

# **CS 123: Data Link Intro and Framing**

**George Varghese**

October 10, 2006

## Issues common to all layers and systems

- **Multiplexing and resource allocation:**  
Handling multiple users economically (e.g., TDM, FDM)
- **Addressing:** Separating out traffic for multiple receivers. (frequencies, time slots)
- **Error Control:** Space levels far apart, send below Nyquist limit, sample at mid bit,
- **Synchronization:** State variables in receiver and sender are kept in sync despite errors and timing uncertainty in timing. (clock synchronization).
- **Interconnection:** Repeaters, T1 Switches, Sonet Rings,

# Data Link Sublayers

**Point-to-point Links (2 nodes)**  
*(e.g., HDLC, Frame Relay)*

ERROR RECOVERY  
*(OPTIONAL)*

ERROR DETECTION

FRAMING



**Broadcast Links (>= 2 nodes)**  
*(e.g., Ethernet, Token Ring)*

MULTIPLEXING

MEDIA ACCESS

ERROR DETECTION

FRAMING



## Five functions of Data Link

Five functions:

- Framing: breaking up a stream of bits into units called frames so that we can extra information like destination addresses and checksums to frames. (Required.)
- Error detection: using extra redundant bits called checksums to detect whether *any* bit in the frame was received incorrectly. (Required).
- Media Access: multiple senders. Need traffic control to decide who sends next. (Required for broadcast links).
- Multiplexing: Allowing multiple clients to use Data Link. Need some info in frame header to identify client. (Optional)
- Error Recovery: Go beyond error *detection* and take recovery action by retransmitting when frames are lost or corrupted. (Optional)

## Why not do error recovery at each hop?

- **End-to-end argument:** Need end-to-end or transport error recovery anyway. Can't trust a series of hop-by-hop schemes because:
  - Crashes and other losses at intermediate nodes.
  - Transport must work over both reliable and unreliable links.
- Thus hop-by-hop is only a performance optimization.
  - Extra cost (ack messages, buffering) not worth it when error rate is low.
  - Worth it (quicker recovery, less wasted resources) when error rate is high.

## Quasi-reliability

- The probability of a receiving Data Link passing up an incorrect frame to the client layer should be very, very small.
  - Undetected error say once in 20 years.
  - Needs to be so small because transport protocols do not insist on end-to-end checksums, a violation of end-to-end argument.
- The probability of a receiving Data Link dropping frames sent by the sender should be small (once a day) to allow good performance.

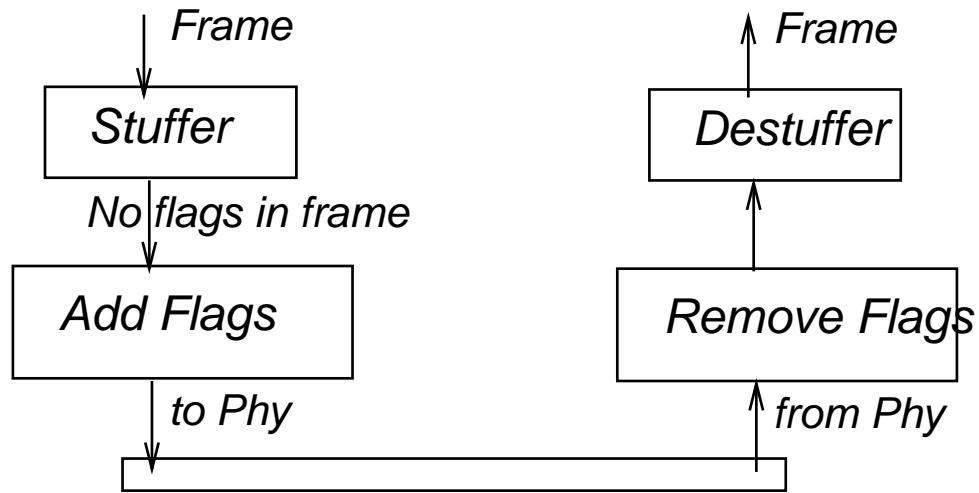
## Why Framing

- Without framing, the bit stream is reserved for one sender and one client per sender. Need frames to add multiplexing information like destination addresses and destination client names.
- Frames offer a small, manageable unit for error recovery and error detection. Add checksums to frames for error detection and sequence numbers etc. for error recovery.

## How Framing

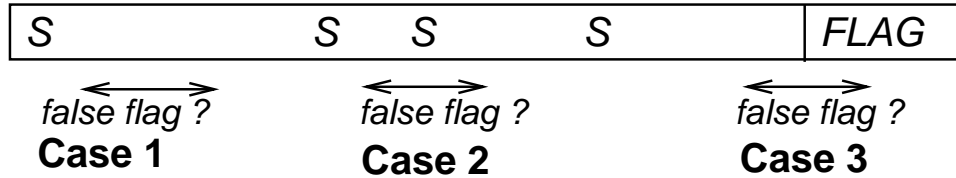
- **Flags and Bit Stuffing:** Use special bit patterns or flags to delimit (i.e., mark boundaries) frames. Need *bit stuffing* to “encode” the user data not to contain the flags. (e.g., HDLC)
- **Start Flags and Character Count:** Use flags to indicate the start of data and a bit count to indicate end of frame. Bit stuffing not needed but less robust. (e.g., DDCMP)
- **Start and End Flags supplied by Physical Layer:** Use special physical layer symbols to delimit frames.

## Framing Design by Sublayering



- Sublayering is a good design technique within layers as well!
- What happens if input data contains 0111110. If receiver gets 111110?

## Correct Bit Stuffing



- **Case 1:** Easy, or we would have added a stuffed bit
- **Case 2:** Stuff after 0 followed by 5 1's does not work!
- **Case 3:** Flag = 01010101 and stuff a 1 after 1 after 010101 does not work!

## Useful Principles

- Each layer or sublayer exacts its penalty: e.g., clock recovery coding, framing bits.
- The end-to-end argument.
- Lower Layers should not depend for correctness on assumptions about higher layers.
- All layers have some common problems to solve: e.g., synchronization, multiplexing.
- The best principles will/should be violated for pragmatic reasons.
- Layering and Sublayering are a good way to understand and design new protocols.