

# Advanced Operating Systems

1st Semester 2023-24

## Assignment 1 - Remote Procedure Calls

[10% weightage - 20 Marks]

Due: 28th Sep 2023 [11:55 PM]


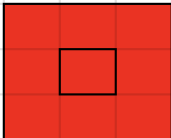
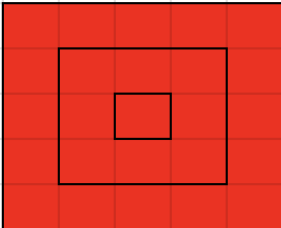
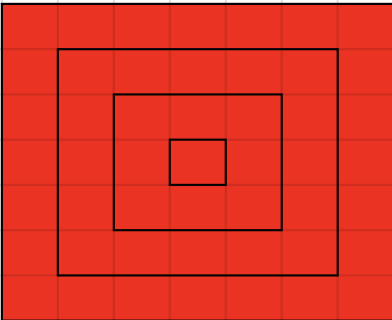
- 
- You are recommended to work in **groups of two** for this programming assignment. If you wish to work individually or in a larger group (of three), please discuss with the IC.
  - This project has two purposes: first to familiarize you with RPCs, processes; second to learn the design and internals of a distributed system.
  - A secondary goal is to familiarize you with source code control (e.g., github) and code testing.
  - You are free to use any programming languages (C++, Java, python, etc) of your choice and any abstractions that might be needed. You are being given considerable flexibility so that you can make appropriate (and creative!) design decisions and implement them in your program.
  - We recommend that you use python, Java or C++ along with any appropriate RPC framework. If you plan to use a language that is different from these choices, please discuss with the IC/TAs to make sure your language of choice supports standard features such as thread, synchronization, and network communication. Make your choice carefully since it will impact how much code you will need to write.

---

### A: Problem statement

- There is an  $N \times N$  field on which  $M$  soldiers have taken some respective positions  $(x,y)$ . One of the  $M$  soldiers is Commander. Commander is initially chosen from among the soldiers, at random. There are thus  $M$  nodes, and whoever is the commander acts as the current server also. Others act as clients. Every soldier, including the commander knows (stores) their respective current location, which is updated when the soldier moves. And there is a specific speed ( $S_i$ ) associated with him. Whenever a soldier is initialized, his speed and current location are also defined. Location is initially defined at random. The speed associated with a soldier will remain a constant value but is also set as a random number between 0 and 4 (both inclusive).

The field is situated in an airstrike battlefield where missiles are launched from drones every  $t$  seconds. The missile is targeted to hit a particular  $x,y$  coordinate. There are 4 categories of missiles based on their impact and radius:

Impact	Name	Radius
	M1	1
	M2	2
	M3	3
	M4	4

Whoever is present in the red zone, when the missile hits, dies. And if the commander dies, leader election must take place. But for simplicity, all you need to do is elect any soldier as the commander at random.

Missile defense system detects the launched attack, and the commander relays this information to all the soldiers present on foot. Each individual soldier receives this threat message and acts accordingly, if required. For simplicity you can assume that the current commander, whoever he is, has access to the defense system and thus is able to detect the initiated attacks.

A soldier's speed is defined as : 0, 1, 2, 3, 4. This means that, if a soldier has speed 1 then, after the notification of the missile, if he has to move (to save himself), he can move at most 1 block before the missile lands. Thus a soldier dies if he is in the RED zone and his speed is less than the radius of the red zone left to cover. It is upto the soldier in which direction he moves.

1 move/hop = a soldier can go in either of 8 neighboring coordinates..

After T time, which is the duration of battle, if >50% soldiers are alive, the battle is won, otherwise not.

- **Your system should implement the following interfaces and components:**
  1. *missile\_approaching(position, time, type)* - commander broadcasts to all. Position is x-y coordinates.
  2. *status(soldierID), status\_all()* after a missile has landed, soldiers are queried to check if they were hit or succeeded in taking evasive action. The status message can query a particular soldierID or a broadcast status\_all message queries all soldiers.
  3. *was\_hit(soldierID, trueFlag)* A reply message to a status request reporting whether the soldier was hit. These responses are used to keep "casualty count".
  4. *take\_shelter(soldierID)*; [local function] This procedure is implemented when a soldier is notified of a missile attack. If he finds himself in a red zone, he moves, else he remains at the same location.

Implement a **printLayout()** function that nicely prints the 2d matrix. It is up to you to decide on the finer details of how to show output layout. But it must include at least:

- Inside matrix: where is each soldier (numbered 1,2,3 etc.), where the recent missile landed (red zone),
- Outside matrix: dead soldiers, current time (iteration - i.e. t).

**There must be a printLayout and status call after every missile drops.**

Hyperparameters: **N, M, t, T, Si**. All of these have been defined above.

Hyperparameters must be programmable! Use various values to make your own test cases and properly document and show the results of them (the war zone at each period, bombing location, casualties, positions, win/loss).

**Other requirements:**

- No GUIs are required. Simple command line interfaces are fine. However, you should print descriptive messages in an output log that allows an evaluator (i.e., TA) to understand what is happening in the war zone.

- A secondary goal of this assignment is to make you familiar with source code control systems (e.g., git, mercurial, svn) which are considered quite essential for any Computer Scientist. Hence, please use a github repository and finally share the repository link in the submission document.
- A related goal is to ensure you properly test your code. You can create specific scenarios and/or inputs to test your code and verify that it works as expected. (This must be done to present the correctness of your solution)

---

## B. Evaluation and Measurement

1. Deploy at least 9 soldiers and one commander. You should use at least two machines for demonstrating your code, but are free to use more as well. Appropriate use of RPCs is a must.
2. Conduct a few experiments to evaluate the behavior of your system under different scenarios. Some scenarios which must be shown are where the soldiers are impacted by the missiles and how the commander stops communicating to such soldiers.

---

## C. Submission

Zip/Tar the entire source and executable files with your id as the zip/tar file name (e.g. 2022h1030004P.tar) and upload on <link to be provided later>. Each program must work correctly and be **documented**. The zip file you upload to should contain (apart from the code):

1. A **readme.txt** file with group details and instructions required to run your code (including any installation instructions).
2. **Output.log** file: The output generated by running your program. Print informative messages when a commander and soldier receives and sends key messages and the casualty count. Do not print multiple flooding messages since it will clutter your output.
3. **Program design** file: A separate document describing the overall program design, a description of "how it works", and design tradeoffs considered. You must clearly describe how we can run your program. If we can't run it, we can't verify that it works.
4. In the same file as above, write a separate description of the tests you ran on your program to convince yourself/us that it is indeed correct. Describe the tests, inputs and outputs of the tests (and how the output shows that the test was a success). Also describe any cases for which your program is known not to work correctly.
5. Please note that there must be appropriate comments in the code.
6. Also include a small video (screen recording) which demos your running code.

**Submit only one zip/tar file per group.**

---

## D. Grading policy for the assignment

1. works correctly ----- 60%
2. Thoroughness of test cases and test output----- 20%
3. Quality of design and creativity ----- 10%
4. Understandability of doc and code comments----- 10%

**Note:** Plagiarism in any form (from online sources or from other batch mates) will fetch ZERO marks and can invite disciplinary action (for both sides - one who copies and one who provides his/her code to copy).