

CSIS0234B Computer and Data Communication (Class B)

Tutorial 3

File transfer using UDP

In this tutorial, we are going to write a simple file transfer program for downloading a file from a UDP server. UDP is a connectionless and unreliable service, so unreliable that if you are not careful, packets can be lost even in a reliable network. We will see tricks to avoid such problems.

In the computer you will find two programs `UDPServer.cc` and `UDPClient.cc`, which makes a UDP socket for communicating with its peer in their `main()` functions. Then they call `dg_ftp_serv()` and `dg_ftp_cli()` respectively, which you must write to achieve the file transfer. The client program needs a command line argument specifying where the server is. Server port number is hard coded as 12345. For simplicity, the server should always read the file `orig.txt`, and the client should always save the received file in `received.txt`.

1. Your tasks

1. Complete the programs by communicating the whole file in a **single** UDP packet. The client should start the file transfer by sending a 0-byte packet to the server, and the server should reply with the content of the file. Use a buffer 128KiB in size for reading and receiving the file. Test your program with a small file (of a few hundred bytes, e.g., “`small.txt`”).
2. Try to use your program to transfer a file of size close to 128KiB (e.g., “`medium.txt`”), to see that it fails because the maximum packet size is exceeded. Modify your programs so that the server reads 4096 bytes from the file at a time, and send it in a separate packet. The server sends a 0-byte UDP packet to notify end of file. Try your program again. Check that the file transferred can now be started, although the file received is not complete.
3. Some packets are lost because the client buffer is not sufficient to hold the file. Implement a simple “flow-control” mechanism to avoid overrunning the client buffer: every time the server sends one packet, it waits for the client’s “acknowledgement” before continuing. The acknowledgement is a 0-byte packet, which is sent by the client whenever it receives a packet. Check that the file is transferred correctly.
4. If time allows, modify the server program so that it allows simultaneous connections from multiple clients. A separate process should be created (using `fork()`) for each client. The parent should return to listening immediately, while the child should create a new socket (with a new port assigned by the server OS) for file transfer. After the file transfer, the child should exit the program. You can test the server using a large-size file (e.g., “`large.txt`”).

2. Note

The loopback interface is reliable, i.e., frames delivered are always received, in correct order. This remains true for a single Ethernet network, so you can test your program in 2 different machines and it will still work. For unreliable network, or network connected by routers, better scheme must be used to guard against out-of-order transmission and lost frames (e.g., the program fails when the server is in virtue). Usually it is much easier to just use TCP instead.