Project Assignment: COMP4621

Build a p2p file sharing application

Spring 2024

TA: Xinyu YANG
E-mail: xinyu.yang@connect.ust.hk

Outline

- Review of concurrent client-server
- Multi-thread method (**pthreads**)
- Programming project introduction

Review

We have learned multiple technique for concurrency:

1. fork(): Creating multiple processes. Fork a new process for each connection.
2. settimeout: Setting a timeout for each connection so that the process wouldn't be occupied by a single connection.
3. poll()/select(): Using an overall mechanism to manage all blocking operations (e.g., recv(), listen()).

Today, we introduce another technique:

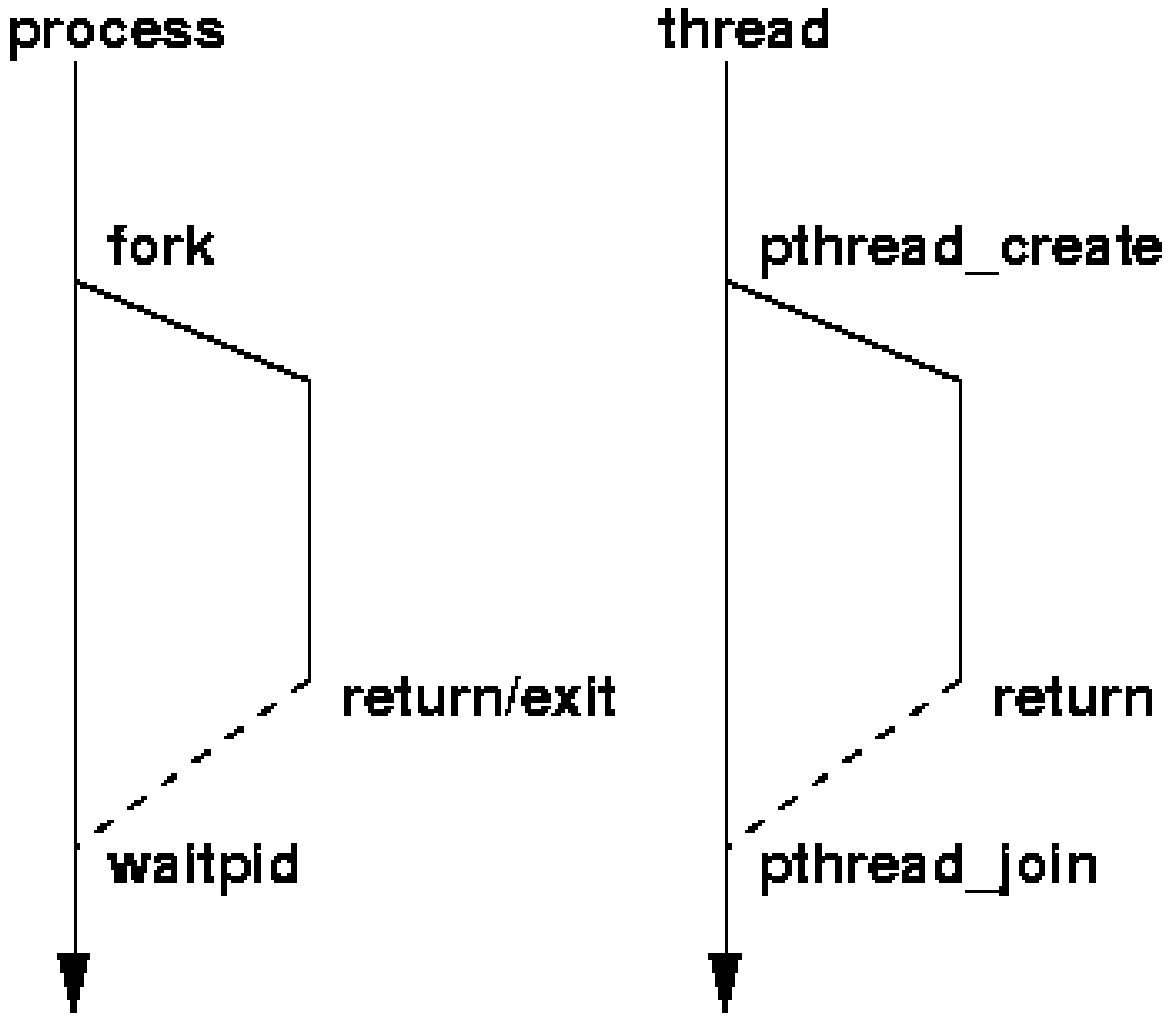
4. multi-threading: a light-weight fork().

Pthreads

Pthreads is a set of threading interfaces developed by IEEE, responsible for specifying the Portable Operating System Interface for UNIX (POSIX).

Pthreads specifies APIs to handle most of the behavior required by threads, including creating and terminating threads, waiting for threads to complete, and managing interactions between threads.

In the Linux/MacOS environments, you can consult the man pages for pthread functions such as pthread_create().



Pthreads

```
int pthread_create (
      pthread_t*   thread,
      const pthread_attr_t*   attr,
      void*  (*start_routine  ) ( void *),
      void*  arg_p) ;
```

Allocate thread before calling.

We won't use and just set NULL.

The function that the thread is to run.

Pointer to the argument that should be passed to the function *start_routine*.

Return value: on success, pthread_create() returns 0; on error, it returns an error number.

Pthreads

```
#include <pthread.h>
int pthread_join(pthread_t thread , void ** retval);
```

- Description:
  - wait for the thread specified by thread to end in a blocking manner. When the function returns, the resources of the waiting thread are reclaimed. If the thread has ended, the function returns immediately.

- Arguments:
  - thread: the thread identifier, which identifies a unique thread
  - retval: a user-defined pointer used to store the return value of the waited thread

- Return value: On success, pthread_join() returns 0; on error, it returns an error number.

Pthreads

#include <pthread.h>

void pthread_exit(void *retval);

- Description:
  - terminate the calling thread and returns a value via retval

- Arguments:
  - retval: a pointer of void* type can point to any type of data, and the data it points to will be used as the return value when the thread exits

- Return value: This function does not return to the caller.

You can find a lot more details about Pthread in this short tutorial on the web

Programming project

Find the requirement on Canvas.

Read the materials carefully, especially the marking scheme.

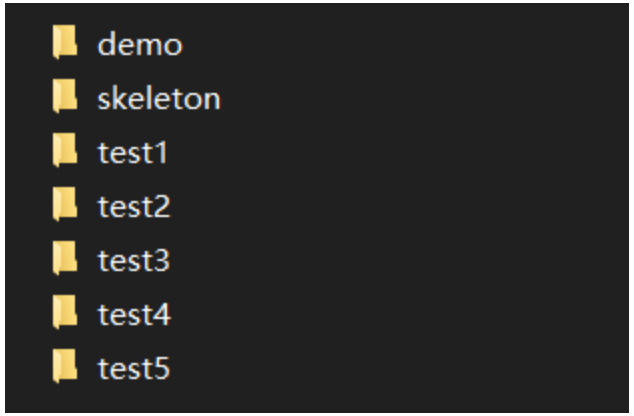
The comments in code is very helpful.

Any questions, post your question on the Canvas or Facebook group.

Lab 9 (Tue Apr 16): Threads, and project demo

code demo video

marking scheme slide



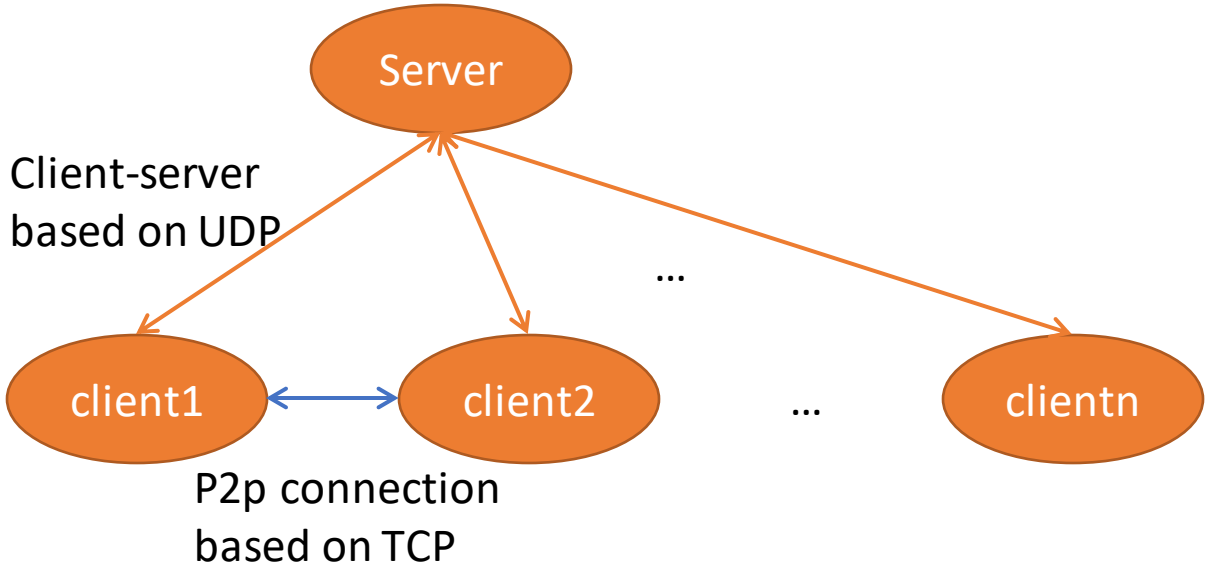
Programming project description

Programming assignment object

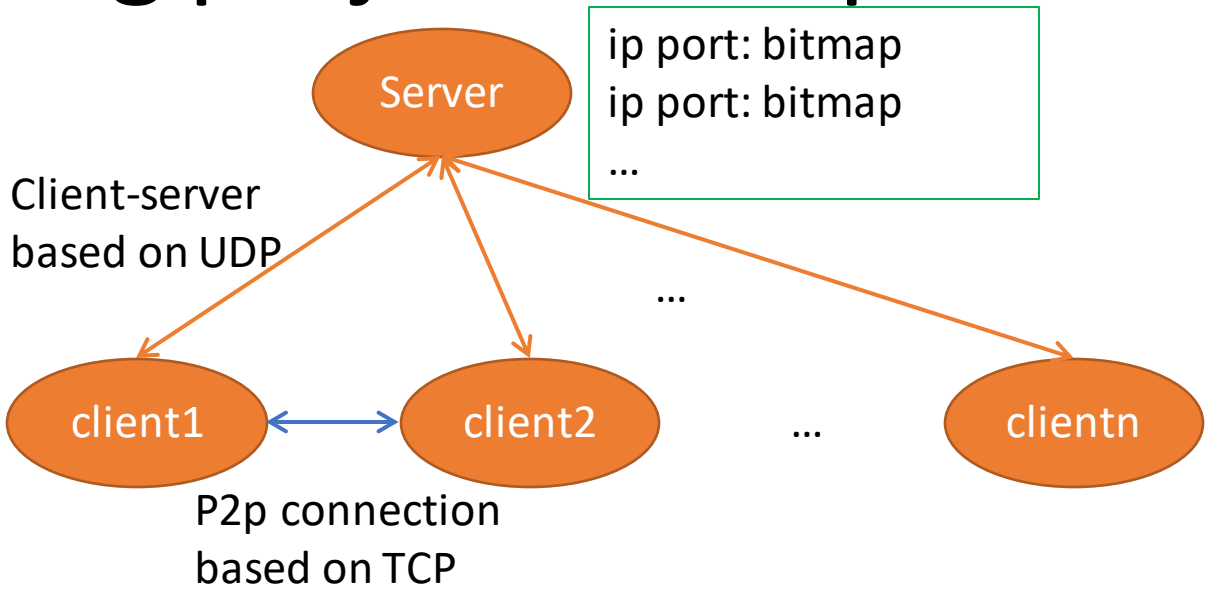
Build a p2p application with a centralized directory server.

Description and requirements

Your objective is to create a p2p application for file sharing and a client-server application for querying the file information from the directory server like napster.



Programming project description

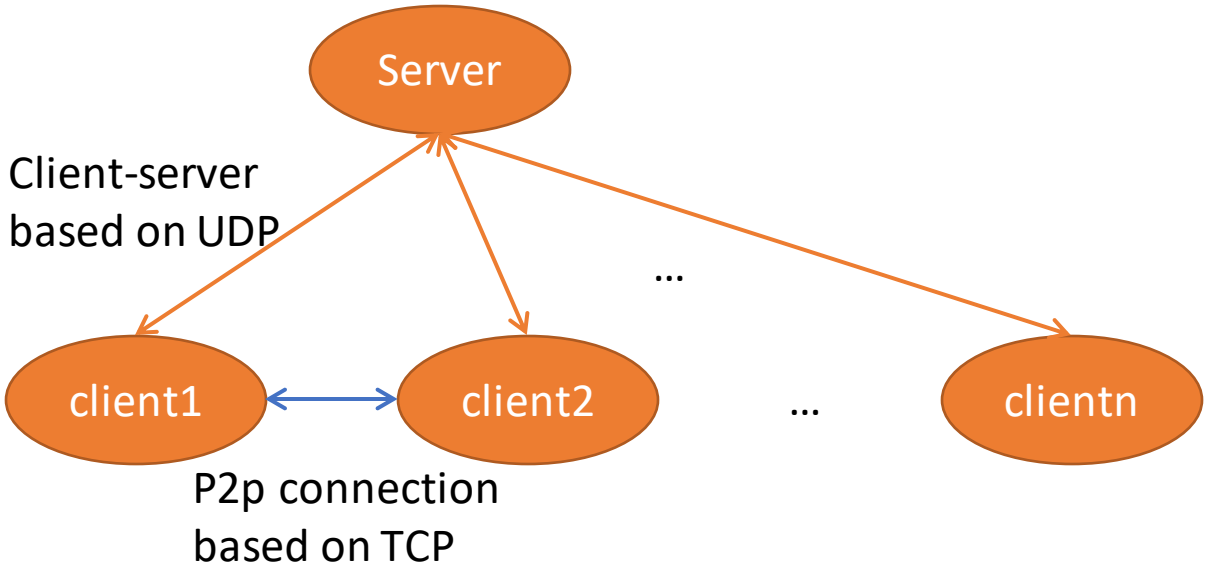


We assume at most 32 files (00.txt … 31.txt). The server maintains an unsigned integer variable for each client, in which each bit indicates whether this client has this file. For example, client1 has 01.txt and 31.txt, then the bitmap corresponds to this client is "0100 0000 0000 0000 0000 0000 0000 0001" in binary format.

After the client starts, it should tell the server what files it has. Then the server will store this information for this client. Whenever there is a client that wants to query for a specific file, it sends a query message to the server and the server returns the addresses of clients that have this file.

Then the client selects one of these addresses and request to download the file from this peer address.

Programming project description

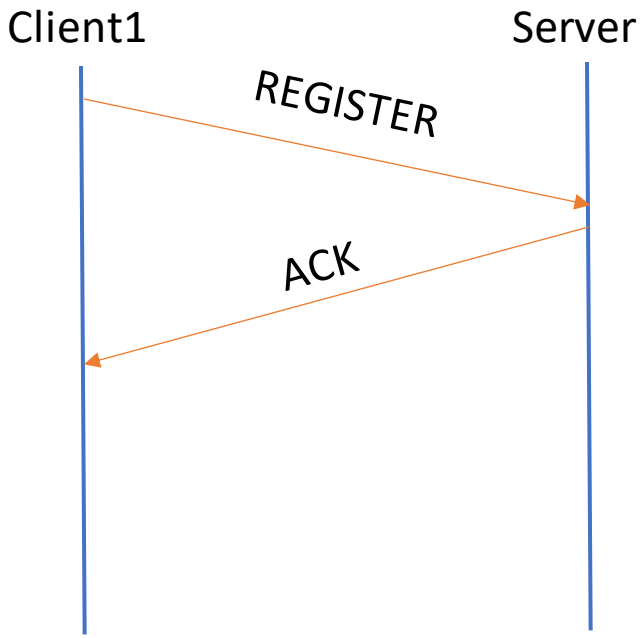


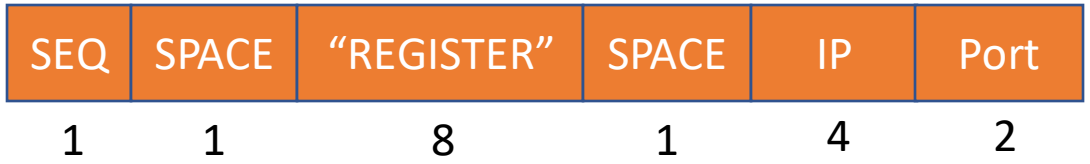
The communication between the client and the server uses rdt3.0 on UDP, i.e., there is no permanent connection between the clients and the directory server.

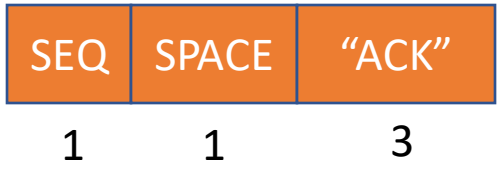
The connection between clients are based on TCP. The clients create a background thread to work as a server to provide file download services to other clients. Additionally, this server uses poll() to achieve concurrency among concurrent clients.

Programming project description

REGISTER: is sent to the server to register the ip and port information. Require ACK.

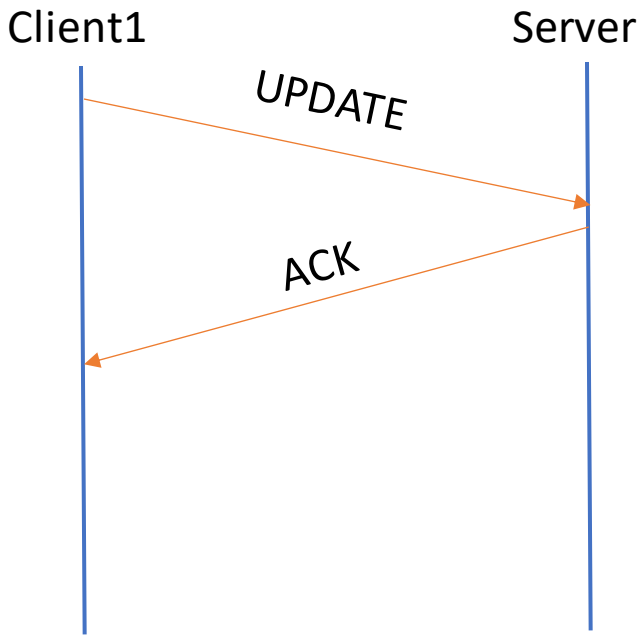






Programming project description

UPDATE: is sent to the server to update the available file bitmap corresponds to the ip and port. Require ACK.



SEQ	SPACE	“UPDATE”	SPACE	IP	Port	bitmap
1	1	6	1	4	2	4

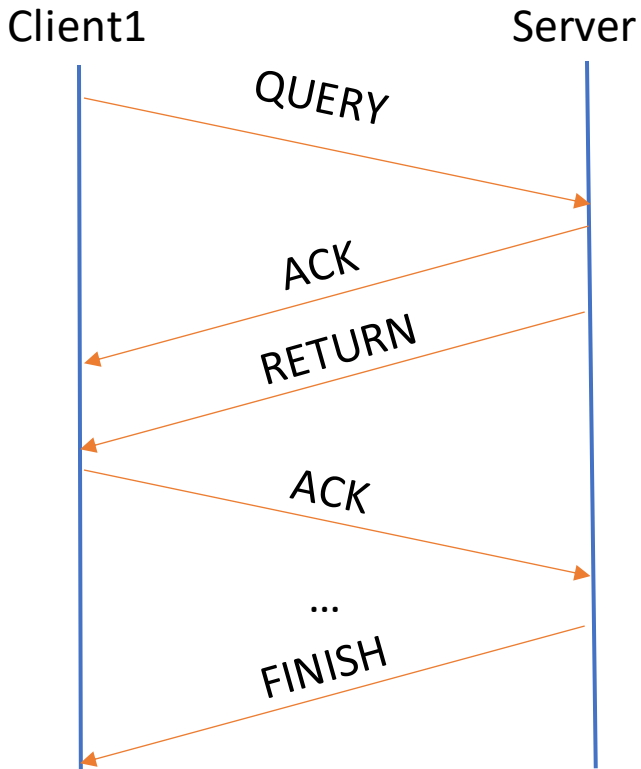
SEQ	SPACE	“ACK”
1	1	3

Programming project description

QUERY: is sent to the server to query the accessible p2p clients for a file.

RESPONSE: is sent to the client to provide available ip and port.

FINISH: is sent to the client to indicate all ip and port are returned.



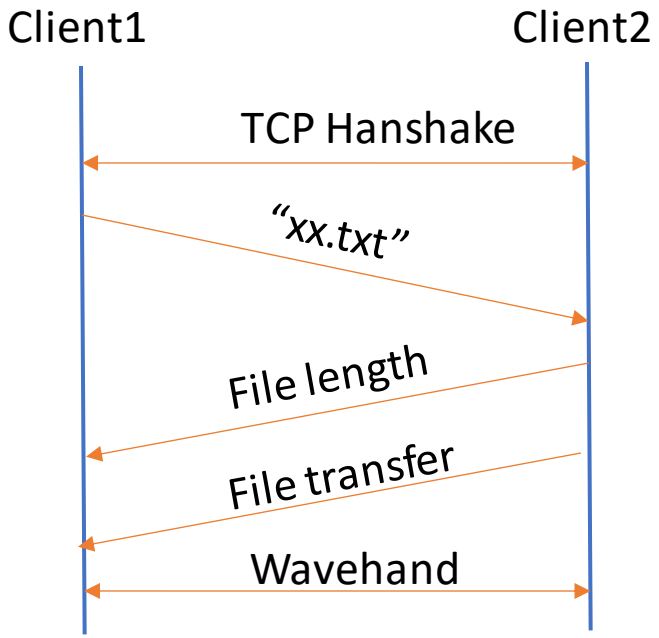
SEQ	SPACE	“QUERY”	SPACE	“xx.txt”
1	1	6	1	6

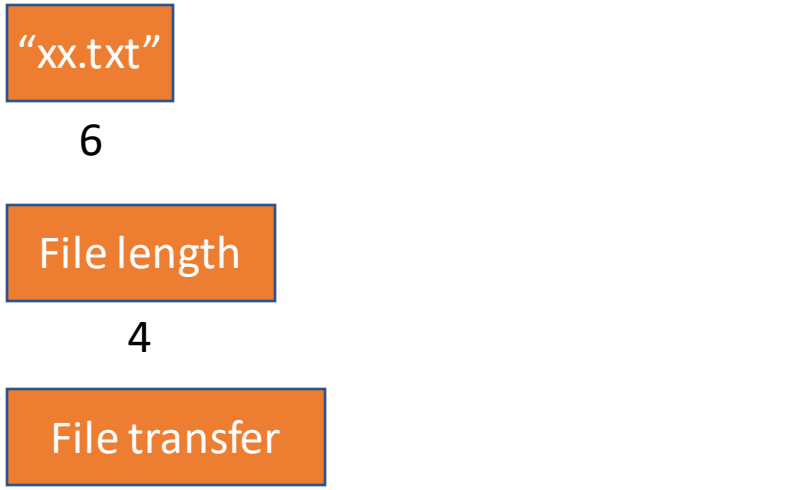
SEQ	SPACE	“RESPONSE”	SPACE	IP	Port
1	1	8	1	4	2

SEQ	SPACE	“FINISH”
1	1	6

Programming project description

GET: is sent to the **p2p client** to get the above **queried file** from one p2p client.

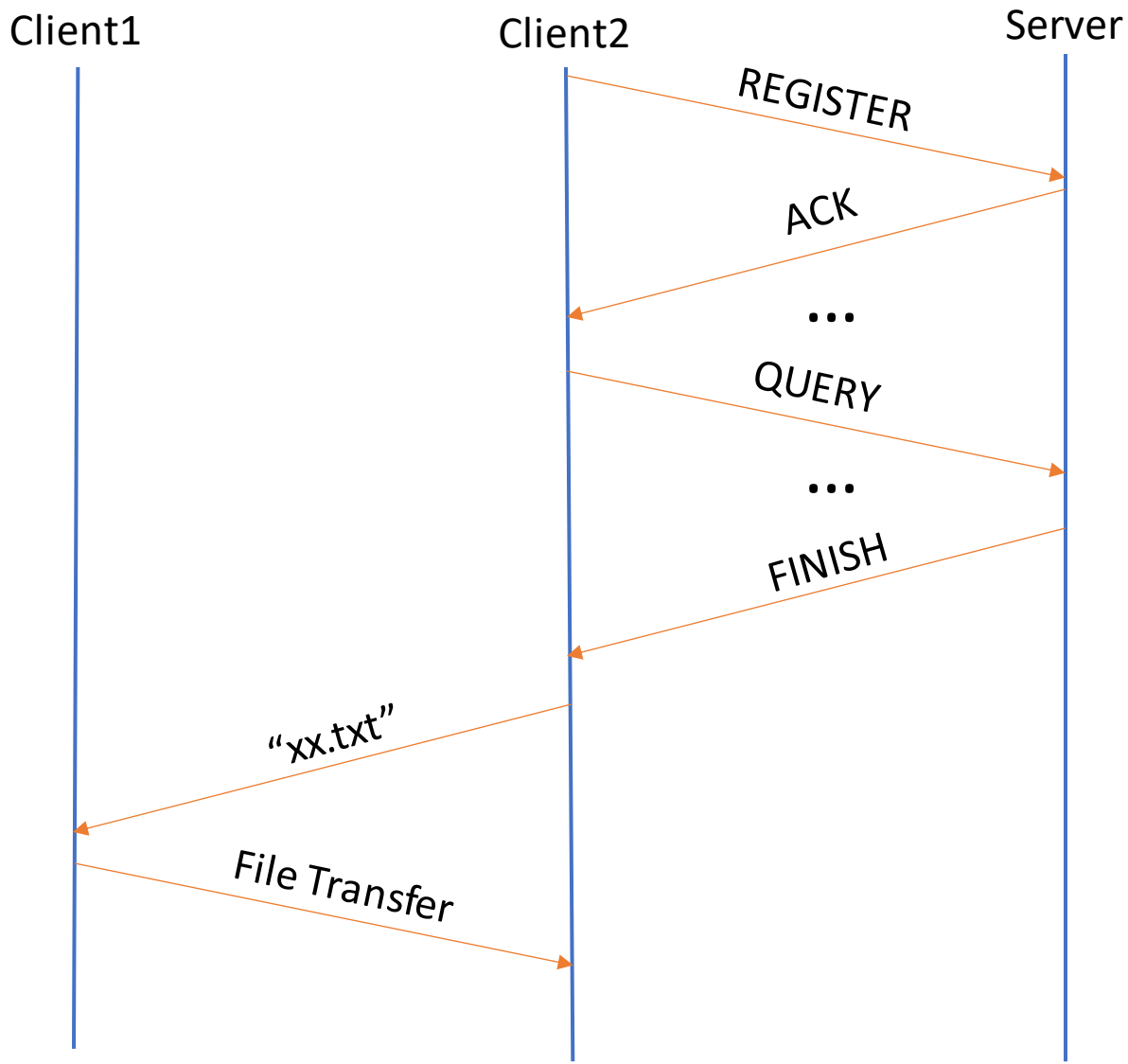




The file content, any reasonable length.

Note that the file transfer is based on TCP.

Programming project description



Programming project description

Requirements:

- Programming must be in C.
- You must implement rdt3.0 based on UDP to deal with all communications between the server and the client.
- The server should be able to provide services to multiple clients simultaneously. That means it should provide a concurrent rdt3.0 based on UDP.
- You must implement a p2p server and p2p client in “client.c” file. Start the p2p server using the multi-thread function when the client get started.
- The p2p server must be based on poll() to enable concurrency.
- The submission contains two files “client.c” and “server.c” plus a short report in pdf to explain your approach.
- Your code must pass the provided test scripts, which will test your code with some simple commands. Note that your code is only required to process valid commands.

Skeletons of codes

- Skeletons for the client and the server are provided for you in the ./skeleton/ folder.

```
/*********************************************
* Refer to REGISTER implementation above and
* the UPDATE protocol description. Dealing with
* UPDATE packet
* START YOUR CODE HERE
*********************************************/

YOUR CODE

/*********************************************
* END OF YOUR CODE
*********************************************/
```

Grading scheme

1. Your code passes the provided test1 for the simplest scenario. This test case will test your code with two clients without packet loss and without latency. Both these two clients connect to the server one by one and one client downloads a file from another client. (10%)

2. Your code passes the provided test2 for concurrency of the server. This test case is similar to the above one. Except that these two clients connect to the server at the same time. (10%)

3. Your code passes the provided test3 for multi-threading of p2p client and server. There are three clients start one by one. Client2 downloads a file from client1 and client3 downloads a file from client2 at the same time. And they are required to finish the transfer within 10s. (10% + another similar case for 10%)

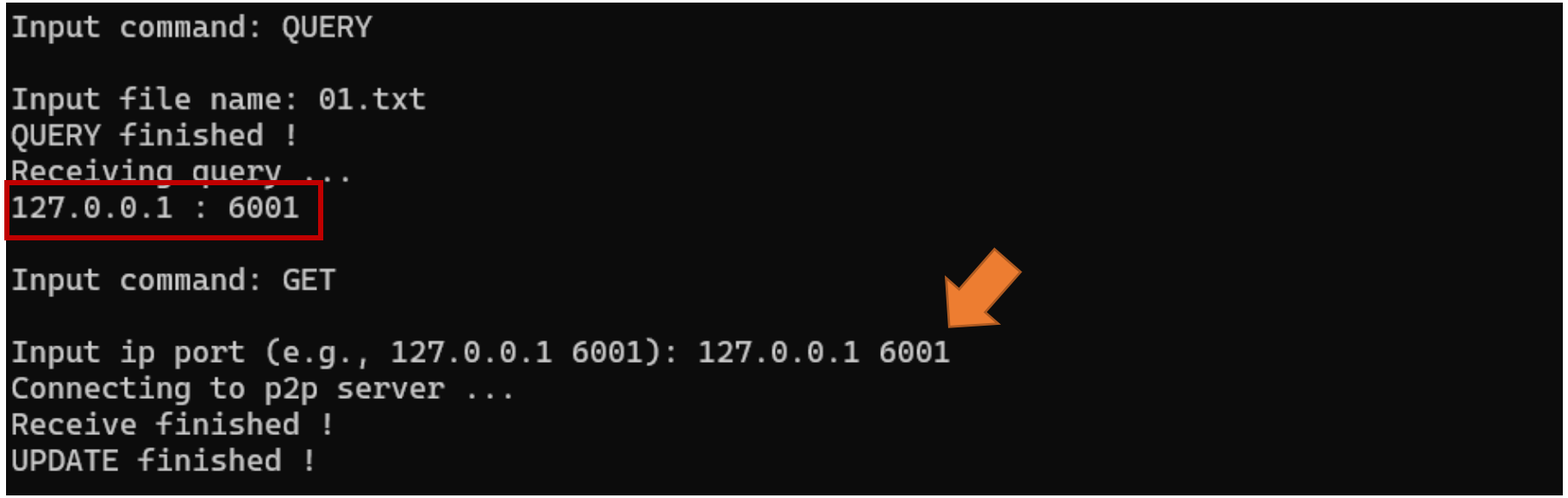
4. Your code passes the provided test4 for packet losses. We will test your server code with a modified client, which will drop some ACKs for RETURN messages. (10% + another similar case for 10%)

5. Your code passes the provided test5 for packet losses. We will test your client code with a modified server, which will drop some ACKs for REGISTER/UPDATE/QUERY messages. (10% + another similar case for 10%)

6. A report for this programming project. (20%)

Bonus

After execute QUERY, return the IP and port number with a sequence number. When you execute GET, you can specify the sequence number instead of the full IP and port number.



If you finish this bonus task, please indicate in your report **apparently**. Upon successful verification, you will receive an additional 10% grade.

Testing your code yourself

All of the provided code is tested on Lab2 machine. How to login? Refer to the first tutorial.

If you want to test your code with test case 1:

1. Copy your code “client.c” and “server.c” to test1
2. $ cd test1
3. $ bash test.sh
4. Similar for test case 2/3/4/5

If your code pass the test case, it will print a “Succeed”. Otherwise, “Fail”.

The “test.sh” script is very simple, it just compiles and run your code. You can also test your code manually.

Note that I may use different hidden test cases (e.g., different txt file name, larger txt file size) when evaluating. But do not worry, the basic operations are the same. If your code works for the provided test case, the hidden test cases may work too.

Reminder, if all of you use the same machine like csl2wk01, you may conflict with each other.

Summary

- We review the concurrency technique.
- We learn how to use pthread library to enable simultaneous sending and writing for the client.
- The above two things are essential in our programming project.

Q&A

Now you have 20 minutes to ask questions about the programming project.