CS3210 - Computer Networks Lab
Even Sem. 2015, Prof. Krishna Sivalingam
Lab 3: Selective Repeat Protocol
Due date: Feb. 22, 2015, 9 PM, On Moodle

Feb. 4, 2015

The objective of this project is to implement the **Selective Repeat** reliable transmission protocol and measure the round-trip delays.

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.

Optionally, **RAW IP** sockets may be used instead of UDP sockets, but this has its own challenges. Refer `http://sock-raw.org/papers/sock_raw` and `http://en.wikipedia.org/wiki/Raw_socket` for more information about raw IP sockets.

1 Sender

The main loop of the sender has these main steps:

1. Generate a packet of length, where the packet length follows a uniform distribution: Uniform(40, MAX_PACKET_LENGTH) bytes, where MAX_PACKET_LENGTH is command-line parameter). The first byte(s) of the packet contains the sequence number (depends on the number of bits in the sequence number field).

   Packets are generated at periodic time intervals specified by the PACKET_GEN_RATE parameter (packets / unit time). The transmit buffer has a capacity specified by the BUFFER_SIZE parameter (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 conditions. Start the timeout timer for this packet's sequence number. The timeout is set to 300 ms for the first 10 packets and then $2 \times RTT_{ave}$ (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. Given an n-bit sequence number, the maximum window size will be 2^{n-1} for Selective Repeat.

4. If an ACK packet arrives, process it, update local state variables and cancel timers corresponding to acknowledged packets. Remove the packet from the Transmit Buffer. Note that selective ACKs are used.

For each sequence number acknowledged, 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, re-transmit only the 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 10.

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
- -n integer – Sequence Number Field Length (in bits)
- -L integer – MAX_PACKET_LENGTH, in bytes
- -R integer – PACKET_GEN_RATE, in packets per second
- -N integer – MAX_PACKETS
- -W integer – WINDOW_SIZE (Assume that SWS = RWS)
- -B integer – 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 by PACKET_ERROR_RATE.

   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 the receiver buffer is FULL, then the received packets are discarded even if they were correctly received. Otherwise, it follows the Selective Repeat protocol for generating ACKs, and buffering out-of-order packets.

   The ACK packets are NOT dropped and are always assumed to be delivered to 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
- -n integer – Sequence Number Field Length (in bits)
- -W integer – WINDOW_SIZE (SWS = RWS)
- -B integer – BUFFER_SIZE
- -e double – PACKET_ERROR_RATE

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. Note that the receiver will print this information ONLY in Sequence Number order. For example, if Seq. No. 3 is received before Seq. No. 2, then the receiver will NOT print information for Seq. No. 3 until Seq. No. 2 is received.

```
Seq #:   Time Received: xx:yy
```

where time is in milliseconds:microseconds format.

3 Sample Session

Assume that you have created the files SenderSR.c and ReceiverSR.c and the corresponding executables in your directory.

```
m1% ./ReceiverSR -p 12345 -N 400 -e 0.00001 -B 100

m2% ./SenderSR -s m2 -p 12345 -L 512 -R 10 -N 400 -W 4 -B 100 -n 8
```

4 What to Submit

The platform for this project will be Linux and C/C++. Create a tar-gz file with name: Lab3-RollNo.tgz (e.g. Lab3-CS10B099.tgz) that will contain a directory named Lab3-RollNo with all relevant files.

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 technical report (in PDF format) that discusses the results obtained by running the programs on any two machines. Report your observations and analyze the results, in 1-2 paragraphs. The report should include your name, roll number, assignment number and title.

  The experiments are to be conducted for: PACKET_GEN_RATE $\in \{20, 300\}$ packets per second; for values of PACKET_LENGTH $\{256, 1500\}$ bytes and RANDOM_DROP_PROB $10^{-3}, \{10^{-5}, 10^{-7}\}$. Prepare TWO tables, one per packet generation rate.

  Plot graphs with all required information: axes labels, legends if there are multiple plots in the same graph, etc.
- 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.

5 Help

1. Ask questions EARLY and start your work NOW. Take advantage of the help of the TAs and the instructor.
2. It will be easier if you start by implementing the Stop-and-Wait protocol and then extend it.
3. 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.
4. Demonstration of code execution to the TAs MUST be done using the student's code uploaded on Moodle.
5. 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.
6. Please protect your Moodle account password. Do not share it with ANYONE. Do not share your academic disk drive space on the Campus LAN.
7. Implement the solutions, step by step. Trying to write the program in one setting may lead to frustration and errors.

6 Grading

- Selective Repeat working correctly: 80 points
- Report: 20 points
- NO README, Typescript or COMMENTS file: -10 points