Tic-Tac-Toe Protocol

**Layer 4 Protocol:** TCP

**Port:** 10101

**Client Game States:**

- Player's Turn
- Computer Turn
- Game Start
- Game Over

**Server Game States:**

- Server's Turn
- Waiting for Client to move
- Game Over

**Client <—> Server Messaging (Type Length Value)** (The Client and Server will both send the entire game information (Python list of Ints) back and forth to each other since the data is small enough to fit into a single TCP Packet consistently)**:**

➔ Type:

- ◆ Size of 1 int
- ◆ Type 0 is the default. The Tic-Tac-Toe protocol. It is the only type for now.

➔ Length:

- ◆ This will indicate how many ***integers*** are present in the data
- ◆ Default: 13 (1 integer tells the Type, 1 integer tells the Length, and 11 integers are left for the Value)

➔ Value:

- Unsigned Integer Order, Values and Meanings (This ∫ will determine what the game state is for the **Default Type (Type 0)**):
    - **Int 1: Turn Count** (How many communications have occurred since the connection. The Client/Server should make sure that the Turn Count it receives is always only 1 more than its lastly sent packet. Turn Count includes all forms of communications even when a "turn" isn't necessarily being taken like when a game restarts into the Game Start Generic Gamestate [See below])
        - Value 0: Connection Start
        - Value {unsigned int}: Turns Occured
    - **Int 2: Generic Gamestate**
        - Value 0: Game Start (All Tic-Tac-Toe Space Values should be 0 [See below])
        - Value 1: Player's Turn Taken, 'O' placed
        - Value 2: Server's Turn Taken, 'X' placed
    - **Int 3 - Int 11: Tic-Tac-Toe Space Values** (Int 3 starts the top left space and subsequent Integers specify the spaces to the right, returning to the next row when reaching the end column)
        - Value 0: Empty Space
        - Value 1: 'O' Space
        - Value 2: 'X' Space

**Example:**

➔ If I'm the client and the game just started, then it's the human player's turn, and the human has to put an "O" down. The player puts the "O" in the middle space. The data sent to the server would be:

- ◆ [0, 13, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0]
- ◆ [Type, Length, Count, Turn, Space_1, Space_2, …, Space_9]

➔ I'm the server and just received this game data from the client:

- [0, 13, 5, 1, 2, 0, 1, 1, 1, 0, 2, 0, 0]
- [Type, Length, Count, Turn, Space_1, Space_2, …, Space_9]

◆ It's the server's turn now, and the server puts down a random "X" in an open space. This is the data it will send back:

- [0, 13, 6, 2, 2, 0, 1, 1, 1, 2, 2, 0, 0]
- [Type, Length, Count, Turn, Space_1, Space_2, …, Space_9]

**Communication Failure Protocol:**

- The Client and Server should continue to resend their current data packet until they receive a packet with an incremented Turn Count.