Full name: Tianbo, Li

SID: 3753751860

c. What you have done in the assignment.

Besides Phase 1 features, I have created TCP connection between five students and one admission office. Then, UDP connections between students and admission office. And UDP connections between admission office and department.

Besides what it is said on the project pdf description, I have added more communications which I think is necessary for stable delivery.

1. According to the description, admission office will firstly receive 3 packets (actually it’s 4 packets includes students’ name) from each student. During the debugging, I noticed that sending several packets together may cost one of them drop randomly. The recv() may not receive all of packets and process next line of code which will produce errors in runtime. Since recv() is a very high level function which provides little hints to me to debug where is wrong, I have to add one more RTT between admission office and each student. So in the program, student will firstly send the size of his application (size of those 4 packets) to the office, then, office reply to the student that his first message is received. After student got the ACK, he can send his packets to the admission office. Using MSG\_WAITALL and size of the packets, there is no need to worry about the message dropping anymore.
2. Same as the above, recvfrom() also needs one more RTT to know how many recvfrom() we need to executes.

d. What your code files are and what each one of them does.

(Please do not repeat the project description, just name your code files and briefly mention what they do).

Student.cpp: program to initialize Student, fork() is used to create five students.

Student.h: header file for student.

Admission.cpp: main file for admission office, it functions as a server.

Admission.h: header file for Admission.cpp

Department.cpp: main file for department, fork() is used to create one more process to represent two departments.

Department.h: header file for department.cpp

makefile: on the terminal, type "make Admission" to compile

Admission.cpp, "make Department” Department.cpp, “make Student” to compile Student.cpp

NOTE \*\*IMPORTANT\*\*: Following two files are input files for department.cpp. I hard coded the file names in the program. So the input file names have to be matched.

departmentA.txt: input files for program requirement of department A;

departmentB.txt: input files for program requirement of department B;

The input files for Student.cpp are (all NOT capitalized):

student1.txt, student2.txt, student3.txt, student4.txt, student5.txt

e. What the TA should do to run your programs. (Any specific order of events should be mentioned.)

On the terminal, type "make Admission" to compile Admission.cpp, "make Department" to compile Department.cpp, “make Student” to compile Student.cpp.

Then, type "Admission" to run the server (Admiision Offce), then, open another terminal, type "Department" to run the client (only call “Department” once!). Open another terminal, type “Student” to run 5 students (only call “Student” once!).

Note: fork() is used in Department.cpp and Student.cpp, so we only need to call it once.

In order to complete one entire communication described in pdf. First, type “Admission” to start admission office, then, type “Department” to start sending program requirement (phase I is done here). Type “Student” to send applications. Phase II is done. We can see the result at the Department’s terminal.

f. The format of all the messages exchanged should follow the ones as given in the table.

Yes, the screen messages follow the requirement exactly.

g. Any idiosyncrasy of your project. It should say under what conditions the project fails, if any.

Everything works fine. I used files to store the packets from two clients, named them "outputA" and "outputB".

The grader/TA doesn't need to worry about them. She can delete those two files when grading is done.

Besides outputA and outputB, the program Student will generate 5 files called Student1, Student2, Student3, Student4, Student5 (capitalize letter and NO postfix “.txt”). To get rid of shared memory, I have to use files to store messages received from child process. TA/grader can delete them after you done grading.

h. Reused Code: Did you use code from anywhere for your project?

If not, say so. If so, say what functions and where they are from.

(Also identify this with a comment in the source code.)

I used server.c and client.c from beej's website as a template to finish phase I.