**Programming Project Stage3**

**Threaded Port Scanner Application**

|  |  |  |
| --- | --- | --- |
|  | Name | ID |
| Student 1 |  |  |
| Student 2 |  |  |

**Guidelines**

* Please find attached the starter code.
* Watch the tutorial videos and start solving the problem.
* It is **not** recommended to make changes in the starter code.
* You don’t need extra functions, but you can add if you need to. Don’t write any code outside a function.
* Please read the comments in the starter code carefully.
* The project might look scary, but it is actually very easy if you know what you need to do.
* Write the code and solve the questions in this document.

**Description**

On this part we are going to build a threaded port scanner. A port scanner tries to connect to a port or a range of ports and see which ones are open. It might take time to scan a huge range of ports, that is why we will make it faster using threads. Each thread will scan a range. We will experiment with different numbers of threads and see how much time is taken to accomplish the task in each case. You are given the code used to check if one port is open.

You need to do the following:

1. Finish the function scan\_range(start, end) which scans a range of ports.
2. Finish the function threaded\_scan(num\_of\_threads), which uses divides the whole range of ports amongst num\_of\_threads threads so that all threads cover almost the same number of ports. Each thread will call the scan\_range(start, end) function. To check that you covered all ports and did the division correclty, the list ports\_scanned which was initially all False, should be all True by the end of this step.
3. In the main function, test the threaded\_scan(num\_of\_threads) with differnet numbers of threads from this list num\_of\_threads = [1, 2, 3, 4, 5, 10, 20, 30, 50, 100, 500, 1000, 2000, 5000, 10000]. Make sure you reset the list ports\_scanned to all False before you call threaded\_scan(), and make sure all is True after the call. Also record the time taken for each call of threaded\_scan().
4. Draw a chart(line chart, or bar chart) showing the time taken to make a full scan for each number of threads tested.
5. Answer the questions at the end of this document.

**Note**

* Times might change based on the load on your machine.
* You might not find any open ports depending on the server you try to scan, but this is fine.
* Don’t test on your localhost or 127.0.0.1. Otherwise, it will finish in zero time even without threads. Use the host in the starter code.
* Test on the server provided in the code or if you have a server you are authorized to scan. It might be illegal to scan any server.
* If you try to scan any server, they might delay the response, which you can avoid by changing the timeout.

**Submission**

* Deadline is 20 April 11:59pm.

**Grading**

* This part of the project is worth 6%.
* The two students will get the same grade.
* There is no partial credit.
* If you follow the tutorial and the starter code, you should be able to solve the project.

**Help**

* Have a look at the first two videos <https://www.youtube.com/playlist?list=PLGKQkV4guDKEv1DoK4LYdo2ZPLo6cyLbm>
* If you have bugs or problems, you can google them and try to find a solution.

**Questions**

1.Draw a chart showing how execution time changes depending on the number of threads used.

2. What is the number of threads that caused the minimum execution time? Which caused the maximum time?

3. Is it the case that increasing the number of threads always reduces execution time? Can you comment on that?

4. Which applications are better suited for threading? Which applications are not?