CS 2110: Computer Programming Lab (Odd 2012)
Simulating a Network Buffer (Part 1)
(Lab 2)

Instructor - John Augustine
Author - Tejas Kulkarni

August 6, 2012

Notes

- We are starting a new series of labs in which we are going to simulate a network buffer. For the next lab, you will be required to build on this assignment. Therefore, make sure that your current assignment is complete and bug-free BEFORE you come for the next lab.
- As this may be be first large program for many of you, optimize it for readability. Use the coding standards mentioned in the associated document in moodle. Also add comments that help readers follow your code.
- To avoid the code duplication, use functions. Each function that you write must be preceded by a comment that explains the input parameters, the output, the side effects (if any) and a brief description of the reasoning behind your solution.
- You are required to do a lot of string comparison in this program.You need not write string-related functions from scratch. Please use the built-in string library functions.
- Make use of *Dynamic memory allocation* and *pointers*. Do not forget to release allocated memory when you are deleting an item.

After successfully finishing your second year, you are hired by a start-up SoftNet solutions as a summer intern. SoftNet is a networking software development company. For the first time you are working in a software company and you are very happy with the friendly work culture there. In one nice afternoon, over a cup of coffee, your boss shares the concern of managing buffers in large networks. As you may know, computers communicate with each others via packets. The packets may arrive faster than they are sent out. So each computer in a network has a buffer that holds all the packets it has received but yet to send. A computer in the network needs to store all the incoming packets in an effective manner so that it can perform standard operations like

- adding a newly received packet to the buffer,
- searching for a particular packet,
- sending a particular packet,
- etc.

After some discussion with your boss, you decide that your program should have the following features.

- Your program should be user friendly and menu-driven. At any given point in the execution, the user must have a list of options to choose from and the program must respond appropriately.

- A network packet is analogous to an envelope in which we insert a letter containing your actual message. On the envelope you write your (i.e., sender's) address and your friend's (i.e., receiver's) address.
- Your program must maintain a list (first singly linked list, then doubly linked list) that will act as the buffer that stores a list of packets.
- A user of your program must be able to simulate receiving a packet by keying in a message, sender's address and receiver's address and the packet must get added to the buffer.
- A user must be able to simulate sending a packet by (i) searching for a particular message, (ii) printing a send message, and then (iii) deleting it from the buffer.

You initially think that it is tough but after some thinking, you realize you can be easily done! So here is how you proceed.

Step 0. Create a project in NetBeans called "NetworkSeries_1" under folder "lab2". Recall that in the previous lab, we had outlined how this can be done in the context of lab 1. Make sure that main.c is created.

Step 1. You create a structure called Packet. It should have following fields.

- Sender's address (string)
- Receiver's address (string)
- Actual message text (string)

As a good programmer, you create a new header file called "packet.h" for it. In NetBeans, to create a header file, you click on File → New File, and then choose "C Header File" and then click next. You will be asked to provide a name (packet). Finally, click Finish.

Step 2. Write a menu driven program to implement a buffer using singly linked list. You should offer the following menu options to the user.

- Receiving a packet.
- Sending a particular message
- Sending all messages from a particular sender
- Sending all messages from a particular receiver
- Printing all the messages currently in the buffer.

You must create source files "singly.h" for declaring all functions and "singly.c" for defining all the functions related to singly linked list.

Step 3. Your mentor in the company who is more experienced than you, suggests you to also implement a circular doubly linked list. At the start of the execution, allow the user to choose either singly linked list or circular doubly linked list. Once the choice is made by the user, the program must execute as defined earlier but using the appropriate data structure as chosen by the user. For this purpose, you will have to create files "doubly.h" and "doubly.c".

Finally you should have 6 files .

1. main.c (Contains the menu and the main function) – in programmer's language it is called as "Driver file"
2. packet.h
3. singly.h
4. singly.c
5. doubly.h
6. doubly.c