NSHG: A Network Simulation Hacking Game

Table of Contents

[Analysis 2](#_Toc7087676)

[1.1, 1.2 Introduction (What is networking and why is it important) 2](#_Toc7087677)

[Interview Summary 5](#_Toc7087678)

[Teacher Interview 5](#_Toc7087679)

[Student Interviews 5](#_Toc7087680)

[The Current system 5](#_Toc7087681)

[What’s the problem? 6](#_Toc7087682)

[1.4 Success Criteria 6](#_Toc7087683)

[Limitations of my system 6](#_Toc7087684)

[Possible Solutions 7](#_Toc7087685)

[1 Revision Game 7](#_Toc7087686)

[2 Simulation game 7](#_Toc7087687)

[3 Story game 7](#_Toc7087688)

[Chosen solution (and why) 8](#_Toc7087689)

[Research 9](#_Toc7087690)

[The OSI model 9](#_Toc7087691)

[Headers 9](#_Toc7087692)

[Routing Tables 18](#_Toc7087693)

[Routing Protocols 19](#_Toc7087694)

[Design 20](#_Toc7087695)

[Overall Summary 20](#_Toc7087696)

[Hierarchy charts 20](#_Toc7087697)

[Server 21](#_Toc7087698)

[Client 23](#_Toc7087699)

[PCL 24](#_Toc7087700)

# Analysis

## 1.1, 1.2 Introduction (What is networking and why is it important)

The fundamentals of communication and networking is a massively important topic in the computer science A-level, with the growing connectivity of the world through the internet of things it is growing ever important that any computer scientists worth their weight will need a strong understanding of what these concepts are and how they ate implemented in the real world.

The topic in the A-Level is split up into 4 topics which are each split into further sub topics, these main and sub topics are as follows.

1. Communication
   1. Communication methods
      1. Serial/parallel
      2. Synchronous/Asynchronous
      3. Start and Stop bits
   2. Communication basics
      1. Baud rate, Bit rate, Bandwidth, Latency, Protocol
      2. Differentiate baud rate and bit rate
      3. Understand the relationship between bit rate, baud rate and bandwidth.
2. Networking
   1. Network Topology
      1. Difference between and how star and bus topologies work
   2. Types of networking between hosts
      1. Explain when peer to peer may be used
      2. Explain when client-server may be used
   3. Wireless networking
      1. Explain the purpose of Wi-Fi
      2. Be familiar with the components required for wireless networking
      3. Be familiar with how wireless networks are secured
      4. Explain the wireless protocol Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) with and without Request to Send/ Clear to Send (RTS/CTS).
      5. Be familiar with the purpose of Service Set Identifiers (SSID)
3. The Internet
   1. The Internet and how it works
      1. Understand the structure of the internet
      2. Understand the role of packet switching and routers
      3. Know the main components of a packet
      4. Define Router and gateway and consider where they are used
      5. Explain how routing is achieved across the internet
      6. Describe the term uniform resource locator in the context of internetworking
      7. Explain the terms fully qualified domain name (FQDM), Domain name and IP address
      8. Describe how domain names are organised
      9. Understand the purpose and function of the domain name service and its reliance on the domain name server (DNS) system
      10. Explain the service provided by internet registries and why they are needed
   2. Internet Security
      1. Understand how a firewall works (packet filtering, proxy server, stateful inspection)
      2. Explain symmetric and asymmetric encryption and key exchange
      3. Explain how digital certificates and digital signatures are obtained and used
      4. Discuss worms, Trojans, and viruses , and the vulnerability’s they exploit
      5. Discuss how improved code quality, monitoring, and projection can be used to address worms, Trojans and viruses.
4. The TCP/IP protocol stack
   1. TCP/IP
      1. Describe the role of the four layers of the TCP/IP stack (application, transport, network and link)
      2. Describe the role of sockets in the TCP/IP stack
      3. Be familiar with the role of MAC addresses
      4. Explain what well known ports and client ports are used for and the differences between them
   2. Standard application layer protocols
      1. Be familiar with the following protocols
         1. FTP
         2. HTTP
         3. HTTPS
         4. POP3
         5. SMTP
         6. SSH
      2. Be familiar with FTP client software and an FTP server, with regard to transferring files using anonymous and non-anonymous access
      3. Be familiar with how SSH is used for remote management
      4. Know how an SSH client is used to make a TCP connection to a remote port for the purpose of sending commands to this port using application level protocols such as GET for HTTP, SMTP commands for sending email and POP3 for retrieving email
      5. Be familiar with using SSH to log in securely to a remote computer and execute commands
      6. Explain the role of an email server in retrieving and sending email
      7. Explain the role of a web server in serving up web pages in text form
      8. Understand the role of a web browser in retrieving web pages and web page resources and rendering these accordingly
   3. IP address structure
      1. Know that an IP address is split into a network identifier part and a host identifier part
   4. Subnet masking
      1. Know that networks can be divided into subnets and know how a subnet mask is used to identify the network identifier part of the IP address
   5. IP standards
      1. Know that there are currently two standards of IP address, v4 and v6.
      2. Know why v6 was introduced
   6. Public and private IP addresses
      1. Distinguish between routable and non-routable IP addresses.
   7. DHCP
      1. Understand the purpose and function of the DHCP system
   8. Network Address Translation (NAT)
      1. Explain the basic concept of NAT and why it is used
   9. Port Forwarding
      1. Explain the basic concept of port forwarding and why it is used
   10. Client server model
       1. Be familiar with the client server model
       2. Be familiar with the Web socket protocol and know why it is used and where it is used.
       3. Be familiar with the principles of Web CRUD Applications and REST:
          1. CRUD is an acronym for:
             1. C – Create
             2. R – Retrieve
             3. U – Update
             4. D – Delete.
          2. REST enables CRUD to be mapped to database functions (SQL) as follows:
             1. GET → SELECT
             2. POST → INSERT
             3. DELETE → DELETE
             4. PUT → UPDATE.
       4. Compare JSON (Java script object notation) with XML.
   11. Thin vs Thick client computing
       1. Compare and contrast thin-client computing with thick-client computing.

This is a vast specification and in order to answer the questions fully the student must understand all of these points and be able to apply them and evaluate aspects of them. For example discuss how the TCP/IP stack works, what information is needed my each layer and what information is passed between them.

To begin my project I am going to interview my end users, whom are teachers and students, to see some of the problems they have with teaching and learning networking concepts, engaging students with the material and the understanding of the abstract concepts covered in networking and the internet.

The interview will be semi-structured having a list of required questions, written before the interview, and asking more questions based of the end users answers. This will allow for more in-depth information to be gathered reducing the risk of a possible re-design when showing the end user the “alpha” version of the project.

During the interview I will be taking notes

In the interviews I will discuss the problems and then possible solutions to those problems. I will focus on managing the expectations of the end user as to not have them expect more than I can produce in the given time.

## Interview Summary

### Teacher Interview

After having an interview with the teacher he has given me a list of lessons/concepts that are covered in networking and the internet when teaching the subject that are the hardest to grasp, this is TCP/UDP DHCP and ICMP. These are the topics that will need to be assessed for my solution for this project as implementing the whole solution would take a vast amount of time

The students find it very difficult when being talked at for the full lesion blocks and then being given a work sheet to complete. He was unsure why this was the case. However he would like to keep the idea of having constant feedback from the students in the form of task results.

At the end of the solution there must be an end of unit test which assesses everything my solution has implemented. This should output the results in a csv file so it can be processed externally. As my system cannot be trusted in a school setting to be secure.

### Student Interviews

The work sheets are boring

I don’t understand why we need to know about this.

When would I ever need to use DHCP in real life?

I want to actually do something to do with networking rather than just looking at PowerPoints and answering questions.

## The Current system

The subject of networking is usually taught in 4 weeks with 5 hours a week. The lessons 1 hour long with 1 block of 3 lessons and 1 block of 2 lessons each week. This results in a total of 20 hours to teach the specification.

The class size currently is 7 people thought this varies from year to year (with a maximum of 15)

The lessons consist of a power point with several task sheets to provoke questions from the students and assess their ability while going through the lesson. These tasks are fast and easy to complete.

At the end of the subject there is a 2 hour subject test which allows the teacher to make sure that the teaching has been successful and that the students have taken in the information this can then be processed to find what parts of the subject were misunderstood and what was received well which is used for revision/ recaps for a re-test if it is deemed necessary by the teacher.

### What’s the problem?

The current problem with the system is the large amount of time and complexity seen when implementing the system, the worksheets are dry and boring, and the long periods of teaching can send the students into a daze.

## 1.4 Success Criteria

|  |  |
| --- | --- |
| Success Criteria | Operationalised |
| The solution must implement and test three of the specification points fully. DHCP, UDP and ICMP | Have a check list of all teaching/information points that need to be taught & check they are all covered. Further check that they someone who hasn’t completed the specification before can use the program, with the test in front of them, and answer all of the questions fully. Given time |
| The solution must be engaging to students to make them want to learn about networking. | Test with the students to make sure that they enjoy the solution and engage with it. Found through observation and a self-report after they finish using the solution |
| The solution must have an “End of unit test” in order for the teacher to be sure all the students fully understand the topic | Have the end of unit test implemented as well as the one in the system to compare results to make sure the system is working, this ensures reliability of the new test. |
| The end of unit test would be self-marked | Have a teacher and the system mark the test to make sure it is marked correctly (interrater reliability) |
| Results should be output as a csv file making it easy to analyse | Have the results of each of the students be output as a CSV value (or another file type that can be imported into google sheets/excel) |

## Limitations of my system

With this system being used in a school environment there are many technical limitations to my solution. Firstly the school network has a large firewall that will blocks lots of network traffic to make sure the school network says secure. This means that any data will have to be hosted within the school system so it can be accessed while in school. However this has the problem that the data would only be available while in school. A possible (though unlikely) solution to this is allowing an exception through the school firewall to allow the data to be accessed from outside the system.

Another limitation is that I will not be able to store information about students as otherwise I wouldn’t be able to be able to ensure its security (as I am not qualified). My solution to this is not storing any personal information about the student meaning it cannot be stolen from me. This does mean that I can’t do any data analysis on the students and instead have that handled by the current data handling system, which has been recently updated.

## Possible Solutions

I brainstormed some ideas with my end users that I thought could solve the problems my end user was having from there I will show those ideas to the teacher and the students and see what idea they will find most engaging and will be most representative of what my end user had in mind

### 1 Revision Game

My first idea was to create a revision game where students would answer questions and be told if their answers were right or wrong. When the student got a question wrong it would be added to a Challenge bank to be used later. The challenge bank would allow students to continue with questions then revisit what they found hard to see if they had learned what they got wrong after a period of time, this is to make sure it goes into their long term memory instead of short term memory.

When a student gets a question correct they are given points in order to associate correct answers with a positive outcome. This is a great idea however many implementations like this already exist online and with the scope of my project I wouldn’t be able to implement anything near that in time. This also fails to make the teaching much more interesting and arguably would be harder to keep their interest as it would just be reading text of a screen, worse than a teacher talking. Lastly this isn’t suitable as it wouldn’t be able to actually teach the student

### 2 Simulation game

This idea would be a simulation of a network that would have the users “hack” into different systems using the required knowledge for the course. They would be hacking to find a flag that would be hex value. The flag would then be sent in as their answers, for each flag they send they would get a mark and a time stamp. Each flag would be attributed to a different specification point.

This solution would be much more interesting for the students as hacking would be much more interesting then filling out a work sheet. Lastly this will have the ability to teach and assess with multiple different networks. For this alpha project I could have 2, one to teach the concepts required and another to assess. It also would have the ability to output to a csv file with the username, mark, and time they gained it, if they gained it. This however would be a lot of work as I would have to implement each of the specification points that would need to be assessed.

### 3 Story game

This would be a story game where the student plays as a packet on a network roaming around and learning about different concepts. There would be different levels where the user would learn about different concepts. This would involve an overarching story and some simple combat/ challenges when traveling over the network. This would keep the students interested with a story and challenges.

This solution is very good for keeping the students interested however it would be difficult to stop the students from skipping text and just doing the challenges. And there would be no way to assess the students abilities in a story game so the problem of the boring task sheets would still be there, however this wouldn’t be a deal breaker.

## Chosen solution (and why)

I proposed my three solutions to some students who are potential end users. And received their feedback about each of the ideas which is included below and asked them which of the three they would prefer to use in the classroom.

“I have the attention span of a fish so the ‘hacking’ game would keep me occupied and engaged for longer. Especially when compared to the first solution and story games are boring”

“Defiantly the hacking game, I’ve always wanted to do something like that in the real world”

“The Hacking game is a good idea, but story games really interesting and I love getting stuck into them”

I conducted a survey of the students in my class to see what the overall results were

|  |  |
| --- | --- |
| Revision Game | 0 |
| Hacking Game | 4 |
| Story Game | 1 |

For me this is good enough evidence that the solution I should implement is the Hacking game.

## Research

Now I will need to conduct some deep research into each of the concepts I will be implementing and how I am going to do so. This is what follows.

### The OSI model

The Open Systems Interconnection model (OSI model) is a way to standardise the communication between systems it is broken down into 7 layers;

1. The physical layer
2. Data Link
3. Network
4. Transport
5. Session
6. Presentation
7. Application

I will go over each of the layers in detail now

#### Layer 1: Physical

This defines the physical, electrical connections between devices e.g. electrical cable or optical fibre. This includes: voltage levels, timing of voltage changes, physical data rates, maximum transmission distances, and physical connectors.

#### Layer 2: Data Link

This layer describes the connections between nodes and doesn’t deal with the physical layers. This layer has basic error correction that will occur on the physical layer. This layer doesn’t deal with connecting multiple nodes together (networking) it is only concerned with the connection between two nodes and terminating it.

This layer can sometimes be divided into two sub layers

##### The Medium access control (Mac) Layer

This is the sub layer responsible for how devices gain access to data and permission to transfer

##### The Logic Link Control (LLC) layer

This layer controls the encapsulation of protocols and error correction

#### Layer 3: Network

#### Layer 4: Transport

#### Layer 5: Session

#### Layer 6: Presentation

#### Layer 7: Application

### Headers

How does data travel across the internet?

I purchased the book The Networking Bible in which I found a reference to something called an RFC so I decided to investigate, the RFC it pointed to was RFC 791 which is the official standard for how the IP Header works and how IP addresses are structured I found that the RFC’s are a great source of info. RFC stands for Request for comments and they are used to show the implementations of systems for example the IP RFC shows how an IP header is formatted and how each field is used.

#### Internet Layer

##### IP [[1]](#footnote-1)

This is the basic framework for the internet, in the A level we are required to know that the IP header contains the source and destination IP addresses. In reality the IP header contains a large amount more than just the source and destination addresses which only take up two fifths of the entire header. 8 of the 20 bytes (minimum) of the header. For my research into how the IP specification and header is formatted I used Wire shark, a piece of software on my pc to intercept all packets coming to and from my system and then using this information along with the RFC 791 which allowed me to see, in the real world, how the IP header Is used and formatted, and what everything meant along with a hex array to see, bitwise, the same thing.

The IP Datagram is formatted in 20 bytes minimum with an optional options datagram at the end of the header. This has been depreciated and isn’t used anymore for this reason I will not be implementing the options and that data will always be length zero. The header is formatted as seen below [[2]](#footnote-2)

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

|Version| IHL |Type of Service| Total Length |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Identification |Flags| Fragment Offset |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Time to Live | Protocol | Header Checksum |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Source Address |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Destination Address |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Options | Padding |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

This shows byte by byte how the header is formatted. This is a good time to point out that there are a few values that are over 1 byte in size. E.g. The header checksum which is 2 bytes large, a unsigned 16 bit integer. This means that I will take into account Evidence when programing my solution. I will most likely use the systems default endianness instead of writing my own code to reverse it to the standard as this will not make a difference to the end solution.

Now I will briefly cover what each of the fields in the header represent.

###### Version (4 bits)

This is the version of the IP Header. For example 4 as this uses IPv4. This only changes when using IPv6. Which my solution will not us.

###### IHL (4 bits)

This is the “Internet Header Length” Measured in 4 byte words. For my purposes this will always be 5 as the header will always be 20 bytes in length

###### Type of Service (8 bits)

This is how the packet is treated. For example high or low latency This is formatted as shown below but will not be implemented in my solution

Bits 0-2: Precedence.

Bit 3: 0 = Normal Delay, 1 = Low Delay.

Bits 4: 0 = Normal Throughput, 1 = High Throughput.

Bits 5: 0 = Normal Relibility, 1 = High Relibility.

Bit 6-7: Reserved for Future Use.

Precedence

111 - Network Control

110 - Internetwork Control

101 - CRITIC/ECP

100 - Flash Override

011 - Flash

010 - Immediate

001 - Priority

000 – Routine

###### Total Length. (16 bits)

This is the total length of the datagram. Including the IP Header This is measured in octets (bytes). The minimum length that a host must be willing to accept is 576 bytes. For a IP Header with no options and no datagram would have a total length of 20.

###### Identification (16 bits)

This is the ID of a datagram and is used for when a data gram is fragmented to put all the pieces back together.

###### Flags (3 bits)

These are 3 flags. The first is reserved, then if the second bit is on it specifies that the datagram cannot be fragmented. And off means it may be fragmented. And the Last bit specifies if there are more fragments to come. A value of 1 specifies that there are more fragments.

###### Fragment Offset (13 bits)

This is the offset of the data of the packet. The first fragment has offset of zero. If this held the first 200 bytes the offset of the second packet would be 200.

###### Time to Live (TTL) (8 bits)

This is the time (in seconds) that the packet can exist before being destroyed. If a packet has TTL of 0 then it is discarded and an ICMP packet will be sent to the source IP Address of the packet specifying that this happened.

This is usually implemented as hops instead of seconds as a packet will be forwarded in a matter of milliseconds and a TTL of 30 seconds would be too long and redundant and is instead decremented by 1 every Hop. So a TTL of 30 Hops would be more appropriate in terms of speeds of the internet.

###### Protocol (8 bits)

The Protocol of the header in the datagram of the packet e.g. 1 – ICMP, 6 – TCP.

###### Header Checksum (16 bits)

The ones compliment of the ones compliment sum for each 16 bit word in the header, not including the datagram.

###### Source Address (32 bits) Destination Address (32 Bits)

These are the source and destination addresses of the systems on the network. For example if a system with the IP address 192.168.1.1 wanted to send a packet to the system with IP address 192.168.2.9 The source field would be the senders IP address and the destination field would be the destination systems IP address.

#### Transport Layer

##### UDP

UDP is a transport layer protocol, similar to TCP but much simpler. This means it will take much less time to implement hence I am only using UDP for my sample project.

UDP is a port oriented protocol that is used to transfer data from one application on a source computer to another application on the destination computer. For example HTTP requests would be sent to the port 80 on the http server as this is where a UDP listener would be listening for packets arriving addressed to that port so it can respond to request. UDP isn’t a connection oriented protocol meaning that it, data is sent in a packet by packet manner and there is no way of making sure all data was received by the destination system. Compare to TCP where a connection is first established and held for the duration of the connection, until one end closes it, or it is closed gracefully by either end. UDP has no system like this and just sends data. This is also mirrored in the simple, small header as shown and labelled below.

0 7 8 15 16 23 24 31

+--------+--------+--------+--------+

| Source | Destination |

| Port | Port |

+--------+--------+--------+--------+

| | |

| Length | Checksum |

+--------+--------+--------+--------+

|

| data octets ...

+---------------- ...

###### Source Port

This is port that the information is being sent from, for example when taking to a host on the port 80 the source port would be chosen as any, random, port that isn’t in use or reserved.

###### Destination Port

This is the port on the destination that the data will be passed to, for example when accessing a web page using HHTP the destination port 80 would be used. There is a list of reserved ports in the Assigned Numbers RFC 762.

###### Length

The Length of the Header and Datagram measured in octets (Bytes), minimum value of 8 for a packet with no datagram.

###### Checksum

The ones compliment of the ones compliment sum of each 16 bit word in the header and data. Calculated the same way as the IP Checksum. If the length of the header isn’t a multiple of 2 it is padded with one octet of zeros.

###### Data octets

This is the data of the underlying packet. For example a DHCP Packet.

#### Application Layer

##### DHCP

Now I have the Headers sorted I need to think about what happens when a system first connects to a network. The first thing a system does when it connects to a network is obtain an IP address and other information through DHCP. This means I will have to implement a version of the DHCP protocol which uses UDP. The basic premise behind DHCP is 4 packets, 1 a broadcast from a DHCP client onto their local network to discover a DHCP server if there is one, 2 A DHCP server Responds multicast with an offer to lease an IP address, 3 the client responds (unicast) with a request to the DHCP server requesting the offered IP Address, 4 The DHCP server responds with a DHCPAck packet, which contains confirmation of the IP address, the subnet mask, Default gateway and server[[3]](#footnote-3) [[4]](#footnote-4) [[5]](#footnote-5)

I will use the RFC1 to form the layout of my packets however with the scope of this project and to reduce the processing overhead I will have to simplify it and it will not be fully featured, only the barebones will be there

This is a good time to note that DHCP is an extension of the older BOOTP Protocol so some of the fields in the datagram are now depreciated.

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| op (1) | htype (1) | hlen (1) | hops (1) |

+---------------+---------------+---------------+---------------+

| xid (4) |

+-------------------------------+-------------------------------+

| secs (2) | flags (2) |

+-------------------------------+-------------------------------+

| ciaddr (4) |

+---------------------------------------------------------------+

| yiaddr (4) |

+---------------------------------------------------------------+

| siaddr (4) |

+---------------------------------------------------------------+

| giaddr (4) |

+---------------------------------------------------------------+

| |

| chaddr (16) |

| |

| |

+---------------------------------------------------------------+

| |

| sname (64) |

+---------------------------------------------------------------+

| |

| file (128) |

+---------------------------------------------------------------+

| |

| options (variable) |

+---------------------------------------------------------------+

The DHCP Packet will be structured as seen above (Excerpt from RFC 2131 Page 34).

The options field is defined with a tag (1 byte) and possibly followed by an option length and/or data.[[6]](#footnote-6)

In the DHCP RFC 2131 Page 10 there is a table of the different fields in the DHCP Datagram seen above. I have copied the table here and will comment in more detail what each of the fields are used for and examples.

###### Field (Length in Octets)

RFC Definition

My Expansion

###### op (1)

Message op code / message type.  
1 = BOOTREQUEST, 2 = BOOTREPLY

Used to specify if a message is going to/from the DHCP server

###### htype (1)

Hardware address type, see ARP section in "Assigned" RFC; e.g., '1' = 10mb Ethernet.

Will only have a value of 1 for my implementation as I am not implementing down to the MAC layer.

###### Hlen (1)

Hardware address length (e.g. '6' for 10mb ethernet).

Length, in bytes, of the hardware (MAC) address.

###### hops (1)

Client sets to zero, optionally used by relay agents when booting via a relay agent.

Will be set to zero when sending DHCP packets, used to track how many, if any, relays were used when sending the packet.

###### xid (4)

Transaction ID, a random number chosen by the client, used by the client and server to associate messages and responses between a client and a server.

###### secs (2)

Filled in by client, seconds elapsed since client began address acquisition or renewal process.

###### flags (2)

Flags

###### ciaddr (4)

Client IP address; only filled in if client is in BOUND, RENEW or REBINDING state and can respond to ARP requests.

###### yiaddr (4)

'your' (client) IP address.

###### siaddr(4)

IP address of next server to use returned in DHCPOFFER, DHCPACK by server.

###### giaddr(4)

Relay agent IP address, used in booting via a relay agent.

###### chaddr(16)

Client hardware address.

###### Sname (64)

Optional server host name, null terminated string.

###### File(128)

Boot file name, null terminated string; "generic" name or null in DHCPDISCOVER, fully qualified directory-path name in DHCPOFFER.

###### Options (var)

Optional parameters field. See the options for a list of defined options.

#### DHCP State Transition Diagram

-------- -------

| | +-------------------------->| |<-------------------+

| INIT- | | +-------------------->| INIT | |

| REBOOT |DHCPNAK/ +---------->| |<---+ |

| |Restart| | ------- | |

-------- | DHCPNAK/ | | |

| Discard offer | -/Send DHCPDISCOVER |

-/Send DHCPREQUEST | | |

| | | DHCPACK v | |

----------- | (not accept.)/ ----------- | |

| | | Send DHCPDECLINE | | |

| REBOOTING | | | | SELECTING |<----+ |

| | | / | | |DHCPOFFER/ |

----------- | / ----------- | |Collect |

| | / | | | replies |

DHCPACK/ | / +----------------+ +-------+ |

Record lease, set| | v Select offer/ |

timers T1, T2 ------------ send DHCPREQUEST | |

| +----->| | DHCPNAK, Lease expired/ |

| | | REQUESTING | Halt network |

DHCPOFFER/ | | | |

Discard ------------ | |

| | | | ----------- |

| +--------+ DHCPACK/ | | |

| Record lease, set -----| REBINDING | |

| timers T1, T2 / | | |

| | DHCPACK/ ----------- |

| v Record lease, set ^ |

+----------------> ------- /timers T1,T2 | |

+----->| |<---+ | |

| | BOUND |<---+ | |

DHCPOFFER, DHCPACK, | | | T2 expires/ DHCPNAK/

DHCPNAK/Discard ------- | Broadcast Halt network

| | | | DHCPREQUEST |

+-------+ | DHCPACK/ | |

T1 expires/ Record lease, set | |

Send DHCPREQUEST timers T1, T2 | |

to leasing server | | |

| ---------- | |

| | |------------+ |

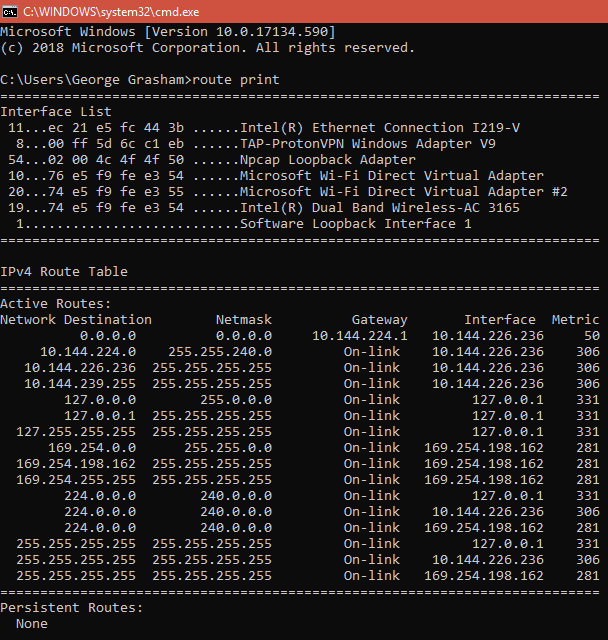
+->| RENEWING | |

| |----------------------------+

----------

### Routing Tables

I will now have to look at how a system decides what adapter to send a packet to for it to forward on. After a brief google search I found that Routing tables are responsible for looking at a packets IP header and deciding what physical connection to send it down. You can access your own routing table in windows by using the “route print” command, I ran this command to see what I could understand from the beginning and what I needed to research more about. The result of my route print command is included below



From here I can see a few familiar things, 1 at the top there is an interface list. This seems to be a list of network adapters that the system has. For me I have a few that stand out, The Inter(R) Ethernet Connection I219-V which is my Lan Ethernet adapter so If I plugged an Ethernet cable into my laptop that adapter would be used, The Intel(R) Dual Band Wireless-AC 3165 which is my current adapter as I am connected to my current network wirelessly. There is also the software loopback which is the loopback adapter for my system, for example if I wanted to ping myself I would use the loopback adapter so I don’t need to be connected to a network.

Next the IPv4 Route Table. This is the routeing table for my system, this confuses me a little but here are my first ideas about each of the columns before research. The destination is the destination address that a packet would have to be addressed to be forwarded to a specific adapter, the Net mask also play a role in this to as the net mask would be applied in the packet that needs to be routed to see if it fits the destination address once it is also had the net mask applied to it. For example for my system if I sent a packet with the address 127.6.1.1 it would be accepted by the 127.0.0.0 entry as if you apply the mask 255.0.0.0 to 127.6.1.1 it yields 127.0.0.0 meaning that is the route it would take. The first entry in these addresses has the 0.0.0.0 address and the 0.0.0.0 mask meaning that any address would be mapped to this entry successfully. This leads me to believe when a system is using the table it doesn’t use a top down approach and instead likely has something to do with the last column “Metric”. The next column is the Gateway. I think this is the address that the packet needs to be forwarded to in order to reach its destination. As of right now I am unsure about on-link but I would like to note that each entry with on-link is a local address. I believe that Interface is the specific adapter that the packet needs to be routed down to reach its destination. The reason it uses the IP address of the interface instead of the mac address is unclear to me but could have something to do with setting the source address of the packet. I will now research to see what I was correct about

From my research I know that what I thought was basically correct, when sending a packet, the routing table is used to know what interface to use. The system will look up the table in a decrementing fashion looking at the highest metric first and getting smaller, the destination and netmask are used to see if the sending destination address is matched by the rule, if the address is a match then the packet is sent on the appropriate interface. The Gateway specifies that the destination isn’t on the local network and needs to be forwarded to that gateway for NAT to take place and then be forwarded from there onto the larger network.[[7]](#footnote-7),[[8]](#footnote-8)

My next question was how Microsoft populates this table

### Routing Protocols

After looking at many routing protocols and how they are used I found that classless routing algorithms are very complex (Especially for what I need them for) but all follow a similar protocol. A hello message sent regularly to make sure that the router is till up, when a router stops responding the routing tables of the surrounding routers are changed. My implementation will follow a similar premise with the hello packets to discover the network and then when a router goes down a message will be sent saying that it is down, updating the network maps of the routers.

The route the router will forward the packet down is found using Dijkstra’s shortest path algorithm to create a forwarding table.

# Design

## Overall Summary

My Project is a Network Simulation Hacking Game or “NSHG” that will aim to interest students in learning networking concepts in networking and the internet in a fun, easy but still academic way. There should be 2 main pieces of software in my package. The client and the server.

The server will be ran by one of the teachers possibly on one of their pc’s for this reason it must be light and easy to run. The server will let the student’s login using their client, run a simulation of the network for the students to “hack” and learn through, and lastly a results collection system will be implemented to collect the results of the tests done by the students. Success will be gauged by which “flags” they were able to collect with each flag requiring a different skill in order to acquire.

The client will be run on the student’s desktop computers and will be used to login to and communicate with the server. The client will be a virtual command line interface to their “system” in the game and will provide them with the material needed in order to progress through the learning levels and then hints/tips for the tests and the correct direction to go.

There will also be a portable class library which will allow for shared code between the server, client and test projects. The PCL will contain the simulation code such as the System code which will act as a pc in my simulation.

## Hierarchy charts

I will now show the hierarchy charts for each of the pieces of software along with a description of what each modules function will be.

### Server

|  |  |
| --- | --- |
| Module | Function |
| **Simulation** | Running the simulation that the clients will interact with, this involves making sure that communications are sent between systems, The AI keeps ticking, and the ability to control and edit each system while the running. |
| Data Transfer | This module will be used to make sure that data is sent, this isn’t specifically done by the simulation but it supports it with the constant ticking which will parse through the list of packets in a system’s packet queue. |
| Run AI | This is a ticking clock that when running will send a pulse every cycle, this will be a delegate which is subscribed to an event that is raised every cycle. The cycle time is variable. There will be a system in place to make sure that the previous tick has finished completing before the next tick start running. This will run the tick commands for each system in the pseudo network. |
| Real Time Control | This is a command line interface that will allow the server operator to start, stop and change the interval of the tick timer. It will also enable the operator to edit each & run commands on each system in the network. This will allow the server operator to assist students who are struggling with what commands to run. |
| **User Interface** | This is a very simple UI for the user and will be a WPF XAML interface which will basically just be a command line however instead of typing into the same window that the log appears you will type into a different input text box and either press return or a command button. |
| System Log | This will be large text box which is a log of all command any users are sending, packets being sent between systems. |
| Command Input | Small text box that can be typed into, will be responsible for cleaning any input, removing whitespace, and logging the commands onto the command log. This text box should accept a return |
| **Client Server** | This is the networked server the clients will connect to, it will deal with them logging in, requests, and results. |
| Receive Commands | This will be responsible for receiving a packet that was issued as a command to the clients system. |
| Login System | The Login System will be just a username and a temporary password. This is because I cannot ensure the security of a user’s password. The password will be stored in plain text and will be one time use for that session. The username will be selected by the user. |
| Results System | This will be responsible for collecting results of all of the clients and shortly before the time end for the lesson/test it will remind the participant to send their results. |
| Sync Systems | This will be responsible for syncing the application logs and system logs for the users system up to date. This will all be managed by the server, for example if a user sent an echo all adapters command to their system their command log wouldn’t change until it was told to update by the sync server. Which would tell it to write the command first, then when there is an output, it will send that too. This is to make the clients as light as possible. |

### Client

|  |  |
| --- | --- |
| Module | Function |
| **Login Portal** | This module will only be used once when connecting to the server. You will need a user name and, if that username already exists the password, otherwise the server will generate one and send it to the user. |
| **User Interface** | The UI for this package will be more complex than any others and will have a main screen for logging in, then progress to a main screen where there will be an interface for their system along with a results entry page |
| Command Log | There will be a main command log for their system. Which will be a large text box that cannot be typed into |
| Command Input | The command input is a text box that can be typed into, that when the return key is pressed will send an appropriate command to the server |
| System Application List | There will be a command log and input screen for each application slot on the system, even those without any applications running this will allow the user to start running new applications. |
| Results Management | This screen will be used to show all flags that have been submitted and will allow you to submit new flags |
| **Server Connection** | This is responsible for all communication, this will be a socket base connection running TCP |
| Synchronisation | Will listen for synchronisation updates from the server and update the relative command log/ logs respectively. |
| Command Output | Will send all commands and the context that they were executed on, for example a command executed in the context of application 8, will be sent straight to application 8 rather than to the system, or rather it will be prefixed with the correct command so that it is router to application 8 without the user having to manually specify that. |

### PCL

|  |  |
| --- | --- |
| Module | Function |
| **Network** |  |
| System List |  |
| Connection List |  |
| Player Slot Management |  |
| XML Load |  |
| XML Save |  |
| **Systems** |  |
| XML Load |  |
| XML Save |  |
| AI |  |
| Application Slots |  |
| **Applications** |  |
| Routing Table |  |
| Packet Sniffer |  |
| DHCP Client |  |
| DHCP Server |  |
| **Headers**  IPv4 |  |
| UDP |  |
| DHCP |  |
| **Types** |  |
| IP Address |  |
| MAC Address |  |
| **Client/Server Communication** |  |
| Packet Header |  |
| Packet Type |  |
| Constants |  |

1. <https://tools.ietf.org/html/rfc791> [↑](#footnote-ref-1)
2. <https://tools.ietf.org/html/rfc791> [↑](#footnote-ref-2)
3. <https://tools.ietf.org/html/rfc2131> [↑](#footnote-ref-3)
4. <https://www.youtube.com/watch?v=S43CFcpOZSI> [↑](#footnote-ref-4)
5. <https://en.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol#Operation> [↑](#footnote-ref-5)
6. <https://www.iana.org/assignments/bootp-dhcp-parameters/bootp-dhcp-parameters.xhtml> [↑](#footnote-ref-6)
7. http://techgenix.com/making-sense-windows-routing-tables/ [↑](#footnote-ref-7)
8. http://library.mobrien.com/Manuals/MPRM\_Group/images/rightframe\_files/routing.htm [↑](#footnote-ref-8)