# 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 lecture, before we dive into the more complicated stuff.

## Sufficient

Review the lecture material and answer the questions below.

1. Why are networks modelled using a layer stack?

|  |
| --- |
| To reduce complexity, larger problems get divided up into smaller problems and are then easier to solve |

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

|  |
| --- |
| A unique ID for an individual device.  Structure: 4 dotted decimals/192.168.0.1, or a 32 bit number: 0Xc0a90001 |

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

|  |
| --- |
| A internal test IP address.  Used for testing purposes or running a local server  Localhost address 127.0.0.1 |

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

|  |
| --- |
| Uniform Resource Locator, an easier way to show IP addresses. Words are memorized easier than a bunch of numbers. Also prevents failure when an IP address changes.  Translation is done by DNS server |

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

|  |
| --- |
| IP protocol = agreement on how packets are routed between *systems*: IP address identifies host  TCP/UDP = agreement on how packets are routed between *applications*: port identifies application |

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | ---------------------------- | TCP/IP | UDP/IP | | Resending data | Application addressing | Application addressing | | Connection type | Connection oriented | Connectionless (connection beforehand not necessary) | | Reliability of message delivery | Guaranteed delivery(if connected), ordered delivery, no duplication | Lossy (might get lost), out of order. Possible packet duplication | | Speed | Slow, a lot of overhead/latency | Very fast | |

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

|  |
| --- |
| An ID for data streams. Only use dynamic port range (49152 – 65535) |

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

|  |
| --- |
| Dedicated: standalone application doing nothing except handling data traffic between clients  Non dedicated: one of the users of an application can also host a server |

## Good

Review the lecture material and answer the questions below.

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Layer** | **Function** | | Applications | An agreement on what sort of messages a game/application is or needs to process | | Sockets | Software. Simple interface to a complicated thing. Kind of like electricity sockets, usable without knowing every detail | | Transport layer (TCP & UDP) | Communication from app to app. Address gets us to the application in the system | | Internet Protocol Layer | The process of moving packets through a network (like a digital postal service). Communication between client to client. The address gets us to the system | | Physical/link layer | Transmits data across a specific medium in ‘frames’: wifi, 4g, cables etc  Defines things like:   * Ways to identify a host * The frame format * How to convert bits into signals | |

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** |
| Player 1: Clicked fire button, doing 50 million damage  Server: Okay, sending data to player 2  Player 2: I’m **dead** dead  Server: registering player 2 as dead | Player 1: trigger “fire bullet event”  Server: event received, subtracting 25 damage from player 2 health.  Player 2: receiving damage information |

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

|  |
| --- |
| Application protocol, they are messages that the game needs to function. |

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

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **P2P** | **Client/server** | | Every connection is a host, if one falls away the connection for the rest of the clients stays up | One central connection point, vulnerable to downtime | | Not efficient, a lot of data for each client to send and receive | Efficient, data is only sent to server and from the server to its clients | | Difficult to manage | Easier to manage | | Vulnerable to malware | Easier to protect from attacks | |

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 ***001\_basic\_tcp\_echo\_server\_commented*** example and answer the questions below:

1. What do we mean with blocking operations?

|  |
| --- |
| The code will not continue until a condition is satisfied |

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

|  |
| --- |
| AcceptTCPClient – blocks until a client connects fully  stream.Read – blocks until some, perhaps not all bytes have been received |

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)

|  |
| --- |
| WSA\_NOT\_ENOUGH\_MEMORY  WSA\_OPERATION\_ABORTED  WSAETIMEDOUT  WSAEHOSTDOWN  https://learn.microsoft.com/en-us/windows/win32/winsock/windows-sockets-error-codes-2 |

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?

|  |
| --- |
| The server. If that does not work properly then all the clients are affected. While if a client has proper error handling then only that client is fixed and all the others connected to the server still don’t work properly (or at least don’t have error handling) |

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

|  |
| --- |
| Since stream.Read() is a blocking operation and the server doesn’t receive any data (an empty string is 0 bytes), it will keep reading forever (or until it times out). Since it doesn’t read anything, it also can’t send anything and the stream.Read() operation in the client code doesn’t execute. |

## Excellent

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

|  |
| --- |
| 1. TcpClient client = listener.AcceptTcpClient (); 2. IPEndPoint endPoint = client.Client.RemoteEndPoint as IPEndPoint; 3. Console.WriteLine($"Client connected from {endPoint.Address}:{endPoint.Port}, waiting to serve ..."); 4. NetworkStream stream = client.GetStream ();   Because it focuses and only listens to the first client that connects. |

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?

|  |
| --- |
| Close the first client. The server starts listening to the 2nd client |

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)

|  |
| --- |
| Set the “backlog” parameter to 1 (Only 1 connection in queue allowed, the one that’s already waiting) |

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

|  |
| --- |
| They stay in the “trying to connect” queue. |

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.  With “TcpClient client = new TcpClient(new IPEndPoint(IPAddress.Any, 55556))” the exception  “Only one usage of each socket address (protocol/network address/port) is normally permitted” is thrown.  Without that line, ports 62655 and 62653 are used. So, on the client each application requires a different port since the IP address (the computer) is the same. |