CSE 3461 Lab 4

Instructor: Adam C. Champion

Due: Tuesday, April 4, 2017, 11:59 p.m. (40 points)

In this lab you will extend Lab 3 to provide reliable file delivery using the Alternating Bit Protocol over UDP in Python 3. Packets will be dropped in both directions using one `troll` process on each machine. A copy of each transmitted data packet needs to be kept until it is ACKed. Note that as ACKs are not ACKed again, so the ACKs are unreliable. You can use a fixed retransmission timeout value of 50 ms.

Your implementation needs to work only for one file transfer. Observe that unlike in a traditional OS, you are implementing the transport layer reliability mechanism within the application layer itself. The file transfer protocol will include a server called `ftps.py` and a client called `ftpc.py`. In the following, we assume the client is running on `beta.cse.ohio-state.edu` and the server is running on `gamma.cse.ohio-state.edu`.[1] First, start the server on `gamma.cse.ohio-state.edu` using the command

```
python3 ftps.py <local-port-on-gamma> <troll-port-on-gamma>
```

Then start `troll` on `beta.cse.ohio-state.edu` with the command (on one line)

```
troll -C <IP-address-of-beta> -S <IP-address-of-gamma> -a
                    <client-port-on-beta>
-b <server-port-on-gamma> <troll-port-on-beta> -t -x <packet-drop-%>
```

On `gamma.cse.ohio-state.edu`, start `troll` with the command (on one line)

```
troll -C <IP-address-of-gamma> -S <IP-address-of-beta> -a
                    <server-port-on-gamma>
-b <client-port-on-beta> <troll-port-on-gamma> -t -x <packet-drop-%>
```

You will then start `ftpc` on `beta.cse.ohio-state.edu` with the command (on one line)

```
python3 ftpc.py <remote-IP-gamma> <remote-port-on-gamma> <troll-port-on-beta>
                    <local-file-to-transfer>
```

The `ftpc.py` client will send all bytes of that local file to the `troll` process, which should forward the packets to `ftps.py`. The `ftps.py` server should receive the file and then store it. Make sure that the new file created by `ftps.py` is *in a different directory* to avoid overwriting the original file since all the CSE machines have your root directory mounted.

The file-transfer application uses a simple protocol. The payload of each UDP segment will contain the remote IP (4 bytes), remote port (2 bytes), a flag (1 byte), a 1-bit sequence number (1 byte) that can take values of 0 or 1, followed by a data/control field as explained below. The flag takes three possible values depending on the data/control field (explained on the next page):

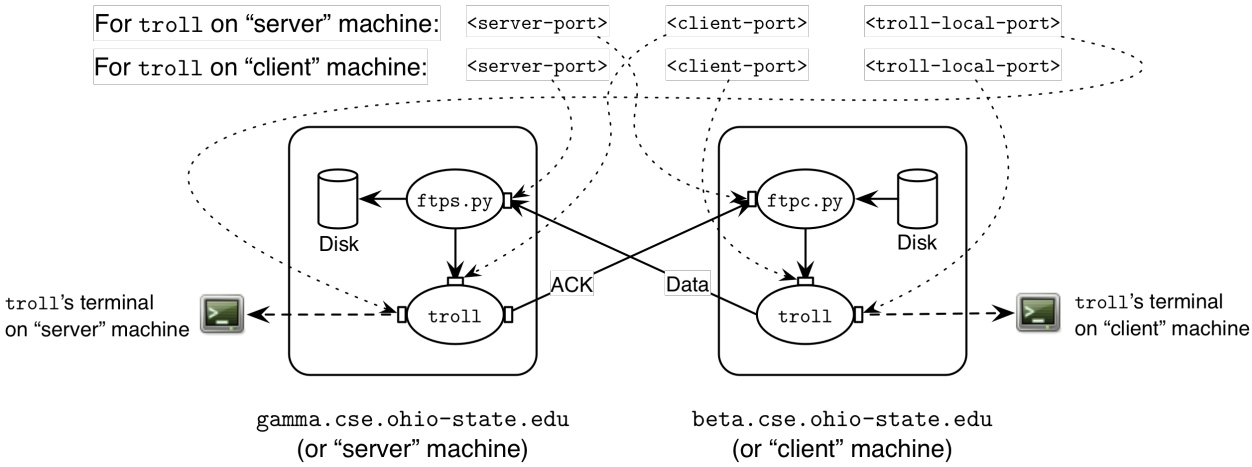
[1] In your lab, you can substitute any machine on `stdlinux` (e.g., `zeta.cse.ohio-state.edu` or `epsilon.cse.ohio-state.edu`) for `beta.cse.ohio-state.edu` and `gamma.cse.ohio-state.edu`. You should do so to avoid "overloading" these machines.

- **First segment (4 bytes):** The first segment should contain the number of bytes in the file to follow (in network byte order). The flag is set to a value of 1.
- **Second segment (20 bytes):** The second segment should contain 20 bytes which is the name of the file (assume the name can fit in 20 bytes). The flag is set to a value of 2.
- **Other segments:** The other segments will contain data bytes from the file to be transferred. Each segment can have up to 1000 data bytes. The flag is set to a value of 3.

Each ACK has 1 byte with a value of 0 or 1. The `troll` process will drop packets with the drop rate percentage `<packet-drop-%>` specified in the command line. Submit well-documented code using the following command:

```
submit c3461ad lab4 <code-directory-name>
```

Your program should work for binary files (images, etc.) of arbitrary size and the client and server should be running on different machines on `stdlinux`. You can use the sample images on the course webpage to test your program. Use `diff` or `md5sum` to ensure that the transferred file is the same as the original one. No buffer, array, or data structure used should exceed 1,200 bytes in size. **You'll need to "chunk" the file into "pieces," each of which is at most 1,000 bytes. Submit a `README.txt` file with your lab.** Please ensure that file transfer succeeds for any `<packet-drop-%>` value less than 20. The three types of processes will function in the following way:



- `troll`: Each machine will run an instance of the `troll` program. `troll` runs continuously and processes each incoming packet. `troll` decides either to drop the packet or to forward it to its intended destination.
- `ftpc.py`: `ftpc.py` sends one packet at a time. After sending the packet, it has to simultaneously wait for two events: either a timeout or receipt of an ACK. Use the `select.select()` function for this purpose. (DO NOT use `socket.settimeout()` or other methods as they will not work correctly.)
- `ftps.py`: `ftps.py` receives one packet at a time and after processing it, it takes the appropriate action (interpreting the size, interpreting the name of the file, and extracting the bytes of the file to write to the local disk). Upon receipt of a packet, it creates an appropriate ACK and sends it back.