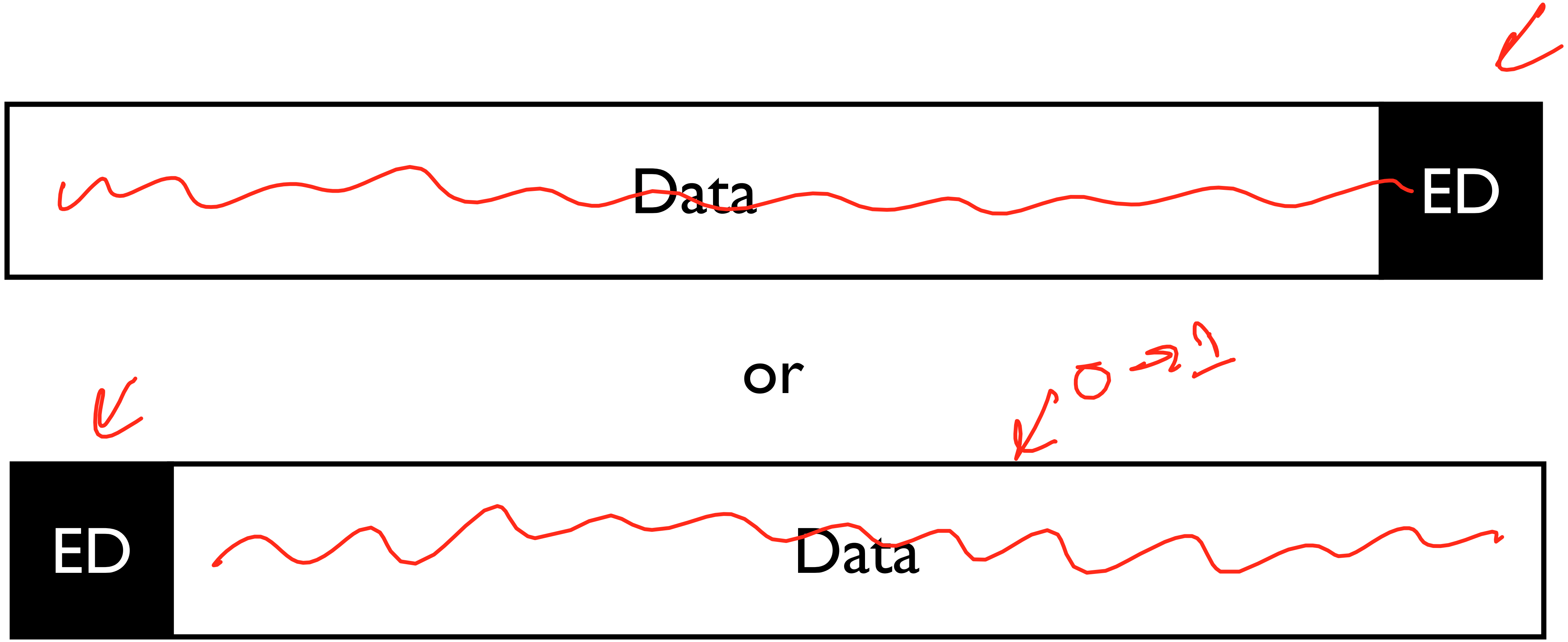
Error Detection: 3 schemes

Checksum, CRC and MAC

Error Detection



(ethernet CRC, TLS MAC)

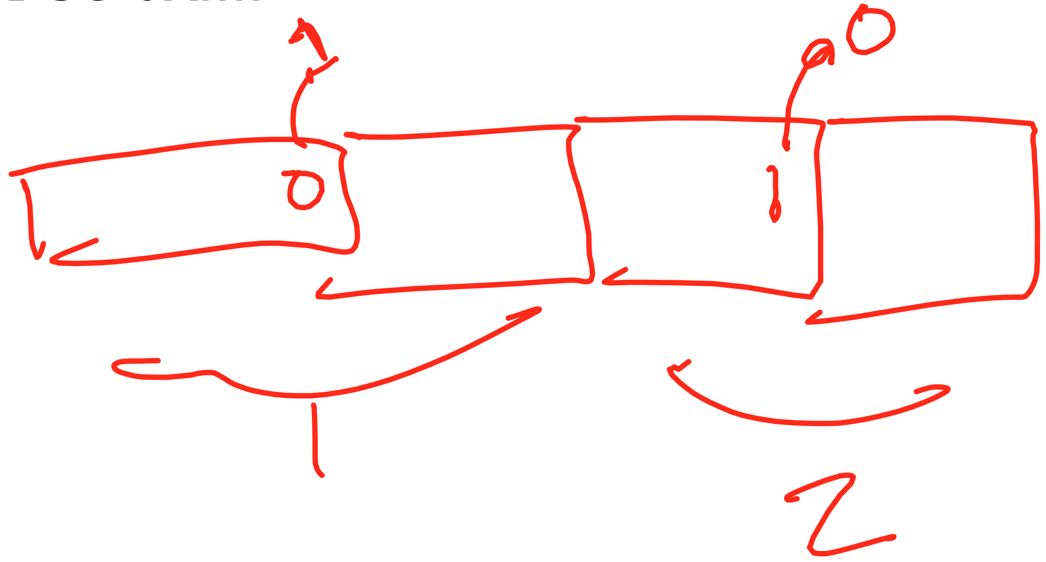
(IP checksum)

Three Error Detection Schemes

- Checksum adds up values in packet (IP, TCP)
  - Very fast, cheap to compute even in software
  - Not very robust
- Cyclic redundancy code computes remainder of a polynomial (Ethernet)
  - More expensive than checksum (easy today, easy in hardware)
  - Protects against any 2 bit error, any burst $\leq c$ bits long, any odd number of errors
- Message authentication code: cryptographic transformation of data
  - Robust to malicious modifications, but not errors
  - If strong, any 2 messages have a 2^{-n} chance of having the same code

IP Checksum

- IP, UDP, and TCP use one's complement checksum algorithm:
  - Set checksum field to 0
  - Sum all 16-bit words in packet
  - Add any carry bits back in: 0x8000 + 0x8000 = 0x0001
  - Flip bits (0xc379 becomes 0x3c86), unless 0xffff, then checksum is 0xffff
  - To check: sum whole packet, including checksum, should be 0xffff
- Benefits: fast, easy to compute and check
  - Motivated by earliest software implementations
- Drawbacks: poor error detection
  - Only guarantees detecting a single bit error



Cyclic Redundancy Check (CRC)

4 1/16

- Cyclic Redundancy Check (CRC): distill n bits of data into c bits, $c \ll n$
  - Can't detect all errors: 2^{-c} chance another packet's CRC matches
- CRC designed to detect certain forms of errors: stronger than checksum
  - Any message with an odd number of errors
  - Any message with 2 bits in error
  - Any message with a single burst of errors $\leq c$ bits long

16 bit

≤16 bit

- Link layers typically use CRCs
  - Fast to compute in hardware (details in a moment)
  - Can be computed incrementally
  - Good error detection for physical layer burst errors

Diversion: CRC Mathematical Basis

- Cyclic Redundancy Check (CRC): distill *n* bits of data into *c* bits, *c* << *n*
- Uses polynomial long division
  - Consider the message M a polynomial with coefficients 0 or 1 (pad with *c* zeroes)
    - E.g., M = 10011101 = $x^7 + x^4 + x^3 + x^2 + 1$
  - Use a generator polynomial G of degree *c* also with coefficients 0 or 1
    - E.g. G = 1011 = $x^3 + x + 1$
  - Divide M by G, the remainder is the CRC
- Append CRC to message M: M' = M + CRC
  - Long division of M' with G has a remainder of 0

MAC

- Message Authentication Code (MAC)
  - Not to be confused with Media Access Control (MAC)!
- Uses cryptography to generate m = MAC(M, s), $|m| \ll |M|$
  - Using M and secret s, can verify m = MAC(M, s)
  - If you don't have s, very very hard to generate $\boxed{m}$
  - Very very hard to generate an M whose MAC is m
  - M + m means the other person probably has the secret (or they're replayed!)
- Cryptograhically strong MAC means flipping one bit of M causes every bit in the new m to be randomly 1 or 0 (no information)
  - Not as good for error detection as a CRC!
  - But protects against adversaries

Three Error Detection Schemes

- Checksum adds up values in packet (IP, TCP)
  - Very fast, cheap to compute even in software
  - Not very robust
- Cyclic redundancy code computes remainder of a polynomial (Ethernet)
  - More expensive than checksum (easy today, easy in hardware)
  - Protects against any 2 bit error, any burst ≤ *c* bits long, any odd number of errors
- Message authentication code: cryptographic transformation of data
  - Robust to malicious modifications, but not errors
  - If strong, any 2 messages have a 2^{-n} chance of having the same code