COMS3200 Assignment 2 2023S1

100 total marks, 25% overall course mark
Due: 15:00 26 May 2023

1 Preface

1.1 Notes

- This document is subject to change for the purposes of clarification. Changes made since the original release will be highlighted in red.
- Please post any questions on the course Ed stem page.

1.2 Revision History

- 4 May, 2023: Version 1.0 Released.
- 12 May, 2023: Version 1.1 - Fixed typos and a few clarifications.

2 Part A: Problem Solving Questions

This section is worth 25% of the assignment.
The question set is located on Blackboard under Assessment $\implies$ Assignment 2 $\implies$ Part A: Problem solving questions.
There is no time limit to submit these answers. Due to the widespread brute-forcing of answers on the previous assignment, only two resubmissions are permitted but working is not required. Only the last submitted attempt will be marked.

3 Part B: Wireshark Questions

This section is worth 25% of the assignment.
This section covers DHCP, IP, DNS and ARP.
The question set is located on Blackboard under Assessment $\implies$ Assignment 2 $\implies$ Part B: Wireshark questions and the capture file is located under Assessment $\implies$ Assignment 2 $\implies$ Part B: Packet capture File.
There is no time limit to submit these answers. For the same reasons as specified within part A, only two resubmissions are permitted. Only the last submitted attempt will be marked.

4 Part C: Socket Programming

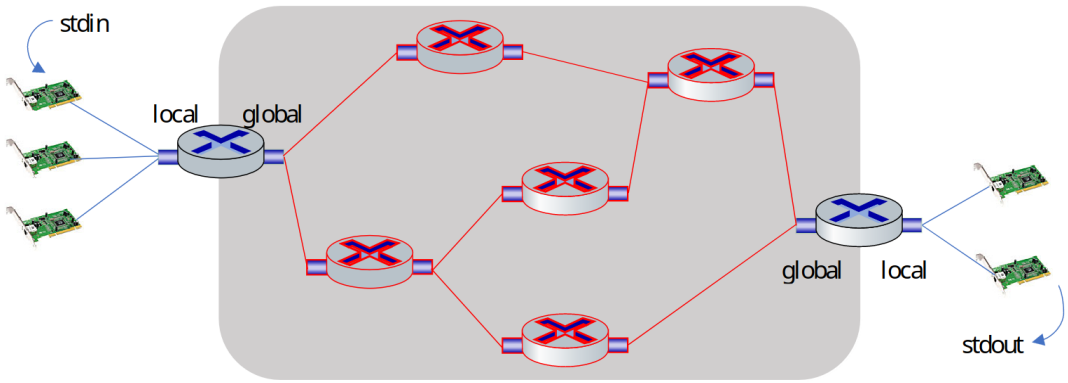
This section is worth 50% of the assignment.

4.1 Goals

You will implement a simulation of a network router in the application layer in Python 3 or C.
This will operate as part of a larger communication system named RUSHB.
This system has a strong emphasis on transmitting data using structured packets, address allocation and efficient routing algorithms.

4.2 Programs

RUSHB is comprised of two main programs: RUSHBAdapter and RUSHBSwitch. From here onwards, RUSHBAdapter processes are referred to as adapters and RUSHBSwitch processes are referred to as switches.
Adapters accept input from `stdin` and transmit this to one corresponding switch over UDP.
Switches can take three forms: local, global and mixed. Local switches open a listening socket on UDP to serve adapters and can connect to global and mixed switches over TCP. Global switches open a listening socket on TCP to service incoming connections with other switches, and can also create outgoing connections to other global/mixed switches. Mixed RUSHBSwitch processes open a listening socket on both UDP and TCP for serving incoming connections from adapters and global/mixed switches respectively, and can also create outgoing connections to other global/mixed switches. This is illustrated in the diagram below:



Code for the RUSHBAdapter is provided, and you do not need to create it yourself. You are required to implement the RUSHBSwitch only.

All adapter and switch processes maintain a virtual IP address used to distinguish each other apart. Adapters and local switches will maintain virtual local IP addresses and global switches will maintain virtual global IP addresses. Mixed switches will maintain a virtual local IP for their local side (facing adapters only) and a virtual global IP (facing other switches only). It is important to note that these addresses are virtual and are not indicative of their real IP addresses. All processes will be run on Moss.

4.3 Packet Structure

All communications to and from adapters and switches follow the required packet structure detailed below:

Bit	0	8	16	24
	Source IP			
	Destination IP			
	Offset			Mode
	Data			

Mode	Value
Discovery	0x01
Offer	0x02
Request	0x03
Acknowledge	0x04
Data	0x05
Query	0x06
Ready	0x07
Location	0x08
Distance	0x09
More Fragments	0x0A
Last Fragment	0x0B

Every packet contains Source IP, Destination IP, Mode, Offset and Data fields. The value in the Mode field is dependent on the packet type. The use cases of each packet type will be detailed in coming sections. The Data field is of variable length and can be omitted entirely in some situations.

4.4 RUSHBSwitch

4.4.1 High-Level Overview

The switch acts like a network router. It can take three forms: local, global and mixed. Local switches open a listening socket on UDP to serve adapters and can create outgoing connections to global and mixed switches over TCP. Global switches open a listening port on TCP to service incoming connections with other switches, and can also create outgoing connections to other global/mixed switches over TCP. Mixed switches open listening sockets on both UDP and TCP for servicing incoming connections from adapters and switches respectively but cannot create outgoing connections to global/mixed switches. By connecting adapters to local and mixed switches, we can simulate a local area network, and by connecting global and mixed switches, we can simulate large scale wide area networks with geographical proximity.
Switches use virtual IP addresses to identify themselves. Local and global switches have one IP address and mixed switches have two IP addresses; one for their local UDP side and one for their global (TCP) side.

4.4.2 Invocation

The switch takes the following commandline arguments in order:

- Its type (local/global)
- Its IP address(es) with CIDR notation
- Its latitude
- Its longitude

where the latitude and longitude are positive nonnegative integers no greater than 32767.

The below commands are used to start a local switch:

```
$ python3 RUSHBSwitch.py local ip_address/cidr latitude longitude
$ ./RUSHBSwitch local ip_address/cidr latitude longitude
```

for example:

```
$ ./RUSHBSwitch local 192.168.0.1/24 50 20
```

The below commands are used to start a mixed switch:

```
$ python3 RUSHBSwitch.py local local_ip_address/cidr global_ip_address/cidr latitude longitude
$ ./RUSHBSwitch local local_ip_address/cidr global_ip_address/cidr latitude longitude
```

for example:

```
$ ./RUSHBSwitch local 192.168.0.1/24 130.102.72.10/24 50 20
```

The below commands are used to start a global switch:

```
$ python3 RUSHBSwitch.py global ip_address/cidr latitude longitude
$ ./RUSHBSwitch global ip_address/cidr latitude longitude
```

for example:

```
$ ./RUSHBSwitch global 130.102.72.10/24 50 20
```

If the incorrect number of arguments are given, or any argument is invalid in any way, the switch will exit immediately.
As soon as the switch process starts it should open the necessary TCP and UDP ports. The ephemeral port should always be used, and the port assigned by the kernel should be immediately displayed to `stdout` on its own line. Mixed switches should display their UDP port first.

4.4.3 Commandline Interface

After opening the required ports, local and global (but not mixed) switches will be able to create outgoing connections to other switches. This is done by typing the below command into `stdin`:

```
connect <port>
```

where `<port>` is the TCP port of the global/mixed switch to connect to.
If any arguments are missing or invalid, the switch will ignore the command and reprompt the user.
Mixed switches do not accept this command. They should still accept input from `stdin` but should never do anything with it.

4.4.4 Greeting Protocol

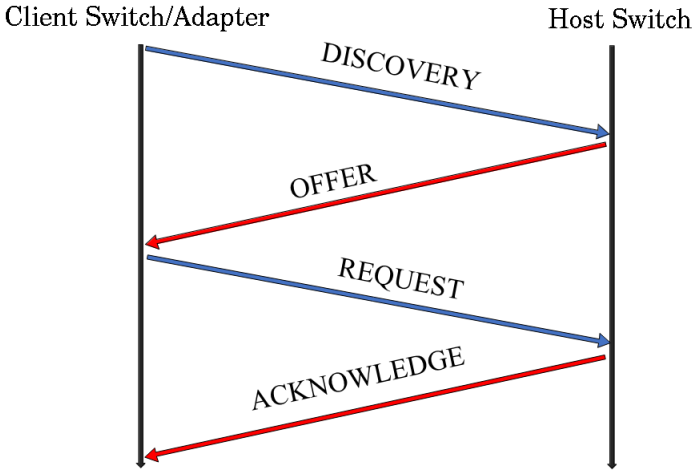
If the switch is able to connect to the port given in the connect command, it (referred to as the client switch) will engage in the Greeting Protocol with the switch it connected to (referred to as the host switch). In the context of the Greeting Protocol, the Data field is always 4B long and is referred to as the Assigned IP field. The protocol works as follows:

- The client switch sends the host switch a Discovery packet (Mode = 0x01). The Source IP, Destination IP, Assigned IP and Offset fields are left at 0.
- The host switch sends the client switch an Offer packet (Mode = 0x02). The Source IP field is set to the global IP of the host switch and the Assigned IP field is set to the IP address the host switch wishes to allocate to the client (further details in Section 4.4.6). The Destination IP and Offset fields should still remain as 0.
- The client switch sends the host switch a Request packet (Mode = 0x03). The Source IP and Offset fields are set to 0. The Destination IP address is set to the global IP of the host switch and the Assigned IP field is set to the IP offered by the host switch in the previous step.
- The host switch sends the client switch an Acknowledgment packet (Mode = 0x04). The Source IP field is set to the global IP of the host switch. The Destination IP and Assigned IP fields are set to the IP offered by the host switch. The Offset field is set to 0.

4.4.5 Connection with Adapters

Adapters will connect to local/mixed switches over UDP upon startup. The exact same protocol as above is followed, except that the client switch is replaced by an adapter and the host switch's local IP address is used in the relevant address fields and in calculating the IP address to allocate to the adapter.

These processes are illustrated in the below timing diagram:



N.B.: Client switches and adapters will always accept the offered address from the host switch. If the host switch cannot allocate an IP address to the adapter/client switch, it will not respond to the initial Discovery packet. In this case, and in any other case where a packet is lost, the Greeting Protocol will not progress past the initial Discovery packet. Client switches will not be connected to the host switch and adapters will hang indefinitely.

Below are depicted some example packets you might see as part of a Greeting Protocol:

0.0.0.0	
0.0.0.0	
0x000000	0x01
0.0.0.0	

Discovery

130.102.72.1	
0.0.0.0	
0x000000	0x02
130.102.72.8	

Offer

0.0.0.0	
130.102.72.1	
0x000000	0x03
130.102.72.8	

Request

130.102.72.1	
130.102.72.8	
0x000000	0x04
130.102.72.8	

Acknowledge

4.4.6 IP Address Allocation

The IP address allocated to the adapter/client switch by the host switch during the Greeting protocol is calculated in accordance with RFCs 1518 and 1519. The host switch will pick the smallest available IP to allocate to each incoming connection. For example, if an adapter were to connect to a mixed switch with the local IP 192.168.0.1/24 and that switch already had two other adapters connected to it, then this new adapter would be allocated the IP address 192.168.0.4 (as the first two adapters take 192.168.0.2 and 192.168.0.3 respectively). Similarly, if a global switch were to connect to a mixed switch with the global IP 130.102.72.01/24 and that switch already had seven other switches connected to it, then this new switch would be allocated the IP address 130.102.72.8. Both of these switches can support a maximum of 254 connections due to its CIDR of 24. If all connections are taken, then the switch will stop responding to incoming connections.

4.4.7 Location Exchange

As soon as the Greeting Protocol is finished, the client switch will send the host switch a Location packet (Mode = 0x08). The Source IP field will be the allocated IP address of the client switch; the Destination IP field will be the global IP address of the host switch and the Offset field will be 0. The first two bytes of the Data field will be the latitude of the client switch and the second two bytes of the Data field will be the longitude of the switch. The Data field must be four bytes long.

The host switch will then reply to the client with a Location packet. The Source IP field will contain the global IP address of the host switch; the Destination IP field will contain the allocated IP address of the client switch and the Offset field will be 0. The first two bytes of the Data field will contain the latitude of the host switch and the second two bytes of the Data field will contain the longitude of the host switch. Again, the Data field must be four bytes long.

Below are depicted two packets you may see as part of a location exchange:

130.102.72.8	
130.102.78.1	
0x000000	0x08
0x04D2	0x11D7

Latitude 1234
Longitude 4567

130.102.78.1	
130.102.78.8	
0x000000	0x08
0x0539	0x01A4

Latitude 1337
Longitude 420

4.4.8 Distance Relaying

Whenever a switch receives a Location packet, it will inform all other neighbouring switches of the distance (rounded down) from the new switch to the respective neighbour on the path going through through the switch which received the location packet.
The switch will send a Distance packet (mode = 0x09) to every connected switch except that which sent it the Location packet. The Source IP field will contain the global IP address of the switch which received the Location packet; the Destination IP field will contain the global or assigned IP (whichever applicable) of the neighbour which the packet will be sent to and the Offset field will be 0. The first four bytes of the Data field will contain the assigned IP of the switch which sent the Location packet and the second four bytes will contain the distance from the switch which sent the Location packet to the neighbouring switch specified in the Destination IP field. This distance is equal to the length of the shortest path from the switch specified in the Data field to the sending switch field plus the Euclidean distance between the sending and receiving switches.

Switches must maintain a record of the length of the shortest path to every other known switch. When a switch receives a Distance packet, if the distance to the switch specified in the Data field is less than its current record, then that record will be updated to the new shortest distance. That switch will then also send a Distance packet to each neighbouring switch except those specified in the Source IP and Data fields. The Data field will contain the same target IP, but will contain the distance from the target IP to the respective neighbour (i.e. the original distance plus the distance to the respective neighbour).
If the distance specified in the packet is greater than or equal to the existing distance record, or if the distance is greater than 1000, the switch will do nothing.

Below is depicted a packet you might see as part of a distance relay after the location exchange depicted in the previous section:

130.102.72.1	
130.102.78.2	
0x000000	0x09
130.102.72.8	
0x00001034	

Distance 4148

4.4.9 Data Forwarding

Data is sent from adapters to switches and other adapters in Data packets (Mode = 0x05). Upon receiving a Data packet, the switch will have to forward it to other switches/adapters until it reaches the destination specified in the Destination IP field. These conditions must be followed when deciding which connection to forward the packet to:

- If the packet is intended for an adapter which the switch is connected to, it will forward the packet to that adapter.
- If the switch is aware of the existence destination IP address, it should forward the packet to whichever connection is on the shortest geographical path to the destination.
- If two or more neighbouring switches are on paths of the same shortest length to the destination, the switch amongst these with the longest matching prefix of the destination IP address should receive the packet.
- If the switch is unaware of the existence of the destination IP address, it should forward the packet to whichever of its neighbouring connections has the IP address with the longest matching prefix with the destination IP address.

4.4.10 Transmitting Data to Adapters

If a local/mixed switch receives data to transmit to a connected adapter, it must first send the adapter a Query packet (Mode = 0x06). The Source IP field contains the local IP of the switch; the Destination IP field contains the IP address of the adapter and the Offset field is 0. The Data field is omitted entirely.

When an adapter receives the Query packet it will respond with a Ready packet (Mode = 0x07). The Source IP field contains the IP address of the adapter and the Destination IP field contains the local IP address of the switch. Again, the Offset field is 0 and the Data field is omitted entirely.

Once the switch has received the Ready packet, it will transmit the data to the adapter using Data packets (Mode = 0x05). The Source IP field contains the IP address of the adapter which originally sent the data. The Destination IP field contains the IP address of the receiving adapter and the Data field contains the data.

If the switch receives more data to forward to the adapter, it can forward it to the adapter immediately so long as it has not been 5 seconds since the last Ready packet was received. If it has been more than 5 seconds, the adapter must be sent another Query packet and the switch must await another Ready packet before forwarding the data to the adapter.

Below is depicted some packets which may be seen as part of an adapter query:

130.102.72.2	
130.102.78.8	
0x000000	0x05
Hello World	

Data packet received by switch 130.102.72.1 for adapter 130.102.72.2

130.102.72.1	
130.102.78.2	
0x000000	0x06

Switch 130.102.72.1 sends Query to 130.102.72.2

130.102.72.2	
130.102.78.1	
0x000000	0x07

Adapter replies Ready to the switch

130.102.72.2	
130.102.78.8	
0x000B88	0x05
Hello World	

Switch forwards the original Data packet to the adapter

4.4.11 Data Fragmentation

In the event that a switch receives a Data packet with a total size greater than 1500B (including the header), and the switch is not the intended recipient of the data, it must fragment the data across multiple packets such that no packet exceeds a total size of 1500B (including the twelve byte header). Each fragment except the last will have the mode 0x0A (More Fragments) and the last fragment will have the mode 0x0B (Last Fragment). The Offset field of each fragment will take the value of the offset within the original data where that fragment's data begins. Each More Fragments packet must contain as much of the data payload as possible. The Source IP and Destination IP fields will contain the addresses of the original packet. These fragments can then be forwarded instead of Data packets as per Sections 4.4.9 and 4.4.10 to the required neighbouring switch or adapter.

For example, if a switch were to receive a packet with 3000B of data, then it would forward two More Fragments packets with 1488B of data each and the Offset values 0 and 1488 respectively. Following this it would send a Last Fragment packet with 24B of data and an offset of 2976.

More Fragments and Last Fragment packets are able to be forwarded the same as Data packets in Section 4.4.9.

Some fragments you may see in the example described above are depicted below:

130.102.72.3	
130.102.78.8	
0x000000	0x05
A x 3000	

Original Data packet received from 130.102.72.3 en route to 130.102.78.8

130.102.72.3	
130.102.78.8	
0x000000	0x0A
A x 1488	

First fragment to forward to 130.102.72.8

130.102.72.3	
130.102.78.8	
0x0005C4	0x0A
A x 1488	

Second fragment to forward to 130.102.72.8

130.102.72.3	
130.102.78.8	
0x000B88	0x0B
A x 24	

Last fragment to forward to 130.102.72.8

4.4.12 Displaying Received Data

If a switch receives Data, More Fragments or Last Fragment packets intended for it (i.e. the Destination IP field is equal to their global or local IP), the below is displayed to the switch's `stdout`:

```
Received from <src_addr>: <data>
```

where `<src_addr>` is given by the Source IP field of the packet and `<data>` is given by the Data field of the packet. If the data in question arrives in several fragments, the above message is displayed only after all fragments are received and the original data has been reassembled.

4.4.13 Miscellaneous Notes

- The switch will ignore any packets with an invalid Mode field. The switch will never receive any packets which are malformed in other ways unless your code generates them.
- If the switch receives a Data packet or Fragment intended for a nonexistent switch or adapter, it will still forward it based on the rules in Section 4.4.9.
- Switches and adapters will never exit, and so you do not have to handle receiving `EOF` over a socket.
- If a switch receives `EOF` from `stdin`, it should stop handling commands but should still process incoming packets and will not exit.
- We will assume that all switches will have a different IP address and CIDR such that no two switches or adapters will have the same IP address.

4.5 RUSHBAdapter

4.5.1 High-level Overview

The adapter connects to one local switch process over UDP. It takes input from `stdin` to transmit from the switch and displays data received from the switch to `stdout`.
You are not required to implement the adapter, and it will be provided to you.

4.5.2 Invocation

The adapter is started using the below command:

```
$ python3 RUSHBAdapter.py port
```

where `port` is the UDP port number of the local/mixed switch it will connect to.

4.5.3 Greeting Protocol and IP Address Allocation

As soon as the adapter is able to open a connection with the desired switch, it will engage in the same Greeting protocol as described in Section 4.4.5. The IP address the switch will allocate to it is calculated in accordance with Section 4.4.7.

4.5.4 Commandline Interface and Sending Data

After finishing the Greeting Protocol, the adapter will be able to send data to switches and other adapters. This is done by typing the below command into `stdin`:

```
send <ip_address> <message>
```

where `<ip_address>` is the IP address of the adapter to send the string `<message>` to. Everything after `<ip_address>` is considered as part of `<message>`.
The adapter will send this data in a Data packet to its switch. The Source IP field will contain the IP address allocated to the adapter by the switch and the Destination IP field will contain the IP address specified in the command. The Offset field will be 0 and the Data field will contain the message specified in the command.
If too few arguments are given, or the IP address is malformed in any way, then the adapter will ignore the command and send nothing to its switch.
Adapters will always send their switch all the data given by the command in one packet. There is no maximum packet size and it will not do fragmentation.

4.5.5 Querying for Data Receipt

A switch which intends to forward data to a connected adapter will first query that adapter if it is ready to receive data. This process is detailed in Section 4.4.10.

4.5.6 Displaying Transmission Results

When an adapter receives a Data packet from a switch, or has received all fragments which comprise a fragmented Data packet, it will display the below to `stdout`:

```
Received from <src_ip>: <message>
```

where `<src_ip>` is the IP address of the adapter which sent this adapter the string `<message>`.

4.5.7 Miscellaneous Notes

- The adapter will ignore any packets with an unrecognised Mode field.
- Once a switch has received a Ready packet from an adapter, it will not have to send another Query packet to that adapter until 5 seconds elapse.
- No extra packets except those described previously should be sent between adapters and switches.
- Threadsafety is crucial to the correct operation of the switch.

4.6 Miscellaneous Requirements

- Your code must be runnable on Moss.
- No third party libraries are permitted.
- If you choose to use C, you must also submit a Makefile. Your code must compile with the `$ make` command, and must compile to the C99 standard. Code which does not compile will receive a mark of 0.
- All your files (.py, .c, .h, makefile) must be within the same subdirectory.
- The `pipe()`, `select()`, `poll()` and `epoll()` C syscalls are expressly forbidden.
- The Python `pipes` and `select` modules are expressly forbidden.
- The packets sent by your switch must match the specified structures exactly.

4.7 Miscellaneous Notes

- It is highly recommended you use C for this assignment.
- All packets follow Nework Byte Order, which is always big endian. This will almost definitely be opposite to your machine's endianness, and is different to Moss' endianness.
- Style is not assessed but it is still highly recommended that you follow some consistent style guide (e.g. KNF, PEP 8 etc).

4.8 Marking Breakdown

Marks will be awarded according to the following breakdown:

Validating commandline arguments	3 marks
Greeting Protocol	9 marks
Location Exchange and Distance Relaying	8 marks
Data Forwarding	7 marks
Switch Routing	8 marks
Complete Communication	15 marks

Total: 50 marks.

5 Integrity

All code submissions will be checked for similarity and plagiarism. All code in your submission which was not written by you (or was written by you for a previous assignment) must be correctly referenced in IEEE format in your code. This includes code written by generative AI.

6 Submission

You must submit all source code files in a zip file. This zip file must not contain subdirectories. This zip file must be named `A2_sxxxxxxx.zip`, where the `sxxxxxxx` is your student ID. This zip file must be submitted to the course Blackboard page under Assessment $\implies$ Assignment 2 $\implies$ Part C: Code Submission.

7 Resources

These resources may be of use to you:

- Joyce, P. (2022). Sockets. In: C and Python Applications. Apress, Berkeley, CA. https://doi-org.ezproxy.library.uq.edu.au/10.1007/978-1-4842-7774-4_6
- Socket Programming HOWTO, available at: https://docs.python.org/3/howto/sockets.html
- Socket Programming with Multi-threading in Python, https://www.tutorialspoint.com/socket-programming-with-multi-threading-in-python
- The Linux Manual Pages