CPE464 Lab 7 – Just UDP part of the lab

Using UDP and sendtoErr library

In this assignment you are going to work with the UDP code and sendtoErr library provided by Prof. Smith. This work will help you in program #3.

When you are done your udpClient/udpServer (rcopy/server) will:

- udpClient (rcopy) must take in an error rate, read in user input, create a PDU, print that PDU and then send (using sendtoErr()) the PDU to the server, recvfrom() the PDU sent back from the server and print out this received PDU.

- udpServer (server) must also take an error rate, recvfrom() the PDU from the udpClient, print out the PDU that it received and send this PDU back to the updClient.

Steps:

a. **Get the UDP code:** Download the UDP code from Canvas and untar the file.
   - Test the code to make sure it works

b. **Install sendtoErr() library:** see directions on Canvas in the program #3 section
   - You want to be in the directory with your UDP code before you install the library. The library makefile (build464lib.mk) creates library ('.a') file.

   - After downloading the library, untar it and build it according to the directions

c. **Verify the sendtoErr() library**. This library is called libcpe464_2_21.a (the 2_21 may change). To verify the library type: ar xv libcpe464_2_21.a

   This should give you a list of the object (.o) files in the .a (library) file.

d. Modify your Makefile to link in the library.

e. Now you are going to integrate your UDP code and the sendtoErr() library.

Overview of steps to implement:

1. (The details of these steps are given in the next section, this is just an overview)

   In file **separate** from your client and server code:

   a. Write a function to create an application level PDU in the proper format for program #3. This includes using the in_cksum() function to generate the PDU checksum which then gets placed into your PDU.

   b. Write a function to print out a PDU.

2. Modify udpClient.c to use your create PDU function and to print out your PDU.

3. Modify udpServer.c to print out the received PDU using your print PDU function.
4. Modify your client and server to use the sendtoErr_init() and sendtoErr() functions.

You will use the **sendtoErr()** function that I provide in program #3 in place of the normal sendto() function. This function drops packets and flips bits but does not tell you. The sendtoErr() function's debugging output expects your packets to be in the format (e.g. PDU header) required by program #3.

Detailed Lab Steps (do these):

a. Change the code and executable names from udpClient (and udpClient.c) to rcopy (rcopy.c) and udpServer (and udpServer.c) to server (server.c). You will also need to update the Makefile.

b. **Modify the rcopy.c and server.c code to take an error rate** (so the rate of dropping/corrupting packets) as a runtime parameter. This error rate will be between 0 and less than 1. The atof() function converts a string to a double. Example runs look like:

Bash$: server **.05** 44444
- This would have a 5% (i.e. ..05) error rate
- The optional server port number (e.g. 44444) follows the error rate

Or

Bash$: rcopy **.1** localhost 44444
- This would have a 10% (i.e. .1) error rate
- The server name/IP and port number follow the error rate

c. **Create a function that will be used to create a PDU.** After you create this PDU, it should contain the seq#, checksum, flag and payload. **This function must be in a separate file.**

The required PDU format (for both the lab and program #3) is:

4-byte sequence number in **network order**	2-byte checksum	1-byte flag	Payload (up to 1400 bytes)

Your function should look similar to:

```
// pduBuffer is the buffer you fill (passes back to calling function)
// sequenceNumber = 32 bit sequence number in network order
// flag = the type of the PDU (e.g. flag = 3 means data)
// payload = payload (data) of the PDU (may NOT be null terminated)
// dataLen = length of the payload (so # of bytes in data),
//           this is used to memcpy the data into the PDU
// returns the length of the created PDU

int createPDU(uint8_t * pduBuffer, uint32_t sequenceNumber, uint8_t flag, uint8_t * payload, int payloadLen)
{
    // create PDU code goes here

    return pduLength;
}
```

Your createPDU() function takes in a 32-bit sequence number, a 1-byte flag, a unit8_t * pointer to a buffer of data and an length (int) of the data buffer. This function puts the sequence number into **network order** prior to putting the number in the PDU. This function also uses the checksum function to calculate a checksum (crc) and put that into the header. This function creates a PDU in the format required by program 3 (seq#, crc, flag, payload). Put this function in a separate .c file and create a .h file for your server and client code.

Note – the checksum function (in_cksum()) is the same checksum function you used in program #1. The sendtoErr() library contains this function and it will be linked into your executable as part of the library. The prototype for this function is:

```
unsigned short in_cksum(unsigned short *addr, int len);
```

d. **Create a print function** that takes in a PDU, verifies the checksum and prints out the sequence number, flag, payload and payload length (note for initial testing the payload will be text). Put this function in the same .c files as the createPDU() (and update your new .h)

void outputPDU(unit8_t * aPDU, int pduLength);

e. **Modify rcopy.c to utilize the createPDU() and outputPDU() functions**.

- Since this is just an example program, for the sequence number just use a counter and increment it every time you create a new PDU.
- The payload of the PDU is the message typed in by the user.
- Use your outputPDU() to verify your code works!

f. Modify server.c to utilize the createPDU() and outputPDU() functions.

g. Integrate the sendtoErr() library into the rcopy.c and server.c.

i. Add a call to sendtoErr_init() in both main() functions. In this example errorRate is the value passed in at run time (see easlier step). You also need to include cpe464.h in your .c file to use this function.

```
sendErr_init(errorRate, DROP_ON, FLIP_ON, DEBUG_ON, RSEED_OFF);
```

ii. Change all sendto() calls used by rcopy/sever to **sendtoErr()**. When using the sendtoErr() function you need to include the cpe464.h in the file.

Demo and what to turn in:

- You will demo this next week in lab.
- You MUST handin your code (Lab7_UDP) on the Unix machines and upload a PDF of the finished lab worksheet to Canvas by 11:59 pm this Friday.