Problem 1: Let f be the frame size, RTT is the round trip time, and T be the timeout. Note that if we assume RTT estimation is perfect, $RTT = T$.

a. $throughput = \frac{f}{RTT+p*T} = \frac{f}{(1+p)RTT}$,

Basically, throughput equals the quotient of the frame size and the (average) total time to transmit the packet. The total time to transmit a packet equals the time of one round to send the packet successfully plus the time caused by the error. In the case error occurs, it wastes a time window T. Since error happens with probability p and the retransmission always succeeds, on average the error time is $p * t$. Thus, the (average) total time $= RTT + p * T = (1 + p)RTT$. So we get a formula for throughput as above.

b. $throughput = \frac{f}{RTT+p*T/(1-p)} = \frac{f(1-p)}{RTT}$

Now, data frames fail with independent (across successive transmission attempts). Let X be a random variable that counts the number of times we have to send a packet until succeeding. Then it means that $E[X] - 1$ first tries get errors and the last try is successful. So the (average) total time to transmit a packet is $(E[X] - 1) * T + RTT$.

Let X_i be a random variable indicator that the packet is successfully sent after exactly i times. It is easy to see that $E[X_i] = i*p^{i-1}(1-p)$. Note that $X = \sum_{i=0}^{\infty} X_i$. By linearity of expectation,

$$E[X] = \sum_{i=0}^{\infty} E[X_i] = \sum_{i=0}^{\infty} i * p^{i-1}(1-p) = \frac{1}{1-p}$$

Hence, the (average) total time to transmit a packet is $RTT + p * T/(1-p) = RTT(1 + \frac{p}{1-p}) = \frac{RTT}{(1-p)}$, and we derive the formula as above.

Problem 2: First we need to find an orthogonal basis of $\mathbb{R}^4$. Here are four orthogonal vectors (ingredients) that will serve as the code vectors of the four users.

$$v_1 = (1, 1, 1, 0), v_2 = (1, -1, 0, 1), v_3 = (1, 0, -1, -1), \text{and } v_4 = (0, 1, -1, 1)$$

To send 1 to the first, 0 to the second, 1 to the third, and 0 to the fourth. The sender just needs to send $(1, 1, 1, -3)$ as the soup, which is derived by computing:

$$\begin{aligned} v &= 1 \cdot v_1 + (-1) \cdot v_2 + 1 \cdot v_3 + (-1) \cdot v_4 \\ &= 1 \cdot (1, 1, 1, 0) + (-1) \cdot (1, -1, 0, 1) + 1 \cdot (1, 0, -1, -1) + (-1) \cdot (0, 1, -1, 1) \\ &= (1, 1, 1, -3) \end{aligned}$$

To decode, each user need to compute the dot product of soup vector with his ingredient vector.

1. Since v_1 is orthogonal to v_2, v_3, v_4, we have $v \circ v_1 = (1 \cdot v_1 + (-1) \cdot v_2 + 1 \cdot v_3 + (-1) \cdot v_4) \circ v_1 = 1 \cdot (v_1 \circ v_1) = (1, 1, 1, 0) \circ (1, 1, 1, 0) = 1 + 1 + 1 + 0 = 3 > 0$, so the first user gets bit 1.

2. Similarly, since v_2 is orthogonal to v_1, v_3, v_4, we have $v \circ v_2 = (1 \cdot v_1 + (-1) \cdot v_2 + 1 \cdot v_3 + (-1) \cdot v_4) \circ v_2 = (-1) \cdot (v_2 \circ v_2) = (1, -1, 0, 1) \circ (1, -1, 0, 1) = -(1+1+0+1) = -3 < 0$, the second user gets bit 0.

3. $(1, 1, 1, -3) \circ (1, 0, -1, -1) = 1 + 0 - 1 + 3 = 3 > 0$, the third user gets bit 1.

4. $(1, 1, 1, -3) \circ (0, 1, -1, 1) = 0 + 1 - 1 - 0 = -3 < 0$, the fourth user gets bit 0.

If the code vectors v_1, v_2, v_3, v_4 are not orthogonal, then when decoding we could not cancel off the dot product of a code vector with each other code vector. So the decoding doesn't work out exactly as expected, and we cannot hide bits in the weights. This is analogous to an issue we encountered when sending bits using amplitude modulation (AM) of electromagnetic waves modeled as complex sinusoids. Sending mutiple bits using traditional FDM is complicated because of inter-channel interference (ICI). The distortion of original weight values introduced by spreading and overlap may result in bit flips or outright failure, so it can make decoding bits difficult. Two approaches used to address the problem:

1. Put neighboring carrier frequencies far apart

2. Use orthogonal FDM in which sinusoids are mutually orthogonal

Problem 3: The carrier frequencies used by AM radio stations are in range from 152 kHz to 1710 kHz (according to wikipedia list of radio stations by frequency) while human hearing range is from 20 Hz to 20 kHz. It means that AM radio has narrow bandwidth, and can only carry from quite low sound to moderately high sound but not extremely loud sound for audio like strong bass and deep. Thus, it helps to explain the perception that AM radio is adequate for talk radio but less suited for music broadcast, FM with bigger range of frequencies is more suitable. The differences between AM radio and FDMA are that AM radio is broadcast to every user using the same channel with small range (or a fix value) of frequency while FDMA is the process of dividing a channel (bandwidth) with medium range of frequency into multiple individual bands, each for use by a single user.

Bonus Problem: My system environment: Windows 10, using Cygwin to compile the client code. I ran the client app on my desktop at home, Crestview apartment, West Lafayette. Here are ping results obtained from my app: