Announcements

- No Sunday afternoon office hours
- Homework due Monday
“I plan to visit my brothel this weekend”

ASCII codes for \texttt{r} and \texttt{l}:
\[
\begin{align*}
l & = 01101100 \\
r & = 01101010
\end{align*}
\]

“He lives near to my mother and I plan to visit her for Mother’s Day”
One Bad Bit

if( safe ) . . .

===> 01101001 01100110 00101000 00100000 01110011
     01100001 01100110 01100101 00100000 00101001 . . .

===> 01101001 01100110 00101000 00100001 01110011
     01100001 01100110 01100101 00100000 00101001 . . .

if(!safe ) . . .
Random Bit Errors

is delivered as

or

with small \((10^{-12} - 10^{-3})\), independent probability \(p_e\) of an error in any given bit
Burst Errors

\[01010001\]

becomes

\[01??1??1\]

where each \( ? \) is randomly replaced by a 0 or 1 with equal probability.
Burst Errors

Given the 5 bits of random interference

\[ 10100 \]

the transmitted message becomes

\[ 010100001 \]

or

\[ 01?????1 \]

or

\[ 011101001 \]
Burst Errors

Given the 5 bits of random interference

\[0\ 1\ 0\ 1\ 0\]

the transmitted message becomes

\[0\ 1\ 0\ 1\ 0\ 0\ 0\ 1\]

or

\[0\ 1\ ?\ ?\ ?\ ?\ ?\ ?\ 1\]

or

\[0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\]
Burst Errors

\[
\begin{array}{cccccccccccccccccccc}
0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
\end{array}
\]

becomes

\[
\begin{array}{cccccccccccccccccccc}
\end{array}
\]
Random Packet Error Rate

Given:
- a packet contains $n$ bits,
- each bit may be damaged with probability $p_e$

What is the probability packet is received incorrectly.
Random Packet Error Rate

Given:
- a packet contains $n$ bits,
- each bit may be damaged with probability $p_e$

What is the probability packet is received incorrectly?

Intuitively:

$$\text{Prob}( \text{packet contains some damaged bit} ) = ???$$
Given:
- a packet contains $n$ bits,
- each bit may be damaged with probability $p_e$

What is the probability packet is received incorrectly.

Intuitively:

\[ \text{Prob( packet contains some damaged bit )} = np_e \]
Random Packet Error Rate

Given:
- a packet contains $n$ bits,
- each bit may be damaged with probability $p_e$

What is the probability packet is received incorrectly.

Prob( a single bit is not damaged ) = ???

Prob( a single bit is not damaged ) = $1 - p_e$
Random Packet Error Rate

Given:
- a packet contains $n$ bits,
- each bit may be damaged with probability $p_e$

What is the probability that the packet is received incorrectly?

$$\text{Prob( a single bit is not damaged )} = (1 - p_e)$$
Random Packet Error Rate

Given:

- A packet contains $n$ bits,
- Each bit may be damaged with probability $p_e$

What is the probability packet is received incorrectly?

$\text{Prob}(\text{a single bit is not damaged}) = (1 - p_e)$

$\text{Prob}(\text{all } n \text{ packets bit arrive correctly}) = ???$
Random Packet Error Rate

Given:

- A packet contains $n$ bits,
- Each bit may be damaged with probability $p_e$

What is the probability packet is received incorrectly.

Prob( a single bit is not damaged ) = $\left( 1 - p_e \right)$

Prob( all $n$ packets bit arrive correctly ) = $\left( 1 - p_e \right)^n$
Random Packet Error Rate

Given:
- A packet contains $n$ bits,
- Each bit may be damaged with probability $p_e$

What is the probability packet is received incorrectly.

\[
\text{Prob( a single bit is not damaged )} = (1 - p_e)
\]

\[
\text{Prob( all n packets bit arrive correctly )} = (1 - p_e)^n
\]

\[
= 1 - n p_e + \binom{n}{2} p_e^2 - \binom{n}{3} p_e^3 + \ldots
\]
The Binomial Theorem

\[(a + b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k} b^k\]

\[
(a + b)^3 = \sum_{k=0}^{3} \binom{3}{k} a^{3-k} b^k \\
= \binom{3}{0} a^{3-0} b^0 + \binom{3}{1} a^{3-1} b^1 + \binom{3}{2} a^{3-2} b^2 + \binom{3}{3} a^{3-3} b^3
\]

\[\binom{n}{k} = \frac{n!}{k!(n-k)!}\]
\[(a + b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k} b^k\]

The Binomial Theorem

\[(a + b)^3 = \sum_{k=0}^{3} \binom{3}{k} a^{3-k} b^k\]

\[= \binom{3}{0} a^3 b^0 + \binom{3}{1} a^2 b^1 + \binom{3}{2} a^1 b^2 + \binom{3}{3} a^0 b^3\]

\[= b^0 + \binom{3}{1} b^1 + \binom{3}{2} b^2 + \binom{3}{3} b^3\]
Random Packet Error Rate

Given:
- A packet contains \( n \) bits,
- Each bit may be damaged with probability \( p_e \)

What is the probability packet is received incorrectly.

\[
\text{Prob( a single bit is not damaged )} = (1 - p_e)
\]
\[
\text{Prob( all n packets bit arrive correctly )} = (1 - p_e)^n
\]
\[
= 1 - np_e + \binom{n}{2} p_e^2 - \binom{n}{3} p_e^3 + \ldots
\]
Random Packet Error Rate

Given:

- a packet contains $n$ bits,
- each bit may be damaged with probability $p_e$

What is the probability packet is received incorrectly.

$$\text{Prob( a single bit is not damaged )} = (1 - p_e)$$

$$\text{Prob( all n packets bit arrive correctly )} = (1 - p_e)^n$$

$$= 1 - np_e + (\binom{n}{2} p_e^2 - (\binom{n}{3} p_e^3 + \ldots$$

$$\text{Prob( some packet bit is damaged )} = ???$$
Random Packet Error Rate

Given:
- a packet contains $n$ bits,
- each bit may be damaged with probability $p_e$

What is the probability packet is received incorrectly.

$$\text{Prob( a single bit is not damaged )} = (1 - p_e)$$
$$\text{Prob( all } n \text{ packets bit arrive correctly )} = (1 - p_e)^n$$
$$= 1 - n p_e + (n_2) p_e^2 - (n_3) p_e^3 + \ldots$$
$$\text{Prob( some packet bit is damaged )} = 1 - (1 - p_e)^n$$
Random Packet Error Rate

Given:
1. a packet contains n bits,
2. each bit may be damaged with probability $p_e$

What is the probability packet is received incorrectly.

$$\text{Prob( a single bit is not damaged )} = (1 - p_e)$$

$$\text{Prob( all n packets bit arrive correctly )} = (1 - p_e)^n = 1 - n p_e + \binom{n}{2} p_e^2 - \binom{n}{3} p_e^3 + \ldots$$

$$\text{Prob( some packet bit is damaged )} = 1 - (1 - p_e)^n = 1 - (1 - n p_e + \binom{n}{2} p_e^2 - \binom{n}{3} p_e^3 + \ldots)$$

$$= n p_e - \binom{n}{2} p_e^2 + \binom{n}{3} p_e^3 - \ldots$$
Random Packet Error Rate

Given:
- a packet contains $n$ bits,
- each bit may be damaged with probability $p_e$

What is the probability packet is received incorrectly.

Intuitively:

\[
\text{Prob( packet contains some damaged bit )} = np_e
\]

Actually:

\[
\text{Prob( some packet bit is damaged )} = 1 - ( 1 - p_e )^n
= np_e - \binom{n}{2} p_e^2 + \binom{n}{3} p_e^3 - \ldots
\approx np_e
\]
Redundancy

F U CN RD THS U CNT SPL WRTH A DM!

Devl kikd outa hevn coz jelus of jesus & strts war. pd'off wiv god so corupts man (md by god) wiv apel. devl stays serpnt 4 hole life & man ruind. Woe un2 mnkind.

Smry of mltn's prdis lost
Redundancy

brothea, brotheb, brothec, brothed, brothee, brothèf, brothèg, brothèh, brothèi, brothèj, bothek, brothem, brothen, brotheo, brothep, brothèq, brothes, brothet, brotheu, brothev, brothew, brothex, brothey, brothez
Parity Check Coding

0 1 1 0 1 0 0 1 0 1 0 1 0 0 0 0 1 0 1 0 0 0 0 1 1
Parity Check Coding

- Divide data into sequences of fixed size $n$ (we will use 8 for examples).

```
0 1 1 0 1 0 0 1 0 1 0 1 0 0 0 1 0 1 0 0 0 0 0 1 1
```
Parity Check Coding

Divide data into sequences of fixed size \( n \) (we will use 8 for examples).

\[
\begin{array}{cccccccccccccc}
0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
\end{array}
\]
Parity Check Coding

For each sequence \(d_1d_2...d_n\) transmit \(n+1\) bits by adding an extra bit, \(d_{n+1}\), chosen so that the total number 1’s is even.

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

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

0 1 1 0 1 0 0 1 1
0 1 0 1 0 0 1 1
0 1 0 0 0 0 1 1
0 1 0 0 0 0 1 1
Parity Check Coding

For each sequence $d_1d_2...d_n$ transmit $n+1$ bits by adding an extra bit, $d_{n+1}$, chosen so that the total number of 1's is even.
Parity Check Coding

For each sequence $d_1d_2...d_n$ transmit $n+1$ bits by adding an extra bit, $d_{n+1}$, chosen so that the total number 1's is even.
Parity Check Coding

For each sequence $d_1d_2...d_n$ transmit $n+1$ bits by adding an extra bit, $d_{n+1}$, chosen so that the total number 1's is even.
Parity Check Coding

For each sequence $d_1d_2...d_n$ transmit $n+1$ bits by adding an extra bit, $d_{n+1}$, chosen so that the total number 1's is even.
Parity Check Coding

Divide received messages into subsequences of size $n+1$. 

011010010010100011110100000111

0110100110
0101001111
0100001111
Parity Check Coding

Reject if *any* subsequence contains an odd number of 1’s.
Reject if *any* subsequence contains an odd number of 1’s.
Parity Check Coding

Reject if **any** subsequence contains an odd number of 1’s.
Parity Check Coding

Reject if any subsequence contains an odd number of 1's.
Parity Check Coding

Reject if **any** subsequence contains an odd number of 1’s.
Parity Check Coding

Reject if **any** subsequence contains an odd number of 1's.
Parity Check Coding

- Divide received messages into subsequences of size $n+1$. 

```
01101001001010000110100000111
```
Parity Check Coding

Divide received messages into subsequences of size n+1.
Parity Check Coding

Reject if any subsequence contains an odd number of 1’s.
Parity Check Coding

Reject if any subsequence contains an odd number of 1’s.
Parity Check Coding

Reject if any subsequence contains an odd number of 1’s.
Parity Check Coding

Reject if any subsequence contains an odd number of 1’s.
Reject if any subsequence contains an odd number of 1’s.
Parity Check Coding

Reject if any subsequence contains an odd number of 1's.
Only treat first $n$ bits of each subsequence as data.
**TCP Segment Format**

<table>
<thead>
<tr>
<th>16</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Source Port</strong></td>
<td><strong>Destination Port</strong></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Sequence Number</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Acknowledgment Number</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Hdr Len</strong></td>
<td><strong>Flags</strong></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Error Check</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>DATA</strong></td>
<td></td>
</tr>
</tbody>
</table>
Random Bit Errors

Suppose \(01010001\) is delivered as

\[
\begin{align*}
01000001 & \quad \text{or} \quad 01011001 \\
01010011 & \quad \text{or} \quad 00010001 \\
11010001 & \quad \text{or} \quad 01000000
\end{align*}
\]

with small \((10^{-12} - 10^{-3})\), independent probability \(p_e\) of an error in any given bit.
Undetected Random Bit Errors

Suppose

is delivered as

or

or

or
Undetected Random Bit Errors

Suppose

is delivered as

or

or

or

or
Undetected Random Bit Errors

\[ \text{Prob( undected bit errors )} = \text{Prob( even \# of errors )} \]
\[ \text{Prob( undected bit errors )} < \text{Prob( more than 1 error)} \]
\[ \text{Prob( undected bit errors )} < 1 - ( \text{Prob( no error )} + \text{Prob( 1 error )} ) \]
Undetected Random Bit Errors

Prob( undected bit errors ) = Prob( even # of errors )
Prob( undected bit errors ) < Prob( more than 1 error)
Prob( undected bit errors ) < 
1 - ( Prob( no error ) + Prob( 1 error ) )
Prob( undected bit errors ) < 
1 - ( ??? + Prob( 1 error ) )
Undetected Random Bit Errors

\[ \text{Prob( undected bit errors )} = \text{Prob( even # of errors )} \]
\[ \text{Prob( undected bit errors )} < \text{Prob( more than 1 error) } \]
\[ \text{Prob( undected bit errors )} < 
1 - ( \text{Prob( no error )} + \text{Prob( 1 error )} ) \]
\[ \text{Prob( undected bit errors )} < 
1 - ( (1 - p_e)^{n+1} + ??? ) \]
Undetected Random Bit Errors

\[ \text{Prob( undected bit errors )} = \text{Prob( even # of errors )} \]

\[ \text{Prob( undected bit errors )} < \text{Prob( more than 1 error) } \]

\[ \text{Prob( undected bit errors )} < 1 - ( \text{Prob( no error )} + \text{Prob( 1 error )} ) \]

\[ \text{Prob( undected bit errors )} < 1 - \left( (1 - p_e)^{n+1} + (n+1) p_e (1 - p_e)^n \right) \]
Undetected Random Bit Errors

010100011

\[
\text{Prob( undetected bit errors)} = \text{Prob( even # of errors)}
\]
\[
\text{Prob( undetected bit errors)} < \text{Prob( more than 1 error)}
\]
\[
\text{Prob( undetected bit errors)} < 1 - ( \text{Prob( no error)} + \text{Prob( 1 error)} )
\]
\[
\text{Prob( undetected bit errors)} < 1 - ( (1 - p_e)^{n+1} + (n+1) p_e (1 - p_e)^n )
\]
\[
\cong n^2 p_e^2
\]
Burst Errors

0110100100101000011010000111

becomes

011?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? 111

\[ P(\text{undetected error}) = ? \]
Burst Errors

\[ 0110100100101000110100000111 \]

becomes

\[ 011?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? 111 \]

\[
P(\text{undetected error}) \approx \left(\frac{1}{2}\right)^{\# \text{ of parity bits involved}}\]
Burst Errors

0 1 1 0 1 0 0 1 0 | 0 1 0 1 0 0 0 0 1 | 1 0 1 0 0 0 0 1 1 1

becomes


\[ P(\text{undetected error}) = \left( \frac{1}{2} \right)^{\text{# of parity bits involved}} - \text{Prob( no error )} \]
Vertical Parity
# Vertical Parity & Burst Errors

## Short burst? (≤ width)

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**P(undetected error)**

\[
P(\text{undetected error}) = 0
\]

## Long burst? (> width)

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>?</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**P(undetected error)**

\[
P(\text{undetected error}) \approx \frac{1}{2^{\text{width}}}
\]
Vertical Parity & Burst Errors

Medium burst?
( width < burst )

\[
P(\text{undetected error}) = P(\text{no error detected}) - P(\text{no error})
\]
Vertical Parity & Burst Errors

Medium burst?
( width < burst )

\[
P(\text{undetected error}) = P(\text{no error detected}) - P(\text{no error})
\]

\[
= \frac{1}{2^{\text{width}}} - \frac{1}{2^{\text{burst}}}
\]
2 Dimensional Parity

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

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

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

0 1 1 0 1 0 0 1 0 0 1 0 1 0 0 0 1 1 0 0 1 0 1 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 0
### Error Correction

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- The table shows a grid with binary values (0s and 1s).
- The 'X' indicates an error that needs to be corrected.
- The corrected values are shown in red.
Error Detection

<table>
<thead>
<tr>
<th>01101Y010</th>
<th>01×010010</th>
</tr>
</thead>
<tbody>
<tr>
<td>010100011</td>
<td>010100011</td>
</tr>
<tr>
<td>01Y0001111</td>
<td>01000×111</td>
</tr>
<tr>
<td>0111110110</td>
<td>0111110110</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0110100010</th>
<th>010×00011</th>
</tr>
</thead>
<tbody>
<tr>
<td>010X0001111</td>
<td>010000111</td>
</tr>
<tr>
<td>0111110110</td>
<td>011×10110</td>
</tr>
<tr>
<td>0111110110</td>
<td>011×10110</td>
</tr>
</tbody>
</table>
Whoops!