For our project, we made our client a “thin” client, performing only the I/O necessary for the user to play the game, in addition to the Network Time Protocol (NTP). The client is responsible for sending its most recent user inputs to the server which determines in which direction that client’s snake should move.

Meanwhile, the server handles all the game logic, and initiates all communications after the initial handshake. It is the server’s responsibility to send the new positions of each of the snakes’ heads (the very first segment) and the previous tails (the very last segment of the previous frame) to be drawn to be drawn on the HTML5 canvas. With the snakes’ heads and previous tails, we can create the illusion that the snake is moving without having to send each position occupied by a segment of a snake. Additionally, the server is also responsible for sending each player's’ score.

Our group achieved artificial latency utilizing a queue that prioritizes messages to be popped of by the lowest timestamp. The game uses one queue for incoming messages and another queue for outgoing messages. Timestamps are computed by taking the time that an incoming or outgoing messages is sent or received on the server and adding a random uniform delay between 0 and 250 milliseconds. The timestamps are then paired with their corresponding message.

Upon implementing artificial latency, we noticed an increase in time between user input and the new resulting game state (which is to be expected). We also discovered a bug in our Milestone 3 implementation where there was a chance that the clients were receiving out of order packets, so there was a chance that the snake was leaving its “tail” behind.

To mitigate the added latency we used, lockstep synchronization to mitigate problems associated with the variation in latency between each client and the server. The most notable problem would be that clients might be faster than one another, thus one client might become a few frames ahead of the other client. Lockstep synchronization guarantees that both clients inputs have been handled by the server before computing the next game state.

As for our team members, Jonathan, Brian, and Yu helped implement both client rendering and prototyped the game logic during Milestone 1. Jonathan helped designed the Lockstep synchronization architecture for Milestone 2 with some feedback of Brain and Tommy in Milestone 2. Tommy wrote the server code to handle Lockstep synchronization and reimplemented the game logic on the server; both Tommy and Brian debugged server code for Milestone 2. Jonathan also helped design the NTP architecture and made the decision to piggyback NTP to the existing packets being sent with the help of Brian and Tommy in Milestone 3. Tommy was the one who implemented NTP for Milestone 3. For Milestone 4 and 5, Tommy polished the game code.