The University of Queensland

School of Information Technology and Electrical Engineering

Semester One, 2022

COMS3200 - Assignment 2

Due: 3:00 pm 1st June 2022

Marks: 100

Weighting: 25% of your final grade

Revision: 1.1

Introduction

This assignment introduces the use of Network Layer and Link Layer (from OSI Models) and socket programming (simulation) on each of these devices: adapters, switches, and routers. After finishing the assignment, you will understand to send data on the Internet.

This assignment contains three parts. You will need to submit your answers through the Blackboard quiz for parts A and B. You will submit your code via COMS3200 subversion (SVN) for Part C. All submitted work will be marked automatically. We recommend that you read this spec thoroughly.

This assignment is individual work. Using answers (or code) that you did not calculate (or write) is against course rules and may lead to a misconduct charge.

Part A (21 marks)

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

You have established a new server for COMS3200 Inc, called RUSHBSvr (in assignment 1). It works great and has received many positive feedbacks. You, a supervisor of this project, were assigned to design two new networks, a branch in Sydney and a local system for working from home. You also moved the File Server to "somewhere" on the Internet to serve as a centralised data centre for your company. The network map of your final design is in Figure 1 on the next page.

You set up only one DNS Server (D2) and one DCHP Server (D1) to minimise cost. You claimed that the system would run efficiently without requiring additional infrastructure.

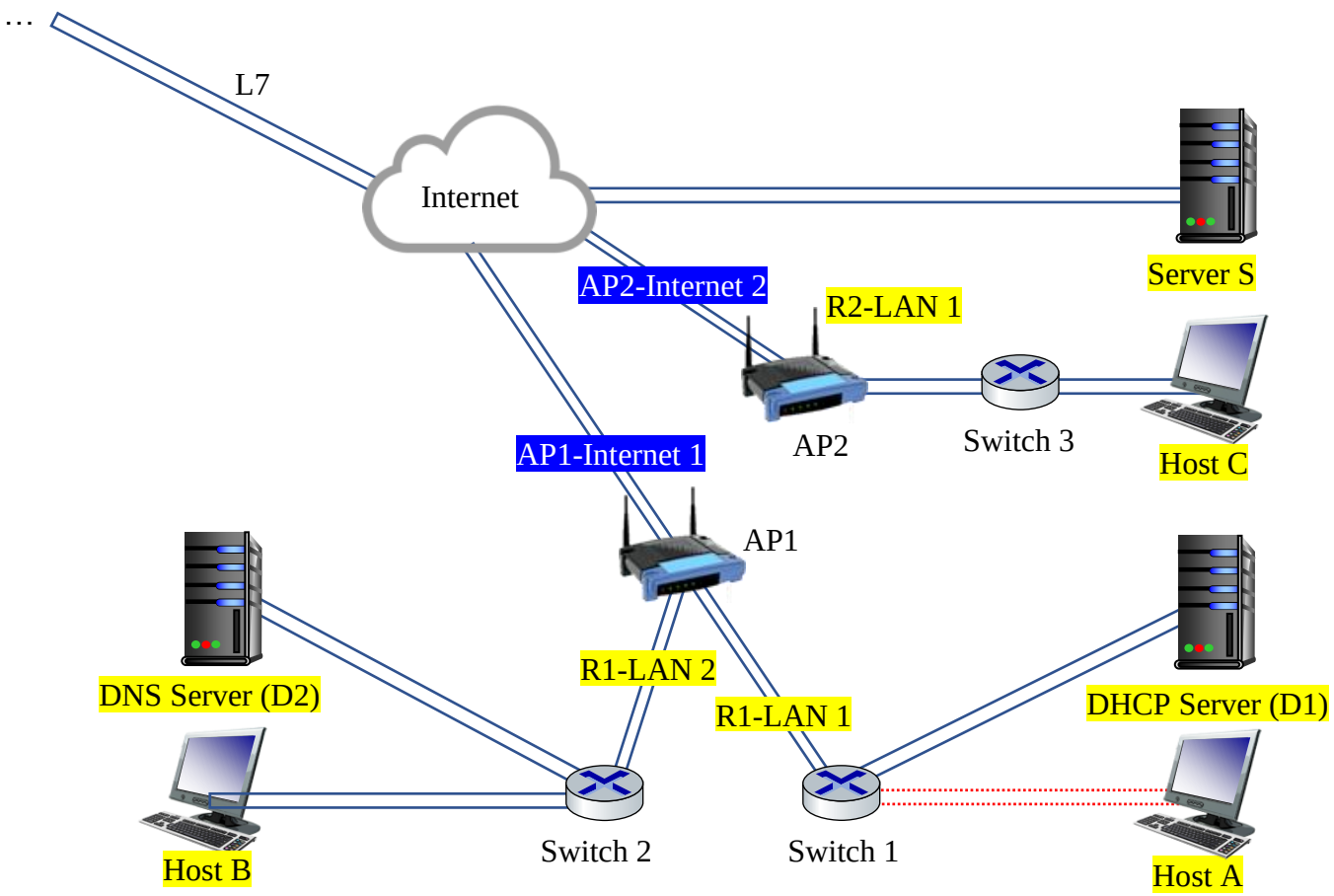


Figure 1. Map of the designed network

You must explain to your colleagues how your system works with some scenarios given below. You can use these assumptions to support your answers:

- The broadcast MAC address of all the LANs is FF:FF:FF:FF:FF: FF
- All forwarding tables in the switches and routers are up to date.
- All valid protocols include HTTP, FTP, SSH, DNS, DHCP, UDP, TCP, ICMP, SNMP, ARP, OSPF, and BGP. If there are two protocols used simultaneously (e.g., SSH and TCP), choose only the highest layer (e.g., SSH).
- D2 is a DNS Server, and D1 is a DHCP Server.

You must submit your answers in the tables and response boxes on the Blackboard quiz. Each student will have different values in the quiz (which replaces the XXX… in this spec).

Please be careful to check before submitting your work on Blackboard thoroughly. You will have one submission only, and we will not clear your attempt once submitted.

Task: Answer the questions below on Blackboard.

Question 1: Host A just joined the network and currently does not have an IP address. Please complete the following table describing the four packets transmitted through the network upon this event (in the order they were sent), assuming that all requests succeed and Host A ends up with IP address **XXX.XXX.XXX.XXX**.

Protocol	Opcode	Source MAC address	Destination MAC address	Source (or sender) IPv4 address	Destination (or target) IPv4 address

Question 2: After ending up with the IP address above, Host A wants to send a standard DNS query to the DNS Server. Host A has already gotten the IPv4 addresses of the DNS Server and its gateway router from the DHCP Server. Router 1's (AP1) ARP table contains all hosts' IP and MAC addresses in the network, and all other ARP tables are empty. Complete the following table describing the first four packets transmitted through the network upon this event, with no additional ongoing communications.

Protocol	Opcode	Source MAC address	Destination MAC address	Source (or sender) IPv4 address	Destination (or target) IPv4 address

Question 3: Host A now wants to establish an HTTP connection with Server S (see figure 1 above). You may assume that Host A knows Server 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 **XXXX** and the following available port number in Router 1's NAT table is **XXXX**. Give the TCP flags as an 8-bit binary number representing the CWR to FIN flags.

Protocol	TCP Flag	Source TCP Port	Destination TCP Port	Source (or sender) IPv4 address	Destination (or target) IPv4 address
TCP					
TCP					

Question 4: In addition to the connection described in question 3, Host B's port **XXXX** 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, use port **XXXX**).

LAN IP	LAN Port	WAN IP	WAN Port

Question 5: Suppose **X** sent a packet with the destination IP address **XXX.XXX.XXX.XXX** and destination MAC address **XX:XX:XX:XX:XX:XX**. What network devices (hosts, servers, routers, switches) in the above diagram would receive this packet? We assume that any message with the IP address of 255.255.255.0 and the MAC address of FF:FF:FF:FF:FF:FF is the broadcast message.

Question 6: Server S sends a packet with the destination IP address **XXX.XXX.XXX.XXX**. What network devices (hosts, servers, routers, switches) in the above diagram would receive this packet?

Part B (29 marks)

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

In this part, you will need to use Wireshark to analyse a packet capture file named **a2.pcap**. This packet capture shows a client connecting to a LAN network; then, a 'traceroute' command is run from the client. Some of the relevant IETF RFCs may help you understand the following questions. In particular:

1. RFC 2131 for Dynamic Host Configuration Protocol (DHCP)
2. RFC 791 for Internet Protocol (IP)
3. RFC 1034 for Domain Name System (DNS)
4. RFC 6747 for Address Resolution Protocol in IPv4 (ARP)

This is a real-life scenario Wireshark trace and has been updated to the newest protocol. We recommend you ask any questions you may have for this trace besides the given questions to understand better how LAN networking works. You need to be familiar with using the Wireshark filter feature to answer these questions.

Task: Answer the questions below on Blackboard.

Question 1: What is the assigned IP address from the DHCP Server to the client? (2 marks)
Question 2: What is the MAC address of the client? (2 marks)
Question 3: Is the MAC address of the client globally unique? (1 mark)
Question 4: Is the MAC address of the DHCP Server globally unique? (1 mark)
Question 5: What is the netmask of this LAN network? (2 marks)
Question 6: What is the DHCP Server's IP address? (1 mark)
Question 7: What is the MAC address of the DHCP Server? (1 mark)
Question 8: Is this LAN network a subset of another LAN network? (2 marks)
Question 9: What is the hostname address of the traceroute command? (2 marks)
Question 10: What is the target IP address of the traceroute command? (2 marks)
Question 11: Did the traceroute run successfully, or give up while waiting for a response? (2 marks)
Question 12: How many routers are between the target server and the host (in traceroute)? (3 marks)
Question 13: How many times did the host check if the client is still alive in the network? (2 marks)
Question 14: Did the client lose its leased IP address, and does it need a new one? (3 marks)
Question 15: How many data packets are fragmented? (3 marks)

Hints:

- For each TTL, the traceroute application sends the packet three times
- The DHCP Server is attached to another device
- Please use the Wireshark filter where appropriate

Part C (50 marks)

Your company is doing well thanks to your RUSHBSvr. Due to a network upgrade in Sydney, your superiors want a new protocol because they feel the current protocols are too convoluted. As seen in figure 1, the RUSHBSvr is now located somewhere else on the Internet, and you need to set up the required protocols to exchange data with the server. There are two virtual protocols that we need to set up, RUSHBAdapter on the link layer and RUSHBSwitch on the network/transport layer. Virtual means that it simulates those protocols in the application layer; network layer addresses will not correspond to physical interfaces. **In this assignment, we will provide a sample RUSHBAdapter, and you need to submit your RUSHBSwitch into your subversion. Tasks with yellow labelled are directly related to your RUSHBSwitch, but we strongly recommend you understand RUSHBAdapter.**

Again, this is an individual assignment. You can feel free to discuss aspects of socket programming and the specification with fellow students. Directly helping (or seeking help from) other students with the actual coding of your solution is considered cheating (even if it's in printed or in electronic form). If you're having trouble, please seek help from a teaching staff member. Don't be tempted to copy another student's code, and don't commit any code to your repository unless it's your work.

Simulation

There are two programs in this part, RUSHBAdapter and RUSHBSwitch, representing the link and network layer RUSHB protocols, respectively. You have already implemented the application layer RUSHB protocol, RUSHBSvr. Since this is a simulated network, they are all implemented in the application layer of the OSI networking model in practice (see figure 2).

<table>
<tr><th colspan="2">OSI MODEL</th><th colspan="2">RUSHB MODEL (SIMULATION)</th></tr>
<tr><td>LINK LAYER</td><td>Ethernet</td><td rowspan="3">LINK LAYER &
NETWORK LAYER</td><td rowspan="2">RUSHBSwitch</td></tr>
<tr><td>NETWORK LAYER</td><td>IP</td></tr>
<tr><td>TRANSPORT LAYER</td><td>UDP/TCP</td><td rowspan="2">RUSHBAdapter</td></tr>
<tr><td rowspan="2">APPLICATION LAYER</td><td rowspan="2">HTTP/…</td><td>APPLICATION LAYER</td><td>RUSHBSvr</td></tr>
</table>

Figure 2. Network layers mapping

RUSHBAdapter and RUSHBSwitch

RUSHBAdapter is supposed to work as an adapter for one process only through TCP (e.g. RUSHBSvr). The process that connects to RUSHBAdapter needs to open a socket under localhost (127.0.0.1) as a listener under an available port (assigned by the kernel). However, in this assignment, RUSHBAdapter is not required to listen to any processes, instead, it will listen to your stdin (see the figure below).

RUSHBSwitch works like a network router, and it can be local or global. A local RUSHBSwitch can listen to many RUSHBAdapter through a UDP and connect to many global RUSHBSwitches through TCP. In the meantime, the global RUSHBSwitch cannot listen to any RUSHBAdapter but can connect to other RUSHBSwitches. This is a diagram explaining how RUSHBAdapters and RUSHBSwitches are connected:

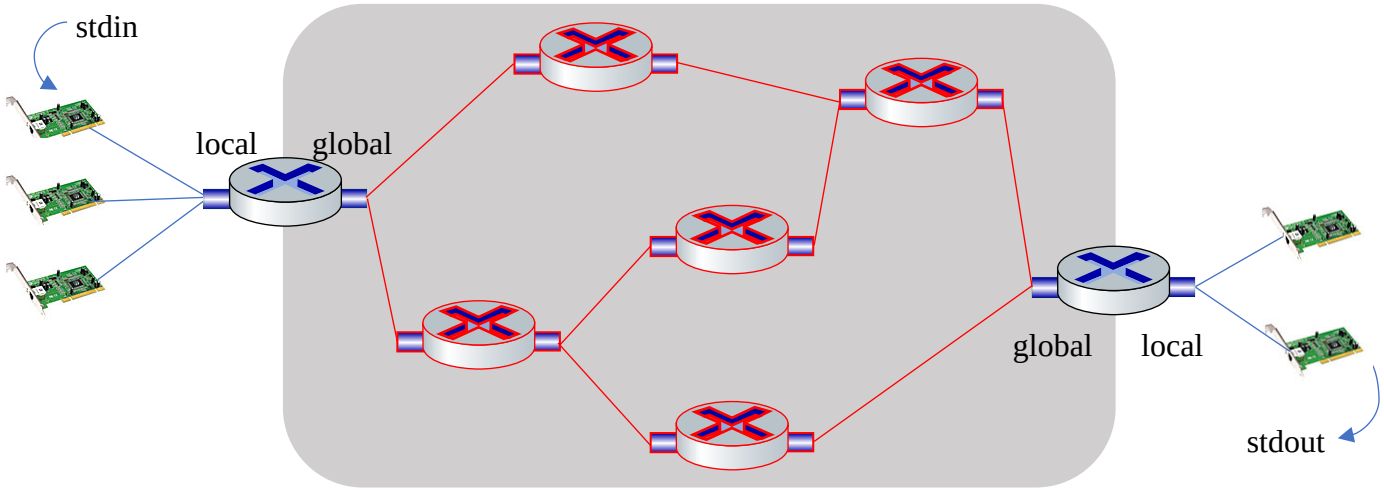


Figure 3. End-to-end connection in RUSHB

The final goal for this programming assignment is to send and receive data across the global network without losses and errors. Data can be anything attached to the adapter, such as netcat, RUSHBSvr, or even stdin (as in this assignment). We refer to calling RUSHBAdapter as the adapter and RUSHBSwitch as the switch.

[Adapter Invocation]

The adapter takes the following command-line arguments (in order):

- the port number of the switch that will connect to in UDP protocol

For example, if the switch is listening on a UDP port 5678, then the adapter runs as:

Python	`python3 RUSHBAdapter.py 5678`
Java	`java RUSHBAdapter 5678`
C or C++	`./RUSHBAdapter 5678`

[Greeting Protocol]

When first executing, the adapter will make "a greeting protocol" to the switch through a UDP port number to get its local IP address; each greeting packet is in the structure below. We will define each block of a packet as 4 bytes (32 bits).

BIT	**0**	**8**	**16**	**24**
	SOURCE IP ADDRESS			
	DESTINATION IP ADDRESS			
	RESERVED (all 0s)			MODE
	ASSIGNED IP ADDRESS			

Figure 4. Greeting structure in RUSHB

MODE = 0x01:	DISCOVERY
MODE = 0x02:	OFFER
MODE = 0x03:	REQUEST
MODE = 0x04:	ACKNOWLEDGE

Figure 5. MODE instruction in RUSHB

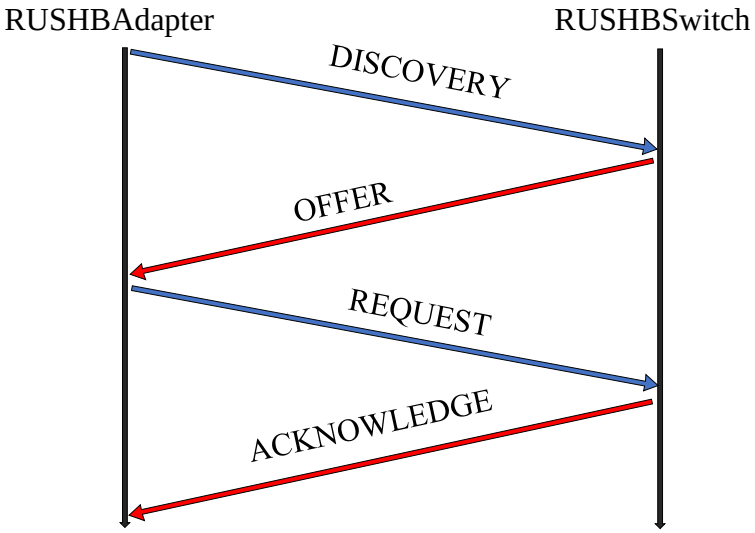


Figure 6. Then greeting process between RUSHBAdapter and RUSHBSwitch

In figure 6, we consider RUSHBSwitch as a host and RUSHBAdapter as a client. The greeting protocol works as follows:

1. In a DISCOVERY message, since the sender has not to know its IP address in the host network yet, then SOURCE IP ADDRESS and ASSIGNED IP ADDRESS should be left at 0x00000000 (let the host fill out the assigned IP address).
2. The OFFER from the host contains the ASSIGNED IP ADDRESS, and the SOURCE IP ADDRESS is the address of the switch itself, but the DESTINATION IP ADDRESS should be left at 0x00000000 (since the client has not accepted the offer yet).
3. After receiving the offer from the host, the adapter sends back the request for the assigned IP address. The REQUEST should contain the value of DESTINATION IP ADDRESS and ASSIGNED IP ADDRESS, while SOURCE IP ADDRESS should be left at 0x00000000.
4. The host sends back an acknowledgment message with all fields filled.

The client must send a discovery message when first connected to the switch, and the host must send back the assigned IP address if there are spaces for the client to join the network. After receiving an acknowledgment from the host, the client should remember its IP address and the router IP address for the upcoming event. The greeting protocol also is applied for two switches. For instance, in figure 3, if you choose two adjacent switches, the one on the left side is a client, and the one on the right side is a host.

[IP Address Allocation]

Your assigned IP addresses with CIDR notation and IP addressing allocation in the subnet must follow the guideline of RFC 1518 and RFC 1519. To be clear, CIDR notation must be valid with the associated IP address. For example, with a range of 192.168.0.1/24, the switch IP address is 192.168.0.1, the following available IP address in this subnet should be 192.168.0.2, and the total hosts can be accepted in this subnet is 254. If more than 254 hosts are connected to your switch, your switch should ignore the incoming connections and not give the [Greeting Protocol] stated on page 7.

[Adapter Commandline Interface]

After finishing the greeting protocol, the adapter prints (to stdout) a character ">" followed by a white space, to tell the user that it is ready for user commands.

The commandline interface for the adapter will only accept the command ”send” with usage:

```
> send {receiver_ip_address} {message}
```

Where {receiver_ip_address} is the IP address of the receiver and {message} is the message that you want to send (separated with white space). For example, if you're going to send "HELLO_WORLD" to the IP address 130.102.71.65, then you would use the command:

```
> send 130.102.71.65 "HELLO_WORLD"
```

To process the user command, the adapter will build a packet with the structure as in figure 7. For example, your adapter was assigned with IP 192.168.0.2 from the host (from 192.168.0.1) and sends "HELLO_WORLD" to 130.102.71.64, your adapter sends to 192.168.0.1 with MODE 0x05 in figure 8 below.

BIT	0	8	16	24
	SOURCE IP ADDRESS			
	DESTINATION IP ADDRESS			
	RESERVED (all 0s)			MODE
	DATA			

Figure 7. Data structure in RUSHB

BIT	0	8	16	24
	192.168.0.2			
	130.102.71.64			
	0x000000			0x05
	HELLO_WORLD			

Figure 8. An example of data packet

Please remember that the network byte order is always defined to be big-endian, and it may differ from your host. After processing a command from a user, your adapter should print out the character ">" followed by a white space again for the following commands.

When your adapter receives an EOF (end of file) from any file or socket descriptors, it should do nothing, not accept input anymore, and works its jobs usually instead of an early exit.

[Data Communication]

The adapter should also be able to receive the packet from its connected switch (in the network). The switch will ask if your adapter is available to receive the packets before sending them. The querying packet will be sent with the structure in figure 9 with MODE 0x06. For example, the packet will be sent as in figure 10.

BIT	0	8	16	24
	SOURCE IP ADDRESS			
	DESTINATION IP ADDRESS			
	RESERVED (all 0s)			MODE

Figure 9. Querying structure in RUSHB

BIT	0	8	16	24
	192.168.0.1			
	192.168.0.2			
	0x000000			0x06

Figure 10. An example of query packet

Your adapter (or client) will respond with MODE 0x07 to tell the server that it is ready to receive the packets (see figure 11 and 12). This is also applied to the transmission between 2 switches, the sender switch will send the packet with MODE 0x06, and the receiver switch will reply with MODE 0x07. However, when an adapter sends a message to a switch, it will not need to ask if it is available.

BIT	0	8	16	24
	SOURCE IP ADDRESS			
	DESTINATION IP ADDRESS			
	RESERVED (all 0s)			MODE

Figure 11. Response structure in RUSHB

BIT	0	8	16	24
	192.168.0.2			
	192.168.0.1			
	0x000000			0x07

Figure 12. An example of response packet

Then, the sender will send packets to the client with MODE 0x05. The receiver needs to print out the data it received. For example, if a host from 130.102.71.64 sends "HELLO_WORLD" after the query protocols are established, the sender will send the packet as figure 11. Please note that even though the switch that sends the message to the adapter has an IP address of 192.168.0.1, the source IP address in the packet is still 130.102.71.64, as it is the origin of the packet.

BIT	0	8	16	24
	130.102.71.64			
	192.168.0.2			
	0x000000			0x05
	HELLO_WORLD			

Figure 13. An example of data packet from a global host

When you receive the data packet, your adapter should print out to **stdout** in this form:

```
Received from {source_ip}: {data}\n
```

For example, with the packet above, the adapter will print this out with a newline:

```
Received from 130.102.71.65: HELLO_WORLD
```

There are a few things you need to remember for sending and receiving packets (for both switches and adapters):

1. The program should remove the characters ">" before printing out the received message; then, it should print the characters ">" again for the following user commands.
2. We send and receive messages through the port number in the localhost (127.0.0.1), assuming that the port number is in the link layer of RUSHB Models.
3. Your switch won't have to send the checking packets (mode 0x06 and 0x07) again if the checking protocol is already done within 5 seconds. After 5 seconds of the checking protocols, if there is an upcoming packet, your switch must establish the checking protocol again.
4. If any packets are not in the given protocols or invalid, your adapter and switch should ignore them. Mode 0x06 and 0x07 are available for sending data packets (mode 0x05, 0x0a and 0x0b) only.
5. **We provided a solution for RUSHBAdapter, but you need to understand the protocols between an adapter and a switch for implementing RUSHBSwitch in this assignment.**

[Switch Invocation]

The switch will take the following command-line arguments (in order):

- the type that the switch will be (either local or global)
- the IP address with the CIDR notation indicating its subset
- (optional for local) the second IP address with the CIDR notation indicating its subset
- the latitude of the switch in the map
- the longitude of the switch in the map

We assume latitude and longitude are positive integers, representing the coordinates on a cartesian plane, where latitude is the horizontal x-axis and longitude is the vertical y-axis.

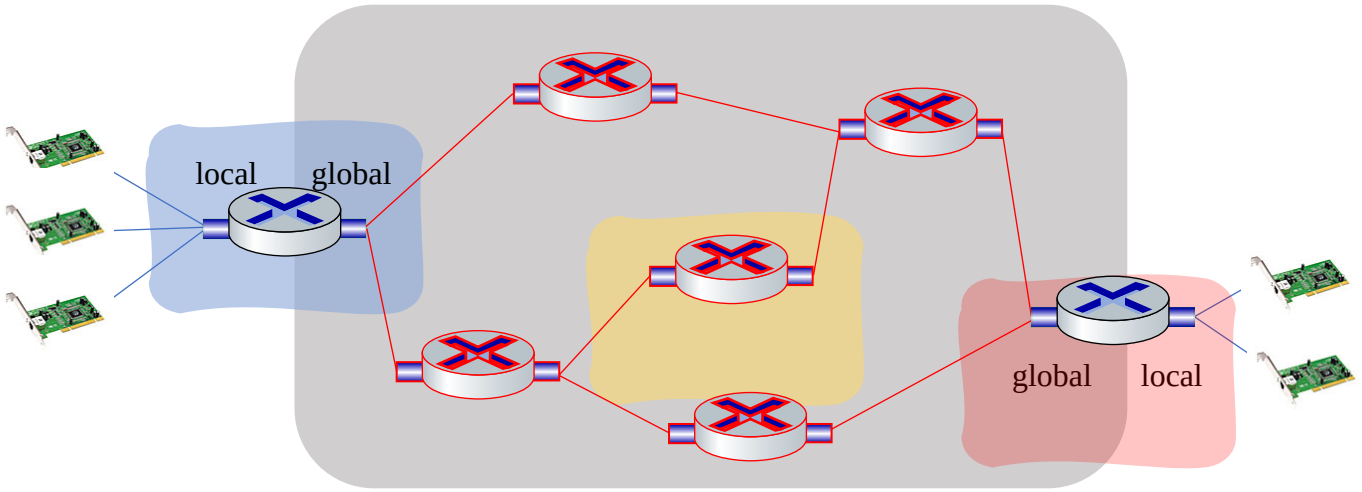


Figure 14. End-to-end connection with the areas

The switch invocation will have three modes:

1. If you want to run the switch in local mode that initially has one listening socket on a UDP port for adapters (the blue area in figure 14), with the LAN IP address is 192.168.0.1/24, and at the location is (2, 5), then the execution is:

Python	`python3 RUSHBSwitch.py local 192.168.0.1/24 2 5`
Java	`java RUSHBSwitch local 192.168.0.1/24 2 5`
C or C++	`./RUSHBSwitch local 192.168.0.1/24 2 5`

2. If you want to run the switch in local mode with two listening sockets, one is for the local adapters through a UDP port, and another one in a global area that listens in the TCP port for other switches (the red zone in figure 14), with the LAN IP address, is 192.168.0.1/24, the global IP address is 44.45.46.0/8, and at the location is (140, 26), then the execution is:

Python	`python3 RUSHBSwitch.py local 192.168.0.1/24 44.46.46.0/8 140 26`
Java	`java RUSHBSwitch local 192.168.0.1/24 44.46.46.0/8 140 26`
C or C++	`./RUSHBSwitch local 192.168.0.1/24 44.46.46.0/8 140 26`

3. If you want to run the switch in global mode (the yellow area in figure 14), with the global IP address being 172.64.2.1/16 and the location is (465, 784), then the execution is:

Python	`python3 RUSHBSwitch.py global 172.64.2.1/16 465 784`
Java	`java RUSHBSwitch global 172.64.2.1/16 465 784`
C or C++	`./RUSHBSwitch global 172.64.2.1/16 465 784`

Your program should print out the port number to stdout with a new line. The switch listening in both TCP and UDP ports should print port on UDP first, and then TCP. It should also print out (to stdout) the character">" followed by white space to tell the user that the adapter is ready for user commands. There are some examples of the switch invocation:

1. The switch is local and listening on a UDP port; the kernel gives it port number 4455:
```
python3 RUSHBSwitch.py local 192.168.0.1/24 2 5
4455
>
```
2. The switch is local and listening on both UDP and TCP ports; the kernel gives it a port number 4455 for TCP and 6677 for UDP:
```
python3 RUSHBSwitch.py local 192.168.0.1/24 22.23.24.1/24 2 5
6677
4455
>
```
3. The switch is global and listening on a TCP port; the kernel gives it port number 1234:
```
python3 RUSHBSwitch.py global 1.2.3.1/24 456 789
1234
>
```

[Switch Connection]

With any incoming connection requests from the adapters or other switches, your switch must be able to run the [Greeting Protocol] that stated on page 7; also, your switch needs to give each of them the smallest available IP address based on its netmask and the running connections. For example, suppose an adapter connecting to your switch with host IP address is 192.168.0.1, and there are two adapters currently connecting. In that case, the available IP address that you will give to the incoming adapter is 192.168.0.4 as in the figure 15 below.

BIT	**0**	**8**	**16**	**24**
	192.168.0.1			
	0x00000000			
	0x000000			0x02
	192.168.0.4			

Figure 15. An example of the offering message

[Switch Commandline Interface & Distance]

The command line for the switch (not applicable for the local switch listening on both UDP and TCP) accepts the command" connect" with usage

```
> connect {port_number}
```

where `{port_number}` is the TCP port number of the host switch it wants to connect. When the command is executed, your switch will connect to that port. Then, it sends a greeting protocol similar to the [Greeting Protocol] defined on page 7. The switch receives the" connect" command (or first establishes the connection) and will first send the greeting message.

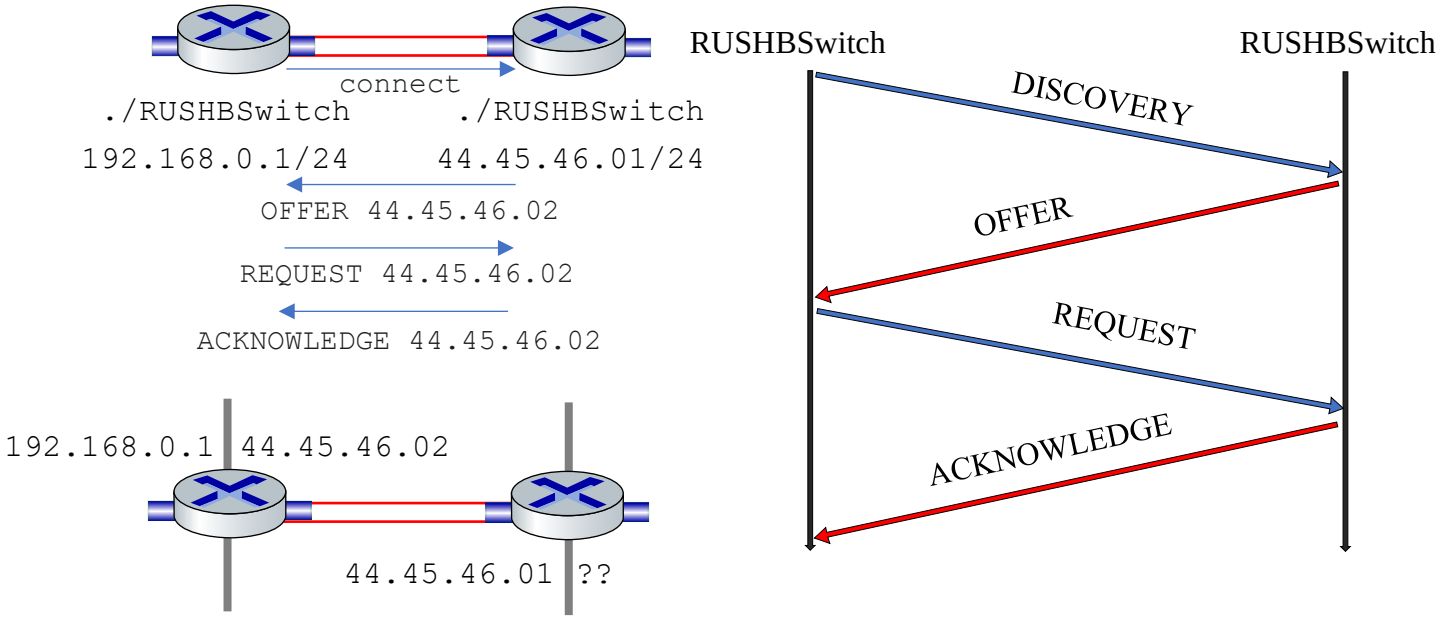


Figure 16. The greeting protocol between a local switch and a global switch

After finishing the greeting between switches, your switch also needs to send a message to tell its location, in MODE 0x08, and within the structure below.

BIT	0	8	16	24
	SOURCE IP ADDRESS			
	DESTINATION IP ADDRESS			
	RESERVED (all 0s)			MODE
	LATITUDE		LONGITUDE	

Figure 17. Location packet in RUSHB

BIT	0	8	16	24
	44.45.46.02			
	44.45.46.01			
	0x000000			0x08
	2		5	

Figure 18. An example of a location packet

The receiver switch will reply to them to tell you where it is (see an example in the figure below).

BIT	0	8	16	24
	44.45.46.01			
	44.45.46.02			
	0x000000			0x08
	465		784	

Figure 19. An example of a location packet

When receiving a location packet, your switch should broadcast the information about the distance of the new neighbour, to the current neighbours (see figure 20). To calculate the distance between you and the new connecting switch, we use this formula:

$$distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

In the example above, the distance will be calculated below and rounded down:

$$distance = \sqrt{(465 - 2)^2 + (784 - 5)^2} = 906.20 \approx 906$$

The broadcast message will be sent in the structure in figure 20 with MODE 0x09, but with the distance added with the distance from the source switch to the destination switch. Suppose you received a packet in figure 19, you have connected to a router with IP address 47.48.49.01, your IP address is 47.48.49.02 in that network, the distance between you and the host is 94, then the message you will send to 47.48.49.01 will be in figure 21.

BIT	0	8	16	24
	SOURCE IP ADDRESS			
	DESTINATION IP ADDRESS			
	RESERVED (all 0s)			MODE
	TARGET IP ADDRESS			
	DISTANCE			

Figure 20. Broadcasts message in RUSHB

BIT	0	8	16	24
	47.48.49.02			
	47.48.49.01			
	0x000000			0x09
	44.45.46.01			
	1000			

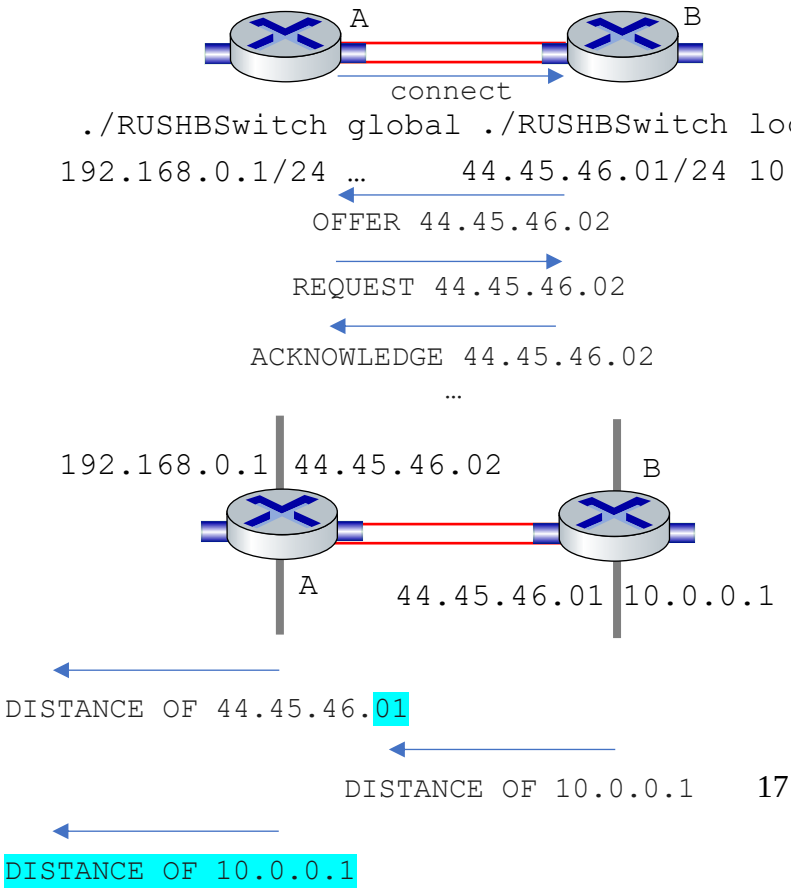
Figure 21. Example of broadcast message

If your switch received a packet with MODE 0x09, you also need to broadcast that message to all your neighbours except the one in SOURCE IP ADDRESS, DESTINATION IP ADDRESS, or TARGET IP ADDRESS. Please remember that you only can broadcast the message to other switches if they are connecting with you. The switch will not send the broadcast message to its adapters.

There are a few things about the command line and broadcasting messages that you need to know:

1. The local switch that is listening on both UDP and TCP won't accept the connect command; thus, it should do nothing if the user inputs a connect request.
2. After a command from a user, your program should print out the character ">" followed by a white space for the following input. If it receives any packet with MODE 0x09 with a distance larger than 1000, your switch should ignore it.
3. We assume that all the switches will have a different CIDR IP address and its range; for example, if there is a switch with IP address 42.43.44.1/8, then there will be no other switches with the IP address starting with 42.XXX.XXX.XXX/8, this also be applied to the local networks.
4. If your switch receives the same MODE 0x09 packet (same SOURCE IP ADDRESS, DESTINATION IP ADDRESS, and TARGET IP ADDRESS) but the distance is larger, it should ignore that packet (and not send to neighbours).

In the case of a switch that connects to a local switch listening on both UDP and TCP (invocation 2); after the greeting protocols, local switch sends a distance packet of its UDP side to the new connected switch with the target IP address is the IP address of the switch in the UDP side. The distance field of the packet is the distance from the local switch to the new connected switch. For example, if switch A connects to switch B (local switch) as in the figure below. After the greeting protocols and the location exchange, B then sends a distance packet to tell A that the distance from A to 10.0.0.1 is 10, suppose that the distance between A and B is 10.



BIT	0	8	16	24
	47.48.49.01			
	47.48.49.02			
	0x000000			0x09
	10.0.0.1			
	10			

Figure 22. Example of distance packet

When your switch receives an EOF (end of file), your switch (or adapter) needs to handle it by stopping accepting more user commands but still works other jobs (such as packet transmitting and packet processing) as usual.

[Fragmentation]

If your switch receives a data packet (MODE 0x05) with a total size larger than 1500 bytes (including header), as in figure 22, it must:

1. Splits the packet into smaller chunks with a maximum size of 1500 bytes
2. Send the chunks with MODE 0x0a (more fragments coming) and the last chunk with mode 0x0b (last fragment packet) with the structure as same as in figure 23. Remember that, before sending any packet with mode (0x0a, 0x0b, and 0x0b), it needs to establish the query to ask if the receiver is ready to receive or not (see [Data Communication] on pages 9-10).
3. Set-up an offset from the second packet.

BIT	0	8	16	24
	192.168.0.2			
	192.168.0.3			
	0x000000			0x05
	"A" x2000			

Figure 23. A 2012 bytes data packet

BIT	0	8	16	24
	SOURCE IP ADDRESS			
	DESTINATION IP ADDRESS			
	OFFSET			MODE
	DATA			

Figure 24. Fragmentation in RUSHB

For the data packet in figure 23, your switch must split it into two packets:

BIT	0	8	16	24
	192.168.0.2			
	192.168.0.3			
	0x000000			0x0a
	"A" x1488			

Figure 25. A 1500-bytes data packet

BIT	0	8	16	24
	192.168.0.2			
	192.168.0.3			
	0x0005d0			0x0b
	"A" x512			

Figure 26. A 524-bytes data packet

As you can see, the packet in figure 26 has an offset of 0x5d0 (1488 in decimal), which denotes that this packet's data starts at index 1488.

When your adapter receives the message with mode 0x0a, it should wait for the next packets until it receives packet mode 0x0b. Then, it prints to stdout the data is received (as normal). For the example above, your adapter will print out:

```
Received from 192.168.0.2: AAA.........AAAA
```

with 2000 letter A's.

Fragmentation is applicable for the switches, and messages send from an adapter to another adapter only. Also, an adapter can send a packet with a maximum size of 55296 bytes.

[Forwarding]

If your switch receives data messages (0x05, 0x0a, and 0x0b), it should forward the message to other routers until it reaches the destination, with these conditions:

1. The router that receives the forward message must be one that could form the shortest path (based on the distances) to the destination.
2. Every message must follow the [Data Communication] on page 9.
3. If the router that receives a message and its IP address is the same as the destination IP address as in the packet, it should print to stdout the message is received, same as [Data Communication] on page 10.
4. If your switch doesn't know the existence of the destination's IP address, it will forward the message to the **longest prefix matching** of the next switch IP address (Kurose, p.344, 4.2.1).
5. In the case of 2 paths that have the (same) shortest distance to the destination, your switch must send the message to the switch with the **longest prefix matching** (Kurose, p.344, 4.2.1). This is only applicable if your switch knows the existence of the destination's IP address; if not, rule number 4 must be applied.

In the example below, A wants to send B a message, and we assume that all routers in a network have their IP address and distances assigned in figure 24. Then, the message will be forwarded to each switch within the green line to get to the destination.

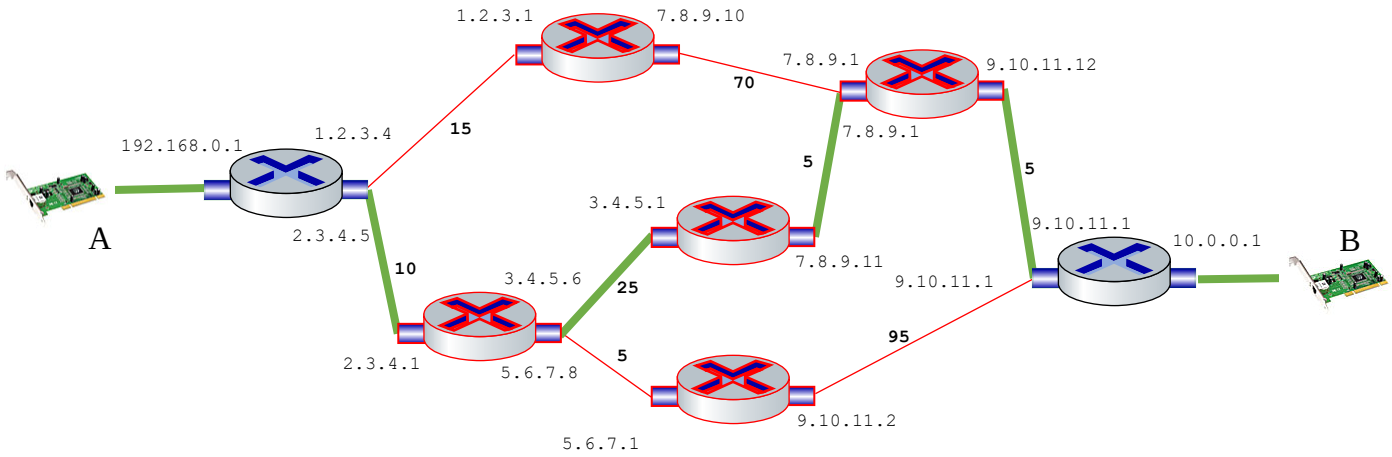


Figure 26. End-to-end connection with the IP addresses and distances

In the case of a coming data packet with the destination IP address is 129.0.0.1 and:

1. Your switch doesn't have any information about the destination IP address from the broadcast messages.
2. Your switch is connecting to a switch that has an IP address of 135.0.0.1, and a switch has an IP address of 136.0.0.1.

Your switch should send the message to the switch with an IP of 135.0.0.1 since it matches the longest IP address prefix (1000 0001 and 1000 0111 rather than 1000 0001 and 1000 1000).

We assume that the connection between switches is from the left-hand side to the right-hand side, in the other words, the message will always start from Adapter A to Adapter B as in figure 26. Only after B received the message from A, B can send the message back to A using the same path used from A to B.

You are free to implement any algorithms you would like to. Still, your adapter and switch must not send any extra packets than the packets that have been stated above, as well as making extra files for the communication between your programs. Also, all the programs must be thread-safe; if race-condition occurs in some tests, you might get 0 for them. As long as your program works as expected, you will receive full marks for that part.

Submission

Your programs must be written in Python, Java, C, or C++. Your program should be able to be invoked from a UNIX command line as follows. It is expected that any Python programs can run with version 3.6, any Java programs can run with version 8, any C programs must be complied with the standard of C99. No external libraries or extensions is permitted for use in your program.

Python	`python3 RUSHBSwitch.py {local|global} {ip} [optional_ip] {x} {y}`
Java	`make` `java RUSHBSwitch {local|global} {ip} [optional_ip] {x} {y}`
C or C++	`make` `./RUSHBSwitch {local|global} {ip} [optional_ip] {x} {y}`

Figure 8. Filename and Command-Line syntax for the submission

No late submissions will be marked for this assignment under any circumstances. Submission must be made electronically by committing using subversion. In order to mark your assignment, the markers will check out `/trunk/ass2/` from your repository on source.eait.uq.edu.au. Please do not create subdirectories under `/trunk/ass2/`. The marking may delete any such directories before attempting to compile them. Code checked into any other part of your repository will not be marked.

IMPORTANT NOTICE: As the assignment is auto-marked, it is essential that the filename and command-line syntax exactly matches the specification above. Specification adherence is critical for passing. If your program fails to execute because of the wrong syntax, you will receive a 0 for the assignment without any exceptions.

Marking

Marks will be awarded for functionality only. The tests will mainly check your program's behaviour and packet structure.

Provided that your code compiles (see above), you will earn marks based on the number of features your program correctly implements. Partial marks may be awarded for partially meeting the functionality requirements. Not all features are of equal difficulty. If your program does not allow a

feature to be tested, you will receive 0 marks for that feature, even if you claim to have implemented it. For example, if your program can never open a file, we cannot determine if your program would have loaded input from it. The markers will make no alterations to your code (other than to remove code without academic merit). Your program should not crash or lock up/loop indefinitely (without any reason). Your program should not delay for unreasonably long times.

For RUSHBAdapter:

- execute and run (0 mark)
- send and receive packets (0 mark)
- packet build and protocols (0 mark)
- command-line interface (0 mark)

For RUSHBSwitch:

- execute and run (2 marks)
- IP validation capacity (5 marks)
- send and receive packets (8 marks)
- packet building and protocols (15 marks)
- command-line interface (5 marks)
- simple communication (5 marks)
- complete communication (10 marks)

Specification updates

It is possible that this specification contains errors or inconsistencies, or missing information. It is possible that clarifications will be issued via the course website. Any such clarifications posted five days or more before the due date will form part of the assignment specification. If you find any inconsistencies or omissions, please notify the teaching staff.

Tips and hints

1. Start your assignment early

Updates

1.1 Fix typos in part C