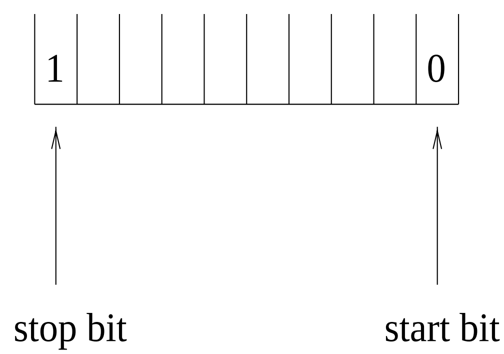
Framing

Asynchronous: e.g., ASCII character transmission between dumb terminal and host computer.



- each character is an independent unit

  → "asynchronous"

- receiver needs to know bit duration

  → bit rate assumed known between sender/receiver

  → in that sense, "synchronous"

Overhead problem; assuming 1 start bit, 1 stop bit, 8 data bits, only 80% efficiency.

$\longrightarrow$ inefficient for long messages

Synchronous: "Byte-oriented"; e.g., PPP, BISYNC

SYN	SYN	SOH	Header	STX	Body	ETX	CRC

$\longrightarrow$ `SYN, SOH, STX, ETX, DLE`: sentinels

Two problems:

- How to maintain synchronization if |Body| is large?
- Control characters within body of message.

$\longrightarrow$ inefficient for short messages

$\longrightarrow$ efficiency approaches 1 as $|\text{Body}| \to \infty$

"Bit-oriented"; e.g., HDLC

$\longrightarrow$ bit is the unit

Use fixed *preamble* and *postamble*; simply a bit pattern.

$\longrightarrow$ 01111110

How to avoid confusing 01111110 in the data part?

$\longrightarrow$ bit stuffing

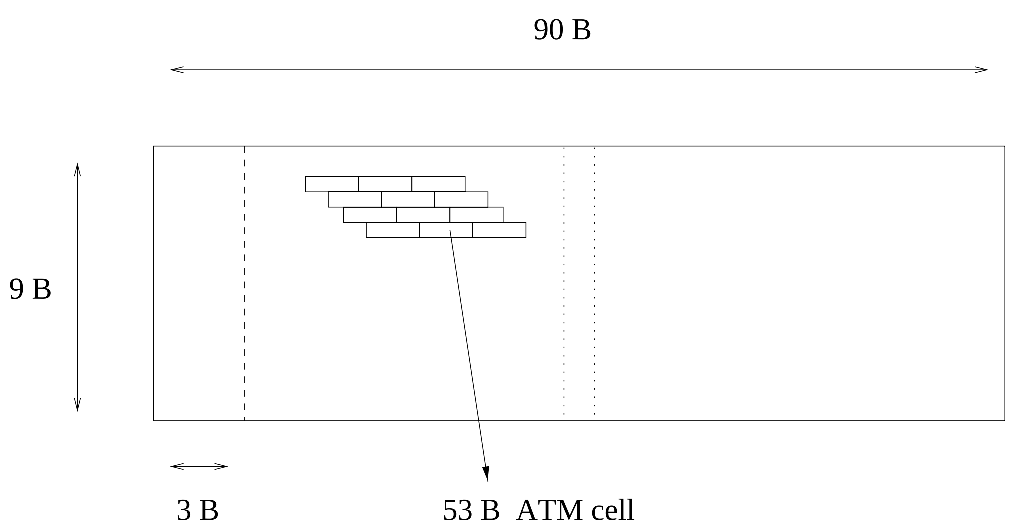
SONET (Synchronous Optical Network)

⟶ framing standard for optical fiber

Rates: OC-1 (51.84 Mbps), OC-3 (155.25 Mbps), OC-3c, OC-12 (622.08 Mbps), OC-24 (1.24416 Gbps), OC-48, etc.

⟶ formally: STS-n

OC-1 frame:

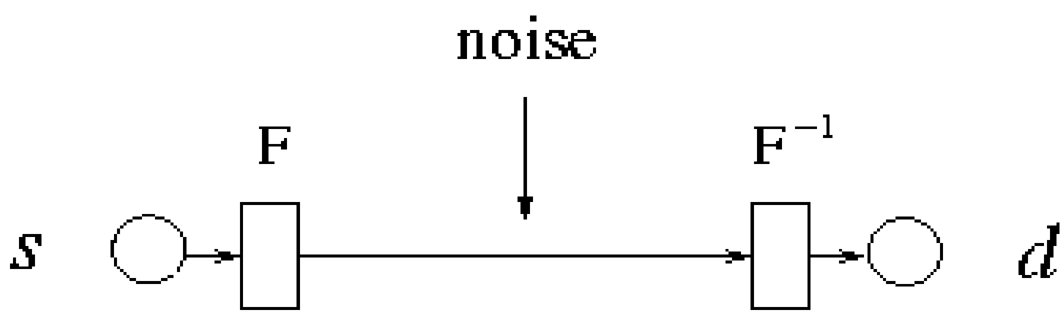


Features:

- 125 μsec frame duration (for all OC-n)
- 51.84 Mbps $= 810 \cdot 8 \cdot 8000$
- $3 + 1$ columns of overhead
- overhead includes synchronization, pointer fields
- overhead encoded using NRZ
- payload scrambled (XOR'ed) to achieve approximate self-clocking

Error-detection and correction

$\longrightarrow$ reliable transmission over noisy channel



Key problem:

- sender wishes to send a; transmits code word w_a
- receiver receives w
- due to noise, w may, or may not, be equal to w_a

$\longrightarrow$ $a \mapsto w_a \mapsto w \mapsto [?]$

Error detection:

- determine if w is a valid code word
- e.g., parity bit in ASCII transmission

  $\longrightarrow$ odd or even parity

  $\longrightarrow$ limitation?

Error correction:

- even if $w \neq w_a$, recover symbol a from scrambled w

  $\longrightarrow$ correction is tougher than detection
- how to correct single errors for ASCII transmission?

  $\longrightarrow$ assume one can use 21 bits

  $\longrightarrow$ what about 14 bits?

Conceptual approach:

Error detection:

- consider legal code word set $S = \{w_a : a \in \Sigma\}$

  $\rightarrow$ take binary alphabet $\Sigma = \{0, 1\}$

- can detect k-bit errors if

  $\rightarrow$ corrupted w does not belong to S

  $\rightarrow$ must hold for all k-bit error patterns

Key question: what kind of S can satisfy these properties

$\longrightarrow$ ASCII with 1-bit flip

$\longrightarrow$ ASCII with 2-bit flips

$\longrightarrow$ brute force approach . . .

Error correction:

- assume w_a has turned into w under k-bit errors
- for all $b \in \Sigma$, calculate $d(w_b, w)$

  $\rightarrow$ Hamming distance; e.g., $d(1011, 1101) = 2$
- pick $c \in \Sigma$ with smallest $d(w_c, w)$ as answer

Ex.: $0 \mapsto 000$ and $1 \mapsto 111$

$\longrightarrow$ want to send 0, hence send 000

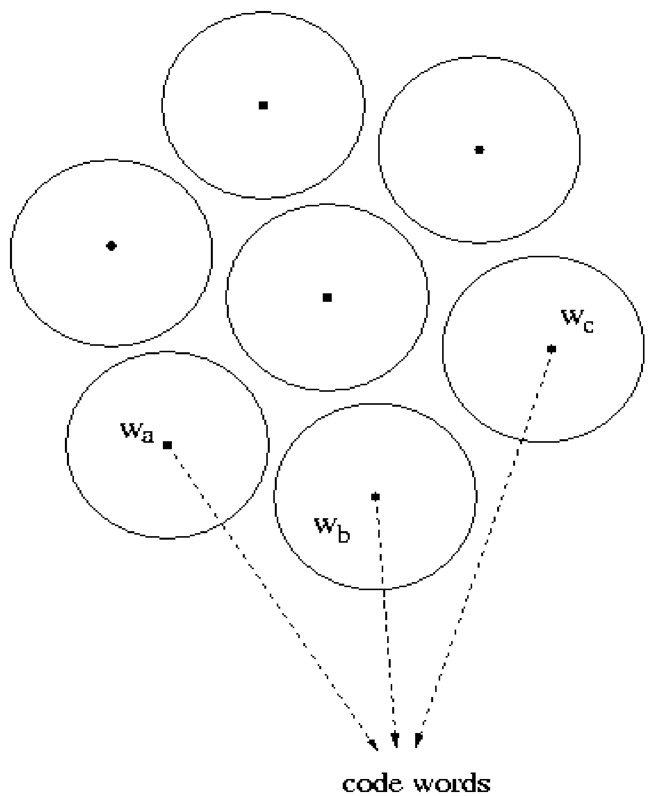
$\longrightarrow$ 010 arrives: $d(010, 000) = 1$ & $d(010, 111) = 2$

$\longrightarrow$ conclude 000 was sent which means 0

Pictorially: consider "ball" of distance r centered at w_a

$$\longrightarrow \quad B_r(w_a) = \{w : d(w_a, w) \le r\}$$

Consider code word set S with "well-separated" layout:



Assuming k bit flips, sufficient conditions for error detection and error correction in terms of $d(w_a, w_b)$?

Network protocol context: different approach to detection vs. correction

$\longrightarrow$ error detection: use checksum and CRC codes

$\longrightarrow$ error correction: use retransmission

$\longrightarrow$ humans?

$\longrightarrow$ can also use FEC; for real-time data

Internet Checksum: Group message into 16-bit words; calculate their sum in one's complement; append "checksum" to message.

$\longrightarrow$ problem?