CS3210 Computer Networks Lab

Even Sem. 2017, Prof. Krishna Sivalingam
Lab 6: Go-back-N Protocol
Due date: March 6, 2017, 11.45 PM, On Moodle
Extension: 15% Penalty for each 24-hr period;
Max. of 48-hrs past the original deadline

Feb. 21, 2017

The objective of this project is to implement the Go-back-N reliable transmission protocol and measure the round-trip delays. **Cumulative ACKs**, as described in class, will be used.

The project requires two separate programs, running at the same time, on two different hosts: a *sender* program that generates and transmits packets; and a *receiver*, that accepts the packets, and transmits the acknowledgments to the sender. Note that the receiver does not send any data packet; it only sends acknowledgments. Communication between the sender and receiver is through UDP sockets. The receiver is set up as a UDP server, and the sender is set up as a UDP client.

1 Sender

The main loop of the sender has these main steps:

1. Generate a packet of length PACKET_LENGTH (command-line parameter) bytes. The first byte(s) of the packet contains the sequence number.

   Packet generation rate is given by the command-line parameter PACKET_GEN_RATE, in packets per second.

   You might need to use a thread that generates packets periodically (based on the above rate) and stores them in a buffer used by the sender's protocol. The maximum size of this sender transmission buffer is given by the command-line parameter, MAX_BUFFER_SIZE (number of packets, not bytes). A newly generated packet will be dropped if the Buffer is full. A sequence number is assigned ONLY if the packet is added to the buffer.

2. Transmit the packet based on the Window Size conditions. Start the timeout timer for this packet's sequence number. The timeout is set to 100 ms for the first 10 packets and then $2 * RTT_{ave}$ (in milliseconds) for all other packets.

3. Process the next packet (when available) and transmit it if the sender window is not exhausted, i.e. the total number of unacknowledged packets is at most WINDOW_SIZE.

4. If an ACK packet arrives, process it, update local state variables and cancel timers corresponding to acknowledged packets. Note that cumulative ACKs are assumed.

   For each packet received, calculate the Round-trip-Time (RTT) for the packet and update the average RTT (RTT_{ave}) for the packets acknowledged so far.

5. If a timer expires, retransmit all packets from the first unacknowledged packet.

The sender terminates after MAX_PACKETS (a command-line parameter) have been successfully ACKNOWLEDGED (OR) if the maximum retransmission attempts for any sequence number exceeds 5.

Summary of Command Line Options: The command line options provided to the sender are listed below:

- -d – Turn ON Debug Mode (OFF if -d flag not present)
- -s string – Receiver Name or IP address.
- -p integer – Receiver's Port Number
- -l integer – PACKET_LENGTH, in bytes
- -r integer – PACKET_GEN_RATE, in packets per second
- -n integer – MAX_PACKETS
- -w integer – WINDOW_SIZE
- -b integer – MAX_BUFFER_SIZE

Output: The sender will operate in TWO modes: DEBUG and NODEBUG. The default operation is NODEBUG mode. A command-line flag of *-d* will turn on DEBUG mode.

For both modes, on termination, the sender will print the following information to the screen:

1. PACKET_GEN_RATE
2. PACKET_LENGTH
3. Retransmission Ratio: Ratio of Total Number of Transmissions (including Retransmissions) to Number of Packets Acknowledged.
4. Average RTT Value for ALL Acknowledged Packets

In DEBUG mode, the Sender will also print the following information for EACH packet when its ACK is received:

```
Seq #:   Time Generated: xx:yy RTT: zz      Number of Attempts: aa
```

where time is in milliseconds:microseconds format.

2 Receiver

The receiver is always waiting to read a packet from the UDP socket it is listening to. Whenever a packet is delivered to the receiver:

1. The receiver **randomly** decides that packet is corrupted and decides to drop the packet; note that you can use *rand, rand48*, etc. The probability of packet drop is specified as a command-line parameter, denoted RANDOM_DROP_PROB.

   This step is used to simulate random network errors.

2. If the packet is NOT corrupted (per step 1 above), the receiver reads the packet and extracts the sequence number. If sequence number matches the NEXT EXPECTED sequence number, it transmits an ACK to the sender, and updates local state variables.

   Note that Cumulative ACKs are used by the receiver.

   The ACK packets are NOT dropped and are always assumed to be delivered to the sender. Thus, RANDOM_DROP_PROB value is not used by the sender.

The receiver terminates after acknowledging MAX_PACKETS (a command-line parameter).

Summary of Command Line Options: The command line options provided to the receiver are listed below:

- -d – Turn ON Debug Mode (OFF if -d flag not present)
- -p integer – Receiver's Port Number
- -n integer – MAX_PACKETS
- -e float – Packet Error Rate (RANDOM_DROP_PROB)

Output: The receiver will operate in TWO modes: DEBUG and NODEBUG. The default operation is NODEBUG mode. A command-line flag of *-d* will turn on DEBUG mode.

In DEBUG mode, the Receiver will also print the following information for EACH packet when it is successfully received:

```
Seq #:   Time Received: xx:yy Packet dropped: false
```

where time is in milliseconds:microseconds format.

3 Sample Session

Assume that you have created the files SenderGBN.c and ReceiverGBN.c and the corresponding executables in your directory.

```
machine1% ./ReceiverGBN -p 12345 -n 400 -e 0.00001 &
```

```
m2% ./SenderGBN -s machine1 -c1 -p 12345 -l 512 -r 10 -n 400 -w 3 -b 10

Output: PktRate = 10, Drop Prob = 0.00001, Length = 512, Retran Ratio
= 1.16, Avg RTT: 100:34
```

4 What to Submit

The platform for this project will be Linux and C/C++/Java. Create a tar-gz file with name: Lab6-RollNo.tgz (e.g. Lab6-CS14B099.tgz).

The directory should contain the following files:

- Source Files for both parts
- Makefile and Script File
  Typing command 'make' or your script program, at the UNIX command prompt, should generate all the required executables.
- A Script file obtained by running UNIX command *script* which will record the way you have finally tested your program.
- a README file containing what port number to use, and instructions to compile, run and test your program.
- a COMMENTS file which describes your experience with the project, suggestions for change, and anything else you may wish to say regarding this project. This is your opportunity for feedback, and will be very helpful.

Hard-Copy Part:

All of the following items should be type-written and STAPLED, with your NAME and Student ID, clearly indicated on the first page.

- A printout of the README and COMMENTS file (which are also submitted on-line)
- A printout of the Report: A technical report that discusses the results obtained by running the programs on any two machines. Report your observations in 1-2 paragraphs. Discuss what happens when the random packet drop probability increases.

  The experiments are to be conducted for: packets per second; for values of PACKET_LENGTH $\in \{128, 1024\}$ bytes and RANDOM_DROP_PROB $\in \{10^{-7}, 10^{-4}\}$. Thus, there are a total of 4 possible combinations. Prepare TWO tables, one per packet drop probability.

5 Help

1. Ask questions EARLY and start your work NOW. Take advantage of the help of the TAs and the instructor.
2. Submissions PAST the extended deadline SHOULD NOT be mailed to the TAs. Only submissions approved by the instructor or uploaded to Moodle within the deadline will be graded.
3. Demonstration of code execution to the TAs MUST be done using the student's code uploaded on Moodle.
4. NO sharing of code between students, submission of downloaded code (from the Internet, Campus LAN, or anywhere else) is allowed. The first instance of code copying will result in ZERO marks for the Lab component of the Course Grade. The second instance of code copying will result in a 'U' Course Grade. Students may also be reported to the Campus Disciplinary Committee, which can impose additional penalties.
5. Please protect your Moodle account password. Do not share it with ANYONE, including your team member. Do not share your academic disk drive space on the Campus LAN.
6. Implement the solutions, step by step. Trying to write the program *in toto* in one setting may lead to frustration, more than anything else.

6 Grading

- Go-back-N working correctly: 80 points
- Report and Viva Voce: 20 points
- NO README, Typescript or COMMENTS file: -10 points