Computer Networks — Assingment #2

September 25, 2025

Assignment #2

Total: **13 points**

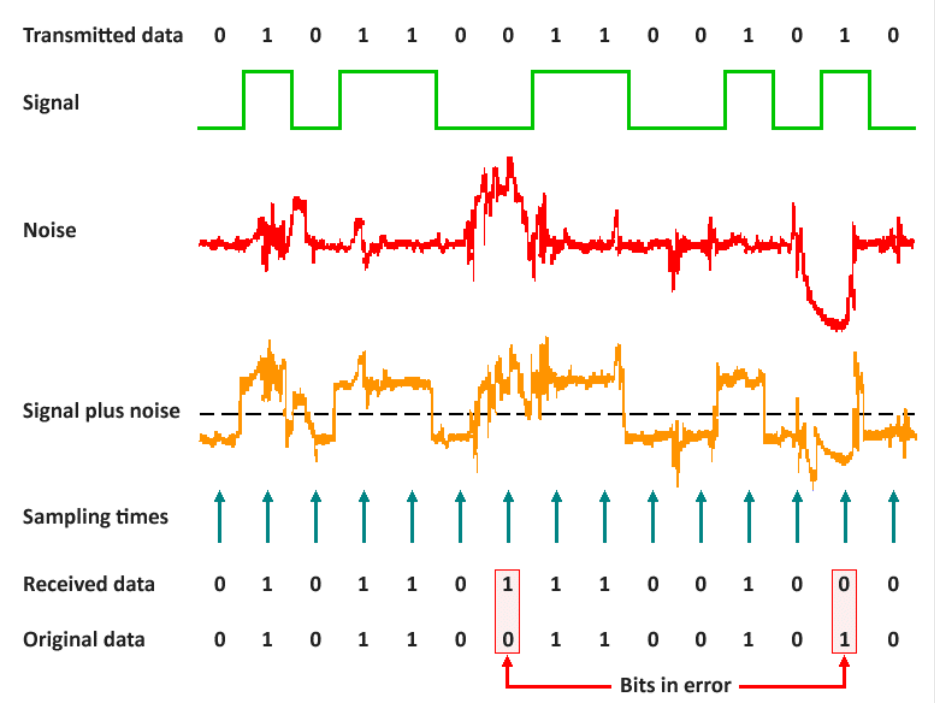
*This assignment must be completed **individually**.*

I.e. you should work alone without any help or collaboration from others.

Please provide detailed answers to the following questions.

Submit a PDF document containing detailed answers and explanation to the questions in the assignment. You can either create a digital document for answering the questions OR write them down in a paper and take photos or scan them – please be sure that the quality of the document is readable.

Learning Outcomes: Encoding, Framing, Error Detection, Hamming codes, Shannon capacity, Ethernet networks – forwarding and routing tables



Problem 1 [1 pt.]

Show (ie. graphically) the NRZ, Machester and NRZI encodings for the following bit pattern:

```
1 0 0 1 1 1 1 1 0 0 0 1 0 0 0 1
```

Asumme that the NRZI signal starts out low.

Problem 2 [1 pt.]

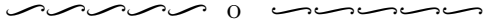
Show the 4B/5B encoding and resulting NRZI signal for the following bit sequence:

```
1101 1110 1010 1101 1011 1110 1110 1111
```

Problem 3 [1 pt.]

The *Data Link Layer* protocol HDLC (High-Level Data Link Control) uses *Bit Stuffing*. If the sender discovers 5 consecutive 1 bits in the bitstream from the Network Layer, it stuffs a single 0 bit into the outgoing bit stream. If the receiver discovers 5 consecutive 1 bits, followed by a single 0 bit in the bit stream from the Physical Layer, it removes (destuffs) the 0 bit. Give the encoding for each one of the following bit sequences, when the sender stuffs after 5 consecutive 1 bits a single 0 bit into the bit stream from the Network Layer.

- 01111110 10100111 11111000 11110010 10011111 10111111 11100101
- 00111111 01110001 11110011 11111100 10101010 11001111 11100001
- 11111111 11111111 11111111 11111111 11111111 11111111 11111111

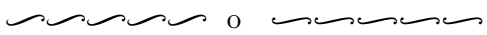


Problem 4 [1pt.]

Suppose that the following sequence of bits arrive over a link:

```
01101011111010100111111011001111110
```

Show the resulting frame after any stuffed bits have been removed. Indicate any errors that might have been introduced into the frame.

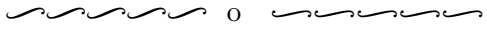


Problem 5 [1 pt.]

Suppose we want to transmit the message `1011001001001011` and protect it from errors using the CRC-8 polynomial $x^8 + x^2 + x^1 + 1$.

a) Use polynomial long division to determine the message that should be transmitted.

b) Suppose the left most bit of the message is inverted due to noise on the transmission link. What is the result of the receiver's CRC calculation? How does the receiver know that an error has occurred?



Problem 6 [2 pt.]

In this problem we will explore how Hamming distance is used as an error detection and error correction mechanism.

In 1950, Richard Hamming an American mathematician working at Bells Lab, published a paper that would serve as the basis for modern coding theory. Hamming postulated that it was possible to not only detect, but correct errors in bit strings by calculating the number of bits disparate between valid codes and the erroneous code. This came to be known as Hamming Distance.

The Hamming distance between two codewords is simply the number of bits that are disparate between two bit strings.

1. According to the previous definition, calculate the Hamming distance, between the following codewords:
   00111100 / 10110101

Typically, Hamming distance is denoted by the function $d(x, y)$ where x and y are codewords. This simple although powerful concept is at the core of Nearest Neighbor error correction.

2. Nearest neighbor error correction involves first defining codewords, typically denoted as C, that are known to both the sender and receiver. Any received codeword not contained in C is obviously the result of an error. Upon identifying an erroneous codeword, nearest neighbor decoding calculates the Hamming distance between it and every codeword contained in C. The codeword with the smallest Hamming distance has a high probability of being correct.

   Let's consider that the sent codeword if "10110101", and $C = \{00001111, 01010101, 00111100, 10000000, 11110111, 11111111\}$; calculate the corresponding Hamming distances for the received codeword and identify which is the most likely transmitted codeword.

   Hamming represented the relationship between minimum Hamming distance and the quality of error correction with two concise equations. A particular code can detect a maximum k errors in a codeword if

$$d(C) \leq k + 1 \tag{1}$$

   and correct a maximum of k errors if

$$d(C) \geq 2k + 1 \tag{2}$$

   For example, a code with $d(C) = 10$ can detect a maximum of nine errors and correct a maximum of four as suggested by the previous equations.

   Redundant bits are extra binary bits that are generated and added to the information-carrying bits of data transfer to ensure that no bits were lost during the data transfer. The number of redundant bits can be calculated using the following formula:

$$2^r \geq m + r + 1 \tag{3}$$

where, r = redundant bit, m = data bit

3. If the number of data bits is 7, compute the number of redundant bits to be appended to the original data.

   A parity bit is a bit appended to a data of binary bits to ensure that the total number of 1's in the data is even or odd. Parity bits are used for error detection.

General Algorithm of Hamming Code

Utilizing Hamming code implies the use of *extra parity bits* to allow the identification of an error.

- Write the bit positions starting from 1 in binary form (1, 10, 11, 100, etc).
- All the bit positions that are a power of 2 are marked as parity bits (1, 2, 4, 8, etc).
- All the other bit positions are marked as data bits.
- Each data bit is included in a unique set of parity bits, as determined its bit position in binary form. See Table I.
  (a) *Parity bit 1* covers all the bits positions whose binary representation includes a 1 in the least significant position (1, 3, 5, 7, 9, 11, etc).
  (b) *Parity bit 2* covers all the bits positions whose binary representation includes a 1 in the second position from the least significant bit (2, 3, 6, 7, 10, 11, etc).
  (c) *Parity bit 4* covers all the bits positions whose binary representation includes a 1 in the third position from the least significant bit (4–7, 12–15, 20–23, etc).
  (d) *Parity bit 8* covers all the bits positions whose binary representation includes a 1 in the fourth position from the least significant bit bits (8–15, 24–31, 40–47, etc).
  (e) *Parity bit n* ...
  (f) In general, each parity bit covers all bits where the bitwise AND of the parity position and the bit position is non-zero.

  Since we check for even parity set a parity bit to 1 if the total number of ones in the positions it checks is odd.

  Set a parity bit to 0 if the total number of ones in the positions it checks is even.

4. Using the previous result from your calculation about the redundant bits, determine the total number of bits and the positions of where these redundant bits will be placed.
5. Employing "placeholders" for the redundant bits, i.e. R1,R2,...; show how the data

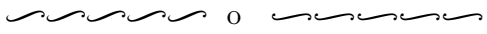
Redundant Parity Bit	Data Bit Position 1	2	3	4	5	6	7	8	9	10	11	...
R1	**1**	0	**1**	0	**1**	0	**1**	0	**1**	0	**1**	...
R2	0	**1**	**1**	0	0	**1**	**1**	0	0	**1**	**1**	...
R4	0	0	0	**1**	**1**	**1**	**1**	0	0	0	0	...
R8	0	0	0	0	0	0	0	**1**	**1**	**1**	**1**	...

TABLE I: Redundant Parity Bits Location –
R1: 1,3,5,7,9,11,... | R2: 2,3,6,7,10,11,... | R3: 4,5,6,7,... | R4: 8,9,10,11,...

and redundant parity bits will look like for "1011001".

6. Now, let's determine each of the parity bits R1, R2,...
7. Write the actual data transferred by combining the original codeword and the corresponding redundant bits.

   Suppose that in the above example the 6th bit is changed from 0 to 1 during data transmission.
8. Recompute the corresponding redundant bits for this changed codeword.
9. Consider the newly updated redundant bits sequence, convert this sequence from binary to decimal, this number represents the bit changed. Do the conversion and see whether it matches your expectations.



Problem 7 [1 pt.]

The *Nyquist theorem* also known as the sampling theorem[1] is a critical component of digital communications.

To convert an analog wave to a digital signal, it must be measured at a regular frequency, which is the *sample rate*. If the sample rate is too low, it will not accurately express the original signal and will be distorted, or show "spurious" effects, when reproduced. If the sample rate is too high, it will needlessly take up extra storage and processing resources. The Nyquist theorem helps to find the perfect sweet spot where all the necessary information is recorded but nothing extra.

[1] The Nyquist theorem is also known as the Nyquist-Shannon theorem or the Whittaker-Nyquist-Shannon sampling theorem.

The Nyquist theorem defines the minimum sample rate for the highest frequency that you want to measure. The Nyquist rate is 2× the given frequency to be measured accurately. The theorem can also be used in reverse. The Nyquist frequency is the highest frequency that equipment of a given sample rate can reliably measure, one-half the given sample rate.

1. Taking this into consideration and the fact that typical phone lines are sampled at a maximum frequency of 8 KHz and that the an average signal-to-noise ratio is around 20 dB. What would be the expected bandwidth for this type of communication channels?
2. Calculate the minimum signal-to-noise ratio corresponding to a transmission at 50 Kbps over a bandwidth of 1MHz. What can you conclude from this?

Problem 8 [1 pt.]

In traditional telephony the usable voice frequency band ranges from 300 to 3400 Hz. Table II shows other windows of frequencies used in other applications in audio bands in modern telephony technologies.

Band	Range (Hz)
Narrowband (traditional)	300-3400
Wideband (eg. VoIP)	50-7000
Super-wideband	50-14000
Fullband	20-20000

TABLE II: Frequency ranges in modern audio communications.

Extend Table II to include *effective* links capacities for these ranges of frequencies for signal-to-noise ratios of 10, 20 and 30 dB.

Problem 9 [1 pt.]

For the network given in Fig. 1, give *i) initial* and *ii) final* ***global*** *distance–vector* tables, like the ones discussed in class, for the following cases:

(a) Each node knows only the distances to its immediate neighbors.

(b) Each node has reported the information it had in the preceding step to its immediate neighbors.

(c) Step *(b)* happens a second time.

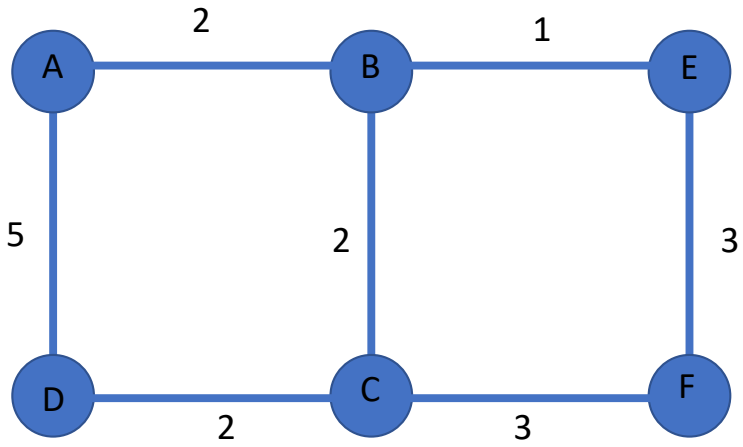


FIG. 1: Network layout for Problem 9.

Problem 10 [1 pt.]

Assuming the we have the following tables shown in Table III for nodes A and F in a network where all links have cost 1. Give a diagram of the smallest network consistent with these tables.

Node	NODE A Cost	Nexthop	Node	NODE F Cost	Nexthop
B	1	B	A	2	C
C	1	C	B	3	C
D	2	B	C	1	C
E	3	C	D	2	C
F	2	C	E	1	E

TABLE III: Forwarding table for Nodes A and F – Problem 10.

Problem 11 [1 pt.]

Given the following routing in Table IV for a router, that can deliver packets directly over interfaces 0 and 1, or it can forward packets to router R2, R3, or R4. Assume the router does the longest prefix match. Describe what the router does with a packet addressed to each of the following destinations:

(a) 128.96.171.92

(b) 128.96.167.151

(c) 128.96.163.151

(d) 128.96.169.192

(e) 128.96.165.121

SubnetNumber	SubnetMask	NextHop
128.96.170.0	255.255.254.0	Interface 0
128.96.168.0	255.255.254.0	Interface 1
128.96.166.0	255.255.254.0	R2
128.96.164.0	255.255.252.0	R3
(default)		R4

TABLE IV: Routing table for Problem 11.

Problem 12 [1 pt.]

Build the shortest path routing table (as shown in slide #29 from the `lecture_week05-IP_Forwarding_vs_Routing` lecture) in the forward search algorithm as it builds the routing database for Node A in the network shown in Fig. 2.

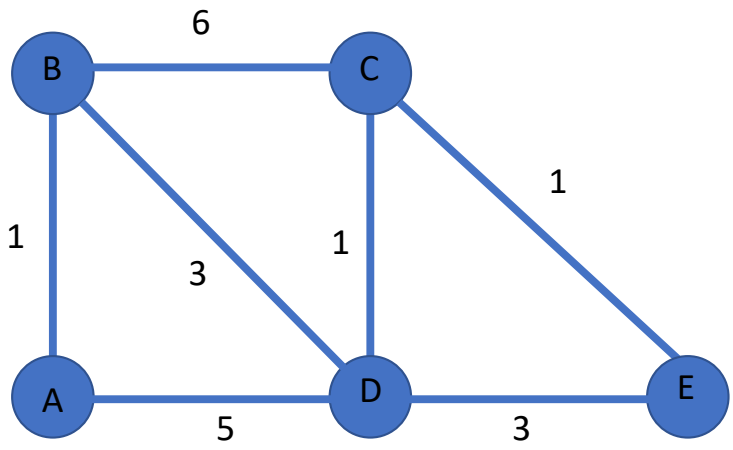


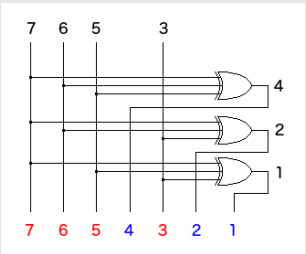
FIG. 2: Network layout for Problem 12.

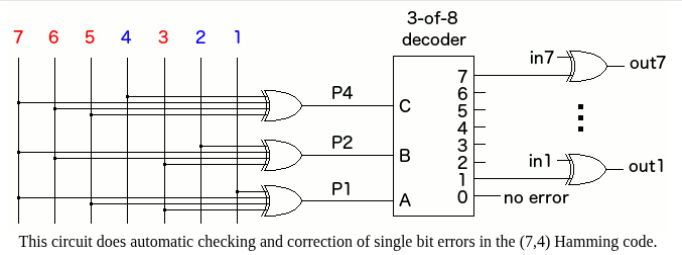
Brain Teasers & Further Readings

Brain Teasers & Further Readings

- Let's consider the *4B5B encoding*.
  - Compute the *Hamming Distance* for the 4B and 5B valid code-words.
  - What do you notice about this? I.e. why do you think the 4B/5B is useful if one just considers **solely** its HD properties?
  - Perhaps, we need to look into a few more details to understand whether the 4B/5B encoding is helpful.
    In many cases, the 4B5B encoding is used in combination with the NRZI encoding (largely used in faster Ethernet networks). Pick two of the closest (i.e. with the shortest HD) code-words in 4B5B and compute the new HD after the code-word are encoded with NRZI.
    What happened?

- Write a program to compute the HD between two binary code-words.
  Can you think of a really "practical" way to do this? Might the figure below be of help... what is it?





This circuit does automatic checking and correction of single bit errors in the (7,4) Hamming code.

- About Hamming Codes:
  "But what are Hamming codes? The origin of error correction"", 3Blue1Brown – https://www.youtube.com/watch?v=X8jsijhllIA
  "Hamming codes part 2: The one-line implementation", 3Blue1Brown – https://www.youtube.com/watch?v=b3NxrZOu_CE

- About redundancy & error detection/error correction:
  "How do QR codes work?", Veritasium – https://youtu.be/w5ebcowAJD8

- On Ethernet & beyond...
  https://www.truecable.com/blogs/cable-academy/powerline-ethernet-adapters-vs-ethernet
  https://planetechusa.com/3-ethernet-cabling-alternatives/
  https://www.lifewire.com/seamless-streaming-with-a-powerline-adapter-8735887