

CS 455 - Computer Networks

Homework 9

Due: See class webpage.

- **Assignment: Reliable Transfer with Stop-and-Go Protocol using UDP**

In this project, you will **implement** the **Stop-and-Wait protocol** to enable **reliable file transfer** between the **send** and **recv** programs **unreliable** UDP sockets.

You will **write** and **turn in two programs**:

- **send.c**

- The **send** program takes **4 arguments**:

```
send filename hostname port# timeout
```

and will **send** the file **filename** to the **socket address (hostname, port#)**

The **parameter timeout** specifies the **timeout value** in **milli-seconds** used in the **stop-and-wait protocol**

- **Requirement:**

- The **send** program must **exit** after **receiving** the **acknowledge** for its **last packet transmission**

Example:

```
send myFile lab1a 8000 5000
```

will send the **file myFile** in the **working directory** to **lab1a** at **port 8000** using **stop-and-wait** with **timeout value 5000 msec (= 5 sec)**

- **recv.c:**

- The **recv** program takes **2 arguments**:

```
recv filename port#
```

and will **receive lines of text** of **port port#** and **save** the **lines of text** in the file **filename**

- **Comment:**

- The **recv** program does **not** need to **exit**.

- You need to **kill** the **recv** program using **control-C** when you see the **send** program has **exited**

The **communication** using **UDP ports** is **unreliable**.

You must **implement** the **stop-and-wait protocol** in the **sender (send.c)** and **receiver recv.c** to make the **communication (= packet transmissions) reliable**

- **Packet structure used in your network programs**

- Use the following **struct data type** in the **communication** between **send.c** and **recv.c**:

```
struct Packet
{
    int done;           // contain 0 if not done, contains 1 if done
    int seqNo;          // The seqNo used in Stop-And-Wait
    char line[1000];    // The next line in the text file
};
```

- **Structure of the sender send.c**

- The **sender send.c** will **open** the **input file** and read the **input file one line at a time** and **send the line** to the **destination**

Here's is the **pseudo code** of **send.c**:

```
1. create an UDP socket s
2. bind socket s to a port of the localport

3. open input file

    FILE *fp = fopen( argv[1], "r"); // CS450 ?

4. read until file is exhausted:

    char buf[1000];

    while ( ( fscanf(fp, "%s\n", buf) > 0 )
    {
        ...
    }

5. Send the line in buf[ ] in a packet (struct Packet) to
   the destination using the Stop-and-Wait algorithm

   Set the done variable in the Packet struct to 0 for these packets !

   That means:

       send the packet
       wait for an ACK
       if ( ACK not received before timeout )
           repeat the send and wait !!!

6. When the file is completely sent, send a Packet with

    done = 1
```

and wait for the ACK

When ACK is received, the sender will exit

Consult the **online nodes** on the **Stop-and-Wait** protocol for **details**: [click here](#)

- **Structure of the receiver `recv.c`**

- The **receiver** will **created** a **UDP socket** on the **specific port** and **write** all the **lines** received on the **port** to an **output file**
- The **pseudo code** for **`recv.c`** is:

```
1. create a UDP socket s
2. bind socket to the specified (local) port # in command argument

3. open output file:

    FILE *fp = fopen( argv[1], "w" );    // CS450 ?

4. loop and receive packets:

    while ( true )
    {
        receive packet P;

        write the line in packet P to file IF packet is new
        You write the line with this call:

            fprintf( fp, "%s\n", line-in-packet );

        if packet P contains done = 1 then
            close the file (use: fclose(fp))

        You will need to implement the Stop-and-wait ACK scheme here
    }
```

The **receiver** will **never exit** (that's because the **ACK messages** sent by the **receiver** can be **lost**).

If the **receiver exits** after sending the **ACK** for the **done = 1** message **and** this **ACK message** is **lost**, then:

- The **sender** will **keep** sending the **done = 1** message and will **never** receive an **ACK** from the **receiver** (because the **receiver** has **exited** !!!)

- **Dropping packets**

- Because you will be running the program in lab machines that are connected by a **highly reliable LAN**, there will be no opportunity to let you find errors in your code if you do have bugs in the **Stop-and-Wait algorithm**.

So I have **rigged** "**`sendto()`**" function to perform "**artificial**" **packet dropping**. I have provided a special library **`libcs455.a`** in the directory **`/home/cs455001/lib`** that you must link into your code.

You **must compile** your **`send.c`** and **`recv.c`** programs **as follows**:

```
gcc -o send send.c -L/home/cs455001/lib -lcs455
gcc -o recv recv.c -L/home/cs455001/lib -lcs455
```

When you **compile** a **UDP network program** in the **above manner**, the **sendto()** function will **exhibit** the **unreliable send behavior** more **frequently**

After **compiling** using the **above commands**, you can **test** your **send** and **recv** programs on **2 lab machines**

Example:

```
Run the receiver on lab1a:

lab1a: cd ~/tmp
lab1a: ~/cs455/recv output 8000

Run the sender on lab1b:

lab1b: cd ~/tmp
lab1b: create a text file "data" (with gedit) - enter some lines and save
lab1b: ~/cs455/send data lab1a 8000 1000
```

The **sender** will **exit** when it finish transmitting. When you **see** the **sender** exit, kill the **receiver** with **control-C**

Compare the files **data** and **output** and they **must** be **identical** (use **cat output** to show the **file content**)

You can use this **command** to **compare** the **content** of **2 text files**:

```
diff file1 file2
```

If the **diff** report **nothing**, then the **files file1** and **file2** are **identical**

- **Warning**

- **Be careful** if you run your **tests** inside your **CS455 project directory** where you **store** your **program files** !!!

Because:

```
■ The receiver will create an output file

■ If you use a filename like recv.c for your receiver, then you will overwrite your project !!!
```

- **Network program examples discussed in class**

- You can start your project using the **UDP sender** and **UDP receiver** discussed in **class**
- I have put the **source code** of the **UDP sender** and **UDP receiver** discussed in **class** here:

```
/home/cs455001/Handouts/udp/sender.c
/home/cs455001/Handouts/udp/receiver.c
```

- You can **experience** the **unreliable communication** of **UDP** by running the **sender and receiver**:

```
To compile:
```

```
gcc -o sender sender.c -L/home/cs455001/lib -lcs455
gcc -o receiver receiver.c -L/home/cs455001/lib -lcs455
```

Run the **sender** and **receiver** as follows:

```
On lab2a: receiver 8000 // Receiver uses port 8000 to rcv data
On lab1a: sender lab2a 8000 // Sender will send data to (lab2a, 8000)
```

- **Test files you can use to transfer**

- I have 2 **text file** that you can use to **test your program**:

```
/home/cs455001/Handouts/udp/Test.small
/home/cs455001/Handouts/udp/Test.large
```

- When you finish your program, you can **test** the **send/rcv** programs using:

```
On labX: rcv output 8000 // Run rcv on labX using port 8000
On labY: send /home/cs455001/Handouts/udp/Test.small labX 8000 5000
```

When the **send** program **exists**, type **control-C** to kill the **rcv** program

Then use the **diff** command to check **difference** between the input file and the received file:

```
cat output // Show content of received file
cat /home/cs455001/Handouts/udp/Test.small // Show input file
diff output /home/cs455001/Handouts/udp/Test.small // Check difference
// between input and output
```

If you pass the **small file** test, try the program on the **large file**.

- **Turnin**

- **Turn in** your **send.c** and **recv.c** programs using these commands

```
/home/cs455001/turnin send.c udp-1
/home/cs455001/turnin recv.c udp-2
```

- **Extension request**

- To request an **extension** for the **UDP network programming project**, use the following command:

```
/home/cs455001/req-ext udp
```