The University Of Queensland
School of Information Technology and Electrical Engineering
Semester One, 2021

COMS3200 - Assignment 2

Due: 3:00pm 19th May, 2021
Marks: 100
Weighting: 20% of your final grade
Revision: 0.7

Introduction

This assignment introduces the use of Network Layer and Link Layer (from OSI Models) as well as socket programming (simulation) on each of these devices: adapters, switches and routers. After finishing the assignment, you will understand how data is sent throughout the internet.

This assignment contains three parts. For Part A and B, you will need to submit your answers through Blackboard Quiz. for Part C, you will submit your code via COMS3200 subversion (SVN). All submitted works are marked automatically. It is strongly recommended that you read this spec carefully before you start working, as to avoiding any mistakes.

This assignment is to be your 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 perfectly and has grown so much in users recently that the company must now upgrade their network.

You, a supervisor of this project, has assigned the task of designing two (2) new networks, a branch in Sydney and a local system for working at home. You also moved the File Server to "somewhere" on the Internet to serve as a centralise data centre for your company. Then, you ended up with the network map on next page.

To minimise spending for this system, you set up for only one DNS Server (D2) and one DCHP Server (D1). You claimed that the system will run smoothly without paying more for building extra servers, and it works.

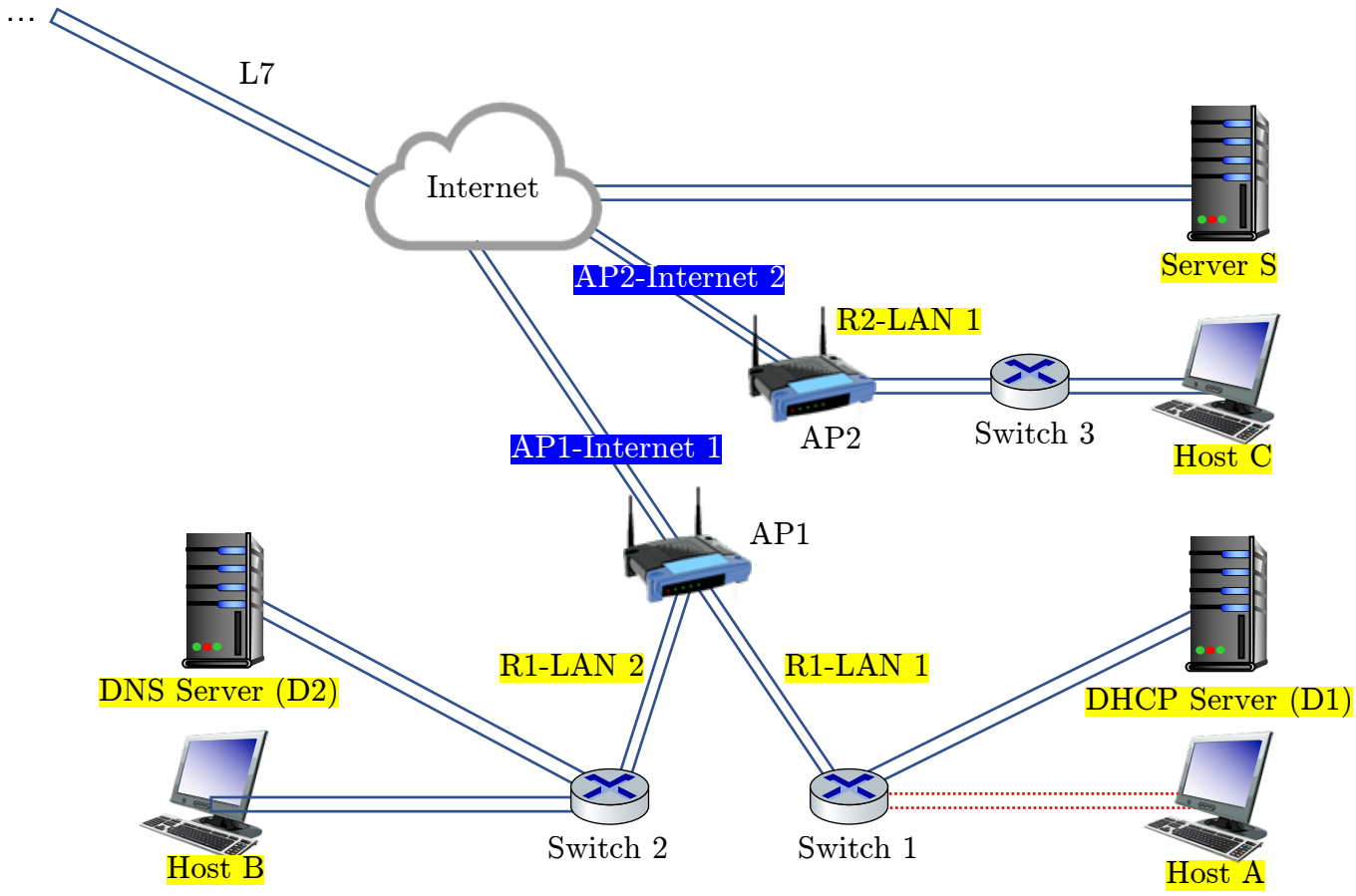


Figure 1. A simple internet map

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

1. The broadcast MAC address of all the LANs is FF:FF:FF:FF:FF:FF.
2. All forwarding tables in the switches and routers are up to date.
3. All valid protocols are HTTP, FTP, SSH, DNS, DHCP, UDP, TCP, ICMP, SNMP, ARP, OSPF, and BGP. If there are 2 protocols used at the same time (e.g., SSH and TCP), choose the highest level (lowest layer) only (e.g., TCP).
4. D2 is a DNS Server, and D1 is a DHCP Server.

You will have to fill out the table and answer some questions in Blackboard. Each student's questions will have different values (that is replaced by XXX...). Your question values will be shown in the quiz. Please be careful to double-check before submitting your work to Blackboard, as you can submit only once.

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 that will be 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 of sender IPv4 address	Destination of 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 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, and no additional ongoing communications.

Protocol	Opcode	Source MAC address	Destination MAC address	Source of sender IPv4 address	Destination of target IPv4 address

Question 3: Host A now wants to establish a HTTP connection with the Server S (see the figure 1 above). 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 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	Opcode	Source MAC address	Destination MAC address	Source of sender IPv4 address	Destination of 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?

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 (9 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 being run from the client. Some of the relevant IETF RFCs may help you in understanding 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 to ask any questions you may have for this trace besides the given questions to provide you with more understanding of 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: The client joined the LAN networking successfully. What is the assigned IP address from the DHCP Server to this client? (1 mark)
Question 2: What is the MAC address of the client? (1 mark)
Question 3: What is the netmask of this LAN network? (1 mark)
Question 4: What is the DHCP Server's IP address? (1 mark)
Question 5: What is the MAC address of the DHCP Server? (1 mark)
Question 6: Is this LAN network a subset of a bigger LAN network? (1 mark)
Question 7: What is the target IP address of the traceroute command? (1 mark)
Question 8: Does the traceroute command run successfully, or give up while waiting for an ICMP response? (1 mark)

Question 9: How many routers are there between the host (where it runs the traceroute command) and the target server? (1 mark)
Question 10 (bonus): The client connects to the LAN network through a password enabled wireless connection, what is the security standard used for this Access Point? (1 mark)

Hints:

1. For each TTL, traceroute sends the packet 3 times.
2. The DHCP Server is attached to another device.
3. Please use the Wireshark filter where appropriate.
4. Available security standards you can choose are WPA/WPA2 (personal or enterprise), WEP, and WPA3.

Part C (70 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 cumbersome and complicated. As seen in Figure 1, the RUSHBSvr is now located somewhere else on the Internet, and you are tasked with setting up the protocols needed to exchange data with the server.

To accomplish this part, you will need to implement two **virtual** protocols: RUSHBAdapter on the link layer, and RUSHBSwitch on network/transport layer protocol. **Virtual** means that your program is simulating those protocols in the application layer; that is, network layer addresses will not correspond to physical interfaces.

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, as well as looking at another student's code (even if it's in printed or in electronic form). All of your submitted code will be subject to automated checks for plagiarism and collusion on moss and detected plagiarism or collusion will be reported for misconduct proceedings. If you're having trouble, please seek help from a member of the teaching staff. 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

You need to implement 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 a simulated network, they are all ***implemented in the application layer of the OSI model of networking*** in practice (see figure below).

<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, your 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:

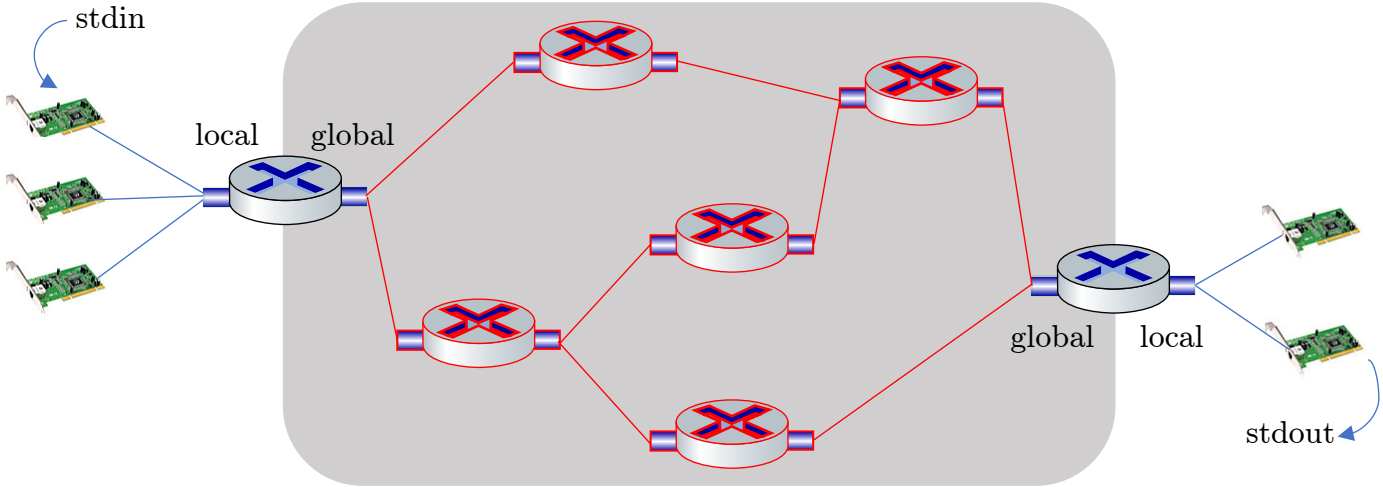


Figure 3. End-to-end connection in RUSHB

The final goal for this programming assignment is sending and receiving data across the global network without losses and errors. Data can be anything that can attached to the

adapter, such as netcat, or RUSHBSvr, or even stdin (as in this assignment). From now, we will refer to 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 an 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 of the greeting packets is in the structure below. Note that from now, we will define each block of a packet has a total of 4 bytes, divided by 4, each one is 8 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

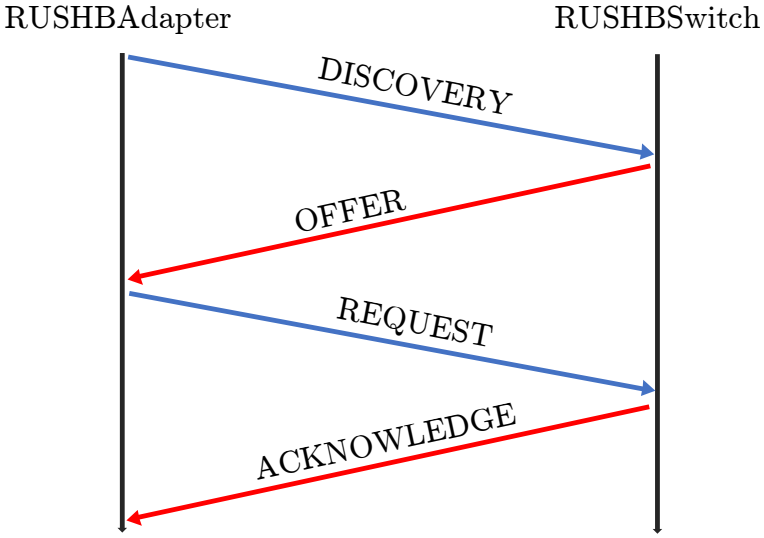


Figure 6. Then greeting process between RUSHBAdapter and RUSHBSwitch

We call RUSHBSwitch in the case above is the host, and RUSHBAdapter is the client. It will work as follow:

1. In a DISCOVERY message, since the sender has not 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 accept the offer yet).
3. After receiving the offer from the host, your adapter (or switch for the upcoming part) now send back the request for the assigned IP address. The REQUEST should contains 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 acknowledgement message with all the fields must be 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 in 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.

[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 192.168.0.1/24, the switch IP address is 192.168.0.1, the next 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 there are more than 254 hosts connected to your switch, your switch should ignore the incoming connections and not giving the [Greeting Protocol] stated in page 7.

[Adapter Commandline Interface]

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

For example, if you make your program using C99, and you want to type "HELLO_WORLD" as a command:

```
./RUSHBAdapter 5678
> HELLO_WORLD
```

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 a white space). For example, if you want 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, your adapter will build a packet with the structure 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 need to send 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, the network byte order is defined to always 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 next command.

[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 for the transmission between 2 switches, the sender switch will send the packet with MODE 0x06, and the receiver switch will reply back 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 established, the sender will send the packet as in figure 11. Please note that even though the switch that sends the message to the adapter has an IP address 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 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 few things you need to remember for sending and receiving packets:

1. Your program should remove the characters "> " before print out the received message, then, it should print the characters "> " again for the next user command.
2. We send and receive messages through the port number in the localhost (127.0.0.1), and assuming that the port number is in link layer of RUSHB Models.
3. Your adapter (or switch) won't have to send the checking packets (mode 0x06 and 0x07) again if the checking protocol already done within 5 seconds. After 5 seconds of the checking protocols, if there is an upcoming packet, your adapter (or switch) has to establish the checking protocol again.
4. If any packets that are not in the given protocols or invalid, your adapter and switch should ignore them.

[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

To simply the calculations, latitude and longitude are denoted the integers, represent the coordinates on a cartesian plane, where latitude is horizontal x-axis and longitude is the vertical y-axis.

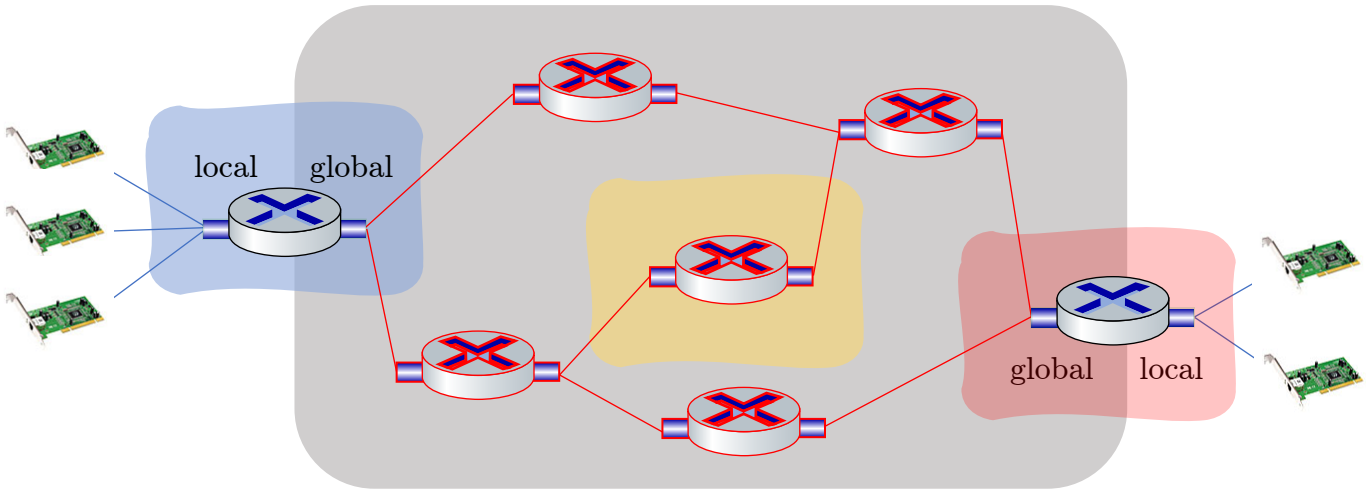


Figure 14. End-to-end connection with the areas

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 socket, one is for the local adapters through a UDP port, and another one in a global area that listen in the TCP port for other switches (the red area 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 is `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. For the switch listening in both TCP and UDP ports, it should print port on UDP first, and then TCP. It should also print out (to stdout) the character ″>″ followed by a 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, if an adapter connecting to your switch with host IP address is 192.168.0.1, and there are two adapters currently connecting, then 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]

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 want connects to. When the command executed, your switch will connect to that port. Then, it sends a greeting protocol as same as the [Greeting Protocol] defined in page 7. The switch receives the `"connect"` command (or first establish the connection) will sends the greeting message first.

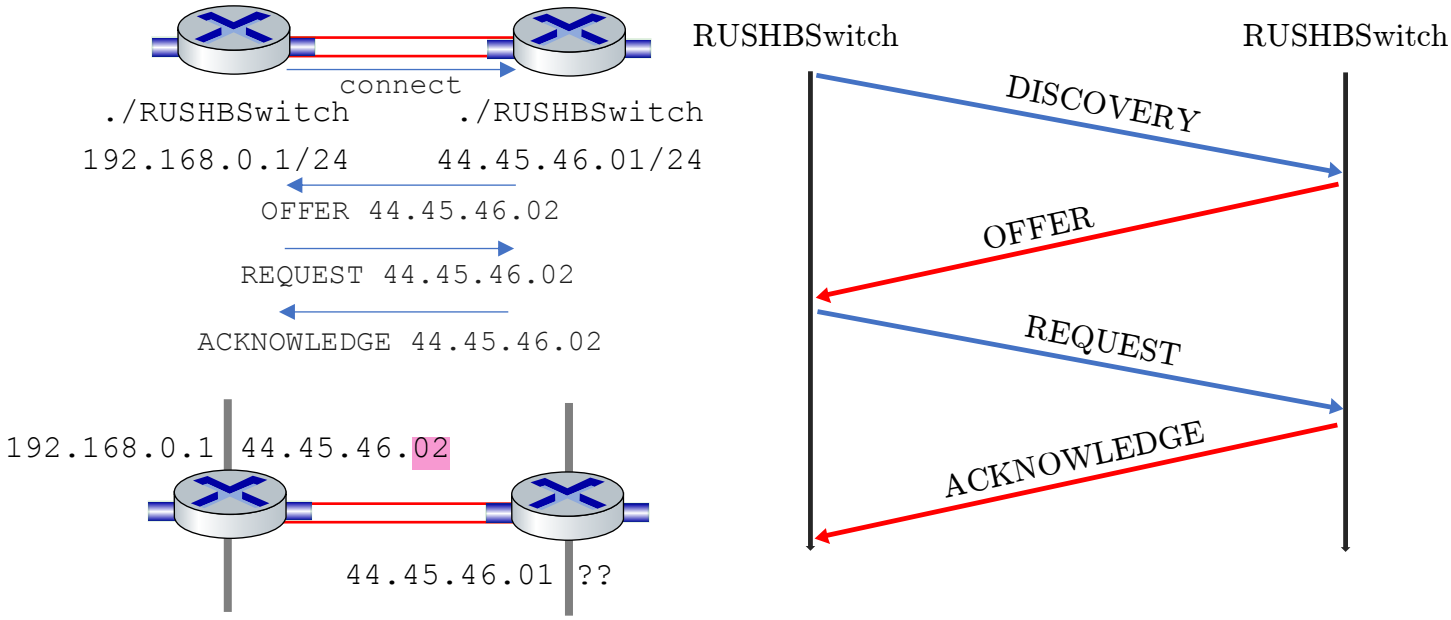


Figure 16. The greeting protocol between a local switch and a global switch

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 location packet

The receiver switch will reply 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 location packet

When receiving a location packet, your switch should broadcasts 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			

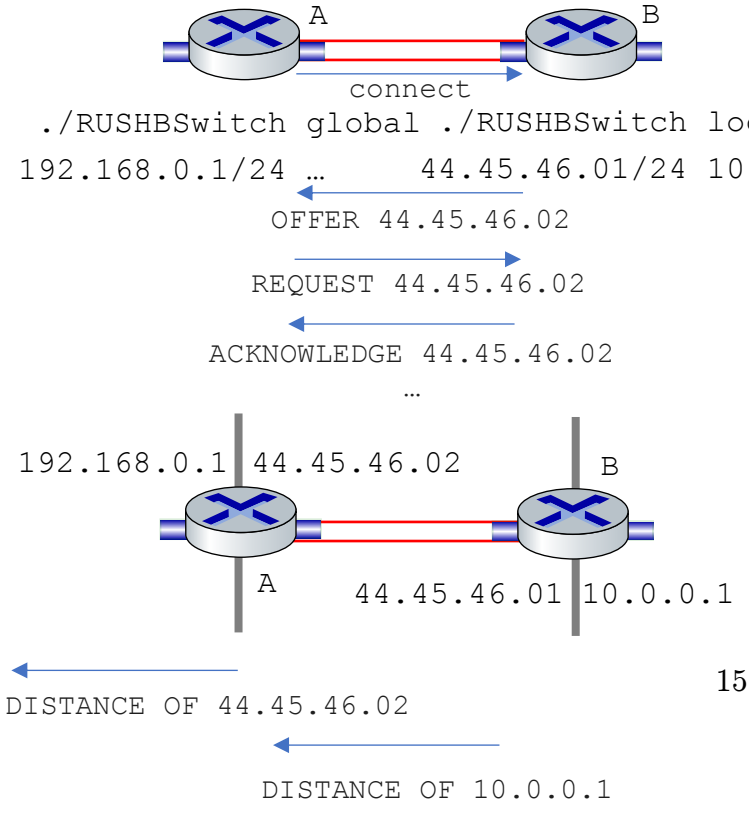
Figure 21. Example of broadcast message

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 boardcast message to its adapters.

There are few things about commandline and broadcasting message that you need to know:

1. For the local switch that listening on both UDP and TCP, it won't accept the connect command, thus if user input a connect request, it should do nothing.
2. After a command from a user, your program should prints out the character ">" followed by a white space for the next input. If it receives any packet with MODE 0x09 with distance larger than 1000, your switch should ignores that packet.
3. We assume that all the switch will have a different CIDR IP address, as well as its range, for example, if there exists a switch with IP address 42.43.44.1/8, then, there will be no other switches that has the IP address start with 42.XXX.XXX.XXX/8, this also be applied for 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, then it should ignore that paket (and not sending to neighbours).

In the case of a global switch (invocation number 3 on page 12) that connects to a local switch (invocation number 2 in page 12); after the greeting protocols, local switch then sends a distance packet of its UDP side to that global switch. The distance field of the packet is the distance from the local switch to the global switch. For example, if switch A (global) connects to switch B (local) 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

[Fragmentation]

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

1. Splits the packet by smaller chunks with the 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.
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 splits it into 2 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, packet in figure 26 has an offset of 0x5d0 (1488 in decimal), denotes that the data in this packet starts at index 1488.

When your adapter receives the message with mode 0x0a it should wait for the next packets until receives packet mode 0x0b. Then, it prints to stdout the data it 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 only, the adapter are able so send the packet with maximun size of 4096 bytes.

[Forward]

If your switch receives data messages (MODE 0x05, 0x0a and 0x0b), it should forwards the message to another routers until it reached the destination, with these conditions:

1. The router that receives the forward message has to be one that could form a shortest path (based on the distances) to the destination.
2. Every message must follow the [Data Communication] in page 9.
3. If the router that receives the 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 it received, as same as [Data Communication] in page 10.

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

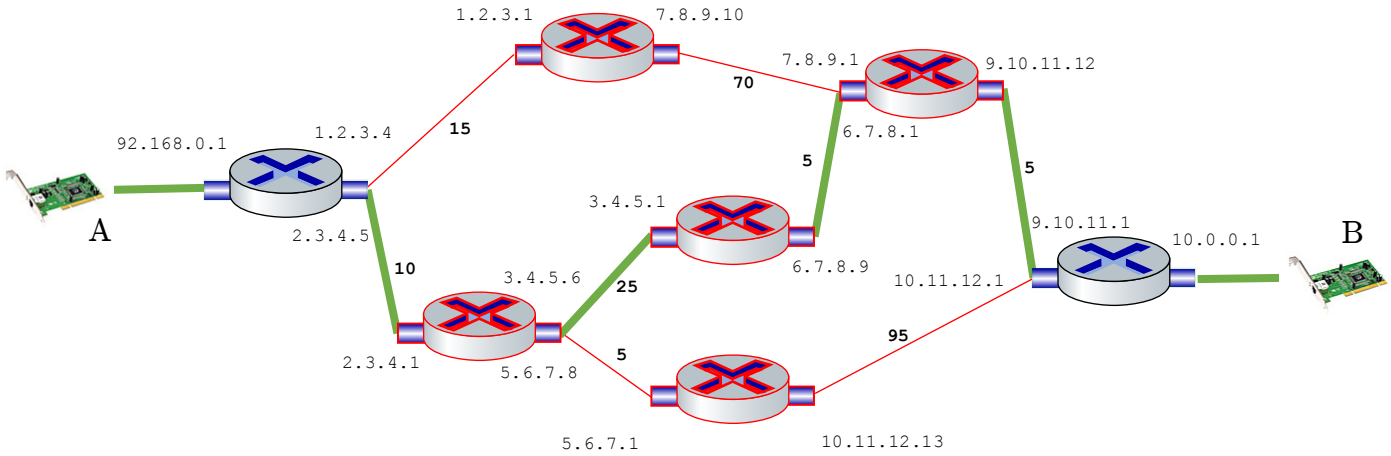


Figure 26. End-to-end connection with the IP addresses and distances

You are free to implement any algorithms you would like to, but your adapter and switch must not send any extra packets than the packets that have stated above. Also, all the programs must be thread-safe, if race-condition occurred in some tests, you might get 0 for these. 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 have to be complied with the standard of C99 or GNU99. No external libraries or extensions is permitted for use in your program.

Python	`python3 RUSHBAdapter.py {port}` `python3 RUSHBSwitch.py {local|global} {ip} [optional_ip] {x} {y}`

Java	`make`
	`java RUSHBAdapter {port}`
	`java RUSHBSwitch {local|global} {ip} [optional_ip] {x} {y}`

C or C++	`make`
	`./RUSHBAdapter {port}`
	`./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. Code checked in to any other part of your repository will not be marked.

IMPORTANT NOTICE: As the assignment is auto-marked, it is very important that the filename and command-line syntax exactly matches the specification above. Specification adherence is critical for passing. If your program is failed to executed 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 mostly 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 (5 marks)
- send and receive packet (5 marks)

- packet build and protocols (5 marks)
- command line interface (5 marks)

For RUSHBSwitch:

- execute and run (10 marks)
- ip validation capacity (5 marks)
- send and receive packet (5 marks)
- packet build and protocols (5 marks)
- command line interface (5 marks)

For the combination of 2 or more processes:

- message transmission in a simple map (10 marks)
- complete transmission in a big map (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 (5) 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

0. Start your assignment early.

Updates

0.1 Initial release
0.2 Add clarification and edit part B
0.3 Edit part B
0.4 Add clarification
0.5 Add clarification
0.6 Add clarification
0.7 Add clarification for Fragmentation