# Assignment 1

## Introduction

Most assignments for this course are practical in nature. This first assignment however is an exception to that rule, since we want to make sure you have a good grasp on the basics and have plenty of time to get used to the quirks of TCP/IP communication using the example provided during the bootcamp, before we dive into the more complicated stuff.

## Sufficient

Review the bootcamp material and answer the questions below.

1. Why are networks modelled using a layer stack?

|  |
| --- |
| Layers split a huge, complicated job (like sending data over the internet) into smaller, simpler tasks. Each layer does one thing well, and they all work together.  It allows for better understanding and management of what happens to the data at what stage of sending or receiving understanding what part is in charge of what task.  Each layer you go down you go deeper into the internet and into primitive forms of data and back up to the device. |

1. What is an IPAddress and what is the valid structure of an IPAddress?

|  |
| --- |
| The location to reach a certain device(local) or network(global)  0.0.0.0 – 255.255.255.255 (IPv4)  There are also other methods like IPv6 but those are used more rarely, the main IP is IPv4 |

1. What is a loopback address and what do you use it for?

|  |
| --- |
| Used to refer to your local device network 127.0.0.1 |

1. What is a URL and what is its use?

|  |
| --- |
| Used to link to websites in an understandable and readable way, also a bit more secure  A URL can directly link to a global IP adress |

1. What is the difference between the IP protocol vs the TCP/UDP protocols?

|  |
| --- |
| Ip protocols adress the IP of the host, TCP/UDP adress the aplication on the host |

1. List 3 differences between the TCP & UDP protocol.

|  |
| --- |
| TCP is always in order, UDP might not be  TCP can’t lose packets, UDP might  TCP can’t receive packets duplicated, UDP might  TCP requires an active connection, UDP doesn’t  TCP waits for the client, UDP does not |

1. What is a port and what port range should you use as an application programmer?

|  |
| --- |
| The port is the identification of what aplication the data goes to  The port range is 0 – 65535  The dynamic range is 49152-65535 you should use this |

1. What is the difference between a dedicated and a non dedicated server?

|  |
| --- |
| It depends on what way you look at it:   * A Dedicated server is Just the server nothing else running from it. * A Non-Dedicated served is hosted on a device were more (maybe all) servers are hosted, you get a part of the device.   OR (the way it is mostly looked at in gaming)   * Dedicated server is just the server. * Non-Dedicated server is both the server and a player |

## Good

Review the bootcamp material and answer the questions below.

1. What are the 5 layers of the discussed networking stack and what is the purpose of each?

|  |
| --- |
| Physical link layer – this is the actual connection between the devices (WiFi, Ethernet, Fiber, 4G)  InternetProtocol Layer – this is the layer that sends the data over the physical layer and locates the host the data is meant for  Transport Layer - this locates the aplication the data is meant for  Sockets – the interface from the Internet data to the Aplication and the other way around  Aplications – handles the data into the aplication and does what it needs |

1. You are playing a network game where one player can shoot another player.  
   a) List the network messages (in regular English/JSON/XML) between client & server for both a client authoritative setup and a server authoritative setup for such an event from the moment a player presses the fire button.

For example if I wanted to describe a login message I could write something like  
<Login name="..." pass="..." /> or Login = { user:"...", pass:"..." } to make it clear what kind of message I want to send and what data it contains.

|  |  |
| --- | --- |
| **Client authoritative** | **Server authoritative** |
| Raycast the bullet and detect “DetectedHit”  And calculate the damage (maybe based on armour)  <DamageTaker=”DetectedHit” damage=”damage”/> to server  <damageTaken=”damage”/> to other client  Other client removes the health from its healthpool  <newHealth=”health”/> to server  <newHealth=”newHealth” player=”damagedPlayer”/> to all other clients to update the health bar | <shoot=”true”/>  Now the server calculates the direction of the shot, the damage it needs to deal, and who to deal it to. In this context the server is the one that keeps track of all data like positions.  Then it deals the damage to that player  <newHealth=”damagedPlayer.health” player=”damagedPlayer”/> to all clients  Updates the health for the damaged player for everyone |

b) Explain whether the messages from the previous question are IP, Transport or Application protocol messages.

|  |
| --- |
| These are Aplication protocol messages, they decide what data is sent |

1. Bootcamp 1 discussed two common setups for building a network game (Peer 2 peer & Client/Server). List/research some advantages/disadvantages of both setups.

|  |
| --- |
| P2P (Peer 2 Peer)  It does not require a separate server aplication, meaning no server costs or server setup for the users.  Becomes more unstable with larger numbers of connections  Less secure  Faster with low number of connections (there is no in-between step from the server on eg. Position changes)  Client/Server  More secure and sturdy connection  Central server that manages all data and connections |

1. List which protocol type is more appropriate for each of the message types below:
   1. Login message UDP / TCP
   2. Fast paced position update messages UDP / TCP
   3. Video stream data UDP / TCP
   4. Player hit messages UDP / TCP
2. For each situation below indicate which protocol has been used (more than 1 correct answer possible):
   1. The client sends 1 message and the server receives it: UDP / TCP
   2. The client sends 1 message and the server receives it twice: UDP / TCP
   3. The client sends 2 messages and the server receives them in order: UDP / TCP
   4. The client sends 2 messages and the server receives them out of order: UDP / TCP

## Very good

Examine the ***starting\_code*** example and answer the questions below:

1. What do we mean with blocking operations?

|  |
| --- |
| An operation that waits for X and then continues. The entire Thread stops here |

1. Name 2 different blocking network operations used in the given example.

|  |
| --- |
| listener.AcceptTcpClient ()  stream.Read(byte[] buffer, int offset, int count) |

1. List some exceptions that might occur while trying to communicate over the network.  
   (Hint: check the code hinting or look up some network calls on MSDN)

|  |
| --- |
| **Generic exceptions you could get**   * HttpRequestException * SocketException * IOException * TimeoutException   **Some exceptions I got playing around**   * You start the client but there is no server that responds: *System.Net.Sockets.SocketException: 'No connection could be made because the target machine actively refused it. [::ffff:127.0.0.1]:55555'* * Trying to connect over the same port: *System.Net.Sockets.SocketException: 'Only one usage of each socket address (protocol/network address/port) is normally permitted.'* * It doesn’t give an exeption but sending an empty message freezes the aplication |

1. Imagine you've been given a client and server without any error handling code and you only have   
   time to fix one of them. Which one would you fix and why?

|  |
| --- |
| Server, the server is in most cases the more important one and more prone to crashing or stuff going wrong. A client can also be restarted a little easier. |

1. Why does the client stop working in the given example if you send an empty string?

|  |
| --- |
| Because there is no bytes to send, so there is nothing to write to the stream. Then the read will block because the server does not send anything back. |

## Excellent

1. Start the server and **two** clients. Why is the server only responding to the first client?

|  |
| --- |
| It has a blocking operation that waits for client one |

1. Start the server and **two** clients. What is the simplest way without making any code changes to have the server respond to the last client?

|  |
| --- |
| Send on client one so it continues after the block |

1. How can you prevent a client from connecting at all if there is already another client waiting to be served? (Hint: research the TcpListener.Start call)

|  |
| --- |
| Stop the listener by using \*.stop() |

1. What happens to clients that are trying to connect, but have not yet been accepted by the TcpListener.AcceptTcpClient call?

|  |
| --- |
| You can send a message but the server will not accept it. |

1. Replace line 17 of the client with:   
   TcpClient client = new TcpClient(new IPEndPoint(IPAddress.Any, 55556));  
   Start the server and two clients again and note what happens.  
   Undo line 17 and repeat, note the port the clients are connecting two.

Select the correct statement below and motivate your answer:

a) It is not possible to bind more than one TcpClient to the same port.

b) Multiple TcpClients can be bound to the same port

c) Multiple TcpClients can be bound to the same port on the server but not on the client

d) Multiple TcpClients can be bound to the same port on the client but not on the server

|  |
| --- |
| C, the client instantly crashes when on the same device  But on a server it can be on the same port as long as the IP is different. |