ELEN4017: Network Fundamentals Lab 3

Benjamin Rosen (858324)

13 March 2018

Question 2

2 a): 10 seconds (the difference between the first and last times printed out amounted to 9 seconds, but it only prints the times after waiting the delay value first). Even though the total waiting time for the program should be 20 seconds (10 for each function call), the threads are executed in parallel, meaning that each waiting period happens concurrently.

2 b):

- **threading.Thread** is used in the class definition (i.e. the class *myThread* inherits from threading) so that all threading functions can be used with *myThread*. It comes with the **threading** library included at the top of the code.
- **threading.active_count()** returns the number of threads in use. It prints 3 in this program, as there is a main thread, and two branched threads (which call *print_time()*).
- **.start()** creates a new thread and calls **run()**, which contains the commands it must execute.
- **.exit()** joins the thread back to the main thread. In this program, it does so when the exit flag is set, to terminate it early. Without this, the thread exits when the *run()* function terminates.

2 c): The counter value for each thread was 2. The program took 10 seconds, since thread 5 had 2 iterations with 5 seconds delay. The number of threads started at 6 (5 threads + main thread), but decreased as each thread finished executing their function (e.g. thread 1 finished after 2 seconds, thread 2 after 4 seconds etc.). *get_name()* does not exist, but *getName()* (or just *.name*) returns the name given to the thread by the user program. *active_count()* is already explained in 2b) and current_thread() returns the thread object, which includes the thread's user given name and when it was started.

Question 3

TCP required multithreading to work with more than one user, but, since UDP is connectionless, it simply receives messages from multiple users without any need to adjust the code. The onus is on the server to echo the message to the right client, by using the address provided in the message from the client, whereas TCP uses the dedicated connection to echo back to the client.

Nevertheless, the UDP server was still implemented using multithreading. This is because a received message might have a more complicated response than simply echoing it back to the client (for example, starting another process such as downloading a file). Therefore, the main thread checks for incoming messages and creates a new thread for each received message, which has its own routine.