# Written Homework 2: Lower Network Layers

## Due Friday, Feb 17

## 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.

Sending data from Application - rdt\_send

Receiving data to Application – rdt\_receive

Receiving data from the application in Transport layer – receive\_data

Sending data to the Application from Transport layer - deliver\_data

Sending data to client in Transport layer - udt\_send

Receiving data from client in Transport layer - udt\_rcv

Time Out = RRT + Congestion Fudge Factor

If the sender receives a duplicate, the it sends back the last send data again.

Checksum – Finding the odd or even 1s the following manner:

Ex-

1011 - 1

1010 - 0

1011 - 1

0011 - 0

------------

1001 | 0 There is a high chance of the final bit (0) getting flipped if there are changes in the incoming bits.

## Question 2: Throttling

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

Flow control is throttling the sender to protect the receiver end.

Congestion control is throttling the sender to protect the network channel.

**Flow Control:**

Flow control is basically changing the size of the sender’s window based on the fluctuation in the size of the receiver’s window.

**Receive Window = Number of free bytes (space) in my receive buffer.**

This information about the receive window is sent by the receiver with the ACK packets.

**Window on Sender Side -> (Send Base) to (Last Acked + Last Receive Window)**

Send Base is determined by Last Acked byte.

When the receive window is 0, the sender sends 1 byte at a time.

**Congestion control:**

Congestion control is done when the sender knows the packets are getting dropped (The sender knows packets are getting dropped when there is a time-out in receiving the ACKs or duplicate ACKs).

There are three modes of congestion control ->

**Slow-Start:** Initially, when the network is starting, packets are sent slowly to determine the network's capacity or when we get time-outs.

**Congestion Window – 1MSS (Maximum Segment Size).**

**Fast Recovery:** The sender data is throttled to fast recovery when we get 3 duplicate ACKs in a row.

**Congestion Avoidance:** The data transfer moves to congestion avoidance when things seem to look good. This is the best state where the congestion window is large.

Every time we get an ACK, the double Rate of transfer (exponential growth).

**MSS \*= 2;**

1 MSS..2 MSS..4 MSS… Basically, we are doubling the congestion window every round-trip time.

**Congestion Window += MSS;**

This exponential growth continues till the congestion window reaches the slow start threshold. Hence, now it’s in Congestion Avoidance mode.

**Congestion window > slow start threshold => transition to Congestion Avoidance mode.**

**In the Congestion Avoidance mode, we increase the Congestion window by 1MSS (linearly).**

We cut the congestion window in half when we switch to fast recovery. On receiving new, we switch back to **Congestion Avoidance mode.**

**Congestion Window /= 2;**

The mechanism for Congestion control is the same as Flow control i.e., controlling the window size of the sender’s end.

Considering both Flow and Congestion control ->

**Window size for sender = min (receiver window, congestion window);**

If the receiver does not have enough buffer to receive the messages, then the sender window will be affected based on the receiver window (flow control). But if the receiver end application is reading messages faster, than the window size will depend on the capacity of the network (congestion window).

## 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 - 10.0.0.1, 35492

Destination - 1.2.3.4, 80

\* from B to X behind the NAT

Source - 10.0.0.2, 32398

Destination - 1.2.3.4, 80

\* from A to X between the NAT and X

Source - 5.6.7.8, 23911

Destination - 1.2.3.4, 80

\* from B to X between the NAT and X

Source - 5.6.7.8, 08321

Destination - 1.2.3.4, 80

\* from X to A between X and the NAT

Source - 1.2.3.4, 80

Destination - 5.6.7.8, 23911

\* from X to A between the NAT and A

Source - 1.2.3.4, 80

Destination - 10.0.0.1, 35492

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

10.0.0.1, 35492 -> 5.6.7.8, 23911

10.0.0.2, 32398 -> 5.6.7.8, 08321

## 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?

Total subnets – 6

AB -> 1.1.4.00 / 31

AC -> 1.1.5.00 / 31

BC -> 1.1.6.00 / 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)?

/22

\* 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.

1.1.1.0/24 -> port 1

1.1.4.1/0 -> port 2

1.1.5.1/0 -> port 3

\*0.0.0.0/32 -> port 4

## Question 5: Routing

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