**Step 1.1**

We noticed that the server received the number the client sent. Upon response, the client received the same number the server received and sent back.

Link to *recording-step1.1*:

**Step 1.2**

Link to *recording-step1.4*:

**Step 1.3**

Link to *recording-step1.4*:

**Step 1.4**

Link to *recording-step1.4*:

**Step 2.1**

We noticed that everything ran smoothly. The client sent 7 and received 16 from the server; just as well, the server received a 7 and sent a 16 back to the client.

Link to *recording-step2.1*:

**Step 2.2**

Link to *recording-step2.2*:

**Step 2.3**

Link to *recording-step2.3*:

**Step 3.1**

We observed there was no problem connecting to the server from the client, and no difficulty sending and receiving data back. However, only the number 21 was sent to the server; indicating that the loop finished before any threads sent their corresponding task count.

Link to *recording-step3.1*:

**Step 3.2**

We observed that the client was able to send numbers 1 through 20 to the server and receive the expected 16 back with no problem; consequently, the server received the expected 1 through 20 numbers and sent 16. Additionally, the processes did not report the sent and received values out of order.

Link to *recording-step3.2*:

**Step 3.3**

First thing we noticed was only a handful of connection requests were received by the server before having to close the socket because of a bad file descriptor; adjacently, the client encountered many network problems when reading from the socket. The client also sent numbers out of order.

Link to *recording-step3.3*:

**Step 3.4**

Link to *recording-step3.4*: