Physical and Link Layers: Forward Error Correction (FEC)

SNR/BER Curves

- For a given modulation scheme and signal-to-noise ratio, you can compute the expected bit error rate
  - Making some mathematical assumptions about noise
  - Bedrock principle of RF communication theory
- Bit error rate can become arbitrarily low, but never reaches zero!
- In practice, the math works out that sending packets as raw bits is very inefficient
  - Expected data throughput is far, far below Shannon limit

Coding

- Adding a little redundancy at the physical layer can greatly improve link layer throughput
  - Both in theory and in practice
- *Coding gain*: the ratio of bits at link layer to bits at physical layer
  - 1/2 code: each link layer bit is 2 physical layer bits
  - 3/4 code: each 3 link layer bits are 4 physical layer bits
- Forward error correction (FEC): proactively adding some additional data (redundancy) so recipient can correct potential errors

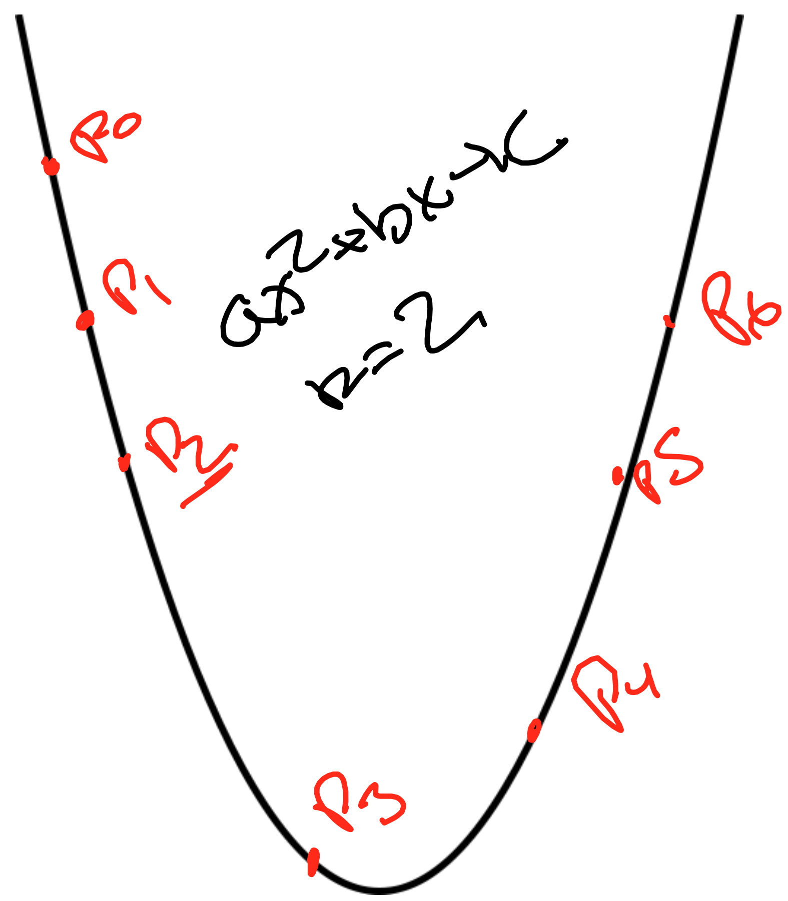
Coding Algorithms

- There are many, many coding algorithms, with different tradeoffs
  - Hamming codes, convolutional codes, LT codes, LDPC, Turbo codes, Tornado codes, Raptor codes...
- Reed-Solomon error correction
  - Tremendously commonly used (e.g., CDs, DVDs, DSL, WiMax, RAID6 storage)
  - Mathematically simple (compared to some of the others)

Reed-Solomon Basic Idea

3 + 10 points

- Take *k* chunks of data
- Make them the coefficients of a *k-1* degree polynomial
- Compute *n* points along the polynomial ($n \geq k$) -1, send as coded data
  - Any *k* of the n points allow you to recover the polynomial coefficients
- Complications: value of n computed points must be in a finite field (limited number of bits)



http://en.wikipedia.org/w/index.php?title=File:Parabola.svg

Details

R = (223, 255)
255 - 223 = 32
32 erasures
16 errors

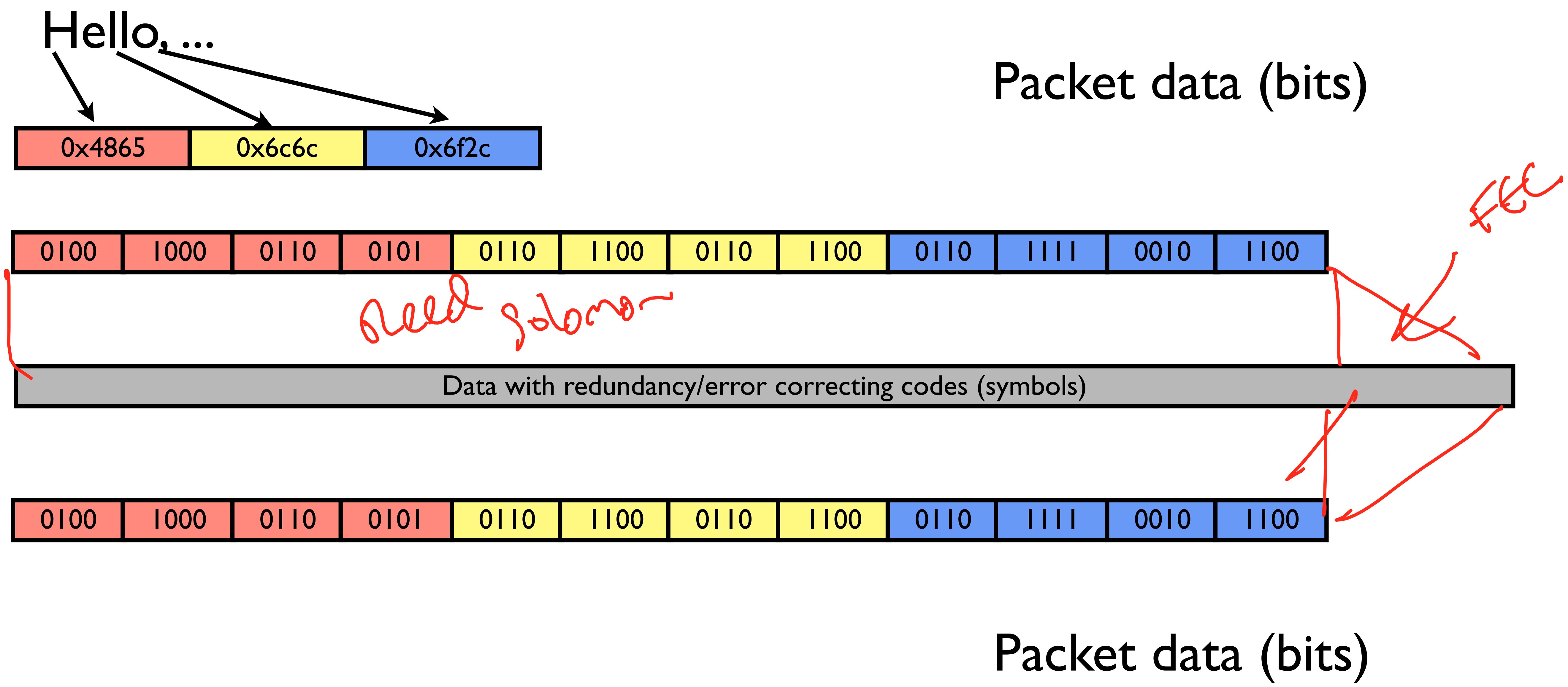
- Two kinds of errors
  - *Erasures:* location of error known ("erased" values)
  - *Errors:* location unknown (e.g., bit error)
- Take k chunks of data, code into n chunks ($n \geq k$)
- Reed-Solomon can correct up to (n - k) erasures (need k points)
- Reed-Solomon can correct up to (n - k)/2 errors

Conceptual Reed-Solomon Code

- Take 223 8-bit values, make coefficients of 222-degree polynomial *p*
- Compute *p*(0), *p*(1), *p*(2), *p*(3), *p*(4).... *p*(254) as 8-bit values (field)
- Send 255 values
- This is a (255,223) code: each 255 *codewords* are from 223 *data words*
  - Can recover from up to 32 erasures or 16 errors

- This isn't what's done in practice today
  - 0-255 are not a field, which is needed for the math to work
  - It's too complex to decode
  - But it gives you the basic idea

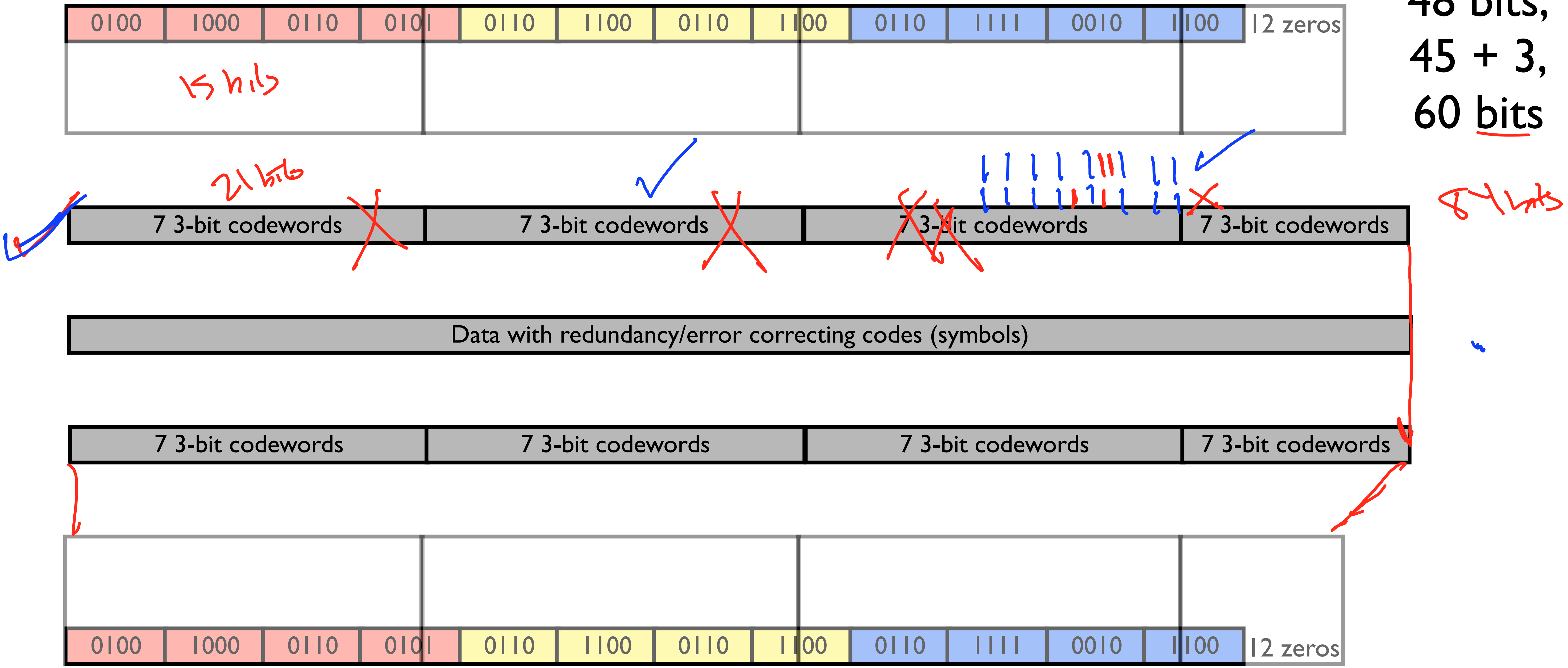
Reed-Solomon Example



Reed-Solomon Example (7,5)

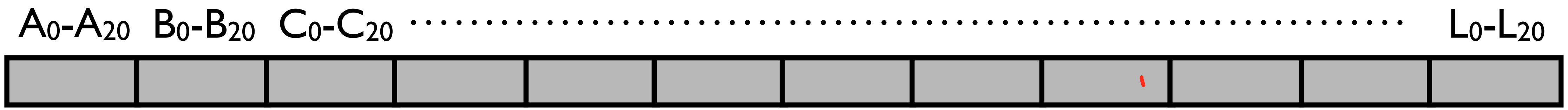
1 errors
2 erasures

48 bits,
45 + 3,
60 bits



Interleaving

- A (n+k, n) Reed-Solomon code protects against k erasures or k/2 errors
- Physical media often have burst errors
- Can make encoding more robust through *interleaving*
- Example: 12 chunks of a (7,5) code
  - Each chunk is 21 bits long: 7 code words of 3 bits each



$A_0 B_0 C_0 D_0$ ..., $A_1 B_1 C_1$ ..., $K_{20} L_{20}$

1 bit error

2 bit errors

6 bit errors

12 bit errors w/ interleaving