Eric Galway
101252535

GAME3110 Assignment 1

Part 3

1. The out parameter is a keyword in C# that allows users to pass parameters to functions by reference instead of by value. Passing by reference means that the variable itself is passed into the function instead of passing a copy of its content which is done when passing by value.
2. The reason an integer would need to be passed as an out parameter is because primitive data types are normally passed by value, not by reference. Passing by value means you are inputting a copy of whatever value the variable is holding into the function. Any change to this value results in no change to the original variable. This is a problem if you want to pass in a variable to a function, alter it, and expect that change to occur in the parameter outside of the function. Passing primitive data types, like an integer, as an out-parameter passes it by reference. This means the contents of the variable will change as the value of the variable is changed inside the function.

   The reason a class instance does not need to be passed as an out parameter is because class instances are already a reference type. Therefore, it would be redundant to specify that you would like a reference type to be passed by reference.

Part 4

1. The QosType enumeration describes the quality of service from a channel. The enumerations are the following:
   - Unreliable: no guarantee of delivery or ordering of messages.

- UnreliableFragmented: no guarantee of delivery or ordering of messages but allows fragmented messages.
- UnreliableSequenced: no guarantee of delivery and all unordered messages will be dropped.
- Reliable: each message is guaranteed but no guarantee of order of messages.
- ReliableFragmented: guarantee of delivery or ordering but allows fragmented messages.
- ReliableSequenced: guarantee of order of messages and of delivery.
- StateUpdate: an unreliable message where only the last message in the buffer is sent. Also, only the most recent message in the receive buffer will be delivered.
- ReliableStateUpdate: A reliable message where only the last message in the buffer is sent. Also, only the most recent message in the receive buffer will be delivered.
- AllCostDelivery: a reliable message that will be recent until acknowledged.
- UnreliableFragmentedSequenced: no guarantee of ordering and delivery of messages but allows fragmented messages.
- ReliableFragmentedSequenced: guarantee of ordering and delivery of messages but allows fragmented messages.

2. The NetworkTransport Init function initializes the Network Transport with default values. Since this creates the NetworkTransport this needs to be called before any other operations are called on NetworkTransport since it will not exist if Init is not called first.

3. NetworkTransport Disconnect sends a signal to the connected peer to disconnect the connection.
4. Encoding.Unicode.GetString returns a string of the decoded results from the specified sequence of bytes from the passed in byte array. Encoding.Unicode.GetBytes returns a byte array which are the results of the encoded string that was passed in as a parameter.
5. Sizeof returns the number of bytes that a variable occupies.
6. NetworkEventType Enumeration is what is returned when calling NetworkTransport Recieve and NetworkTransport RecieveFromHost functions. The enumerations are the following:
    - DataEvent: Indicates data was received.
    - ConnectEvent: Indicates connection event was received.
    - DisconnectEvent: Indicates disconnection event was received.
    - Nothing: Indicates no new event.
    - BroadcastEvent: Indicates broadcast discovery event was received.
7. Network Transport Receive returns a NetworkEventType. This function essentially delivers network events to users. The data is returned in the buffer argument. The parameters of Network.Transport.Receive do the following:
    - hostid: is of udp socket where event happened.
    - connectionId: device connected to.
    - channelId: channel id for data event.
    - buffer: data received over the network.

- bufferSize: size of buffer.
- receivedSize: received length.
- error: possible returned error.

8. Network Transport Send essentially sends data to a peer. The parameters of Network.Transport Send do the following:
    - hostId: Id of udp socket used for sending.
    - connectionId: Id of connection.
    - channelId: Id for channel.
    - buffer: Binary buffer containing data for sending.
    - size: Buffer Size.
    - error: possible error.
9. The ConnectionConfig class sets out parameters that are used in the connection between two peers. Some of parameters that are set out include timeouts, sizes and channel configurations. The AddChannel function adds a new channel to the configuration while returning the id of the channel. This returned id can be used to send messages via this channel. Channels are essentially boundaries that divide traffic between peers.
10. The HostTopology class defines how many default config connections will be supported by the host and what the special connection will be. For example, this class would define a default connection for ever member of a p2p players connection to the chat server as well as creating one special connection. The AddHost function creates a Host or open

socket, with a topology that was passed as a parameter. AddHost then returns an integer containing the host ID that was just created.

11. I had some idea as to what NetworkTransport.Connect did on a general level but was not sure of all the details and parameters so I did some further research. This function tries to connect to a peer and will return either a non-zero value when an error occurs or a zero value on success. It can still fail even when returning a zero id so polling NetworkTransport Receive or ReceiveFromHost should be used to listen for events. The parameters are as follows:

    - hostId: Host ID associated with the connection.
    - Address: IPv4 Address of the other peer.
    - Port: port of the peer.
    - exceptionConnectionId: Set to 0 in the case of a default connection.
    - Error: an error.