Zixuan Cui (Elle) & Vladyslav Polishchuk (Vlad)

3/7/2019

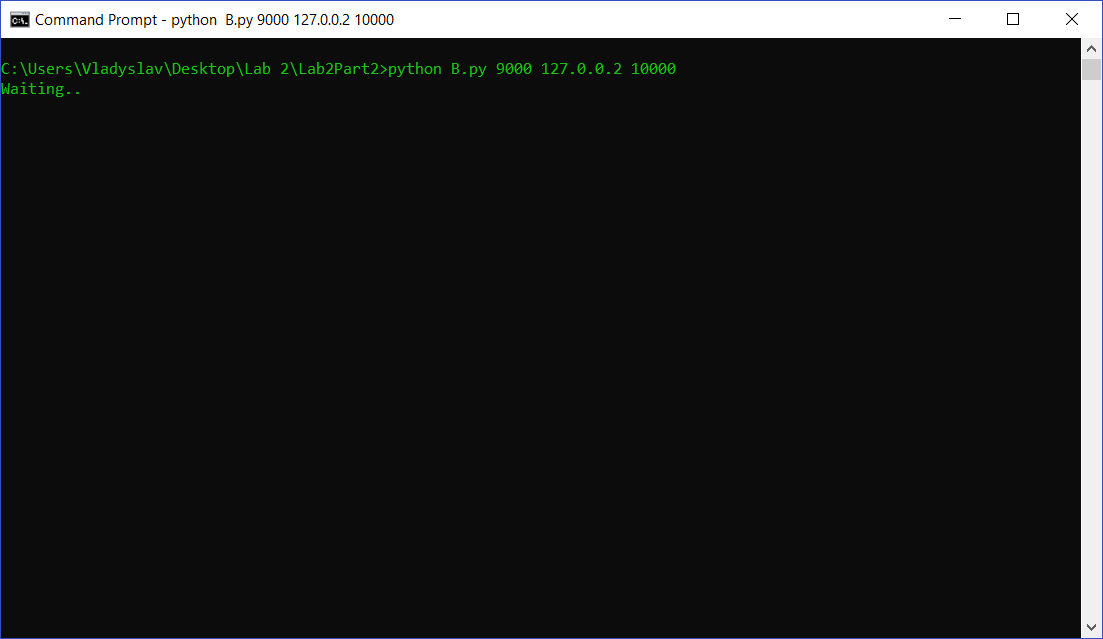
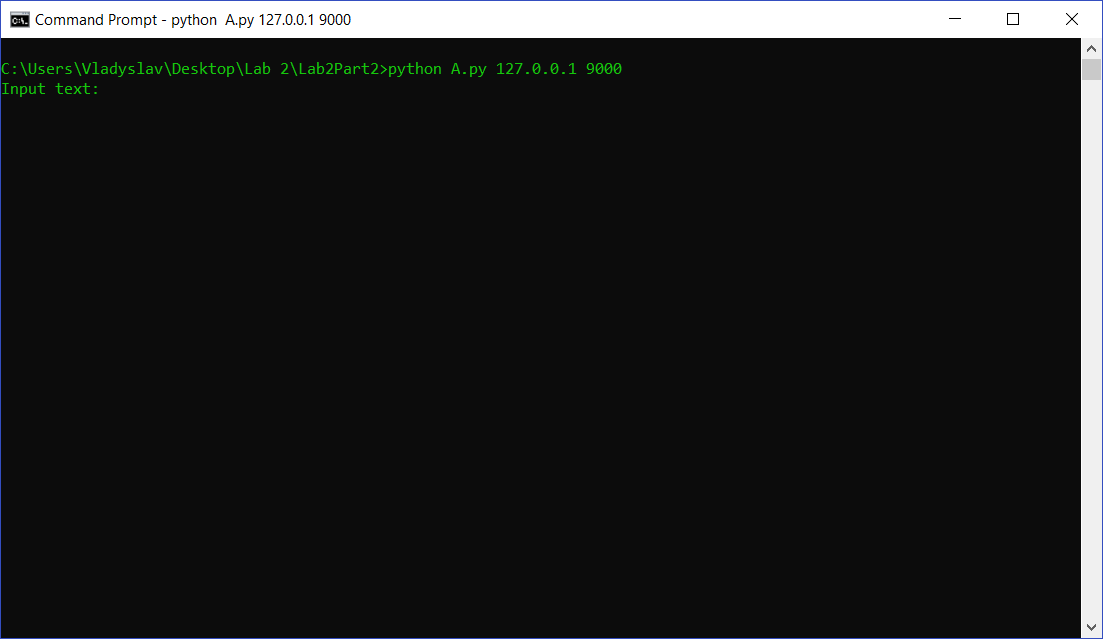
Data Communications

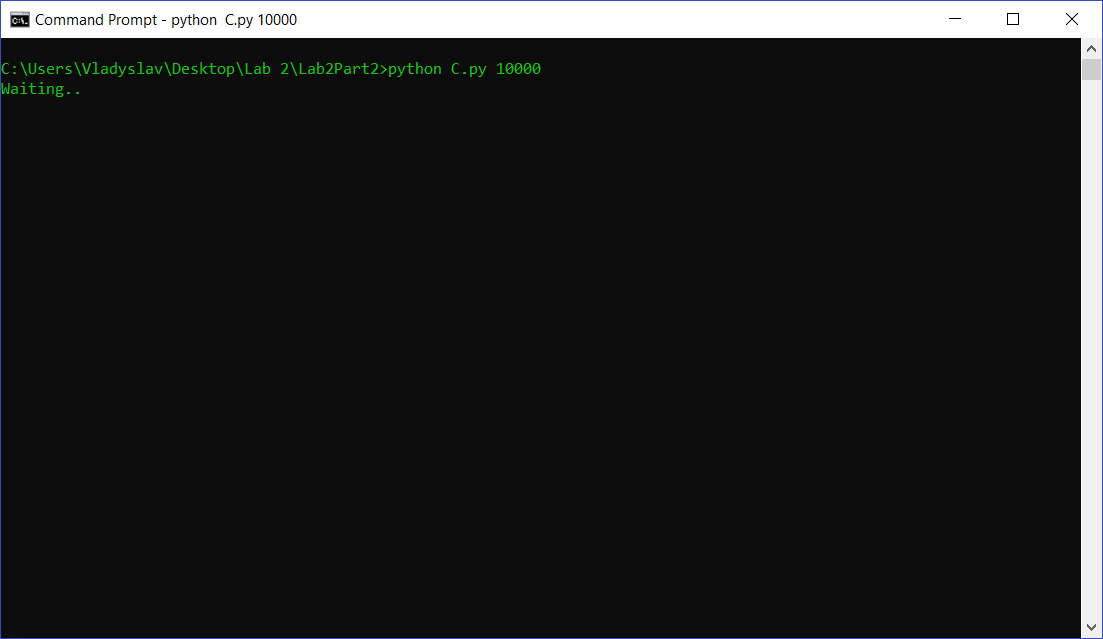
Professor Ying Mao

Lab 2 Part 2 Report

* We copied over the structure used in part 1 of Lab 2 since the premise is largely the same - A sends to B, B forward to C. To start off, we used the code directly from client.py and server.py of sample-3 to fill in our A.py and C.py. A.py remains unchanged, but B.py (both receiving and sending) had to be constructed and C.py had to be altered for a crc comparison.
* The number of arguments for B was set to 4, just like in part 1, since B.py is acting as a forwarder. We also carried over the “s1” and “s2” concept for the same reasons. B was meant to receive the message and crc32 code, follow a 40% probability of altering the message (and only the message), and then send the message and original crc32 code to C. We made B receive the message and crc code, unpack it, and create a backup of the original for our break statement (in case the original message is altered). We used “import random” at the top of B.py and ran the following if statement for our 40% probability: “if random.randint(0, 100) < 40:”. If the message was changed, we made sure that B announces that the message was altered. Finally, whether or not the message was altered, we repacked the ss struct and forwarded the message to C.
* C.py was meant to receive the message and code from B, unpack the message and code, check that the received code is correct using the crc32 function, and then print the result of the check. C.py was kept similar to server.py from sample-3, but the message was unpacked and decoded (2 lines of code, like in part B). For our comparison, we realized that we could simply re-use the crc32 function that was defined in the sample-3 files to compare “crc32(str)” and “crc” (received from B).

**Pre:**





**Post:**