CSE 3461: Taming `troll`

Instructor: Adam C. Champion

The advice below on `troll` comes from previous CSE 3461 students. Labs 3 and 4 are more difficult than Labs 1 and 2 as three or four processes communicate with each other: the client, the server, and one or two `troll`s. **Do not procrastinate**; Lab 4 builds on Lab 3.

Thoughts on Troll

There has been confusion about how to use `troll`. This section is to provide insight into `troll`'s behavior and usage and hopefully clarify numerous questions.

UDP 101

In UDP network programming, you send datagrams from a source host to a destination host. Here, I'll use several terms; be sure to get them straightened out:

Datagram:
Another name for a packet sent using UDP (the data you pass to Python's `sendto()` method)

Client machine:
The machine executing `ftpc.py`. The lab document uses `beta.cse.ohio-state.edu`.

`client_ip`**:**
The IP address of the client machine

`client_port:`
The port used by `ftpc.py` to send datagrams.[1] You can pick this.

Server machine:
The machine executing `ftps.py`. The lab document uses `gamma.cse.ohio-state.edu`.

`server_ip`**:**
The IP address of the server machine

`server_port:`
The port used by `ftps.py` to receive incoming datagrams. You can pick this.

`troll` **machine:**
The machine executing `troll`. The lab document uses `beta.cse.ohio-state.edu`.[2]

`troll_ip`**:**
`troll`'s IP address.[3]

`troll_port`**:**
The port used by the `troll` program. You can pick this.[3]

[1] In typical UDP programming, you don't specify a port on the client side; it's sufficient to call `sendto()` without `bind()`. In this case, the OS will pick any available port for you to use. However, `troll` only accepts packets from a specified port, so we have to manually tell the OS which port to use by `bind()`ing the socket to `client_ip` and `client_port`.

[2] The lab document demonstrates using the program by running `troll` and `ftpc.py` on the same machine (`beta.cse.ohio-state.edu`). This is not strictly necessary: `troll` could run on a third host if we wanted.

[3] There are a lot of IPs and ports floating around. It's helpful to pair them up and associate them in your head with one of the three programs being used in this lab. For example, the server has `server_ip` and listens on `server_port`. `troll` lives on `troll_ip` and listens on `troll_port`. Don't worry if `troll_ip` is the same as `client_ip`, because you can just pick different ports for them to use. These values are specified in pairs so frequently that a popular shorthand is to simply concatenate them with a "`:`". For example, the troll machine is `client_ip:client_port`.

Troll

`troll` is a network program with pretty simple behavior.[4] It listens for datagrams coming in on `troll_port` and forwards them somewhere else. More specifically, it will only accept datagrams from a single IP/port combination, and it will only send them to a single IP/port combination.

Let's define some more terms:

`source_ip`:
The IP from which `troll` will accept datagrams

`source_port`:
The port from which `troll` will accept datagrams

`dest_ip`:
The IP to which `troll` will forward datagrams

`dest_port`
The port to which `troll` will forward datagrams

`troll`'s usage looks like this (on one line):[5]

```
[me@beta ~/Lab3]$ ./troll -C source_ip -a source\_port -S dest_ip
-b dest_port -r -t -x 0 -s 1
```

Troll Use in Lab 3

Your first priority for Lab 3 should be to get the transfer process to work using datagrams (UDP) instead of a TCP stream. If this doesn't work, you might as well forget about using `troll`.

Then, you should make a small modification that allows the client port to be specified on the command line (footnote 1).

In this section, I'm going to assume that your implementation works like this:

```
# Start the server on gamma, binding to server_ip:server_port
[me@gamma Lab3]\$ python3 ftps.py server_port

# Start the client on beta, sending datagrams to server_ip:server_port
# bound client_address:client_port (ON ONE LINE)
[me@beta Lab3]\$ python3 ftpc.py server_ip server_port client_ip client_port
../image1.jpg
```

This should look almost exactly like Lab 2 with one exception: you are required to specify `client_ip` and `client_port` when executing `ftpc.py`.

To properly implement this lab, you need to be able to route your file transfer through `troll`. This is deceptively simple! `troll` should be configured to **accept** packets from the **source** `client_ip:client_port` and send these packets to the **destination** `server_ip:server_port` (if you're confused about how to configure `troll` for a specific source and destination, then re-read the previous section).

[4] It's not actually this simple . . . It can be configured to "troll" you by randomly dropping or rearranging packets. Lol.

[5] Using the `-s 1` option decreases the `troll` delay from 10 ms to 1 ms, which should literally save you an order of magnitude of time when transferring large files.

Then, instead of directing the client program ftpc.py to send datagrams to server_ip:server_port, you should direct the client to send to troll_ip:troll_port.

If all goes well (and your ftps.py/ftpc.py worked without troll), your server should receive datagrams from troll.

Running troll

Some students have had trouble getting troll to run. When I taught CSE 3461 in Summer 2015, my students found that if you run

```
sock.sendto(data,('',port))
```

on the client, then you must run troll as follows (on one line):

```
troll -C 127.0.0.1 -S <IP-address-of-gamma> -a <client-port-on-beta>
-b <server-port-on-gamma> -r -t -x 0 <troll-port-on-beta>
```

But if you run

```
ip = socket.gethostbyname(socket.gethostname())
sock.sendto(data, (ip, port))
```

on the client, then you must run (on one line):

```
troll -C <IP-address-of-beta> -S <IP-address-of-gamma>
-a <client-port-on-beta> -b <server-port-on-gamma> -r -t -x 0
<troll-port-on-beta>
```

If you hard-code the client port using bind(), you need to run troll as follows (on one line):

```
troll -C <IP-address-of-beta OR 127.0.0.1> -S <IP-address-of-gamma>
-a <VALUE_YOU_HARDCODED> -b <server-port-on-gamma> -r -t -x 0
<troll-port-on-beta>
```

In all cases, **please document how to run the lab in the README file** so the graders can run your lab!