

CSIS0234B Computer and Communication Networks (Class B)

Tutorial 10

Multicast file transfer

In tutorial 3, you've written a program to download a file from a UDP server. In this tutorial, you're going to do a similar task, but using multicasting. As our program will not run for a long time (they do one thing and terminate), it would be confusing to call them "client" and "server", so let's call them "sender" and "receiver" respectively. The **receiver** starts first, listening to a well-known port 12345 after joining a multicast group as specified on the command line. Then the **sender** will run, multicasting a file to all listening receivers.

For reference, sample programs of tutorial 3 (`UDPServer.c` and `UDPClient.c`) are placed in the computer, under the network user. You will find that many parts of these programs can be copied to the ones we are building. In particular, the receiver program will have network code of `UDPServer.c` and the file manipulation code of `UDPClient.c`, while the sender program will have network code of `UDPClient.c` and file manipulation code of `UDPServer.c`. The code snippet in the tutorial reading to join a multicast group will also be useful.

Here are the detailed steps to guide you through the process to make sure you know whether each step is done correctly.

1. **Create the sender program.** It should take two command line arguments, the file to send and the multicast address to send to. The program should get a datagram socket and an address structure to that multicast address at port 12345, open the specified file, and send the file to the other end **without waiting for a datagram from the other side**.
2. **Test your sender program** by sending to the group address 239.1.0.x (where x = the computer number of your computer). Use `ethereal` to capture all multicast packets to `eth0` with that address as destination (using a `dst host` filter), so as to make sure the multicast is being sent. Compare the Ethernet addresses with the IP addresses.
3. **Create the receiver program.** It should take two command line arguments, the multicast address (group) to listen to, and the port number to bind to. The program should get a datagram socket with binds to that multicast group and port. Then it should join the multicast group, and wait until a datagram is received. The received datagram should be written to standard output, as in the original `UDPClient.c`.
4. **Test your receiver program** using 239.1.0.x as the group address and 12345 as port number, without running the server. Check `/proc/net/igmp` with and without the receiver program running, in order to make sure you have joined the right group.
5. Now create a small file. Run the receiver program, redirecting the output to a file. Run the sender program with the receiver running. **See if the receiver get the file** correctly. Compare the original and the received files to make sure the file is sent correctly. Try also sending the file to multiple computers at the same time, and make sure that all of them receives the same file.
6. Again use **Ethereal** to capture the network packets on the multicast address, this time **when the receiver is started**. Note that some IGMP packets are being sent. Run the receiver program again, and capture packets with the "update list of packets in real time" option turned on to see the timing of these packets sent. Repeatedly look at the `/proc/net/igmp` file as you capture packets to see how to predict when the OS will send these packets.

7. **Try running two copies of the receiver program** (without running the sender program), using the same and different (1) multicast addresses, and (2) port numbers. Note which of the combinations leads to successful execution of both programs. If both program can be executed successfully, note also the content of `/proc/net/igmp`, and see whether each copy of the program will cause IGMP packets to be sent.
8. Try to **repeat the steps using IPv6 interface**. Since the computer in our lab are not fully configured for IPv6, we cannot use it to send datagrams across computers. But you should at least be able to send datagrams using IPv6 multicasting by a local IPv6 multicast address like `ff01::1`. Before doing so, you will need to load the `ipv6` kernel module using `insmod`. You will not see any packets captured by `ethereal`. Use `/proc/net/igmp6` to see the membership information. Fix any problem you find.