CS456 Fall 2019

Assignment 2 - Recovering from Packet Loss and Delay

Due: November 8th, Midnight
Language: Any

1 Purpose

The purpose of this assignment is to implement the **Go-Back-N** protocol, which could be used to transfer a text file from one host to another across an unreliable network. The protocol should be able to handle network errors such as packet loss and duplicate packets.

For simplicity, your protocol is unidirectional, i.e., data will flow in one direction (from the sender to the receiver) and the acknowledgments (ACKs) in the opposite direction. To implement this protocol, you will write two programs: a sender and a receiver, with the specifications given below. You will test your implementation using the provided network link emulator.

When the sender needs to send packets to the receiver, it sends them to the network emulator instead of sending them directly to the receiver. The network emulator then forwards the received packets to the receiver. However, **it may randomly discard and/or delay received packets.** The **same scenario happens when the receiver sends ACKs to the sender**.

2 Specifications

Note: these specifications are provided for Java, but you are free to use other languages.

2.1 Packet Format

All packets exchanged between the sender and the receiver should have the following structure:

```
public class packet
{
      private int type;  // 0: ACK, 1: Data, 2: EOT
      private int seqnum; // Modulo 32
      private int length; // Length of the String variable ‘data’
      private String data; // String with Max Length 500
}
```

The `type` field indicates the type of the packet. It is set to 0 if it is an ACK, 1 if it is a data packet, 2 if it is an end-of-transmission (EOT) packet (see the definition and use of an end-of-transmission packet below). For data packets, `seqnum` is the modulo 32 sequence number of the packet. The sequence number of the first packet should be zero. For ACK packets, `seqnum` is the sequence number of the packet being acknowledged. The `length` field

specifies the number of characters carried in the data field. It should be in the range of 0 to 500. For ACK packets, `length` should be set to zero.

2.2 Sender Program (sender)

You should implement a sender program, named **sender**. Its command-line input includes the following:

<host address of the network emulator>
<UDP port number used by the emulator to receive data from the sender>
<UDP port number used by the sender to receive ACKs from the emulator>, and
<name of the file to be transferred> in the given order.

Upon execution, the sender program should be able to read data from the specified file and send it using the Go-Back-N protocol to the receiver via the network emulator. The window size should be set to N=10. After all contents of the file have been transmitted successfully to the receiver (and corresponding ACKs have been received), the sender should send an EOT packet to the receiver. The EOT packet is in the same format as a regular data packet, except that its type field is set to 2 and its length is set to zero. Once the sender has received ACKs for all data packets it has sent, it will send an EOT to the receiver. When the sender receives an EOT from the receiver, the sender can close its connection and exit. **To keep the project simple, you can assume that the end-of-transmission packet never gets lost in the network.**

In order to ensure reliable transmission, your program should implement the Go-Back-N protocol as follows:

- If the sender has a packet to send, it first checks to see if the window is full, that is, whether there are N outstanding, unacknowledged packets.
- If the window is not full, the packet is sent and the appropriate variables are updated and a timer is started if not done before. The sender will use only a single timer that will be set for the oldest transmitted-but-not-yet-acknowledged packet.
- If the window is full, the sender will try sending the packet later.
- When the sender receives an acknowledgment packet with sequence number n, the ACK will be taken to be a cumulative acknowledgment, indicating that all packets with a sequence number up to and including n have been correctly received at the receiver.
- If a timeout occurs, the sender resends all packets that have been previously sent but that have not yet been acknowledged.
- If an ACK is received but there are still additional transmitted-but-yet-to-be-acknowledged packets, the timer is restarted. If there are no outstanding packets, the timer is stopped.

Further description of the GBN sender and receiver can be found in slides 47-50 of Chapter 3 Lecture Notes.

2.2.1 Timing

This assignment includes an experiment. In order to perform this experiment, the sender needs to record how long it takes for the receiver to successfully acquire the entire file. To do this, record the time just before sending the first packet and the time immediately after receiving the EOT from the receiver. From these two recorded times, compute the transmission time in milliseconds.

2.2.2 Output

For both testing and grading purposes, your sender program should be able to generate three log files, named as **seqnum.log, ack.log, and time.log**. Whenever a packet is sent, its sequence number should be recorded in seqnum.log. The file ack.log should record the sequence numbers of all the ACK packets that the sender receives during the entire period of transmission. The format for these two log files is one number per line. You must follow this format to avoid losing marks.

Place the transmission time in **time.log**. Do not place anything else in this file.

2.3 Receiver Program (receiver)

You should implement the receiver program, named **receiver**. Its command-line input includes the following:

<hostname for the network emulator>,
<UDP port number used by the link emulator to receive ACKs from the receiver>,
<UDP port number used by the receiver to receive data from the emulator>, and
<name of the file into which the received data is written> in the given order.

When receiving packets sent by the sender via the network emulator, it should execute the following:

- check the sequence number of the packet;
- if the sequence number is the one that it is expecting, it should send an ACK packet back to the sender with the sequence number equal to the sequence number of the received packet;
- in all other cases, it should discard the received packet and resends an ACK packet for the most recently received in-order packet;

After the receiver has received all data packets and an EOT from the sender, it should send an EOT packet then exit.

2.3.1 Output

The receiver program is also required to generate a log file, named as **arrival.log**. The file arrival.log should record the sequence numbers of all the data packets that the receiver receives during the entire period of transmission. The format for the log file is one number per line. You must follow the format to avoid losing marks.

Network Emulator (nEmulator)

You will be given the executable code for the network emulator. To run it, you need to supply the following command line parameters in the given order:

<emulator's receiving UDP port number in the forward (sender) direction>,
<receiver's network address>,
<receiver's receiving UDP port number>,
<emulator's receiving UDP port number in the backward (receiver) direction>,
<sender's network address>,
<sender's receiving UDP port number>,
<maximum delay of the link in units of millisecond>,
<packet discard probability 0 <= p <= 1 >,
<verbose-mode> (Boolean: if 1, the network emulator outputs its internal processing).

When the link emulator receives a packet from the sender, it will discard it with the specified probability. Otherwise, it stores the packet in its buffer, and later forwards the packet to the receiver with a random amount of delay (less than the specified maximum delay).

3 Experiment

After implementing the GBN protocol, measure the impact of packet loss and delay on transmission time.

For each combination parameters, execute your program THREE times and record the average transmission time (Section 2.2.1).

Set	**Maximum Delay**	**Packet Discard Probability**
Baseline	0	0
No Delay 1	0	0.1
No Delay 2	0	0.2
No Delay 3	0	0.3
No Delay 4	0	0.4
No Delay 5	0	0.5
No Discard 1	10	0
No Discard 2	20	0
No Discard 3	30	0

No Discard 4	40	0
No Discard 5	50	0
Both 1	20	0.1
Both 2	20	0.2
Both 3	20	0.3
Both 4	40	0.1
Both 5	40	0.2
Both 6	40	0.3

Repeat these experiments for the provided small input file, medium, and large input file.
Plot the results.

Produce a report of your findings, including the plots.

- How do changes in packet delay, probability of loss impact the transmission time?
- How do changes in file size (number of packets sent) impact transmission time?

4 Hints

- Use the packet class given in packet.java containing necessary declarations, definitions, and helper methods
- All the packets must be sent and received as byte arrays instead of as Java packet objects. Since the network emulator is written in C/C++, it cannot read Java objects. Necessary code to convert the packet class into byte array and vice versa is provided in the packet.java file.
- You must run the programs in the CS Undergrad Environment in order to allow nEmulator to work.
- Run nEmulator, receiver, and sender on three different machines in this order to obtain meaningful results.

Example Execution

1. On the host host1: nEmulator 9991 host2 9994 9993 host3 9992 1 0.2 0
2. On the host host2: java receiver host1 9993 9994 <output File>
3. On the host host3: java sender host1 9991 9992 <input file>

5 Hand-in Instructions

Submit all your files in a single compressed file (.zip, .tar, etc.) using LEARN, in the dedicated Dropbox. You must hand in the following files/documents:

1. all source code files

2. Makefile (your code **must** compile and link by typing **make** or **gmake**)
3. README file containing compile and runtime instructions
4. Modified server/client.sh scripts

Your implementation will be tested on the undergrad environment machines. **Please compile and test your code on these machines prior to submission.**

5.1 Late Policy

10% penalty for each day (24hrs) late up to a maximum of 3 days. Submissions will not be accepted beyond 3 late days.

6 Evaluation

See attached marking scheme.