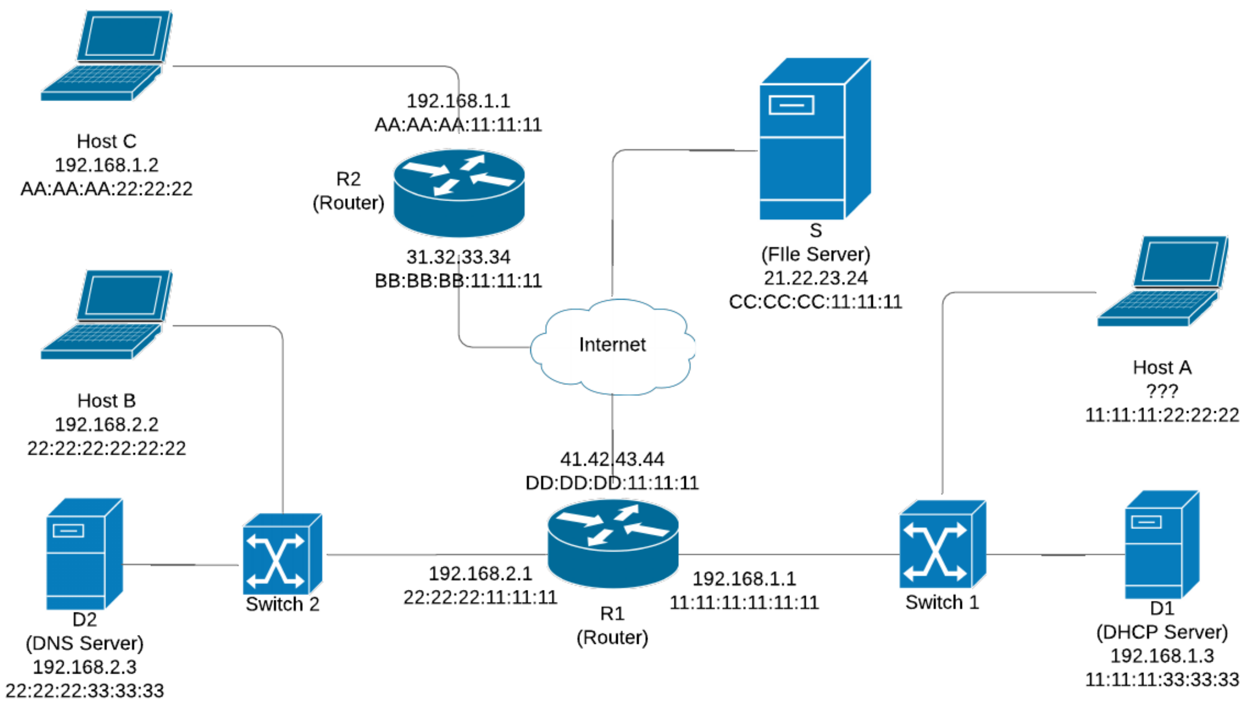
# COMS3200/7201 Assignment 3

Due: Friday 31st May 2019, 20:00

100 marks total (scaled to 15% of final grade)

## Part A (25 marks)

Part A asks some questions on packet behavior in the following network, which depicts three LANs and a file server connected to the internet.



Answer each of the following questions in the associated quiz on blackboard, following the specified instructions. All questions will be automatically marked. Each question follows on from the previous question - that is, for each question you can assume the tasks described in the previous question(s) have been fully completed.

**NOTE:** In any column that asks you for a protocol, the following options will be allowed: HTTP, FTP, SSH, DNS, DHCP, UDP, TCP, ICMP, SNMP, ARP, OSPF, BGP. In any case where multiple protocols are used, pick the highest-layer protocol (eg. if SSH was used, you would select it over TCP, even though SSH packets use TCP as a transport layer protocol). The broadcast IP address of all LANs depicted in this network is 255.255.255.0, and the broadcast MAC address of all LANs is FF:FF:FF:FF:FF:FF. Assume all forwarding tables in the switches and routers are up to date.

1. (6 marks) Suppose Host A has just joined the network and does not currently have an IP address. Complete the following table describing the four packets that will be transmitted through the network upon this event (in the order they were sent), assuming that all requests succeed and that Host A ends up with the IP address 192.168.1.2.

| Protocol | Opcode | Src MAC addr | Dst MAC addr | Src/Sender IPv4 addr | Dst/Target IPv4 addr |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

2. (6 marks) Now suppose Host A wants to send a standard DNS query to D2. Host A has already gotten the IPv4 addresses of D2 and its gateway router from D1. R1's ARP table contains the IP and MAC addresses of all hosts in the network, and all other ARP tables are empty. Complete the following table describing the first four packets that will be transmitted through the network upon this event, assuming that there are no other ongoing communications.

| Protocol | Opcode | Src MAC addr | Dst MAC addr | Src/Sender IPv4 addr | Dst/Target IPv4 addr |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

3. (5 marks) Host A now wants to establish a HTTP connection with a server (S). You may assume that Host A knows S's IP address and that all ARP tables contain all required information to transmit any packet. Complete the following table describing the packets that are sent to transmit Host A's request to initialise the connection, assuming the request is being sent from TCP port 4001 and the next available port number in R1's NAT table is 5001. Give the TCP flags as an 8-bit binary number representing the CWR to FIN flags.

| Protocol | TCP Flags | Src TCP port | Dst TCP port | Src/Sender IPv4 addr | Dst/Target IPv4 addr |
|---|---|---|---|---|---|
| TCP | | | | | |
| TCP | | | | | |

4. (4 marks) Suppose in addition to the connection described in (3), Host B port 3200 is also currently connected to S. What are the contents of the NAT table in R1 while these connections remain alive? (if a new port is required, you can choose any valid port).

| LAN IP | LAN Port | WAN IP | WAN Port |
|---|---|---|---|
| | | | |
| | | | |

5. (2 marks) Suppose host B sent a packet with destination IP address 255.255.255.0 and destination MAC address FF:FF:FF:FF:FF:FF. What network devices (hosts, servers, routers, switches) in the above diagram would receive this packet?

6. (2 marks) Suppose S sent a packet with destination IP address 192.168.1.2. What network devices (hosts, servers, routers, switches) in the above diagram would receive this packet?

## Part B (25 marks)

This question requires you to analyse the Wireshark file `assign3.pcapng`. This packet capture shows a client using the `tracert/traceroute` command to find the path that is taken from their machine to an external server. Note that the trace has been slightly modified, so not all information conveyed in this trace is representative of real-world information.

Answer each of the following questions in the associated quiz on blackboard, following the specified instructions. All questions will be automatically marked.

1. What is the IPv4 address of...
   (a) The network's gateway interface? (1 mark)
   (b) The network's DHCP server that offers the client's IP address? (1 mark)
   (c) The network's primary DNS server? (1 mark)
2. How many possible IPv4 hosts (including existing hosts) could fit inside the client's subnet? (2 marks)
3. Consider the ARP packet shown in frame 26
   (a) Who is the vendor of the receiver's NIC? (1 mark)
   (b) Is this a broadcast, multicast, or unicast message? (1 mark)
4. Which of the following Ethernet fields are not captured in any of the Wireshark frames in this trace? (there may be multiple) (1 mark)

   i. SFD
   ii. Source MAC address
   iii. Destination MAC address
   iv. Ethertype
   v. Payload
   vi. CRC

   (Optional extension question (not marked): Can you explain why not all fields are not captured?)
5. What is the hostname of the server being tracerouted? (2 marks)
6. How many hops are taken to get to the server being tracerouted? (2 marks)
7. What is the hop number of the first ping in the traceroute sequence that does not receive a response? (2 marks)
8. What is the IPv4 address of `lichen.labs.eait.uq.edu.au`, according to the Wireshark trace? (3 marks)
9. What is the IPv4 address of the DNS server used to retrieve the information in (8)? (1 mark)

The following questions each ask you to provide a **Windows** command-line command. In any case where multiple commands exist, you should use the one taught in the Wireshark labs for this course (we will of course be flexible on the order of parameters in the commands when marking). Assume for each of the following scenarios, the user has waited until the command has finished its execution (without manually terminating it early).

10. What command was likely used by the user to initiate the packet transmission captured in the following frames?
    (a) Frame 16 (1 mark)
    (b) Frames 19 to 23 (1 mark)
    (c) Frames 381 to 394 (ignoring ARP packets) (2 marks)
    (d) Frame 455 (3 marks)

## Part C (50 marks)

In this part of the assignment you will be required to implement the network layer for a host running on a virtual IP network. You should be familiar with the typical TCP/IP stack (A.K.A. the DoD model or internet protocol suite), where Ethernet is used as a link-layer protocol and IPv4 used as a network layer protocol. The protocols we will be using will be *virtual* - that is, network layer addresses won't actually correspond to physical interfaces. To do this, the network stack will be redefined to that shown in the table below.

| **TCP/IP Stack** | **Protocols** | **Virtual Stack** |
|---|---|---|
| | | Network Layer |
| Transport layer | *UDP* | Link layer |
| Network layer | *IPv4* | |
| Link Layer | *Ethernet* | |

As described in the above figure, UDP will be used as your virtual network's link layer, and you will be required to implement a virtualisation of IPv4 ontop of this UDP-based link layer. Each UDP socket will correspond to a link layer interface, and localhost UDP port numbers will be used as the link layer addressing system of this virtual network.

By the end of this assignment, your implemented host program should be able to:

- Accept simple user commands through a basic command line interface (CLI)
- Send and receive messages across this virtual network layer
- Handle fragmentation of virtual IP packets

### Program Invocation

Your program should be able to be invoked from a UNIX or UNIX-like command line as follows. It is expected that any Python programs can run with version 3.6, and any Java programs can run with version 8. The `ip-addr` and `ll-addr` parameters correspond to the IPv4 address in CIDR notation (indicating the client's subnet) and link layer address (UDP port number) of your host program respectively.

#### Python

```
python3 assign3.py ip-addr ll-addr
```

#### C/C++

```
make
./assign3 ip-addr ll-addr
```

#### Java

```
make
java Assign3 ip-addr ll-addr
```

## Your Task

### Command Line Interface (5 marks)

To start with, you should implement a basic command line interface that will allow the user to supply basic information about the network. Your CLI should prompt users with a single `>` character, followed by a space. For the rest of the assignment we define anything wrapped in square brackets as a parameter or field that needs replacing (NOT as an optional parameter). The CLI should persistently prompt the user for another command until the program is terminated. For full marks in this section your CLI needs to accommodate the following commands:

- `gw set [ip-addr]` : set the gateway IP address of the subnet the client is a part of to `[ip-addr]` (overriding any existing gateway address)
- `gw get` : print the currently stored gateway IP address to stdout, or `None` if no gateway address has been specified
- `arp set [ip-addr] [ll-addr]` : insert a mapping from `[ip-addr]` to `[ll-addr]` in the host's ARP table (overriding any existing entries for `[ip-addr]`)
- `arp get [ip-addr]` : print the currently stored link layer address mapped to `[ip-addr]` to stdout, or `None` if no mapping exists
- `exit` : terminate the program

Error-handling of user input is optional - you won't receive any marks for this but you can choose to implement it provided it doesn't impact on the rest of the assignment.

### Sending Messages (15 marks)

To receive marks in this section, your CLI should be able to handle the following additional command:

- `msg [ip-addr] "[payload]"` : send a virtual IPv4 packet to `[ip-addr]` with the given payload (which will be supplied as a string)

Any packets sent should have a meaningful identifier and a protocol number of 0. You may assume the that all payloads will be less than or equal to the MTU of the network's links, but the DF flag should still be set to 0. Your program should support any IP address, regardless of whether it is a part of your subnet or not. If your program needs to send a packet to the gateway address and no gateway address has been specified, your program should print `No gateway found` to stdout. If no required address mapping can be found in the host's ARP table, your program should print `No ARP entry found` to stdout.

### Receiving Messages (15 marks)

In addition to sending packets, your program should be able to receive them from other hosts. When your program receives an IPv4 packet with the protocol indicator set to 0, it should print the payload of that packet to stdout, in the below format (`[ip-addr]` should be replaced with the sender's IPv4 address and `[message]` should be replaced with the string encoding of the payload):

```
Message received from [ip-addr]: "[message]"
```

Note that there is only a single space after the colon. When your program receives an IPv4 packet with a non-zero protocol number, it should print the following message to stdout (`[proto-num]` should be the hexadecimal representation of the protocol formatted as 0x??):

```
Message received from [ip-addr] with protocol [proto-num]
```

You can assume all packets sent to your program are valid IPv4 packets. You should be able to receive messages at any time without blocking the CLI, and any messages should be printed cleanly (without any CLI prompts or responses disrupting the message contents). Remember to backspace the current prompt (the `>` character) before printing the output.

**IP Fragmentation (15 marks)**

To receive marks in this section your program should be able to handle IP fragmentation. Your CLI should support the following extra commands:

- `mtu set [value]` : set the MTU of the network's links as the specified `[value]`
- `mtu get` : print the currently stored MTU (the default MTU should be 1500)

Virtual packets that are longer than the specified MTU (or the default MTU if none has been specified) should be fragmented before transmission. The length of a virtual packet is equal to the length of the IPv4 header added to the length of the IPv4 payload (i.e. you don't need to consider the length of the non-virtual headers). You can assume the value of the MTU will never be smaller than 100.

Your program should also be able to receive packets that have been fragmented (and display them as a single message).

## Example CLI output

```
python3 assign2.py 192.168.1.1/24 1024
> gw get
None
> gw set 192.168.1.30
> gw get
192.168.1.30
> msg 192.168.1.2 "hello"
No ARP entry found
> arp get 192.168.1.2
None
> arp set 192.168.1.2 2222
> arp get 192.168.1.2
2222
> msg 192.168.1.2 "hello"
> mtu get
1500
> mtu set 1600
> mtu get
1600
Message received from 192.168.1.2:  "hello there, thankyou for your message"
Message received from 192.168.1.3 with protocol 0x06
> exit
```

## Packet Formatting

Your IPv4 packets don't need to contain meaningful values for the DCSP/ECN fields, nor are you required to compute a correct checksum. You can assume packets will never contain any options and subsequently that the IHL will always be 5. The TTL field should contain some meaningful value (i.e. to allow the packet to reach its destination).

### Tips for Success

- Revisit the lectures and labs on the internet layer of the TCP/IP stack (in particular the lectures on IPv4)
- Make sure you fully understand the requirements of this assignment before commencing work
- Ensure you always exercise good thread safety if using threads to implement concurrency
- Frequently test your code
- Ensure your base functionality is working (such as the CLI) before attempting the more difficult tasks
- Start early and ask any questions you might have early

### Library Restrictions

- The only communication libraries you may use are standard socket libraries which open UDP sockets
- You can't use any libraries that aren't considered standard for your language (i.e. if you have to download a library to use it it would be considered as non-standard)
- If you are unsure about whether you may use a certain library, please ask the course staff on Piazza

### Submission

Submit all files necessary to run your program. At a minimum, you should submit a file named `assign3.py`, `assign3.c`, `assign3.cpp` or `Assign3.java`. If you submit a C/C++ or Java program, you should also submit a makefile to compile your code into a binary named `assign3` or a .class file named `Assign3.class`.

**IMPORTANT:** If you do not adhere to this (eg. submitting a C/C++/Java program without a Makefile, or a .class file instead of a .java file), *you will receive 0 for this part of the assignment.*

### Marking

Your code will be automatically marked on a UNIX machine, so it is essential that your program's behaviour is exactly as specified above. Your program should complete all tasks within a reasonable time (no more that one second) - there will be timeouts on all tests and it is your responsibility to make sure your code is not overly inefficient. It is expected that you will receive a small sample of tests and a basic network emulator before the submission deadline.

There are no marks for coding style, however you may lose marks in relevant sections for poor management of concurrency.

## Academic Misconduct

Students are reminded of the University's policy on student misconduct, including plagiarism. See the course profile and the School web page `http://www.itee.uq.edu.au/itee-student-misconduct-including-plagiarism`.

## Late Submission and Extensions

Please view the ECP for policies on late submissions and extensions.

## Version History

### v1 (11/05/2019)

- Released assignment

### v2 (16/05/2019)

- Fixed typos

### v3 (24/05/2019)

- Part A Q5 will *not* be manually marked

### v4 (25/05/2019)

- Minor change to the part B marking scheme - the original mark count did not add up to 25

### v5 (26/05/2019)

- Clarified that there is only one space after colons in output