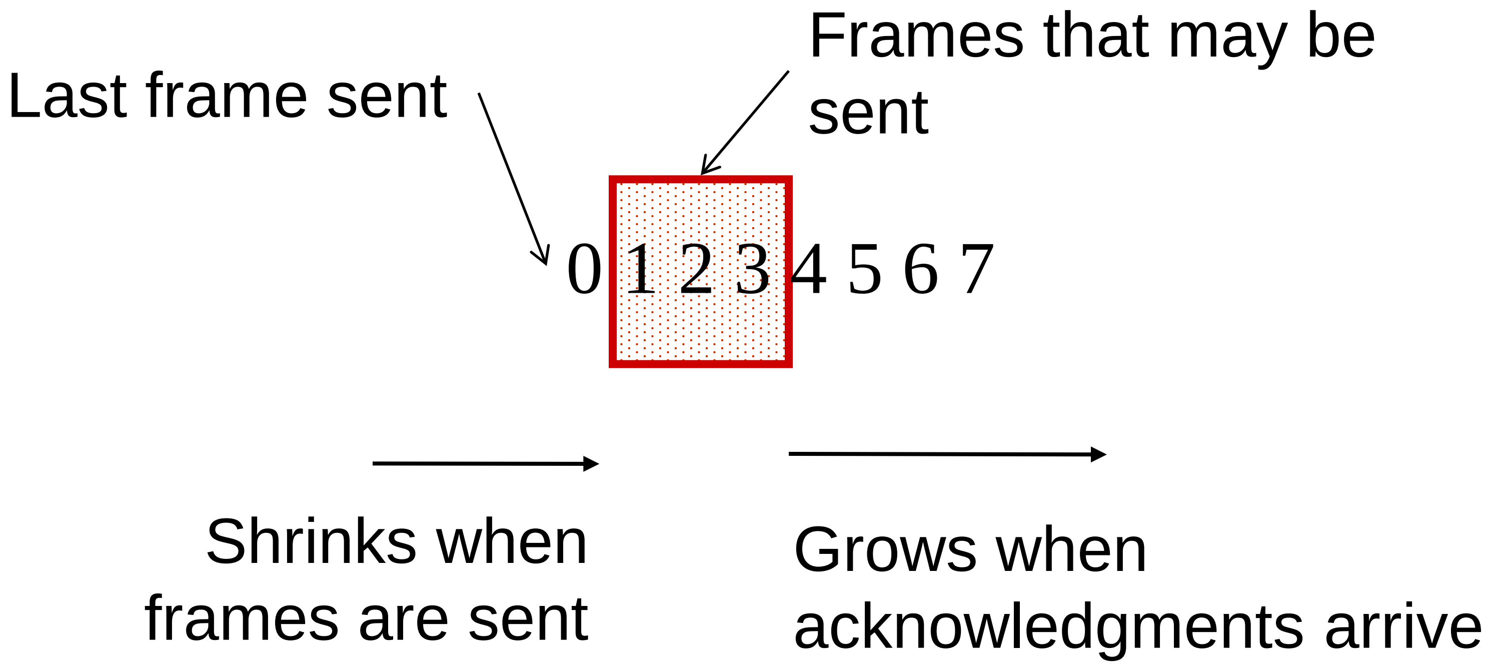
Sliding Window



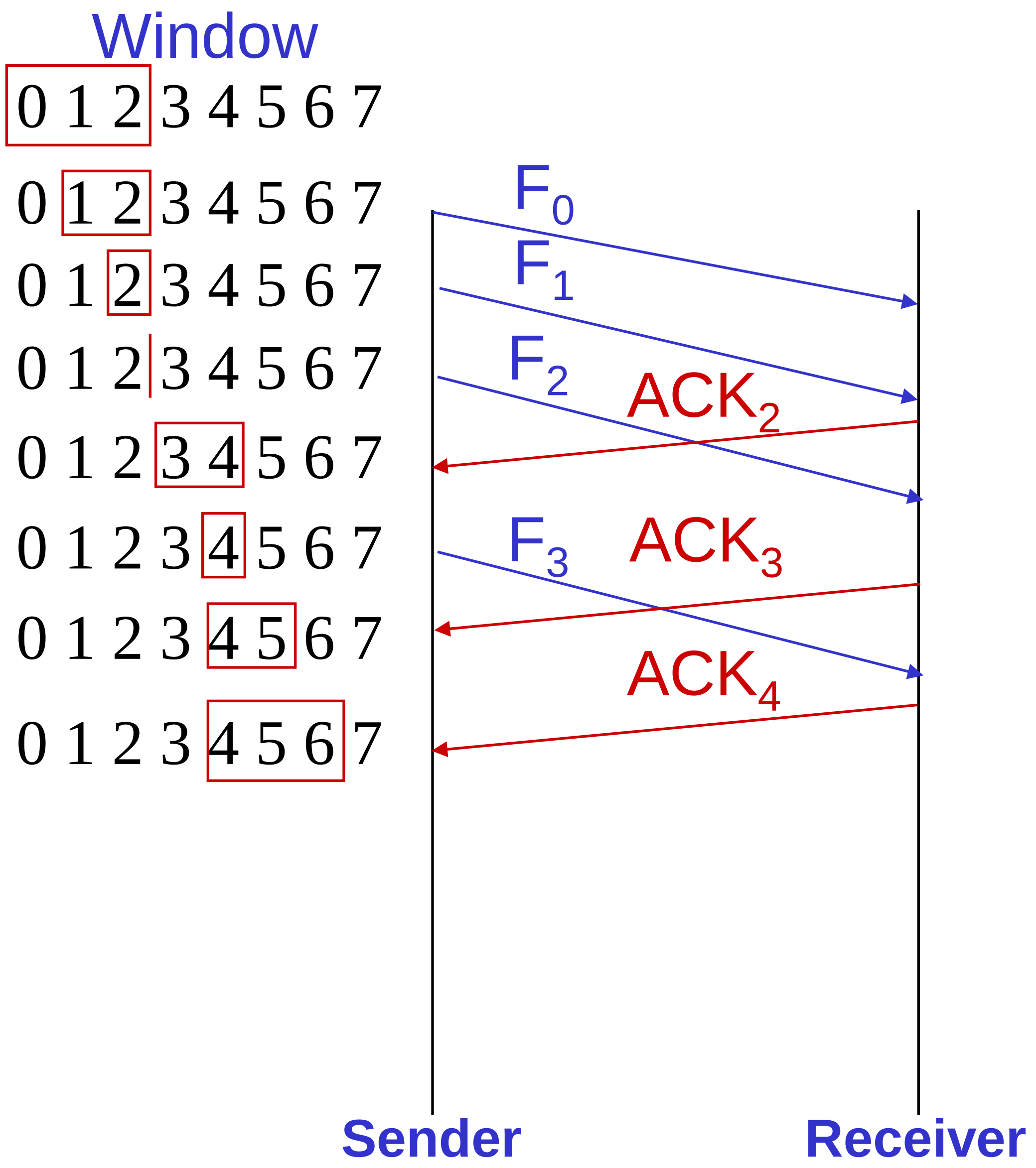
- Send *N* frames before waiting
  - *N* is ”window size”
    - Matched by buffer space at receiver
  - Choose N to get utilization close to 100%
- Sequence numbers
  - All frames, both data and ACKs
  - Acknowledgments
    - Sequence number of the next frame the receiver is ready to accept
    - ACK of all frames with lower sequence numbers

How Does it Work?

At the sender ($N = 3$)

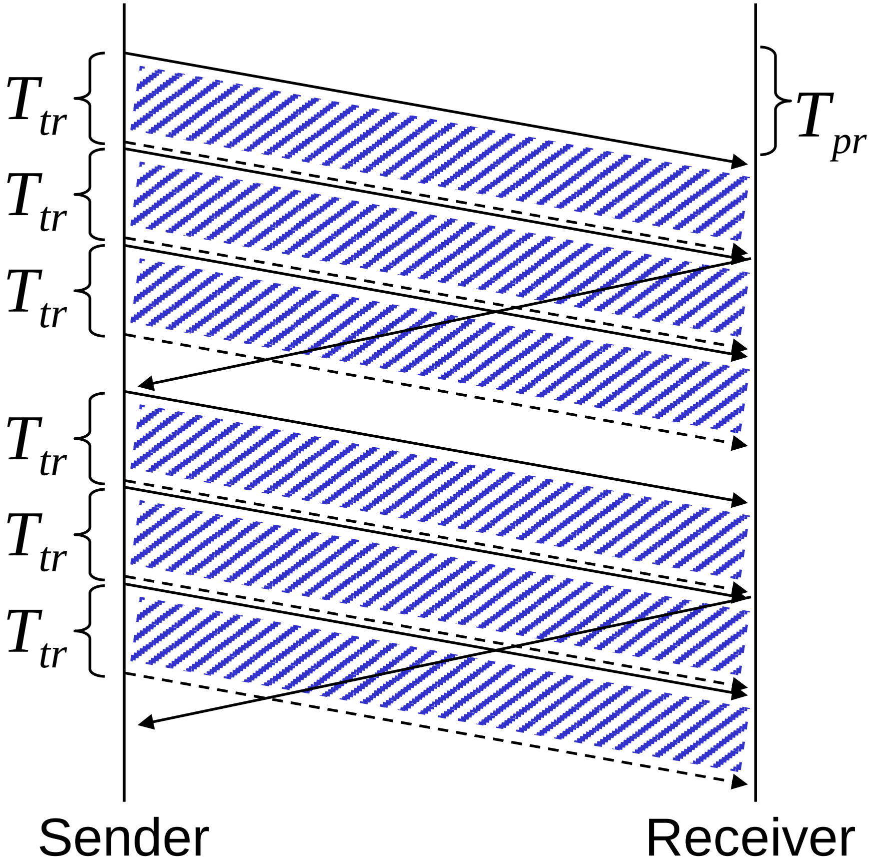


Example (N = 3)


Window
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
F_0
F_1
F_2
ACK_2
F_3
ACK_3
ACK_4
Sender
Receiver

Sliding Window Utilization

$$U = \frac{NT_{tr}}{T_{tr} + 2T_{pr}} = \frac{N}{1+2a}$$



- $U \geq 1$
  - Sender receives acknowledgment before window is closed
  - Sender may send without stopping
  - True utilization can never be more than 100%
- $U < 1$
  - Window closes after N frames
  - Sender must stop and wait for acknowledgment
  - Utilization is the fraction of the time when the sender does not wait

How Large Window?

- $N = 1 \Rightarrow$ stop-and-wait
- Small $a \Rightarrow$ small N
  - Local area network: $N = 8 \Rightarrow 3$ bits
- Large $a \Rightarrow$ large N
  - TCP uses 32-bit sequence number
    - Byte number
    - Propagation times for global distances

Acknowledgments

- Types of acknowledgments
  - Positive
    - ACK (*acknowledgment*)
      - — receiver ready
  - Negative
    - NACK (*negative acknowledgment*)
      - — receiver not ready
- Indicates sequence number of next expected frame
  - Cumulative acknowledgment of all frames with lower sequence numbers
- When and how is the acknowledgment sent?
  - As a separate frame
  - Together with data from the receiver to the sender
    - “Piggybacking”

Error control

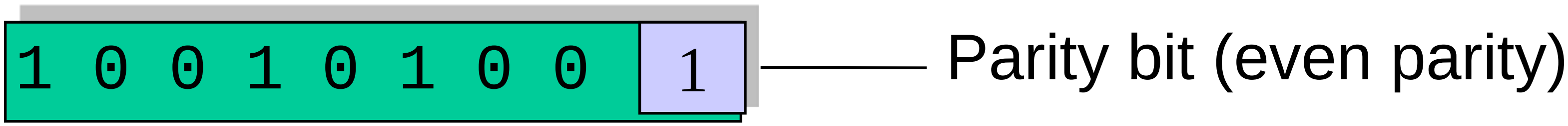
Transmission Errors

- Lost frame
    - Framing error
- Corrupted frame (bit errors)
    - Single bit error
    - Burst errors
        - Whole sequences of bits are corrupted
        - External noise, for example power surges
    - Must be discarded! Why?

Error Detection—Basic Idea

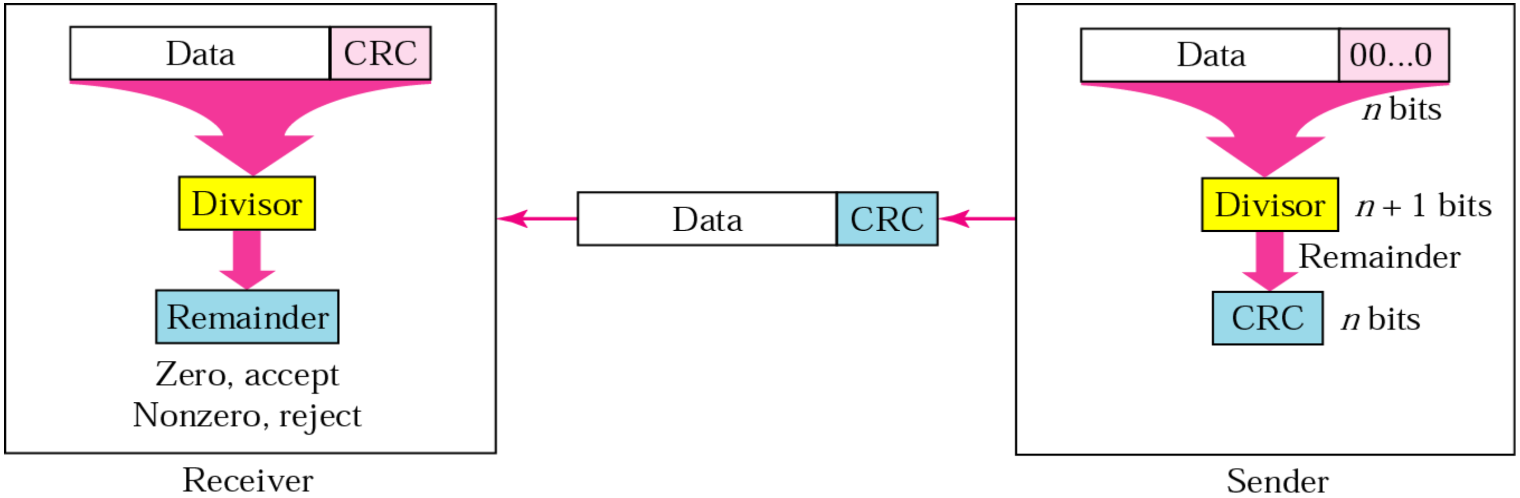
- Add extra (redundant) information for detecting errors
  - Parity check
  - Checksum
  - Cyclic redundancy check (CRC)
- Sender computes function over data, and appends result
- Receiver computes same function, and compares the results
- If the results differ, there was an error

Parity Check



- Simple parity check
  - Extra bit (parity bit) is added to the data unit
  - Numbers of 1s should be even (”even parity”) or odd (”odd parity”)
  - Receiver checks the number of 1s
- Advantages
  - Simple: $P = 1 \oplus 0 \oplus 0 \oplus 1 \oplus \ldots \oplus 1 \oplus 0$ for even parity
  - Inexpensive: only one extra bit per data unit
- Disadvantage
  - Only detects single bit errors, and burst errors with odd number of bit errors

Cyclic Redundancy Check (CRC)



- The data *M* is treated as a sequence of bits
- Predefined binary word *P* (generator) of length $n+1$
- Sender generates *M'* by adding *n* CRC bits to *M*
  - Such that *M'* is evenly divided by *P*
  - *M'* is sent
- Receiver receives *M''*
  - If remainder of *M''* divided by *P* is zero then *M''* = *M'*
  - Otherwise: bit error detected, discard the data

CRC Calculation Using Binary Division

- Append ’00’ to *M*
- Binary subtraction (xor) of 3 bits
  - If first bit is ’1’
    - subtract *P*
    - (Put ’1’ in quotient)
    - Copy down next bit
  - If first bit is ’0’
    - subtract ’000’
    - (Put ’0’ in quotient)
    - Copy down next bit
- Append remainder to data as checksum

```
          10110
    101 | 1001100
          101
           011
           000
            111
            101
             100
             101
              010
              000
               10
```

$n = 2$

$P = 101$

$M = 10011$

$M' = 1001110$

CRC Control at Receiver

- Divide received data with *P*
- If remainder is '00', data is OK
  - Strip off CRC bits
- Otherwise discard data

```
          10110
     -----------
101 | 1001110
      101
      ---
       011
       000
       ---
        111
        101
        ---
         101
         101
         ---
          000
          000
          ---
           00
```

$n = 2$

$P = 101$

$M = 10011$

$M' = 1001110$

Generator Polynomials

- Binary numbers can be represented as polynomials
  - Bit value is coefficient of a term
  - Exponent indicates the bit position, starting at 0
  - Example: 100111 $\Rightarrow$

$P(X) = 1 \times X^5 + 0 \times X^4 + 0 \times X^3 + 1 \times X^2 + 1 \times X^1 + 1 \times X^0$

$P(X) = X^5 + X^2 + X + 1$

- Standard polynomials

ITU-16: $X^{16} + X^{12} + X^5 + 1$

ITU-32: $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$

CRC

- Effective error detection
  - All burst errors that affect an odd number of bits
  - All burst errors of length less than or equal to degree of polynomial
  - With high probability longer errors
- Simple implementation in hardware
  - Shift register circuit
  - CRC often appended to the data (trailer)

Checksum

- Treat the data as a sequence of integer numbers in binary format
- Compute the sum of the integer numbers
  - (In one's complement arithmetic)
  - Use the result for error detection

Checksum

- Less effective than CRC
  - Easier to implement in software
- Detects
  - all errors involving an odd number of bits
  - Most errors involving an even number of bits
    - Two opposite bit inversions may balance out each other
- Used in IP, TCP and UDP

Correction of Errors

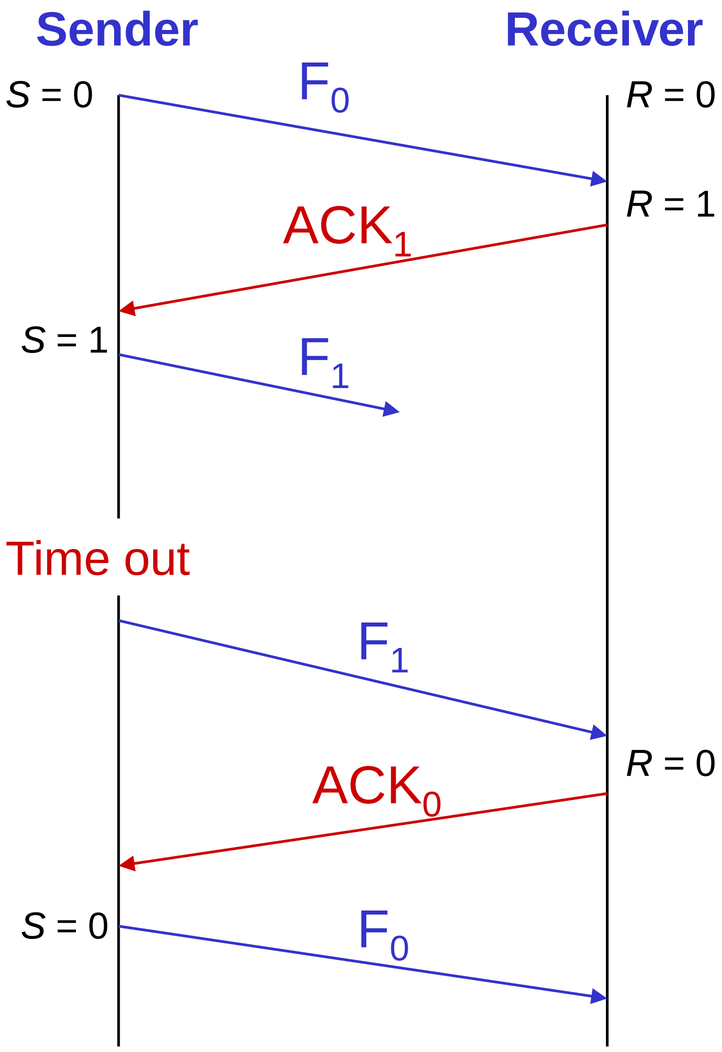
- Forward Error Correction (FEC)
  - Error-correcting codes
  - Replace CRC, checksum etc with a code that can automatically correct the error
  - Needs more redundancy bits
- Retransmission (ARQ)
  - Can be used both for bit errors and frame loss
  - A frame with bit errors is dropped (lost)
- Hybrid ARQ
  - Include FEC
  - Correct error if possible, retransmit otherwise

Automatic Repeat Request (ARQ)

- Error control—when frames or acknowledgments are lost
  - Assume that receiver detects and discards frames with errors
- Based on flow control to combine error and flow control
  - Stop-and-wait flow control
    - Stop-and-wait ARQ
    - “Alternating Bit Protocol”
      - Two sequence numbers—0 and 1
  - Sliding window flow control
    - Go-back-N ARQ
    - Selective-reject ARQ

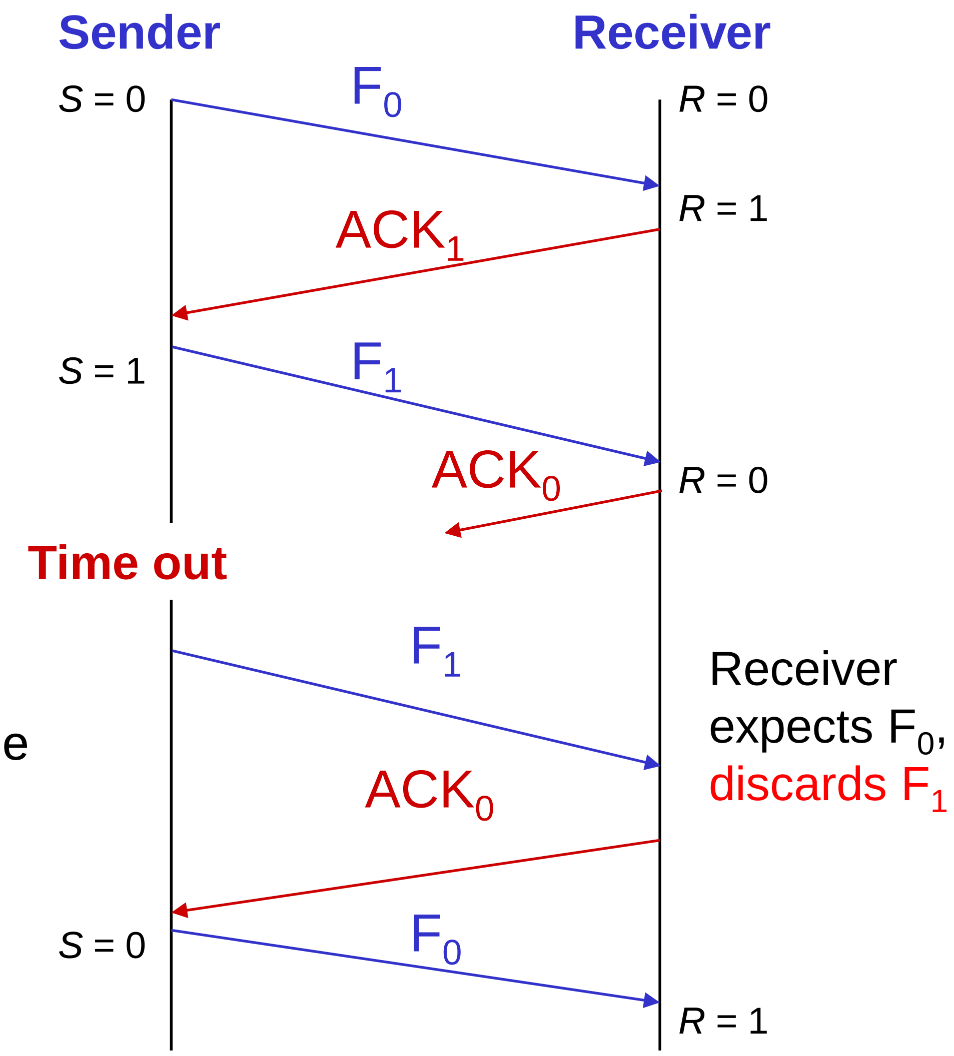
Stop-and-wait ARQ

- Sender
  - Variable S: sequence number of last frame sent
  - Keeps a copy of last frame sent
  - Starts a timer when a frame is sent
  - Stops timer when ACK is received
  - Retransmits frame if time out (restarts timer)
- Receiver
  - Variable R: next expected sequence number
  - When a frame is received, sends an ACK with next expected sequence number
    - Ready to receive
    - Receiver not ready (flow control)
  - Drops received packet if wrong sequence number



Stop-and-wait ARQ: Lost Acknowledgment

- No ACK received
  - Sender times out
  - Retransmits frame
- Receiver receives wrong sequence number
  - Discards frame
  - Sends ACK with expected sequence number (0)
- Sender may send next frame



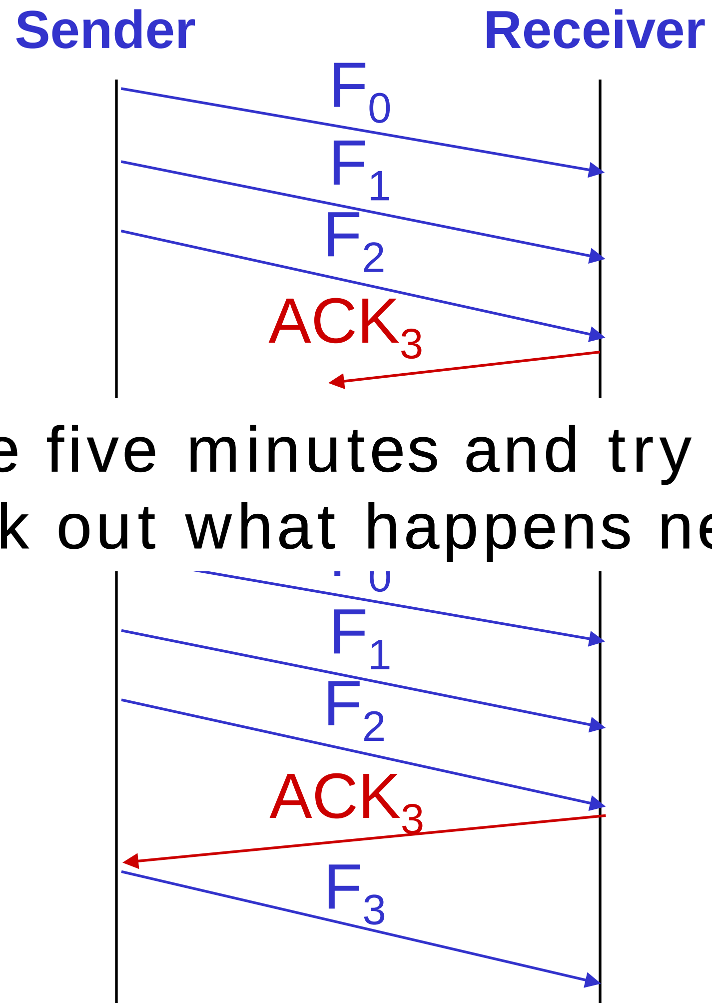
Continuous ARQ

- Stop-and-wait ARQ is simple but inefficient
  - same reason as for stop-and-wait flow control
- Continuous ARQ
  - Sequence numbers with sliding window
  - ACK and NACK
  - Time out

Go-back-N ARQ

- Based on sliding window flow control
- Sender
  - May send *N* frames without acknowledgment
    - Copies of all unacknowledged frames kept in a buffer
  - Time out
    - retransmits *all* unacknowledged frames
- Receiver
  - Discards frames with unexpected sequence numbers

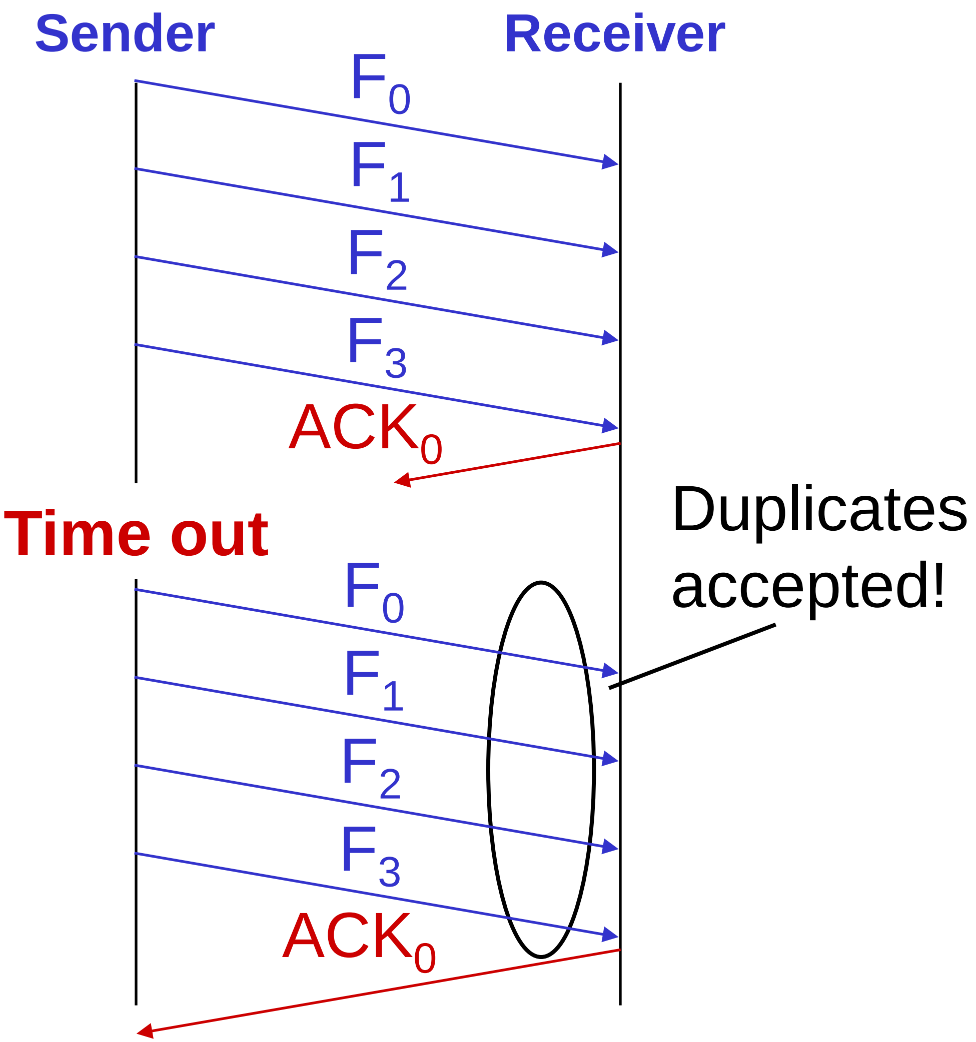
Example: Lost Acknowledgment (N = 3)



Take five minutes and try to work out what happens next!

Window Size Versus Sequence Numbers

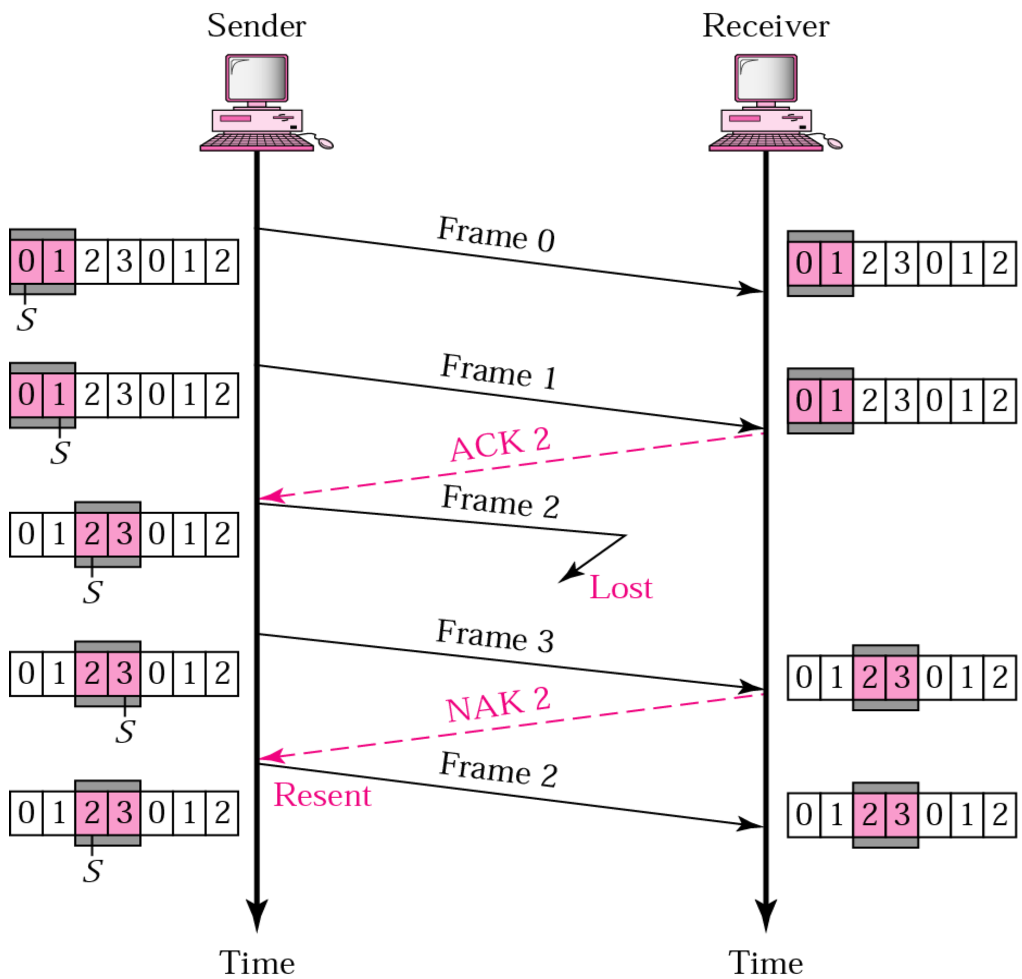
- With *k*-bit sequence numbers, window size can be at most 2^k-1
- Otherwise frames can be duplicated at receiver
- For example:
  - Sequence numbers 0-3 (k = 2)
  - Window size 2^k = 4 (incorrectly)



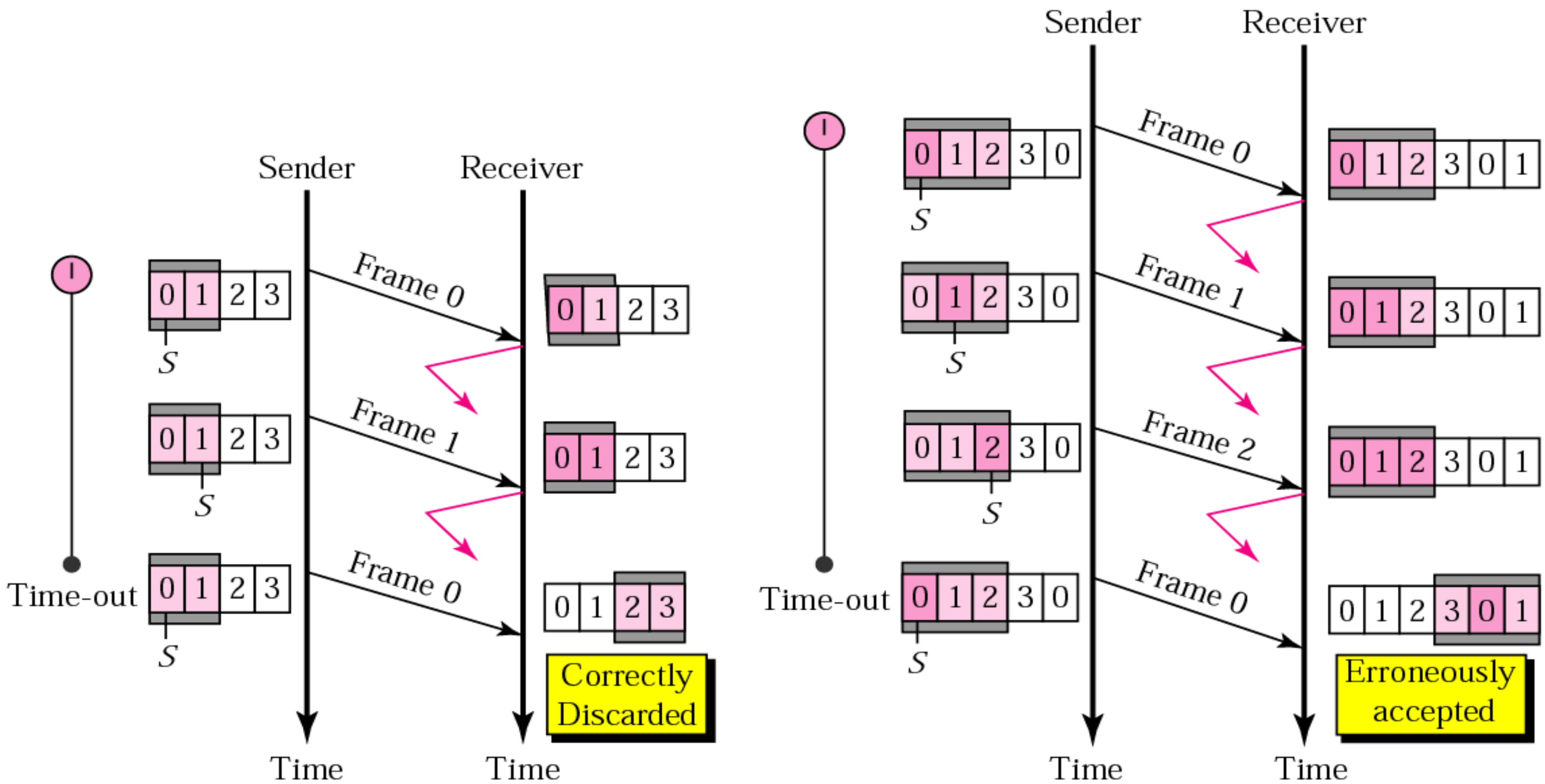
Selective Repeat ARQ

- Sometimes also called Selective Reject ARQ (SREJ)
- Only retransmit frames that are lost
  - Negative acknowledgment NAK (SREJ)
  - Time out
- Receiver has a receiver window
  - Only frames with sequence number within receive window are accepted
  - Sorts accepted frame into correct order

Selective Repeat ARQ

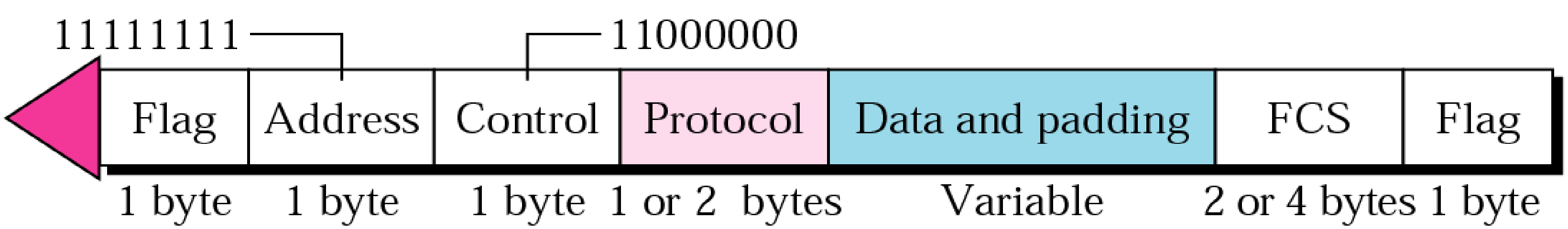


Window Size in Selective Repeat ARQ



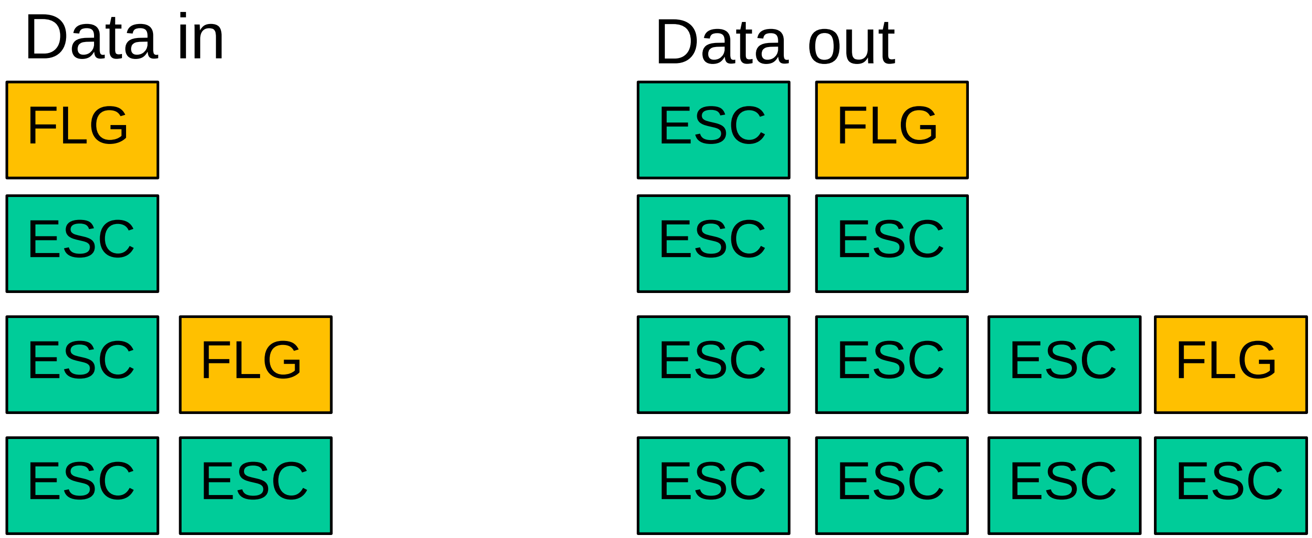
Data Link Example: (PPP)

- Point-to-point Protocol
- Control and management of data transfer over physical (point-to-point) links
  - Dedicated link with two stations
  - Traditional modem, DSL, SDH
- Based on HDLC frame format



Character stuffing

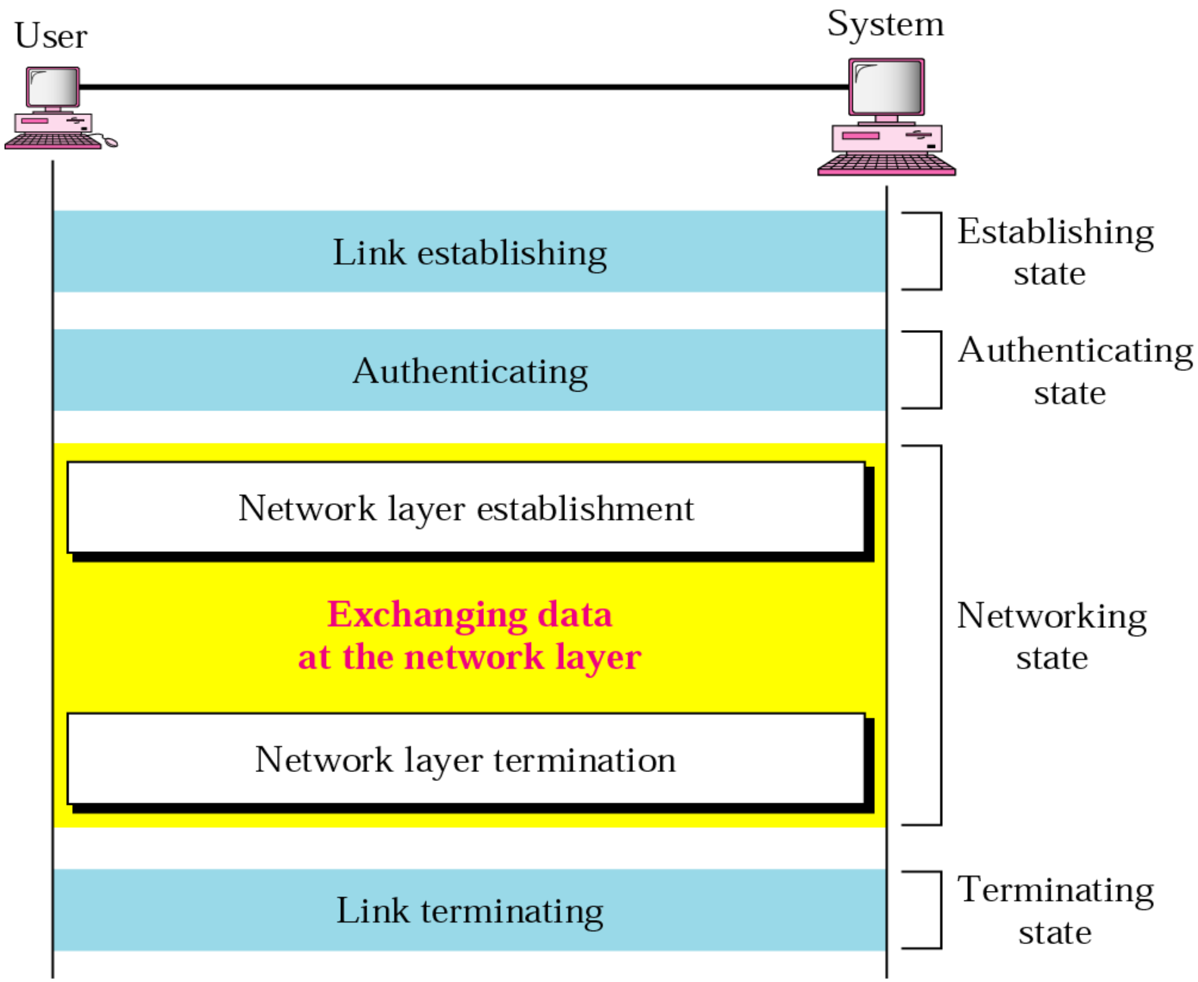
- Transparency of data
  - Flags may appear in data field
  - Truncation of frame and
  - loss of frame boundaries
- Solution: prevent mistakes
  - Precede the flag (FLG, x7E) with an escape code (ESC, x7D)
  - Precede ESC by ESC
  - XOR 02 to all bytes in frame; why?



PPP Protocol Family

- Link Control Protocol (LCP)
  - Establish, disconnect link
  - Negotiate options—maximum receive unit, authentication, omit address and control fields
- Authentication
  - Password Authentication Protocol (PAP)
  - Challenge Handshake Authentication Protocol (CHAP)
- Network Control Protocol (NCP)
  - Internetwork Protocol Control Protocol (IPCP)

PPP Example


User
System
Link establishing
Establishing state
Authenticating
Authenticating state
Network layer establishment
Exchanging data at the network layer
Network layer termination
Networking state
Link terminating
Terminating state

Summary

- Flow control
  - Stop and wait
  - Sliding window
- Bit error detection
  - Parity control
  - Checksum
  - Cyclic redundancy check (CRC)
- Detecting frame loss: sequence numbers
- Error control: retransmission (ARQ)
  - Stop and wait ARQ
  - Go-back-N ARQ
  - Selective reject ARQ
- Example: PPP