Homework 2: Lower Network Layers

Question 1: Reliable Data Transfer

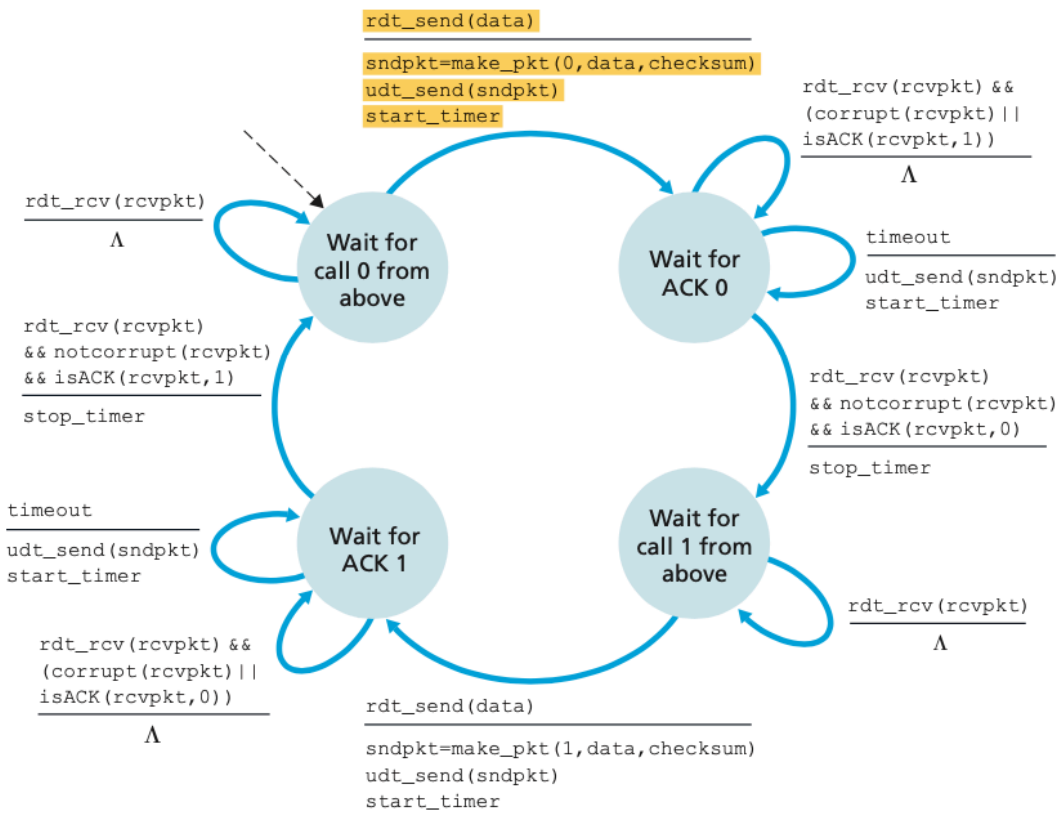
We want to send data from one node to two other nodes using over a simple broadcast channel. Specifically, we want to design a protocol for reliably sending data from host S to hosts R1 and R2 over this channel. The channel can lose or corrupt packets independently. For example, a packet sent by S might 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



Rdt_send(data)

- Udt_send packet
- Start timer

Rdt_rcv

- isAck
- timeout
- Start timer

rdt_rec

- isAck
- Stop_timer

Rat_send

- udt_send
- Start timer

Rdt_rec

- isAck

Timeout

- udt_sent
- Start timer

rdt_rec

- isAck
- Stop timer

Receiver

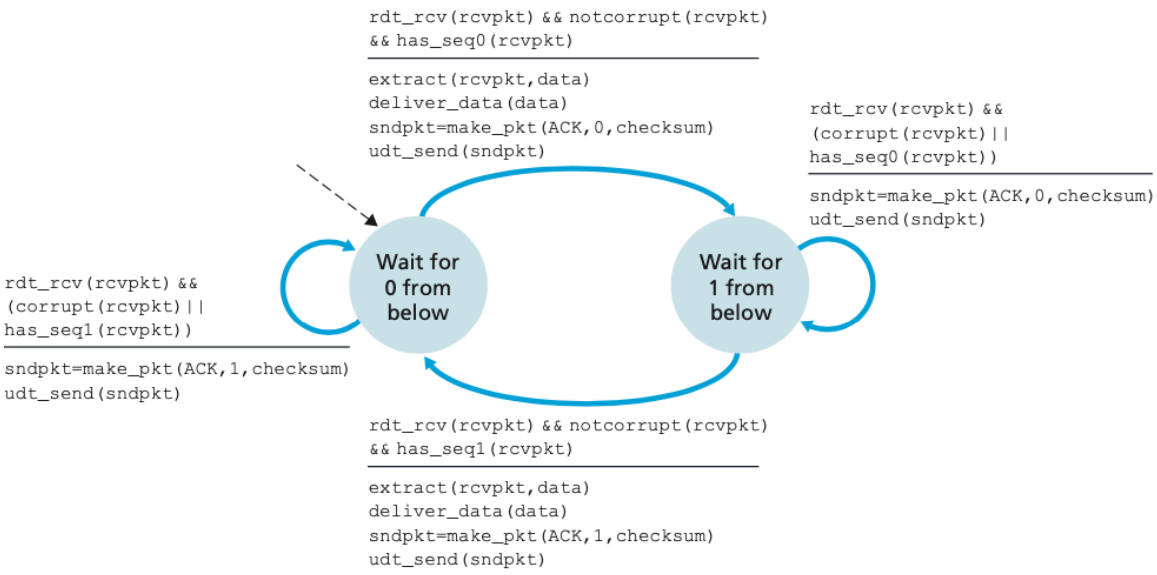


Figure 3.14 ♦ `rdt2.2` receiver

Rdt_rec

- extract
- Deliver data
- Make packet
- udt_send

rdt_rcv

- Has seq
- Make packet
- send

Rat_rcv

- extract
- deliver
- Udt_send

Rat_rcv

- Has seq
- Udt_send

Question 2: Throttling

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

Flow Control

Flow control is the mechanism that controls the traffic between sender and receiver (end to end). Handled at the data link layer and transport layer. The sender is responsible for how much traffic is sent to the receiver. Takes place on the receiver side (receiver reports status). Receive Window (Sender is rate limited by receiver). This mechanism prevents the receiver from getting overloaded with data. There are two ways it achieves this:

1. Feedback-based control
2. Rate-based flow control.

Feedback-based flow control

The receiver sends it's status to the sender and uses either a sliding window protocol or stop-and-wait protocol to determine how much data is sent.

Rate-based flow control

A built-in mechanism limits the rate of transmission without using feedback from the receiver.

Congestion Control

Congestion control is used to prevent too many packets being transmitted by the transport layer into the network. Handled at the network layer and transport layer (node to node). The transport layer is responsible for how much traffic is sent to the network. Takes place on the sender side (uses things like time to detect congestion). Tell if congestion by looking at dropped packets. Three methods are typically used to control congestion:

1. Provisioning
    - Network built to match traffic
2. Traffic aware routing
    - Routes tailored according to traffic
3. Admission control
    - Refuses new connections causing congestion (slow start and exponential backoff)

Sources

https://techdifferences.com/difference-between-flow-control-and-congestion-control.html

https://www.difference.wiki/flow-control-vs-congestion-control/

https://www.differencebetween.com/difference-between-flow-control-and-vs-congestion-control/

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
    - A sends on 10.0.0.1 on randomly assigned port > 1024 to the router 5.6.7.8 and then the router forwards the request to 1.2.3.4 on port 80
- from B to X behind the NAT
    - B sends to 10.0.0.1 on randomly assigned port (typically > 1024) to the router 5.6.7.8 and then the router forwards the request to 1.2.3.4 on port 80
- from A to X between X and the NAT
    - The NAT sends from 5.6.7.8 on port 80 to X on 1.2.3.4
- from B to X between X and the NAT
    - The NAT sends from 5.6.7.8 on port 80 to X on 1.2.3.4
- from X to A between X and the NAT
    - X sends from 1.2.3.4 to 5.6.7.8 on port 80 and the router forwards the request to port previously assigned to 10.0.0.1
- from X to A between the NAT and A
    - X sends from 1.2.3.4 to 5.6.7.8 on port 80 and the router forwards the request to 10.0.0.1 to port previously assigned to 10.0.0.1

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

Local IP Address	**Internal Port**	**Router Public IP address**	**Remote Host**	**Port**
10.0.0.1	Randomly Assigned (typically > 1024) *	5.6.7.8	1.2.3.4	80
10.0.0.2	Randomly Assigned (typically > 1024) *	5.6.7.8	1.2.3.4	80

The NAT table keeps track of all incoming and outoing packets and associates them with it's private IP addresses and the public addresses where they are being sent or received.

*https://stevessmarthomeguide.com/understanding-port-forwarding/

Question 4: Routers

A company has 3 groups that each hava 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 3 subnets on this network: 1.1.1.0/24, 1.1.2.0/24, 1.1.3.0/24. The smallest IP prefix to describe each one is 24 bits.
- 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?
  - 256 (addresses available at each 24 bit prefix) * 3 (subnets) = 768 IP Addresses
- 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.
  -

Next Hop	Destination
Port 1 (Deliver Direct)	1.1.1.0/24
1.1.4.1- Port 2 (Router B)	1.1.2.0/24
1.1.5.1- Port 3 (Router C)	1.1.3.0/24
Port 4 (ISP)	0.0.0.0/0

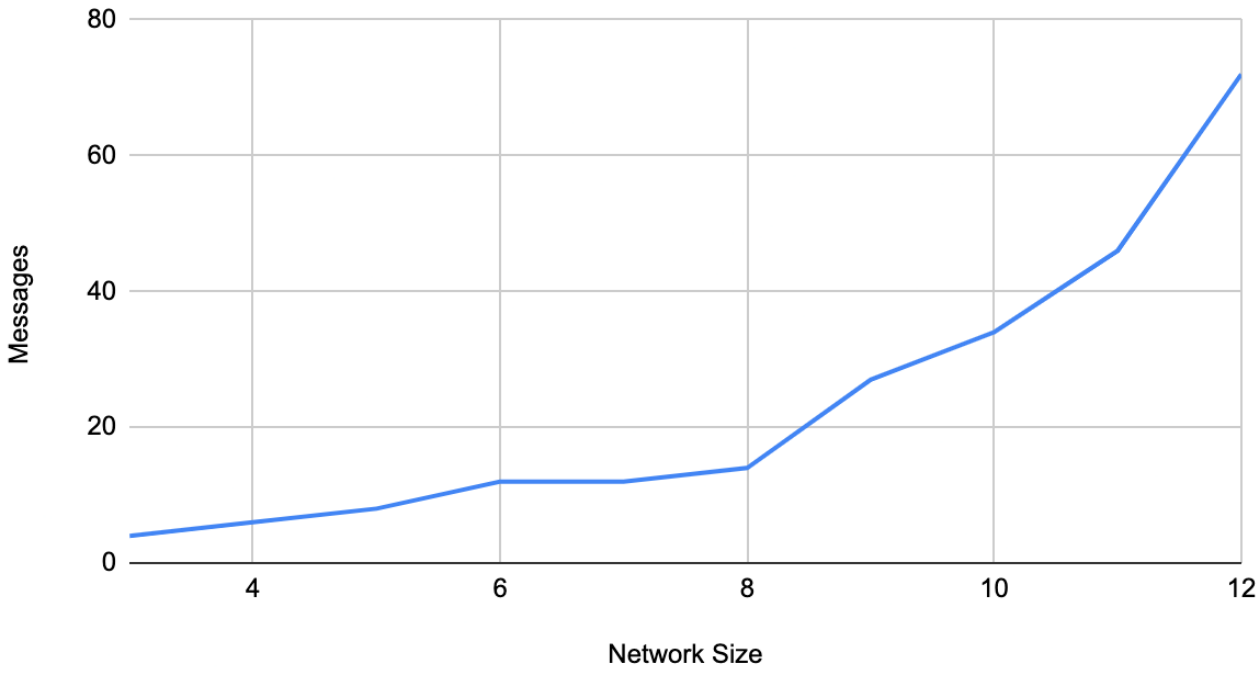
Question 5: Routing

Implement the `onInit` and `onDistanceMessage` methods in the Router class of the code provided in this directory 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 probablisitically generated, you might want to try several networks of each size to get a sense of the distribution.

```
    public void onInit() throws InterruptedException {
        //As soon as the network is online,
        //fill in your initial distance table and broadcast it to your neighbors
        HashSet<Neighbor> neighbors = Network.getNeighbors(this);

        for (Neighbor neighbor: neighbors) {
```

```
            distances.put(neighbor.router,neighbor.cost);
        }

        for (Neighbor neighbor: neighbors) {
            Message newMessage = new Message(this, neighbor.router, distances);
            Network.sendDistanceMessage(newMessage);
        }
    }

    public void onDistanceMessage(Message message) throws InterruptedException {

        int dis_to_sender = distances.get(message.sender);
        boolean update_table = false;
        HashSet<Neighbor> neighbors = Network.getNeighbors(this);

        for (Neighbor neighbor: neighbors) {

            Integer dis_sender_to_router = message.distances.get(neighbor.router);
            if (dis_sender_to_router != null) {
                int dis_through_neighbor = dis_to_sender + dis_sender_to_router;
                if (dis_through_neighbor < distances.get(neighbor.router)) {
                    System.out.println("Faster Route Available");
                    update_table = true;
                    distances.put(neighbor.router, dis_through_neighbor);
                }
            }
        }
        if (update_table) {
            for (Neighbor neighbor: neighbors) {
                Message newMessage = new Message(this, neighbor.router,
distances);
                Network.sendDistanceMessage(newMessage);
            }
        }
```


Messages vs. Network Size
80
60
40
20
0
Messages
4
6
8
10
12
Network Size