Note: We will start at 12:53 pm ET

Course Summary:

Date	Details	
Mon Feb 1, 2021	18-441/741 Lecture 1	12:50pm to 2:50pm
Wed Feb 3, 2021	18-441/741 Lecture 2	12:50pm to 2:50pm
Mon Feb 8, 2021	18-441/741 Lecture 3	12:50pm to 2:50pm
Wed Feb 10, 2021	18-441/741 Lecture 4	12:50pm to 2:50pm
Fri Feb 12, 2021	18-441/741 Recitation 1 (Hybrid) -- Project-1 Intro -- Zoom / In-person (A-L)	12:50pm to 1:40pm
Sun Feb 14, 2021	Quiz 1	due by 11:59pm
→ Mon Feb 15, 2021	18-441/741 Lecture 5	12:50pm to 2:50pm
Wed Feb 17, 2021	18-441/741 Lecture 6	12:50pm to 2:50pm
Mon Feb 22, 2021	18-441/741 Lecture 7	12:50pm to 2:50pm
Wed Feb 24, 2021	18-441/741 Lecture 8	12:50pm to 2:50pm
Fri Feb 26, 2021	18-441/741 Recitation 2 (Hybrid) -- Project 2 Intro -- Zoom / In-person (M-Z)	12:50pm to 1:40pm
Sun Feb 28, 2021	Quiz 2	due by 11:59pm
	Project 1	due by 11:59pm

18-441/741: Computer Networks

Lecture 5: Physical Layer III

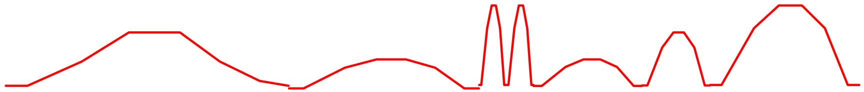
Swarun Kumar

Physical Layer: Outline

- Digital networks
- Characterization of Communication Channels
- Fundamental Limits in Digital Transmission
- Modems and Digital Modulation
- Line Coding
- Error Detection and Correction
- Wired PHY 101 (if time permits)
- Wireless PHY 101

From Signals to Packets

Analog Signal



“Digital” Signal

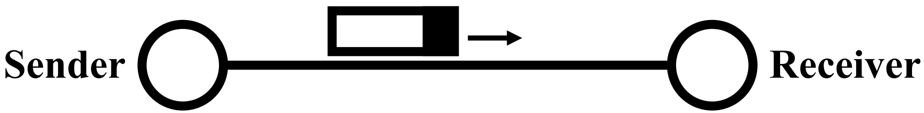
Bit Stream 0 0 1 0 1 1 1 0 0 0 1

Packets

0100010101011100101010101011101110000001111010101110101010101101011010111001

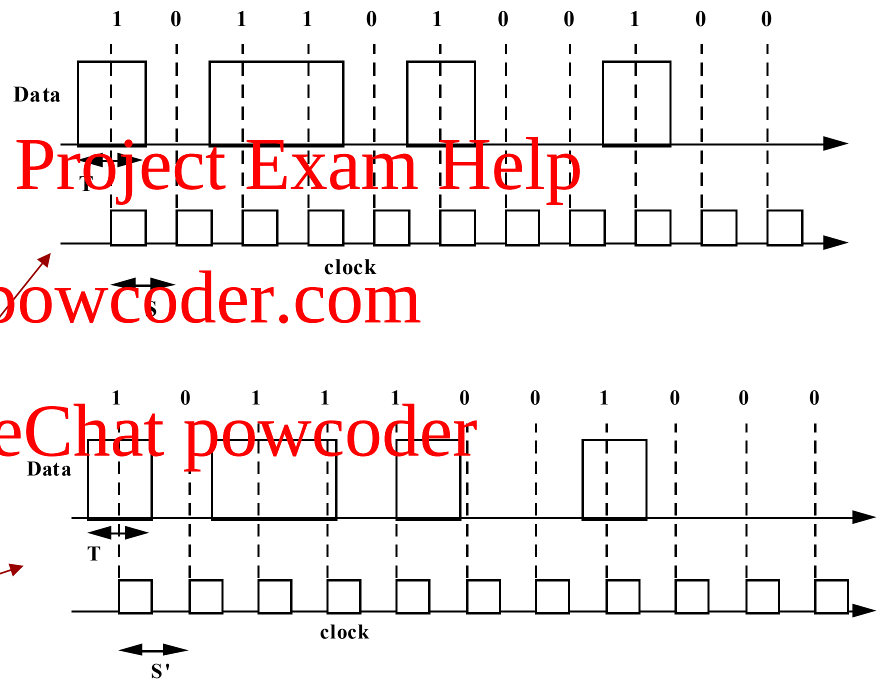
Header/Body Header/Body Header/Body

Packet Transmission



Synchronization

- Synchronization of clocks in transmitters and receivers.
  - clock drift causes a loss of synchronization
- Example: assume '1' and '0' are represented by V volts and 0 volts respectively
  - Correct reception
  - Incorrect reception due to slow clock at the receiver

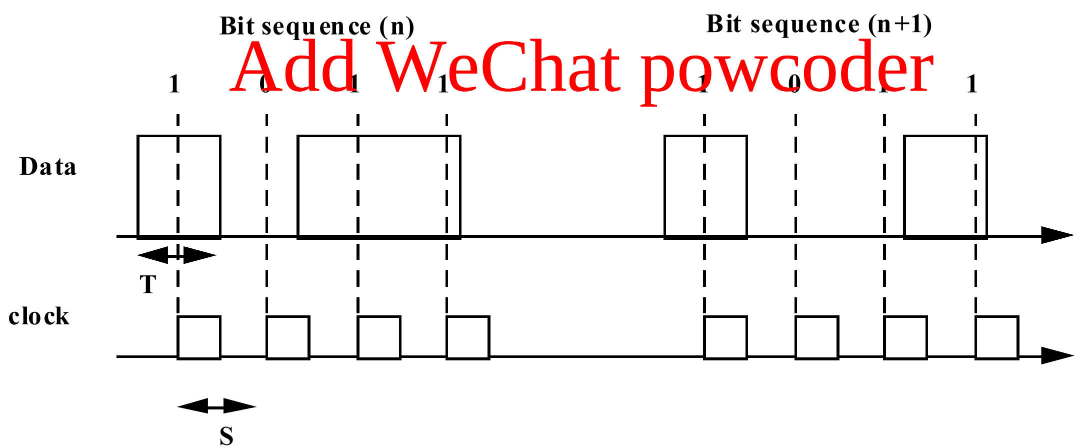


Synchronization (cont')

- How to avoid a loss of synchronization?
  - Asynchronous transmission
  - Synchronous transmission

Asynchronous Transmission

- Avoids synchronization loss by
  - specifying a short maximum length for the bit sequences (so that clock doesn't drift much within sequence)
  - and resetting the clock in the beginning of each bit sequence (by using a 'start bit')
- Accuracy of the clock?

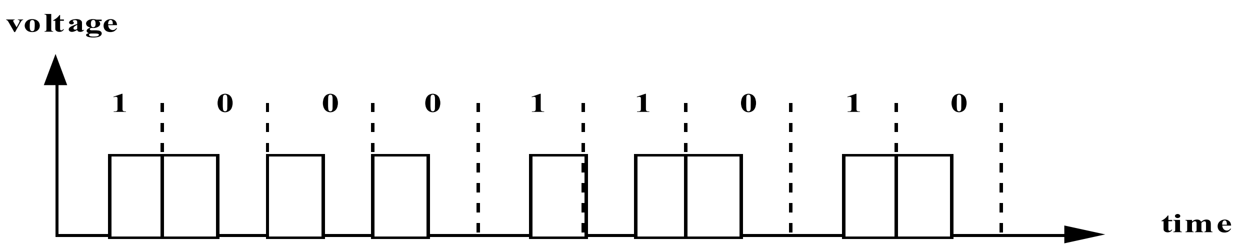


Asynchronous transmission: ASCII code

- ASCII (American National Standard Code for Information Interchange) code
  - 7 bits to represent 128 letters, symbols, and control characters. (i.e. A='1000001', CR(Carriage Return)='0001101')
  - Asynchronous transmission sends sequences of 8 bits=one start bit + 7 ASCII bits.
  - some systems add one parity bit to make number of '1' to be even number
  - i.e. '1100111'1 or '1010110'0

Synchronous Transmission

- Overcomes the inefficiency of asynchronous transmission.
- Improves efficiency by transmitting longer sequences of bits, called packets (variable length).
- Requires extra information to indicate the end of the packet.



Encoding

- Encoding converts a binary information sequence into a digital signal
  - Sender then uses the digital signal to modulate the signal in a way that the receiver can recognize
- Encoding can be done one bit at a time or in blocks of multiple bits called a symbol
  - Example: a symbol with 8 values means that 3 bits are sent in each time slot
- Transmission is synchronous, i.e., a clock is used to sample the signal.
  - Receiver's clock must be synchronized with the sender's clock

Why Do We Need Encoding?

- To meets certain electrical constraints.
  - Many of them! See next slide

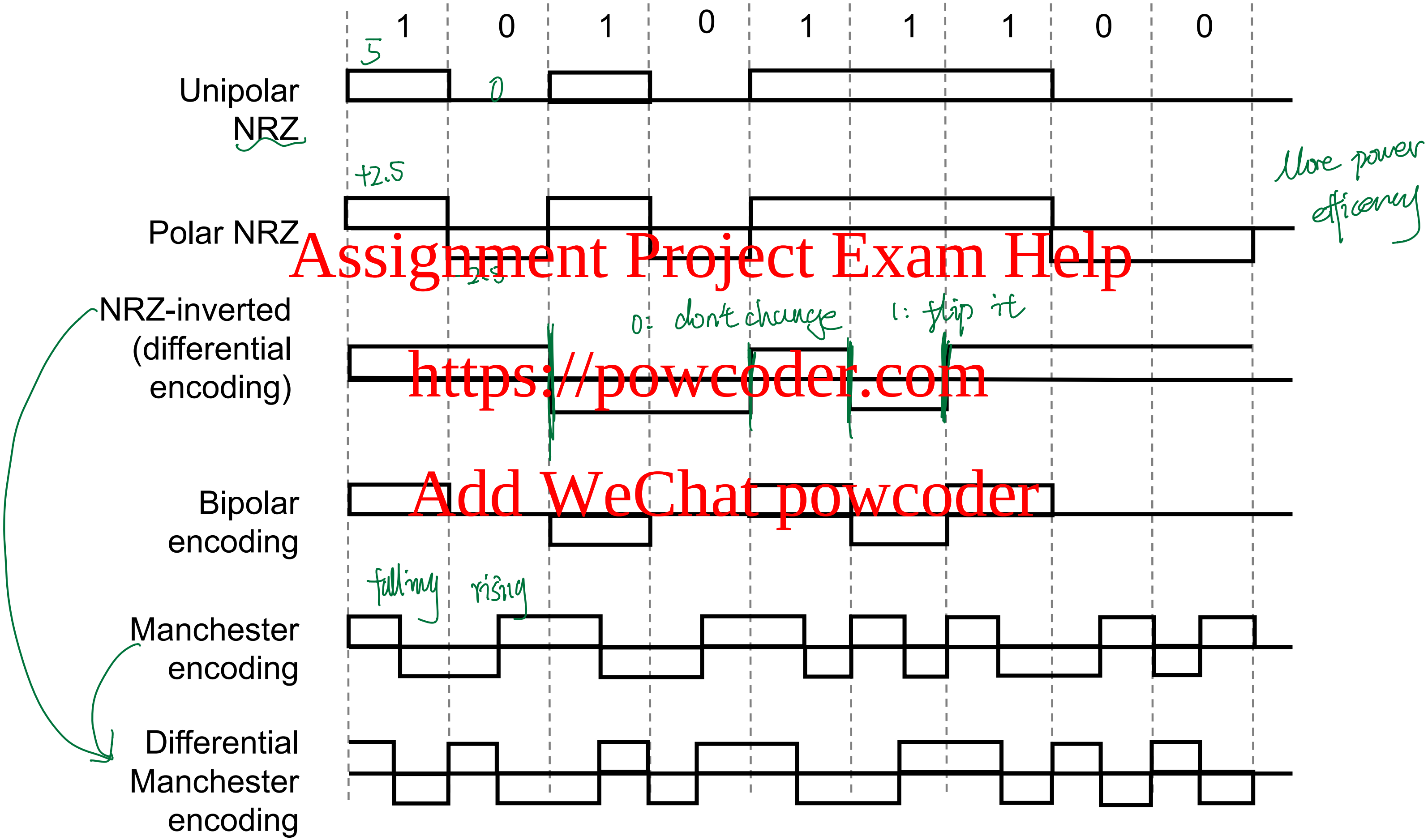
- Creates control symbols, besides regular data symbols.

can not appear in data

  - E.g. start or end of frame, escape, ...
- Can do error detection or error correction

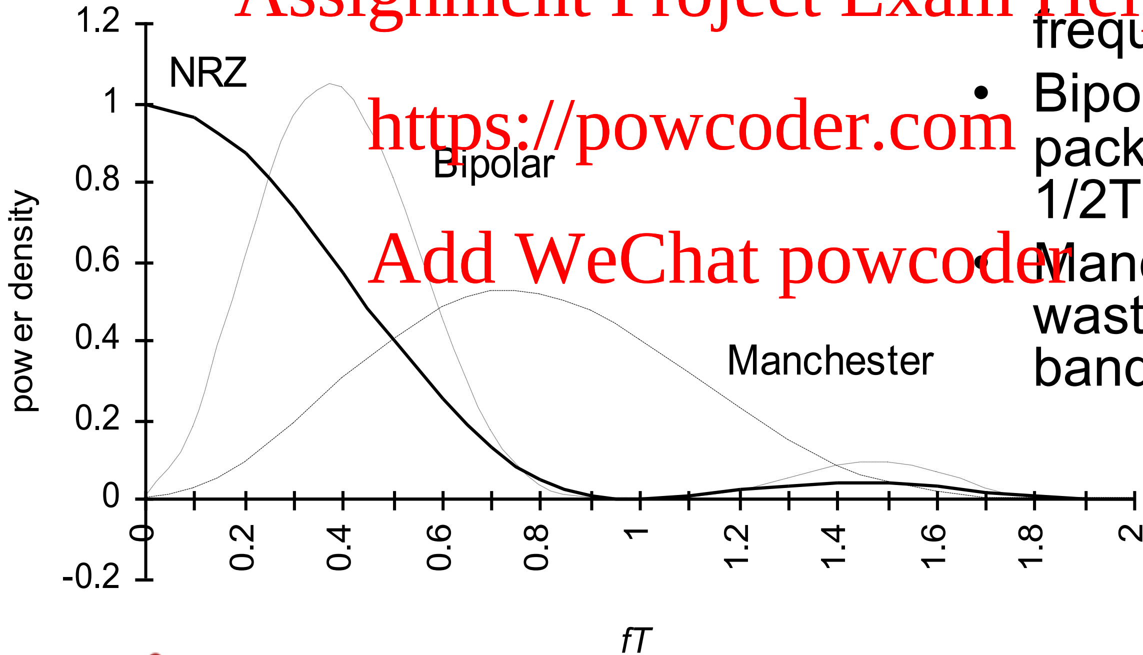
  - Some codes are illegal so receiver can detect certain classes of errors
  - Minor errors can be corrected by having multiple adjacent signals mapped to the same data symbol

Line coding examples



Spectrum of Line codes

- Assume 1's & 0's independent & equiprobable



- NRZ has high content at low frequencies
- Bipolar tightly packed around 1/2T
- Manchester wasteful of bandwidth

mB/nB Encoding

$n \geq m$

- m data bits are coded as *symbols* of n line bits
- Example: FDDI uses 4B5B
  - 4 data bits for 5 line bits, so 100 Mbps uses 125 MHz.
  - Uses less frequency than Manchester encoding (1B2B)
- Each valid symbol has at least two 1s: get dense transitions.
- 16 data symbols, 8 control symbols
  - Data symbols: 4 data bits
  - Control symbols: idle, begin frame, etc.
- Also: 8B10B (Gigabit Ethernet, Fiber channel) and 64B66B code (10G Ethernet)

4B/5B Encoding

Data	Code	Data	Code
0000	11110	1000	10010
0001	01001	1001	10011
0010	10100	1010	10110
0011	10101	1011	10111
0100	01010	1100	11010
0101	01011	1101	11011
0110	01110	1110	11100
0111	01111	1111	11101

Quiz Question

- The following are notable absentees in 4B/5B encoding.. Why?
  - 00000 ➢ Need transitions No transitions.
  - 00001 ➢ Need at least two ones
  - 11111 ➢ Need transitions
  - 10001 ➢ Control symbol! (start delimiter)

Physical Layer: Outline

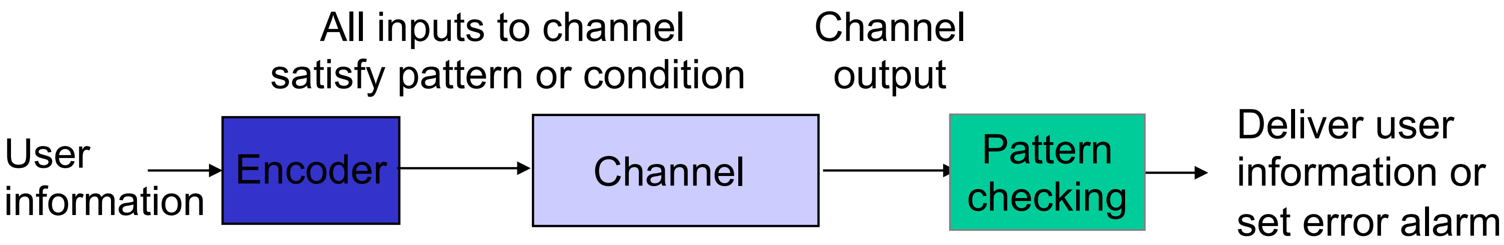
- Digital networks
- Characterization of Communication Channels
- Fundamental Limits in Digital Transmission
- Line Coding
- Modems and Digital Modulation
- Error Detection and Correction
- Wired PHY 101
- Wireless PHY 101

Error Control

- Channels introduce errors in digital communications
- Applications require certain reliability level
  - Data applications require error-free transfer
  - Voice & video applications tolerate some errors
- Error control may be needed to meet application requirement
- Error control ensures a data stream is transmitted to a certain level of accuracy despite errors
- Two basic approaches:
  - Error ***detection*** & retransmission (ARQ)
  - Forward error ***correction*** (FEC)

Key Idea

- All transmitted data blocks ("codewords") are chosen so that they satisfy a pattern
- If received block doesn't satisfy pattern, it is in error
- Redundancy: Only a subset of all possible blocks can be valid codewords
- Undetectable Error: When channel transforms a codeword into another valid codeword



Single Parity Check

- Append an parity bit to *k* information bits

Info Bits: $b_1, b_2, b_3, \ldots, b_k$

1s $\begin{cases} \text{even} & 0 \\ \text{odd} & 1 \end{cases}$

Check Bit: $b_{k+1} = b_1 + b_2 + b_3 + \ldots + b_k$ modulo 2

Codeword: $(b_1, b_2, b_3, \ldots, b_{k,}, b_{k+1})$

- All codewords have even # of 1s

- Receiver checks to see if # of 1s is even
  - All error patterns that create an odd # of 1 bits are detectable
  - All even-numbered error patterns are undetectable
- ASCII code is precisely such as code (7+1 bits)

Quiz Question: Single Parity Code

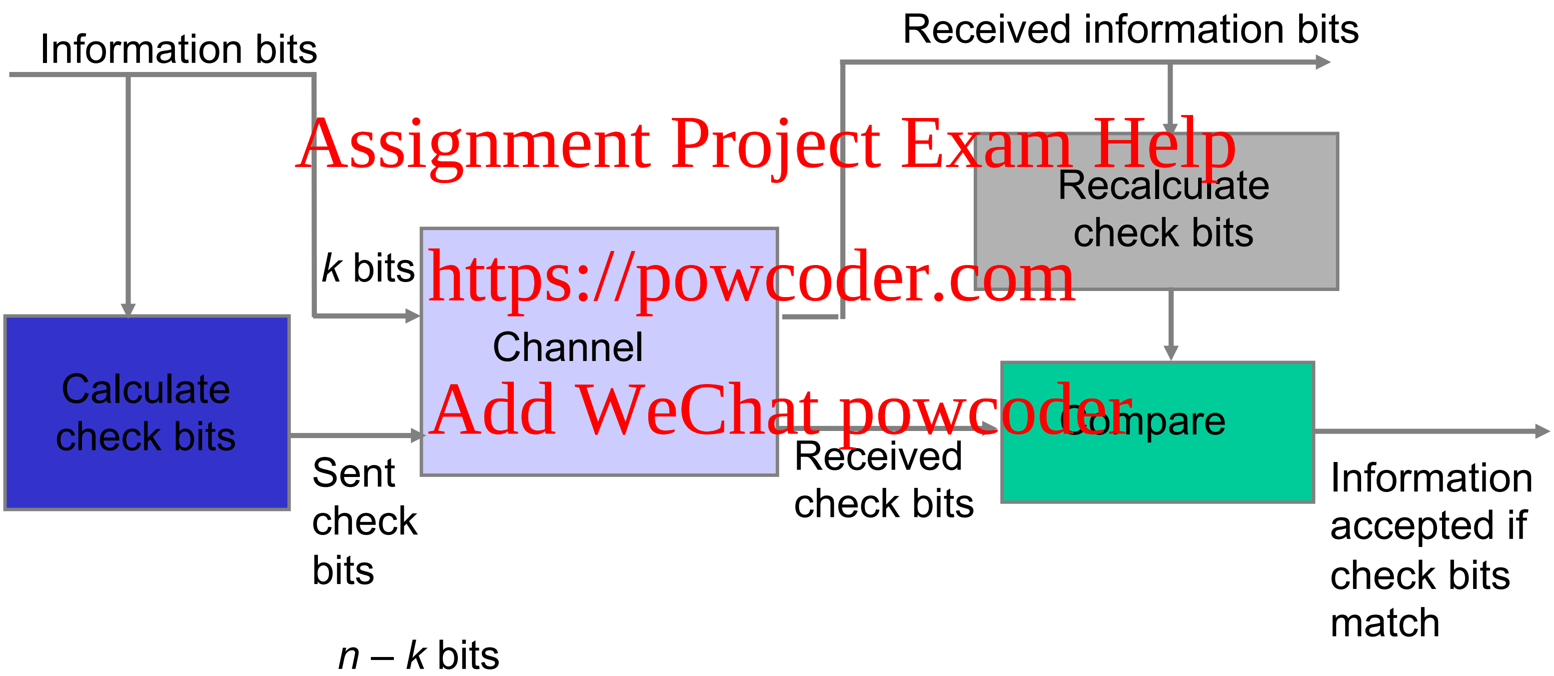
- Information (7 bits): (0, 1, 0, 1, 1, 0, 0)
- Parity Bit? ("True" → 1, "False" → 0)

  $b_8 = 0 + 1 + 0 + 1 + 1 + 0 = 1$

  Codeword (8 bits): (0, 1, 0, 1, 1, 0, 0, 1)

- If single error in bit 3? (0, 1, 1, 1, 1, 0, 0, 1)
  - # of 1's =5, odd => Error detected
- If errors in bits 3 and 5? (0, 1, 1, 1, 0, 0, 0, 1)
  - # of 1's =4, even => Error not detected

Parity Checkbits & Error Detection



How good is the single parity check code?

- *Redundancy*: Single parity check code adds 1 redundant bit per k information bits: overhead = $1/(k+1)$
- *Coverage*: all error patterns with odd # of errors can be detected
  - An error pattern is a binary (k+1)-tuple with 1's where errors occur and 0's elsewhere
  - Of 2^{k+1} binary (k+1)-tuples, ½ are odd, so 50% of error patterns can be detected
- Is it possible to detect more errors if we add more check bits?
- Yes, with the right codes

What if bit errors are random?

- Many transmission channels introduce bit errors at random, independently of each other, and with probability *p*
- Some error patterns are more probable than others:

$$P[10000000] = p(1-p)^7 = (1-p)^8\left(\frac{p}{1-p}\right) \text{ and}$$

$$P[11000000] = p^2(1-p)^6 = (1-p)^8\left(\frac{p}{1-p}\right)^2$$

- In any worthwhile channel $p<0.5$, and so $(p/(1-p))<1$
- It follows that patterns with 1 error are more likely than patterns with 2 errors and so forth
- What is the probability that an undetectable error pattern occurs?

Single parity check code with random bit errors

- Undetectable error pattern if even # of bit errors:

$$P[\text{error detection failure}] = P[\text{undetectable error pattern}]$$

$$= P[\text{error patterns with even number of 1s}]$$

$$= \binom{n}{2} p^2 (1-p)^{n-2} + \binom{n}{4} p^4 (1-p)^{n-4} + \ldots$$

- Quiz! What's the probability for n=32, $p=10^{-3}$?

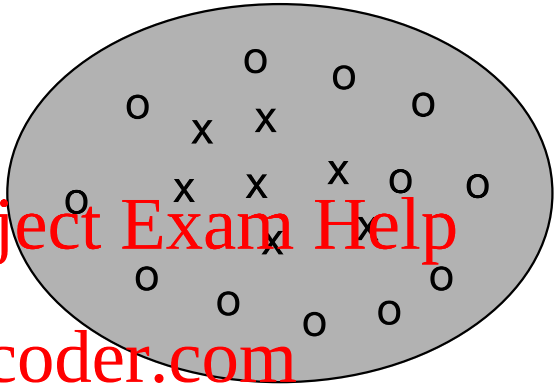
$$P[\text{undetectable error}] = \binom{32}{2}(10^{-3})^2(1-10^{-3})^{30} + \binom{32}{4}(10^{-3})^4(1-10^{-3})^{28}$$

$$\approx 496\,(10^{-6}) + 35960(10^{-12}) \approx 4.96(10^{-4})$$

- For this example, roughly 1 in 2000 transmissions will result in an undetectable error

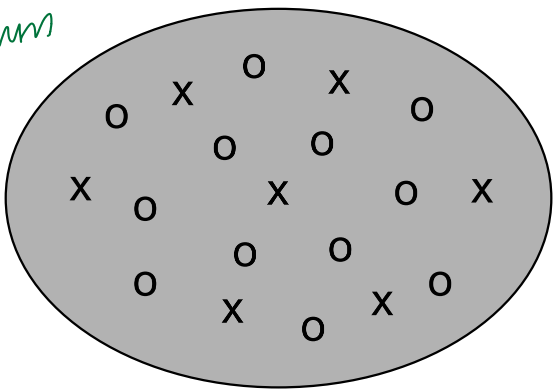
What is a good code?

- Most channels will have relatively few bit errors
- Erroneous codewords transmitted over those channels will map to nearby n-tuples
- If valid codewords are close to each other, then detection failures may occur
- Good codes should the minimum maximize separation between valid codewords



Poor distance properties

x = valid codewords
o = non-codewords



Good distance properties

Two-Dimensional Parity Check

- More parity bits to improve coverage
- Arrange information as columns
- Add single parity bit to each column
- Add a final "parity" column
- Used in early error control systems

1	0	0	1	0	0
0	1	0	0	0	1
1	0	0	1	0	0
1	1	0	1	1	0
1	0	0	1	1	1

Last column consists of check bits for each row

Bottom row consists of check bit for each column

Error-detecting capability

1	0	0	1	0	0
0	0	0	0	0	1
1	0	0	1	0	0
1	1	0	1	1	0
1	0	0	1	1	1

detect
+
correct

One error

1	0	0	1	0	0
0	0	0	0	0	1
1	0	0	1	0	0
1	0	0	1	1	0
1	0	0	1	1	1

detect only

Two errors

1, 2, or 3 errors can always be detected; Not all patterns >4 errors can be detected

1	0	0	1	0	0
0	0	0	1	0	1
1	0	0	1	0	0
1	0	0	1	1	0
1	0	0	1	1	1

detect
only

Three errors

1	0	0	1	0	0
0	0	0	1	0	1
1	0	0	1	0	0
1	0	0	0	1	0
1	0	0	1	1	1

Four errors

rectangler shaped.
error canot be detect

Arrows indicate failed check bits

Other Error Detection Codes

- Many applications require very low error rate
- Need codes that detect more number of errors
- Single parity check codes do not detect enough errors
- Two-dimensional codes require too many check bits
- The following error detecting codes are widely used in practice:
  - Internet Check Sums
  - CRC Polynomial Codes

Internet Checksum

- Several Internet protocols (e.g. IP, TCP, UDP) use check bits to detect errors in the ***header***
- A checksum is calculated for header contents and included in a special field.
- Checksum is potentially recalculated at every router, so algorithm selected for ease of implementation in software
- Let header consist of L, 16-bit words,

  $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{L-1}$
- The algorithm appends a 16-bit checksum $\mathbf{b}_L$

Checksum Calculation

The checksum $\mathbf{b}_L$ is calculated as follows:

- Treating each 16-bit word as an integer, find

$$\mathbf{x} = \mathbf{b}_0 + \mathbf{b}_1 + \mathbf{b}_2 + \ldots + \mathbf{b}_{L-1} \text{ modulo } 2^{16}-1$$

- The checksum is then given by:

$$\mathbf{b}_L = -\,\mathbf{x} \quad \text{modulo } 2^{16}-1$$

Thus, the headers must satisfy the following ***pattern*** at the receiver:

$$\mathbf{0} = \mathbf{b}_0 + \mathbf{b}_1 + \mathbf{b}_2 + \ldots + \mathbf{b}_{L-1} + \mathbf{b}_L \text{ modulo } 2^{16}-1$$

- The checksum calculation is carried out in software using one’s complement arithmetic

Internet Checksum Example

Use Modulo Arithmetic

- Assume 4-bit words
- Use mod 2^4-1 (= 15) arithmetic
- $\underline{b}_0$=1100 = 12
- $\underline{b}_1$=1010 = 10
- $\underline{b}_0$+$\underline{b}_1$=12+10=7 mod15
- $\underline{b}_2$ = -7 = 8 mod15
- Therefore
- $\underline{b}_2$=1000

Use Binary Arithmetic

- Note 16 =1 mod15
- So: 10000 = 0001 mod15
- leading bit wraps around

$b_0 + b_1$ = 1100+1010
=10110
=10000+0110
=0001+0110
=0111
=7

Take 1's complement
b_2 = -0111 =1000

Polynomial Codes

- Polynomials instead of vectors for codewords
- Polynomial arithmetic instead of check sums
- Implemented using shift-register circuits
- Also called *cyclic redundancy check (CRC)*
- Most data communications standards use polynomial codes for error detection
  – Have very simple hardware implementations
- Polynomial codes also basis for powerful error-correction methods

Binary Polynomial Arithmetic

- Binary vectors map to polynomials

$$(i_{k-1}, i_{k-2}, \ldots, i_2, i_1, i_0) \rightarrow i_{k-1}x^{k-1} + i_{k-2}x^{k-2} + \ldots + i_2x^2 + i_1x^1 + i_0$$

Addition:

$$(x^7 + x^6 + 1) + (x^6 + x^5) = x^7 + x^6 + x^6 + x^5 + 1$$

$$= x^7 + (1+1)x^6 + x^5 + 1$$

$$= x^7 + x^5 + 1 \quad \text{since} \quad 1 + 1 = 0 \bmod 2$$

Multiplication:

$$(x+1)(x^2 + x + 1) = x(x^2 + x + 1) + 1(x^2 + x + 1)$$

$$= (x^3 + x^2 + x) + (x^2 + x + 1)$$

$$= x^3 + 1$$

Binary Polynomial Division

- Division with Decimal Numbers

$$
\begin{array}{r@{\,}l}
 & 34 \\ \cline{2-2}
35\;) & 1222 \\
 & 105 \\ \cline{2-2}
 & 172 \\
 & 140 \\ \cline{2-2}
 & 32
\end{array}
$$

quotient

dividend

divisor

remainder

dividend = quotient x divisor + remainder

1222 = 34 x 35 + 32

- Polynomial Division

$$
\begin{array}{r@{\,}l}
 & x^3 + x^2 + x \quad = q(x) \text{ quotient} \\ \cline{2-2}
x^3 + x + 1\;) & x^6 + x^5 \\
 & x^6 + \quad x^4 + x^3 \\ \cline{2-2}
 & x^5 + x^4 + x^3 \\
 & x^5 + \quad x^3 + x^2 \\ \cline{2-2}
 & x^4 + \quad x^2 \\
 & x^4 + \quad x^2 + x \\ \cline{2-2}
 & x \quad = r(x) \text{ remainder}
\end{array}
$$

divisor

dividend

Note: Degree of r(x) is less than degree of divisor

$$
\begin{array}{r@{\,}l}
 & x^3+x^2+x \\ \cline{2-2}
x^3+x+1\;) & x^6+x^5 \\
 & x^6 \quad +x^4+x^3 \\ \cline{2-2}
 & x^5+x^4+x^3 \\
 & x^5 \quad +x^3+x^2 \\ \cline{2-2}
 & x^4 \quad +x^2 \\
 & x^4 \quad +x^2+x \\ \cline{2-2}
 & x
\end{array}
$$

Polynomial Coding

- k information bits define polynomial of degree k-1

$$i(x) = i_{k-1}x^{k-1} + i_{k-2}x^{k-2} + \ldots + i_2x^2 + i_1x + i_0$$

- Code has binary *generating polynomial* of degree n-k

$$g(x) = x^{n-k} + g_{n-k-1}x^{n-k-1} + \ldots + g_2x^2 + g_1x + 1$$

- Find *remainder polynomial* of at most degree n-k-1

$$\begin{array}{r|l} & q(x) \\ \hline g(x) & x^{n-k}\, i(x) \\ & r(x) \end{array}$$

$x^{n-k}i(x) = q(x)g(x) + r(x)$

- Define the *codeword polynomial* of degree n-1

$$\boxed{\underbrace{b(x)}_{\text{n bits}} = \underbrace{x^{n-k}i(x)}_{\text{k bits}} + \underbrace{r(x)}_{\text{n-k bits}}}$$

is always a multiple of g(x)

Quiz Q: Find codeword if k=4, n-k=3

And: Generator polynomial: $g(x) = x^3 + x + 1$

Information: (1,1,0,0) $i(x) = x^3 + x^2$

Encoding: $x^3 i(x) = x^6 + x^5$

Quiz Q: Find codeword if k=4, n-k=3

And: Generator polynomial: $g(x) = x^3 + x + 1$

Information: (1,1,0,0) $i(x) = x^3 + x^2$

Encoding: $x^3 i(x) = x^6 + x^5$

$$
\begin{array}{r}
x^3 + x^2 + x \\
x^3 + x + 1 \;\big)\; \overline{\boxed{x^6 + x^5}} \\
x^6 + \phantom{x^5 +{}} x^4 + x^3 \\
\hline
\boxed{x^5 + x^4 + x^3} \\
x^5 + \phantom{x^4 +{}} x^3 + x^2 \\
\hline
\boxed{x^4 + \phantom{x^3 +{}} x^2} \\
x^4 + \phantom{x^3 +{}} x^2 + x \\
\hline
\boxed{x}
\end{array}
$$

Quiz Q: Find codeword if k=4, n-k=3

And: Generator polynomial: $g(x) = x^3 + x + 1$ 1 0 0 1

Information: (1,1,0,0) $i(x) = x^3 + x^2$

Encoding: $x^3 i(x) = x^6 + x^5$

$$
\begin{array}{rl}
 & x^3 + x^2 + x \\
\cline{2-2}
x^3 + x + 1 \;) & x^6 + x^5 \\
 & x^6 + \phantom{x^5 +{}} x^4 + x^3 \\
\cline{2-2}
 & \phantom{x^6 +{}} x^5 + x^4 + x^3 \\
 & \phantom{x^6 +{}} x^5 + \phantom{x^4 +{}} x^3 + x^2 \\
\cline{2-2}
 & \phantom{x^6 + x^5 +{}} x^4 + \phantom{x^3 +{}} x^2 \\
 & \phantom{x^6 + x^5 +{}} x^4 + \phantom{x^3 +{}} x^2 + x \\
\cline{2-2}
 & \phantom{x^6 + x^5 + x^4 + x^3 + x^2 +{}} x
\end{array}
$$

$$
\begin{array}{rl}
 & 1110 \\
\cline{2-2}
1011 \;) & 1100000 \\
 & 1011 \\
 & 1110 \\
 & 1011 \\
 & 1010 \\
 & 1011 \\
 & 010
\end{array}
$$

Transmitted codeword:

$b(x) = x^6 + x^5 + x$

$\underline{b} = (1,1,0,0,0,1,0)$

The *Pattern* in Polynomial Coding

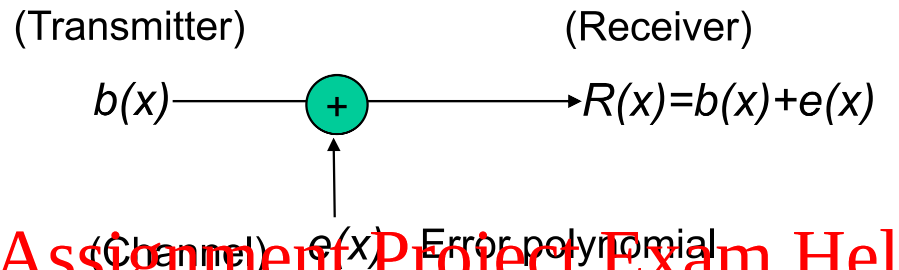
- All codewords satisfy the following **pattern**:

$$b(x) = x^{n-k}i(x) + r(x) = q(x)g(x) + r(x) + r(x) = q(x)g(x)$$

- All codewords are a multiple of $g(x)$!
- Receiver should divide received n-tuple by g(x) and check if remainder is zero

- If remainder is non-zero, then received n-tuple is not a codeword

Undetectable error patterns



- *e(x)* has 1's in error locations & 0's elsewhere
- Receiver divides the received polynomial *R(x)* by *g(x)*
- Undetectable error: If *e(x)* is a multiple of *g(x)*, that is, *e(x)* is a non-zero codeword, then

$$R(x) = b(x) + e(x) = q(x)g(x) + q'(x)g(x)$$

- *The set of undetectable error polynomials is the set of nonzero code polynomials*
- *Choose the generator polynomial so that selected error patterns can be detected.*

Designing good polynomial codes

- Select generator polynomial so that likely error patterns are not multiples of $g(x)$
- *Detecting Single Errors*

  - $e(x) = x^i$ for error in location $i+1$
  - If $g(x)$ has more than 1 term, it cannot divide x^i

- *Detecting Double Errors*
  - $e(x) = x^i + x^j = x^i(x^{j-i}+1)$ where $j>i$

  - If $g(x)$ has more than 1 term, it cannot divide x^i
  - If g(x) is a *primitive* polynomial, it cannot divide x^m+1 for all $m<2^{n-k}-1$ (Need to keep codeword length less than $2^{n-k}-1$)
  - Primitive polynomials can be found by consulting coding theory books

Standard Generator Polynomials

CRC = cyclic redundancy check

- CRC-8: $= x^8 + x^2 + x + 1$ ATM

- CRC-16: $= x^{16} + x^{15} + x^2 + 1$

$= (x+1)(x^{15} + x + 1)$ Bisync

- CCITT-16: $= x^{16} + x^{12} + x^5 + 1$ HDLC, XMODEM, V.41

- CCITT-32: IEEE 802, DoD, V.42

$$= x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Hamming Codes

- Class of *error-correcting* codes
- Capable of correcting all *single-error* patterns

- Provably optimal for 1-bit errors
- Very less redundancy, e.g. 1-bit error proof – adds O(log n) bits of redundancy for n bit sequences

m=3 Hamming Code

- Information bits are b_1, b_2, b_3, b_4
- Equations for parity checks b_5, b_6, b_7

$$b_5 = b_1 \qquad\quad + b_3 + b_4$$

$$b_6 = b_1 + b_2 \qquad\quad + b_4$$

$$b_7 = \qquad + b_2 + b_3 + b_4$$

- There are 2^4=16 codewords
- (0,0,0,0,0,0,0) is a codeword

My "simple" proof of optimality

Assume you got the following 7 bit sequences and make the following checks:

$$b_5 = b_1 + b_3 + b_4$$
$$b_6 = b_1 + b_2 + b_4$$
$$b_7 = b_2 + b_3 + b_4$$

Case	b_5 match	b_6 match	b_7 match
No error			
b_1 flipped			
b_2 flipped			
b_3 flipped			
b_4 flipped			
b_5 flipped			
b_6 flipped			
b_7 flipped			

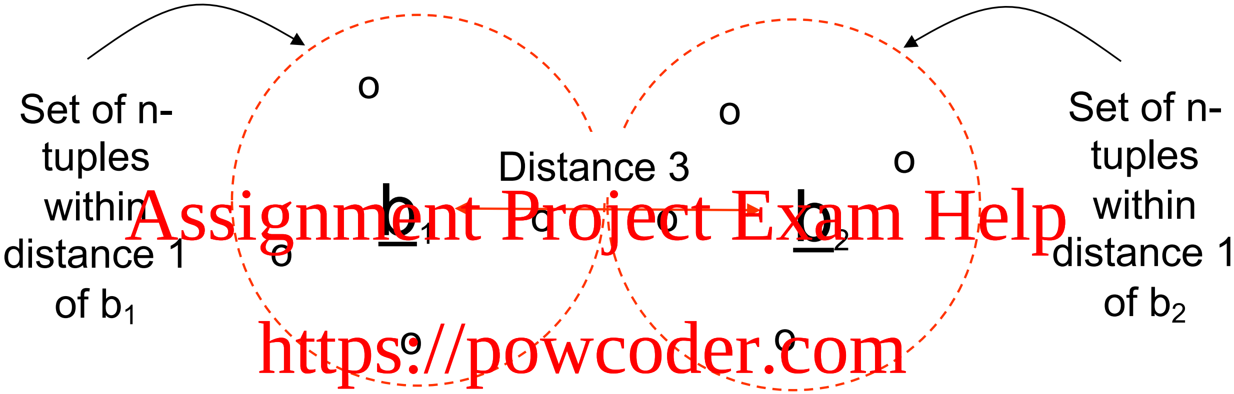
My ”simple” proof of optimality

Assume you got the following 7 bit sequences and make the following checks:

$$
\begin{aligned}
b_5 &= b_1 \qquad\quad + b_3 + b_4 \\
b_6 &= b_1 + b_2 \qquad\quad + b_4 \\
b_7 &= \qquad + b_2 + b_3 + b_4
\end{aligned}
$$

Case	b_5 match	b_6 match	b_7 match
No error	✓	✓	✓
b_1 flipped	X	X	✓
b_2 flipped	✓	X	X
b_3 flipped	X	✓	X
b_4 flipped	X	X	X
b_5 flipped	X	✓	✓
b_6 flipped	✓	X	✓
b_7 flipped	✓	✓	X

Why is Hamming a "good code"?



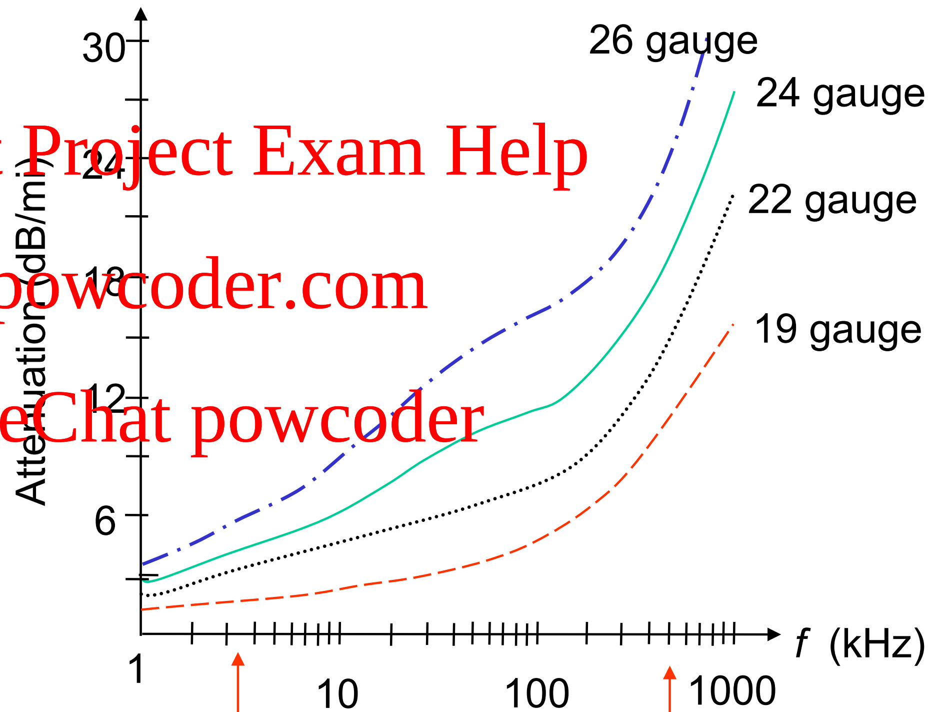
- Two valid bit sequences have a minimum distance of 3 bit flips
- Spheres of distance 1 around each codeword do not overlap
- If a single error occurs, the resulting n-tuple will be in a unique sphere around the original codeword
- Thus, receiver can correct erroneous reception back to original codeword

Physical Layer: Outline

- Digital networks
- Characterization of Communication Channels
- Fundamental Limits in Digital Transmission
- Line Coding
- Modems and Digital Modulation
- Error Detection and Correction
- Wired PHY 101
- Wireless PHY 101

Twisted Pair

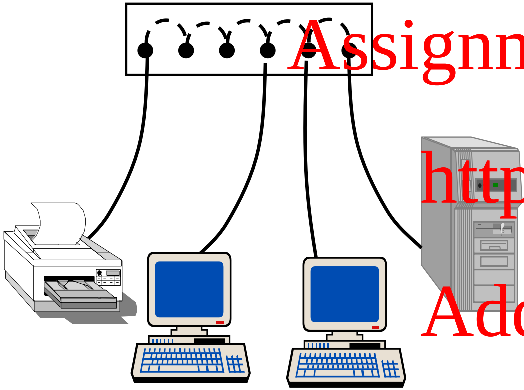
- Two insulated copper wires arranged in a regular spiral pattern to minimize interference
- Various thicknesses, e.g. 0.016 inch (24 gauge)
- Low cost
- Telephone subscriber loop from customer to CO
- Old trunk plant connecting telephone COs
- Intra-building telephone from wiring closet to desktop



Lower attenuation rate for analog telephone

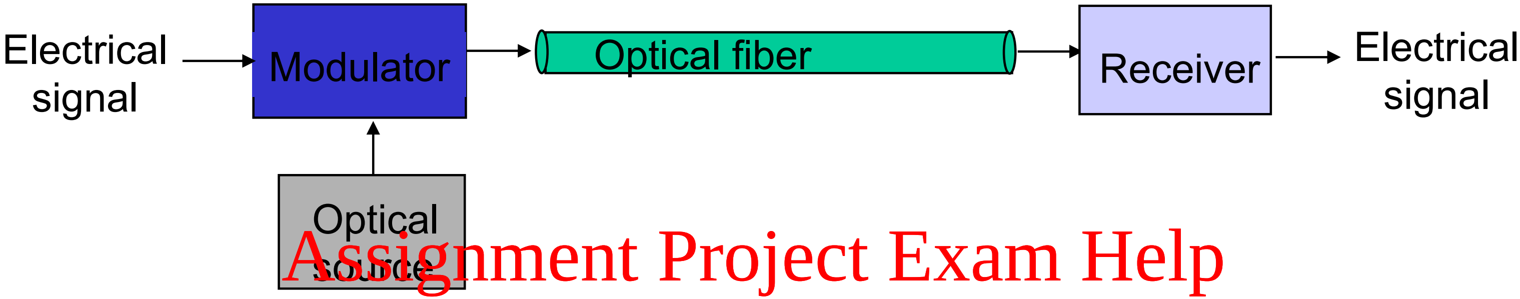
Higher Attenuation rate for DSL

Ethernet LANs



- Evolved from 10 -> 100 → 1000 Mbps to now 10Gbps
- All use twisted pair in some form!
- 10BASE-T Ethernet
  - 10 Mbps, Baseband, Twisted pair
  - Two Cat3 pairs
  - Manchester coding, 100 meters
- 100BASE-T4 *Fast* Ethernet
  - 100 Mbps, Baseband, Twisted pair
  - Four Cat3 pairs
  - Three pairs for one direction at-a-time
  - 100/3 Mbps per pair;
  - 3B6T line code, 100 meters
- 1000BASE-T
  - 8b10b encoding, Four pairs

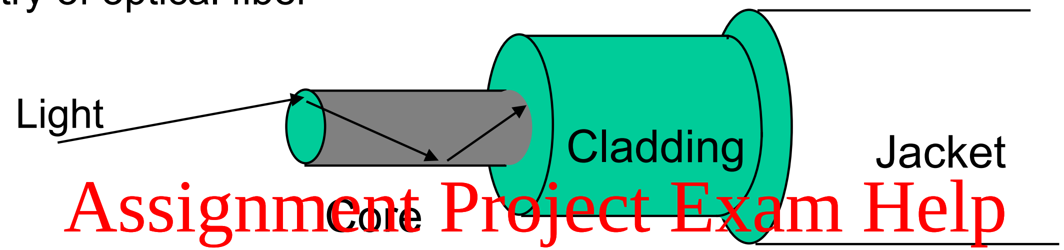
Optical Fiber



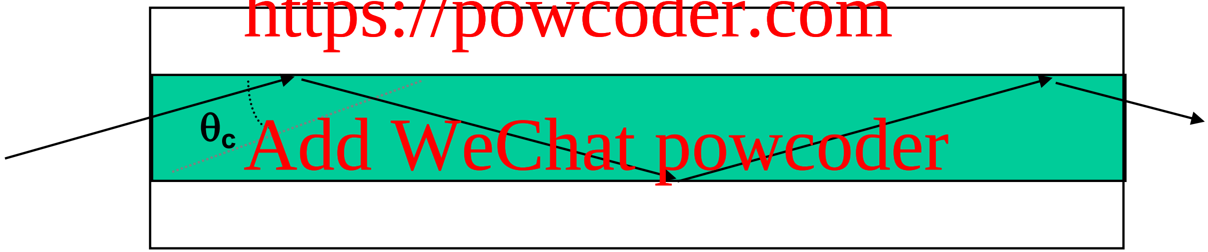
- Light sources (lasers, LEDs) generate pulses of light that are transmitted on optical fiber
  - Very long distances (>1000 km)
  - Very high speeds (>40 Gbps/wavelength)
  - Nearly error-free (BER of 10^{-15})
- Profound influence on network architecture
  - Dominates long distance transmission
  - Distance less of a cost factor in communications
  - Plentiful bandwidth for new services

Transmission in Optical Fiber

Geometry of optical fiber

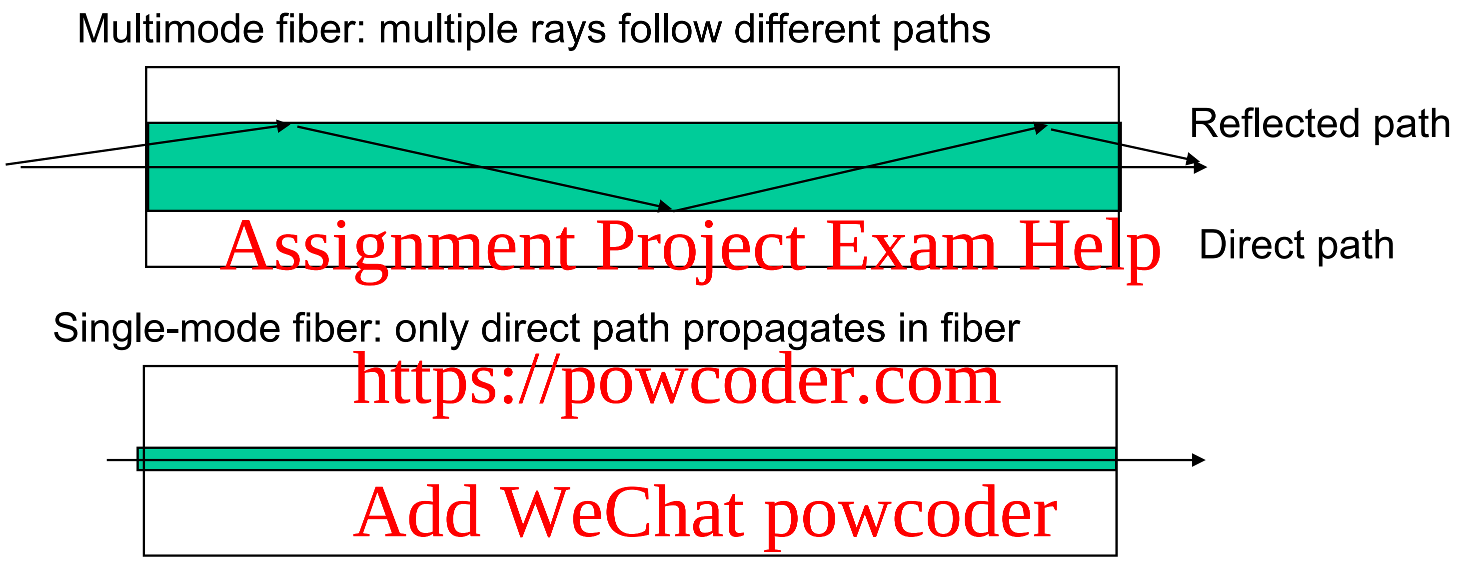


Total Internal Reflection in optical fiber



- Very fine glass cylindrical core surrounded by concentric layer of glass (cladding)
- Core has higher index of refraction than cladding
- Light rays incident at less than critical angle θ_c is completely reflected back into the core

Multimode & Single-mode Fiber



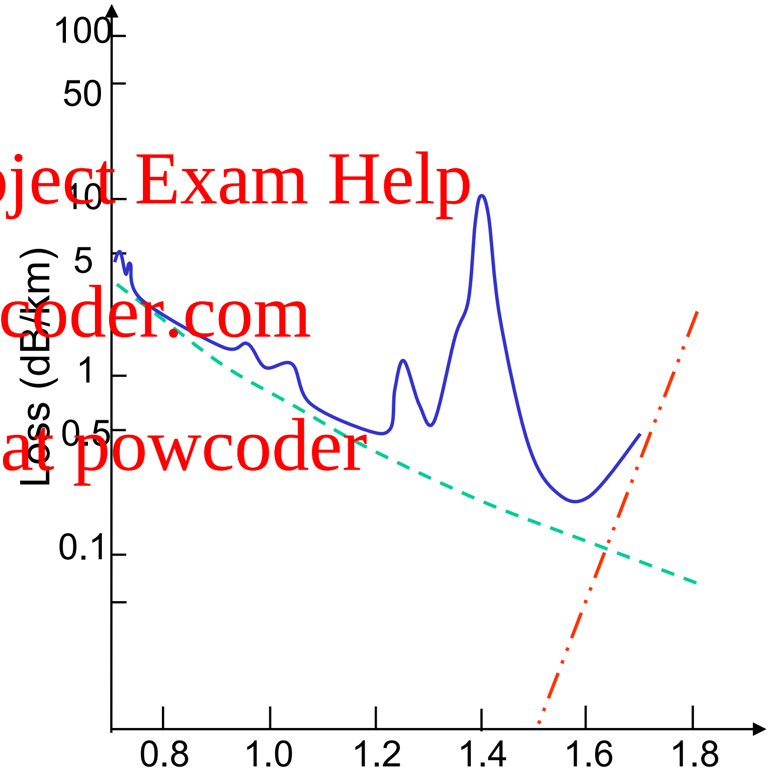
- Multi Mode: Thicker core, shorter reach
  - Rays on different paths interfere causing dispersion & limiting bit rate
- Single Mode: Very thin core supports only one mode (path)
  - More expensive lasers, but achieves very high speeds

Huge Available Bandwidth

- Optical range from λ_1 to $\lambda_1+\Delta\lambda$ contains bandwidth

$$B = f_1 - f_2 = \frac{\nu}{\lambda_1} - \frac{\nu}{\lambda_1 + \Delta\lambda}$$

$$= \frac{\nu}{\lambda_1}\left\{\frac{\Delta\lambda/\lambda_1}{1+\Delta\lambda/\lambda_1}\right\} \cong \frac{\nu\Delta\lambda}{\lambda_1^2}$$



Quiz Question

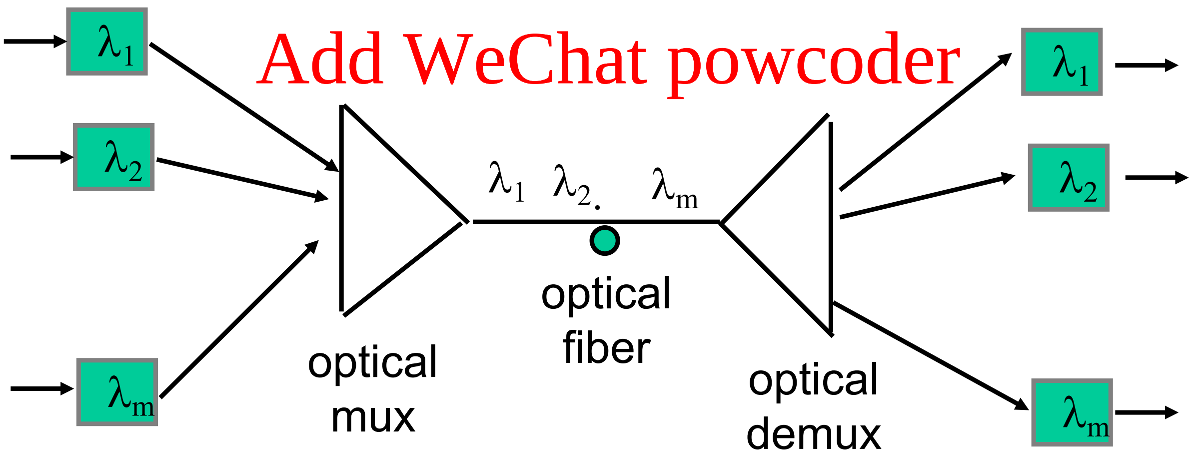
How much optical fiber bandwidth is available between:

λ_1 = 1450 nm and $\lambda_1 + \Delta\lambda$ =1650 nm:

Answer: $$B = \frac{2(10^8)\,\mathrm{m/s}\;200\,\mathrm{nm}}{(1450\,\mathrm{nm})^2} \approx 19\ \mathrm{THz}$$

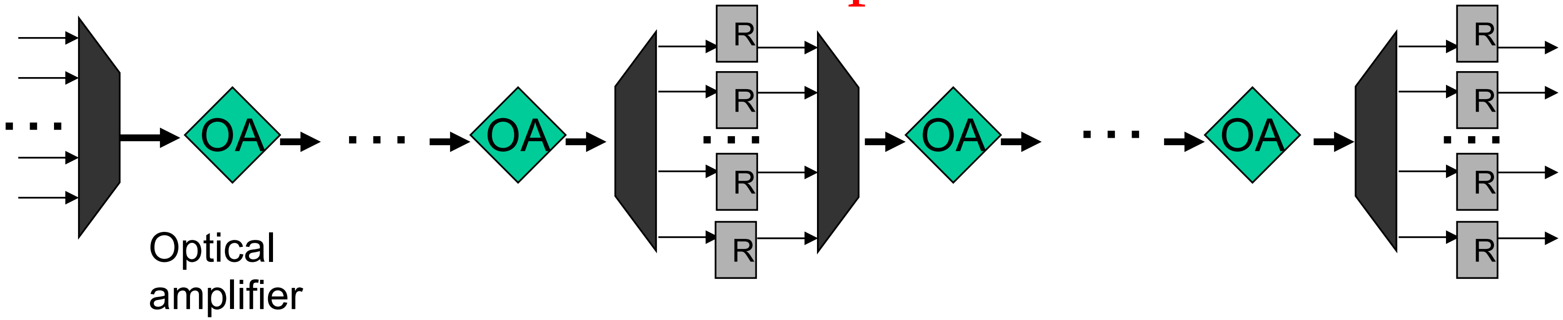
Wavelength-Division Multiplexing

- Different wavelengths carry separate signals
- Multiplex into shared optical fiber
- Each wavelength like a separate circuit
- A single fiber can carry 160 wavelengths, 10 Gbps per wavelength: 1.6 Tbps!



How Do We Extend Range

- Use combinations of optical amplifiers and regenerators
- More amplifiers than regenerators (why?)



Physical Layer: Outline

- Digital networks
- Characterization of Communication Channels
- Fundamental Limits in Digital Transmission
- Line Coding
- Modems and Digital Modulation
- Error Detection and Correction
- Wired PHY 101
- Wireless PHY 101