1 Framing & Error Correction

1. The message `0110 0111 1110 0000 0111 1110 1111 1001` is to be transmitted. Show the bit sequence for the message if it is encoded using the following framing methods:

   (a) Flag bytes with byte stuffing. `<FLAG> = 0111 1110, ESC = 1110 0000`

   **Solution:** The message contains an occurrence of `<FLAG>` pattern at the second byte and an occurrence of the `<ESC>` pattern at the third byte:

   `0110 0111 1110 0000 0111 1110 1111 1001`

   We escape each of these by inserting the `<ESC>` pattern directly before them, resulting in the stuffed byte string:

   `0110 0111 1110 0000 1110 0000 1110 0000 0111 1110 1111 1001`

   Finally, we surround the stuffed message with the flag bytes, to get the final framing:

   `<FLAG> 0110 0111 1110 0000 1110 0000 1110 0000 0111 1110 1111 1001 ...`

   (b) Starting and ending flag bits with bit stuffing, `<FLAG> = 0111 1110`

   **Solution:** Recall that in bit stuffing, we insert a zero-bit after every sequence of 5 one-bits. The above message would thus be stuffed as:

   `0110 0111 11010 0000 0111 11010 1111 10001`

   This flag byte is then prepended and appended to the stuffed string.

   `<FLAG> 0111 1110 0110 0111 11010 0000 0111 11010 1111 10001 0111 1110`

2. A 12-bit hamming code whose hexadecimal value is `0xE4F` arrives at a receiver. What was the original value in hexadecimal? Assume that no more than 1 bit is in error.
**Solution:** When checking the Hamming code from left → right, the 12-bit hamming code in binary notation reads

<p>| | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>b_1</td>
<td>b_2</td>
<td>b_3</td>
<td>b_4</td>
<td>b_5</td>
<td>b_6</td>
<td>b_7</td>
<td>b_8</td>
<td>b_9</td>
<td>b_{10}</td>
<td>b_{11}</td>
<td>b_{12}</td>
</tr>
</tbody>
</table>

The bits are split into parity bits and data bits as follows

<p>| | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>p_1</td>
<td>p_2</td>
<td>p_3</td>
<td>p_4</td>
<td>d_1</td>
<td>d_2</td>
<td>d_3</td>
<td>d_4</td>
<td>d_5</td>
<td>d_6</td>
<td>d_7</td>
<td>d_8</td>
<td></td>
</tr>
</tbody>
</table>

We should have \( p_1 = b_3 \oplus b_5 \oplus b_7 \oplus b_9 \oplus b_{11} = 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 = 1 \), which is indeed the case.

Moreover, we should have \( p_2 = b_3 \oplus b_6 \oplus b_7 \oplus b_{10} \oplus b_{11} = 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 = 0 \). However, we see that \( p_2 = 1 \). Before we guess the error, let’s go on first.

For \( p_3 \), we should have \( p_3 = b_5 \oplus b_6 \oplus b_7 \oplus b_{12} = 0 \oplus 1 \oplus 0 \oplus 1 = 0 \), which is indeed the case.

We conclude that the value of the parity bit \( p_2 \), hence bit \( b_2 \) has accidentally been flipped. After correction, we obtain

<p>| | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>b_1</td>
<td>b_2</td>
<td>b_3</td>
<td>b_4</td>
<td>b_5</td>
<td>b_6</td>
<td>b_7</td>
<td>b_8</td>
<td>b_9</td>
<td>b_{10}</td>
<td>b_{11}</td>
<td>b_{12}</td>
</tr>
</tbody>
</table>

which corresponds to subtracting \( 0x400 \) from \( 0xE \) and yields thus \( 0xA4F \).

By contrast, checking the code from right → left, via the method demonstrated in the lecture proceeds as follows. Recall that a Hamming code with 4 check bits can detect and correct 1 bit of error in a message of length \( n = 2^4 - 1 = 11 \), for a total code length of 15. By contrast, a code with 3 check bits would only detect and correct a bit of error in a 4 bit message. Thus we infer that the Hamming code contains 4 check bits.

Padding the 12-bit Hamming code to 15 bits in binary notation reads

<p>| | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>b_{15}</td>
<td>b_{14}</td>
<td>b_{13}</td>
<td>b_{12}</td>
<td>b_{11}</td>
<td>b_{10}</td>
<td>b_9</td>
<td>b_8</td>
<td>b_7</td>
<td>b_6</td>
<td>b_5</td>
<td>b_4</td>
<td>b_3</td>
</tr>
</tbody>
</table>

The bits are split into parity bits and data bits as follows

<p>| | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>d_{11}</td>
<td>d_{10}</td>
<td>d_9</td>
<td>d_8</td>
<td>d_7</td>
<td>d_6</td>
<td>d_5</td>
<td>d_4</td>
<td>d_3</td>
<td>d_2</td>
<td>d_1</td>
<td>p_4</td>
<td>p_3</td>
</tr>
</tbody>
</table>
To determine the position (if any) of an invalid bit, we compute its binary digits.

\[ q_1 = p_1 \oplus b_3 \oplus b_5 \oplus b_7 \oplus b_9 \oplus b_{11} \oplus b_{13} \oplus b_{15} = 1 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 0 = 0 \]

\[ q_2 = p_2 \oplus b_3 \oplus b_6 \oplus b_7 \oplus b_{10} \oplus b_{11} \oplus b_{14} \oplus b_{15} = 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 \oplus 0 = 1 \]

\[ q_3 = p_3 \oplus b_5 \oplus b_6 \oplus b_7 \oplus b_{12} \oplus b_{13} \oplus b_{14} \oplus b_{15} = 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 1 \]

\[ q_4 = p_4 \oplus b_9 \oplus b_{10} \oplus b_{11} \oplus b_{12} \oplus b_{13} \oplus b_{14} \oplus b_{15} = 0 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 1 \]

Taking the bits \( q_i \) and arranging them as a binary number, we get 1110 = 14 indicating an error in \( b_{14} \). After correction, we obtain

\[
\begin{array}{cccccccccccc}
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
\hline
b_{15} & b_{14} & b_{13} & b_{12} & b_{11} & b_{10} & b_9 & b_8 & b_7 & b_6 & b_5 & b_4 \n\end{array}
\]

which is a value of 0x2A4F.

3. A 32 bit message is transmitted using a Hamming code. How many check bits are needed to properly detect and correct 1-bit errors in this 32 bit message?

**Solution:** For data of length \( n \), the length of the Hamming code is given by \( n + k = 2^k - 1 \) where \( k \) is the number of check bits to detect and correct an error in one bit (minimum Hamming distance of 3). Thus 5 check bits can support correcting \( 2^5 - 5 - 1 = 26 \) message bits, and 6 check bits can support correcting one error in \( 2^6 - 6 - 1 = 57 \) bits. Thus to correct an error in 32 bits we would need 6 check bits.

4. You wish to transmit the frame containing 10010110010 with a CRC attached. Your generator polynomial is \( x^3 + 1 \). What bit string would be transmitted?

**Solution:** The generator polynomial converted to binary is

\[ 1 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0 = 1001 \]

The corresponding CRC code is of length 3 for a polynomial of degree \( k = 3 \). This code takes the value of the remainder of 10010110010 left shifted by 3 bits, and divided by the generator polynomial. Thus we have

\[
\frac{10010110010000}{1001} = 100
\]

And a transmitted bit string of 10010110010100.
2 BGP

1. In this problem, we study the convergence of the BGP routing protocol. We will consider BGP routers that implement two policies: the route selection policy and the export policy. The route selection policy is used to select the preferred route to the destination, based on the routes learned from the neighboring routers. The export policy is used to select which routes would be announced to each neighboring router.

![AS topology with five ASes.](image)

Figure 1: AS topology with five ASes.

For this problem and the ASes in Figure 1, we make the following assumptions.

- All ASes in Figure 1 have the same route selection and export policies, as follows:
  - **Route Selection Policy:** The boxes with the dotted lines in Figure 1 show the preferred routes of each AS towards the destination \( D \). If there are two routes in the box, ASes prefer the upper route over the lower route. For example, AS 3 prefers route \( \{321D\} \) over route \( \{34D\} \) to reach AS \( D \). Moreover, if no preferred route is available, ASes do not select any route to the destination, and the corresponding routing table becomes empty. For example, if no preferred routes for AS 2 is available, then AS 2 keeps its routing table empty rather than using route \( \{2D\} \).
  - **Export Policy:** Announce the route that has been selected by the route selection policy to all neighboring ASes. For example, AS 1 announces the route \( \{1D\} \) to AS 2.

- Each AS has one BGP router. When we refer to an AS, we refer to its BGP router.
- All BGP announcements are received by ASes in a synchronized manner, and upon reception, ASes immediately run their route selection policies. You do not have to consider race conditions among the events of sending, receiving, and computing routes.

(a) Initially, assume that the BGP routing tables at ASes 1, 2, 3, and 4 are empty. Then, AS \( D \) announces itself to its neighbors. What routes do ASes 1, 2, 3, 4 use to reach AS \( D \)? (2p)

| Solution: | \{1D\}, \{None\}, \{None\}, \{4D\} |

(b) Then all ASes (i.e., ASes 1, 2, 3, 4, and \( D \)) simultaneously make BGP announcements based on the Export Policy. What routes do ASes 1, 2, 3, and 4 use to reach AS \( D \)? (3p)
(c) Again, all ASes simultaneously make BGP announcements based on the Export Policy. What routes do ASes 1, 2, 3, and 4 use to reach AS D? (3p)

**Solution:** \{1D\}, \{21D\}, \{34D\}, \{4D\}

(d) Again, all ASes simultaneously make BGP announcements based on the Export Policy. What routes do ASes 1, 2, 3, and 4 use to reach AS D? (1p)

**Solution:** \{1D\}, \{21D\}, \{34D\}, \{4D\}

(e) Again, all ASes simultaneously make BGP announcements based on the Export Policy. What routes do ASes 1, 2, 3, and 4 use to reach AS D? (1p)

**Solution:** \{1D\}, \{234D\}, \{321D\}, \{4D\}

(f) Do you see any pattern? Comment about BGP convergence. (1p)

**Solution:** Routes for 2 and 3 keep oscillating. It suggests that BGP does not converge.