UNIVERSITY of STIRLING

CSCU9V6 Concurrent and Distributed Systems

Assignment 2: Distributed Systems

Project Task: Basic Problem (40%)

This assignment covers material presented during the lectures on distributed systems and builds upon the work in the two distributed practicals focussing on sockets. This assignment will extend the theme of a Distributed Mutual Exclusion (DME) system based on sockets. In the practicals you have developed a solution based on a token ring. In the lab you had a class implementing a *ring node* and a *start manager* class to inject a token to start off the system.

In this assignment you will develop *a socket-based centralised DME* using a coordinator node. The *coordinator* program runs on a JVM on one of the computers in the system. The *coordinator* sequentially passes a unique *token* to each single *node* in the system that requests it. When granted the unique token, a node will be allowed to execute its critical section, and on completion of the critical section code will then return the token to the coordinator. All nodes are connected to the coordinator.

You will be issued with a skeleton solution as a starting point. You **MUST** use this starter code as a base in your solution. It is strongly recommended that you start by studying the details of the problem and the general functioning of the provided implementation.

Each *node* will be running on a specific *host* and *port* (host and port must be configured at launch time, e.g. by using a command line parameter). You may run the different applications for the nodes and the coordinator on the same machine, in separate JVMs, for ease of use. Each (non-coordinator) node will perform the following repeated sequence:

1. *request the token from the coordinator* by passing the node's IP and port to the coordinator (the coordinator listens on port 7000 at a known IP address).
2. *wait until the token is granted by the coordinator.* Granting the token can be implemented by a simple message the coordinator sends to the node, analogously to what you did in the token ring-based solution.

3. *execute the critical region*, simulated by sleeping for about 3-5 secs, and outputting indicative messages marking the start and end of the critical section. **Important**: with multiple nodes running on the same machine, in different windows, it must be evident from the messages that only a single node at any one time is executing the critical session. Implement randomised sleep durations to experiment.
4. *return the token to the coordinator.* This can also be done through a simple message (the coordinator listens on port 7001).

The *coordinator* consists of two concurrent tasks that share a buffer data structure:

- a *receiver* that listens for requests from nodes. On connection from a client, the receiver will spawn a thread (C_connection_r) that receives IP and port and stores these in the buffer using a *request* data structure, defined in the code.
- a *mutex* thread that continuously fetches available requests from the buffer in a FIFO order. For each request, the *mutex* thread issues the token to the requesting node, by connecting to the node's indicated port. It then waits for the token to be returned by means of a synchronisation (on port 7001).

All sockets/servers must be suitably closed. All exception catches must print appropriate messages, declaring occurred exceptions. There shouldn't be any! All nodes must print suitable messages showing their activities, including the start and end of executing a critical section for nodes, and granting and receiving the token back for the coordinator as the minimum. Use randomised sleep times so the order of granting the token to nodes varies fixed.

Advanced Features (40%)

You can enhance the basic solution by implementing additional features:

1. Implement a file logging mechanism using a single text file for all nodes and the coordinator. That is, nodes log their start of critical section and return of token to the coordinator in the file using timestamps. The coordinator may log token requests and issuing and queue length.
2. Implement a priority discipline in the request queue. Imagine a critical process that will have priority in acquiring the token over all the other nodes.
3. Implement a solution to avoid starvation of low priority nodes.
4. Modify the nodes so that they can deal with the coordinator being closed down (or crashing!)
5. Implement full gracefully closing down of the system initiated by a closing down request initiated by a node.

Report (20%) and Submission

The deadline for handing this project is **Friday 5 April 2024**. Separately, you will be asked to demonstrate your solution to the given problem. This will take place in the usual practical sessions from **Tuesday, 9 April**. A separate demo schedule will be distributed nearer the time. For the Canvas submission you should prepare a single document which includes a report (roughly four pages plus a cover sheet with your student number) discussing any assumptions you have made, and a description of your solution, **as well as** the code listings of your program. Make sure the source code is formatted appropriately and is readable. Please use appropriate report headings.

The report should include appropriate diagrams of the design and screen shots of your application. The report should discuss any assumptions you made, and **your solution**. **Importantly**, it should **critically reflect** on your application and the functionality you implemented and the design you adopted. It should further include appropriate diagrams of the design and screen shots of your application in operation. Describe the various classes, their relationships, and outline the class methods. The document also should provide details as to how complete your solution is, and if applicable, any special cases when your program is not working correctly.

In short, your assignment should consist of (in a single document):

- Cover sheet giving the module, title, and student number
- Report of about 4 pages discussing the problem and your solution, plus a complete code listing (make sure this is readable)
- Zip file of your **source** code (for reference purposes only)
- Separately, a demo of your solution (from 9 April)

You should submit your document **via Canvas**. You are expected to **demonstrate** your solution, so please do test out the final version. Make sure that what you submit does work in some fashion. You can delete or comment out incomplete code before submission. After submission, you should leave your files in the folder **untouched**, until you are notified of your grade for this assignment.

Demos

Demonstrations will be seen in the lab sessions from **Tuesday, 9 April** in the usual practical class. Demos must be using the same code base which was submitted. Demoing an improved version of your software since submission is **not acceptable** and seen as a form of **academic misconduct**. We may check the dates your files were last edited at the demo. During the demo we will ask you to explain/talk-through specific parts of your code. **Please note, the demos form part of your submission, and so missing the demo constitutes a non-submission of your whole assignment.** If you cannot attend the demos due to good reason (such as medical) please see Dr Mario Kolberg to explain the situation and arrange a new time for your demo.

Generative AI

It is important to read the University guidance on the use of generative Artificial Intelligence (AI), for example ChatGPT, at the University.

Some limited used of generative AI, such as ChatGPT is allowed, e.g. to get you started with the report. **Any use of AI material kept within the text or your code must be appropriately identified, placed in quotations, and cited**. See https://libguides.stir.ac.uk/Referencing/AI for information on referencing AI and read the section on 'Use of Generative AI in Academic Work'.

You are NOT ALLOWED to use generative AI to solely produce substantial parts (or the whole) of the assignment report or code.

You must include a **cover sheet** in your report with student number and a generative AI statement explaining whether, and in what way, generative AI has been used to help with the assignment. The statement is provided at the end of this assignment description.

Plagiarism

Work which is submitted for assessment must be your own work. All students should note that the University has a formal policy on Academic Integrity and Academic Misconduct (including plagiarism) which can be found at https://www.stir.ac.uk/media/stirling/services/academic-registry/documents/policy-procedure-academic-integrity-v5.docx

Plagiarism: We are aware that assignment solutions by previous students can sometimes be found posted on GitHub or other public repositories. Do not be tempted to include any such code in your submission. Using code that is not your own will be treated as "poor academic practice" or "plagiarism" and will be penalized.

To avoid the risk of your own work being plagiarised by others, do not share copies of your solution, and keep your work secure both during and after the assignment period.

Collusion: This is an individual assignment: working together with other students is not permitted. If students submit the same, or very similar work, this will be treated as "collusion" and all students involved will be penalized.

Contract cheating: Asking or paying someone else to do assignment work for you (contract cheating) is considered gross academic misconduct, and will result in termination of your studies with no award.

Plagiarism means presenting the work of others as though it were your own. The University takes a very serious view of plagiarism, and the penalties can be severe. We check submissions carefully for evidence of plagiarism, and pursue the cases found. Penalties range from a reduced grade, through a Fail grade for the module, to being required to withdraw from studies.

Assessment Criteria

In this assignment we shall be assessing your work with respect to various criteria, the most important of which are:

- Correctness of operation
- Appropriate use of programming constructs
- Intelligent code comments and consistency, legibility and tidiness of program layout
- Clear and comprehensive report

The marks for the assignment count for 50% of the module grade with the remaining 50% being allocated to the classtest. The split for the assignment is basic solution (40%), advanced features (40%) and report (20%).

Late submission

If you cannot meet the assignment hand in deadline and have good cause, please see Dr Mario Kolberg to explain the situation and ask for an extension. Submissions will be accepted up to **seven days** after the hand in deadline (or expiry of any agreed extension), with the mark being lowered by three points per day. After seven days the work will be deemed a non-submission and will receive an X (no grade).

Backups: You are advised to make backup copies of your work regularly.

Generative AI statement (to be included on the cover sheet of your assignment report)

Use of AI

I acknowledge that :

1. No content generated by AI technology has been knowingly presented as my own work in this submission;

or

2. I used <insert AI tool(s)/link/date of access> to generate materials that are included within my submission.

(delete and fill in as required)

Presentation, structure, and proofreading

I acknowledge that:

1. I did not use AI technology to assist in structuring or presenting my submission;

or

2. I used <insert AI tool(s)/link/date of access> in structuring or presenting my submission;
3. My final submission has been proofread by a dedicated proofreading software or AI tool (*please specify*).

(delete and fill in as required)