Programming Assignment #2
Ordered Delivery of Packets

April 9, 2013

1 Objective

- to implement a simple version of in-order delivery service required in many networking protocols and applications.

2 Submission Instruction

You are expected to submit using the online submission system using the upload file(s) link.

The submitted code file should be named A2.cpp
if your file has a different name, it will not be considered in the evaluation.

Submission Deadline is April 24 @ 3PM.

- Missing the deadline == No Marks for this assignment
- Submit even if your code is partially working
- Write the code yourself. Plagiarism (code copying) in any of the assignments ==> **-5 Marks**

3 Assignment Overview

- This assignment will be based on inorder linked list application in communications protocols.
- In communication networks, a sender can send a sequence of packets (i.e. units of information transported over the network) but due to unpredictable network conditions, these packets can be lost or arrive out-of-order at the receiver.
- We focus here on the case of out-of-order delivery of packets at the receiver. The receiver has a buffer in which it temporarily stores the incoming packets. It only delivers packets that are in proper sequence to the final application that needs the information sent by the sender. For example assume the sender sends packets 0, 1, 2, ..., 9. The receiver receives the packets in the following time instants:

  Time Received Packet

  1 2

  2 0

  3 1 –> 0, 1, 2 are now in order –> Deliver packet 0, 1, 2

  4 3 –> 3 in order with 0,1,2 –> Deliver 3

5 5

6 5 (duplicate reception, ignore)

7 4 –> 4 and 5 are inoder –> Deliver 4, 5

8 4 (duplicate reception, ignore)

9 7

10 8

11 7 (duplicate reception, ignore)

12 6 –> 6, 7, 8 are inorder –> Deliver 6, 7, 8

and so on.

- The packets will be stored in an inorder list and when a sequence of expected ordered packets are received they are removed from the inrorder list.
- The receiver is initialized with the sequence of next expected received packet and a maximum number of outstanding packets it can store (this is known as window size in networks terminology).
- The sequence number is represented by a finite number of bits. So for example, if the number of bits of the sequence number is 8, then we can have the sequence space 0, 1, . . . , 63. In this case, we define a parameter called MAXSEQNU which will be equal to 64.
- The sequence number can wrap and restart from 0. So, the sequence of 61, 62, 63, 0, 1 is actually perfectly in sequence.
- The maximum window size cannot exceed the size of the sequence space (MAXSEQNU) divided by 2.
- If the receiver receives a packet with sequence number outside of allowable sequence space, then it is dropped. For example if the receiver's window size is 32 and the next expected received packet is 0 and it receives packets 1 and 2 and 33, then it will have to drop packet 33. If it receives packet 17, then 17 is stored. Moreover, when a set of inorder packets are delivered to the application, the next expected received packet is set to the sequence number of the last delivered inorder packet + 1 (Modulu MAXSEQNU).

4 Typical Operaiton

4.1 Input and Output lines

- In all the following, a2.exe is assumed to be the name of your executable file
- Input line include
  - a2.exe seqNumBits initSeq pid1 pid2 .....
    - where seqNumBits represents the number of bits used by the sequence number,
    - initSeq represents the first packet expected to be received
    - pidx are packet ids received by the end host.
  - OUTPUT: should be R [list of in order received packet IDs] E [next expected packet ID] W [orderd list of out of order packets] D [list of dropped packets]

4.2 ERROR Handling

- You should check for the correctness of every command
  - Wrong seqNumBits should print "**Invalid seqNumBits**"
  - Wrong initSeq should print "**Invalid initSeq**"
  - Wrong packet ID should print "**Invaid packet ID**"
- Note that Error messages are case sensitive.

4.3 Example Test Cases

- Input: a2.exe 4 3 3 4 5 4 9 7 7 6 10 15
  Output: R 3 4 5 6 7 E 8 W 9 10 D 4 7 15
  - R 3 4 5 6 7 indicates the reception of pacekts 3 - 7
  - E 8 indicates that the application is expecting packet 8
  - W 9 10 indicates that 9 and 10 are out of order packets waitinf for delivery
  - D 4 7 15 indicates that the repeated packet 4, 7, and 15 are dropped. Notes that packets may be dropped if they are out of window or due to duplicate reception
- Input: a2.exe 4 3 3 4 5 4 4 9 6 7 8
  Output: R 3 4 5 6 7 8 9 E 10 W D 4 4
- Input: a2.exe 8 58 59 58 60 61 63 0 2 4 32
  Output: R 58 59 60 61 E 62 W 63 0 2 4 D 32
- Input: a2.exe 4 3 4 5 6 7 a 4
  Output: **Invaid Packet ID**
- Input: a2.exe 4 3 4 5 6 7 100 4
  Output: **Invaid Packet ID**
  - $100 >$ max seqNumber $2^4 - 1 = 15$
- Input: a2.exe m 4 5 6 7 a 4
  Output: **Invaid seqNumBits**
- Input: a2.exe 3 r 5 6 7 a 4
  Output: **Invaid initSeq**