Computer Networks HW2 Report

B03901078 蔡承佑

January 10, 2017

1 Overview – Data Structure

In the header file `mySocket.h` contains the backbone class definition for three programs, `myServer` and `myClient`. This way provides a rather simple interfacefor higher level programs. Besides, there are still two classes to be introduced, `Data` and `Buffer`. `Data` encapsulates the information (similar to header) with which we need to transfer with the content of the package. `Buffer` provides some useful member function for `receiver.cpp` to call to maintain its buffer.
Three executable files `sender`, `agent`, `receiver` are further introduced by the state diagram in the following page.

2 Execution environment and Library

C++11 under UNIX environment. Only STL are used. Only OOP of C++ is used. No C++-only function or library used.

3 Difficulties and Solution

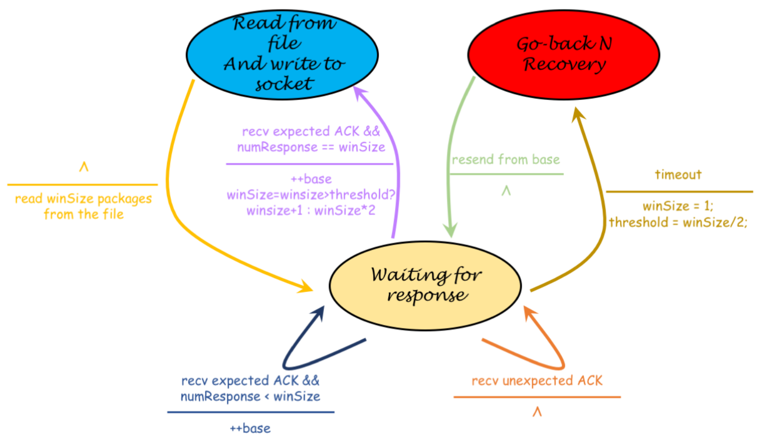
3.1 UDP

UDP is quite a different concept from TCP, and therefore a brand new topic from the HW1 and assignments in System Programming. Honestly, it did take much time to realize the operation of UDP, and to design a data structure for it. After having got those codes understood, since there are minor difference between a server and a client in UDP, I decided to seperate them into two classes, though it can be more clear using inheritance.

3.2 Pipeline

One-by-one transmission is rather simple. But pipeline transmission requires more issues to be maintained. To understand all principles in TCP also takes quite long time. At last, I found that the most complex part lies in sender. What should be handled in agent and receiver is rather simple. Agent just foward the messages come to it, and receiver just replies ACK if buffer is not full. All problems comes from how sender should response to these information. Just figure out conditions that the sender should face, this problem becomes simple, too. Details are explained in graphs in the follow page.

4 FSM of three programs



Read from file And write to socket
Go-back N Recovery
Waiting for response
Λ
read winSize packages from the file
recv expected ACK && numResponse == winSize
++base winSize=winsize>threshold? winsize+1 : winSize*2
resend from base
Λ
timeout
winSize = 1; threshold = winSize/2;
recv expected ACK && numResponse < winSize
++base
recv unexpected ACK
Λ

Figure 1: Sender

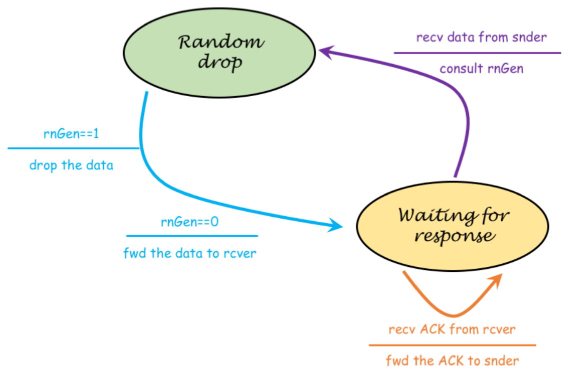


Figure 2: agent

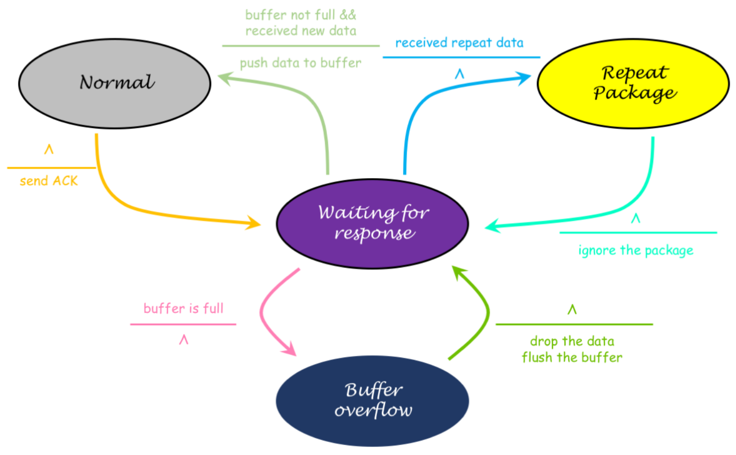


Figure 3: receiver