**CS6014 Written Homework 2: Lower Network Layers**

**U1426365 Jin-Ching Jeng**

**Question 1: Reliable Data Transfer**

We want to send data from one node to two other nodes using over a simple broadcast channel. Specifically, we want to design a protocol for reliably sending data from host S to hosts R1 and R2 over this channel. The channel can lose or corrupt packets for independently. For example, a packet sent by S might be received by R1 but not R2.

When there are collisions on the broadcast channel, you can assume that the receiving hosts will detect them as corrupt packets. If data needs to be resent, you can ignore random backoffs, etc, and assume that eventually the colliding hosts will be able to resend their data without interference.

**Design the protocol state machines for S and R (both R1 and R2 should use the same protocol).**

Use the primitives we discussed in the notes (udt\_send and receive, etc). Don't consider pipelining. The RDT protocol we developed with sequence numbers 0 or 1 + timeouts is a good starting point.

Sender:

Diagram

Description automatically generated

Receiver:

Text

Description automatically generated

**Question 2: Throttling**

What is the difference between flow control and congestion control? Describe the way TCP implements each of these features.

In TCP, it throttles senders to protect receiver (flow control) and network (congestion control).

1. Flow control: Sender reduces its sending rate so receiver can keep up.

* In TCP, flow control is implemented by a sliding window. The receiver sends the size of the receive window rwnd , which indicates the amount of data that the sender can transmit before receiving an acknowledgement.
* The rwnd increases when the application reads data out of the buffer and decreases when it receives a message.
* If the receiver's buffer fills up, it will reduce the size of the window, effectively slowing down the sender.

1. Congestion control: Prevent a high volume of traffic from causing network congestion and packet loss. If the network is congested, packets will be delayed or dropped.

* Congestion control in TCP is implemented using a variant of the Additive Increase Multiplicative Decrease (AIMD) algorithm. It increases the congestion window cwnd slowly while it can successfully transmit, and then decreases it rapidly when it detects congestion. The cwnd size determines the number of packets that can be in flight at any given time.
* When a TCP sender detects congestion, it reduces its transmission rate by reducing its congestion window size.
* As congestion decreases, the sender slowly increases its congestion window size, allowing it to transmit more packets.

**Question 3: NAT**

Two hosts (IPs A: 10.0.0.1 and B: 10.0.0.2) sit behind a NAT enabled router (public IP 5.6.7.8). They're both communicating with a remote host X, 1.2.3.4 on port 80. What are *possible* values for the source and destination addresses and ports for packets:

* from A to X behind the NAT

|  |  |  |  |
| --- | --- | --- | --- |
| Source Address | Source Port # | Destination address | Destination Port # |
| 10.0.0.1 | 7777 | 1.2.3.4 | 80 |

* from B to X behind the NAT

|  |  |  |  |
| --- | --- | --- | --- |
| Source Address | Source Port # | Destination address | Destination Port # |
| 10.0.0.2 | 8888 | 1.2.3.4 | 80 |

* from A to X between the NAT and X

|  |  |  |  |
| --- | --- | --- | --- |
| Source Address | Source Port # | Destination address | Destination Port # |
| 5.6.7.8 | 8080 | 1.2.3.4 | 80 |

* from B to X between the NAT and X

|  |  |  |  |
| --- | --- | --- | --- |
| Source Address | Source Port # | Destination address | Destination Port # |
| 5.6.7.8 | 8080 | 1.2.3.4 | 80 |

* from X to A between X and the NAT

|  |  |  |  |
| --- | --- | --- | --- |
| Source Address | Source Port # | Destination address | Destination Port # |
| 1.2.3.4 | 80 | 5.6.7.8 | 8080 |

* from X to A between the NAT and A

|  |  |  |  |
| --- | --- | --- | --- |
| Source Address | Source Port # | Destination address | Destination Port # |
| 1.2.3.4 | 80 | 10.0.0.1 | 7777 |

What there corresponding contents of the router's NAT translation table?

|  |  |  |  |
| --- | --- | --- | --- |
| Local IP | Local Port # | Internet IP | Internet Port # |
| 10.0.0.1 | 7777 | 5.6.7.8 | 8080 |
| 10.0.0.2 | 8888 | 5.6.7.8 | 8080 |

## Question 4: Routers

A company has 3 groups that each have a subnet on the corporate network.

Group A uses subnet 1.1.1.0/24.   
Group B uses 1.1.2.0/24.   
Group C uses subnet 1.1.3.0/24.

Each group has a router. There is a link between each pair of routers.

A and B have a link: 1.1.4.0 (on A) to 1.1.4.1 (on B)   
A and C have a link: 1.1.5.0 (on A) to 1.1.5.1 (on C)   
B and C have a link: 1.1.6.0 (on B) to 1.1.6.1 (on C)

* How many subnets are a part of this network, and what is the smallest IP prefix (i.e. most fixed bits) that can be used to describe each one?

There are 6 subnets in this network.

The smallest IP prefix to describe each one:

A: 1.1.1.0/24

B: 1.1.2.0/24

C: 1.1.3.0/24

A-B: 1.1.4.0/31

A-C: 1.1.5.0/31

B-C: 1.1.6.0/31

* If this network is somehow connected to the internet, what is the cheapest (i.e. smallest number of address) IP prefix the company could have purchased (without using NAT)?

The cheapest IP prefix is /22

Explanation: They need 774 IP addresses ( 256(A) + 256(B) + 256(C) + 2(A-B) + 2(A-C) + 2(B-C) = 774 ), and /22 has 1024 usable addresses. (/21 has 512 addresses which is not enough.)

* Assume the router for group A has 4 ports: port 1 is connected to the group subnet, port 2 is connected to router B, port 3 is connected to router C, and port D is connected to the ISP. Write out router A's forwarding table.

|  |  |
| --- | --- |
| Destination Prefix | Output Link (# port) |
| 1.1.1.0/24 (group A subnet) | 1 |
| 1.1.4.1 (router B) | 2 |
| 1.1.5.1 (router C) | 3 |
| 0.0.0.0/0 (ISP Gateway) | D |

## Question 5: Routing

Implement the onInit and onDistanceMessage methods in the Router class of the code provided in [this directory](https://github.com/UtahMSD/CS6014_2023/blob/main/homeworks/bellmanFord) so that the routers use the Bellman Ford algorithm to compute routing information. Use the static methods in the Network class to help you with this. These methods shouldn't be more than ~20 lines of code or so!

Once you have a working implementation, test your algorithm on a variety of network sizes. Plot the number of messages required to converge as a function of network size. Since the networks are probabilistically generated, you might want to try several networks of each size to get a sense of the distribution.