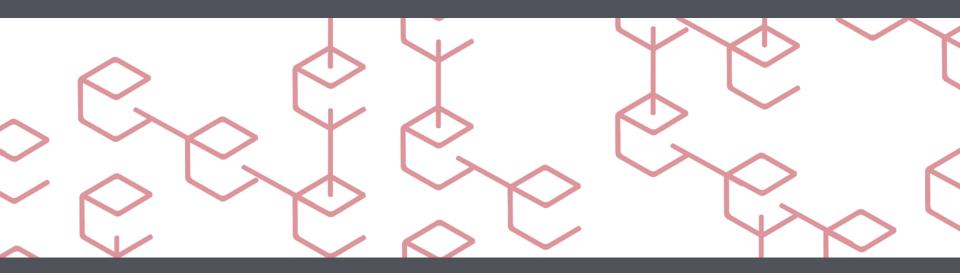
Practical Exercises and Coursework Support



Coursework Assignment:
Hints and Tips



Assignment Tasks

Options

Choose ¾ of the following options:

- 1. Threading during Proof-of-Work
- 2. Adjusting the Difficulty Level in Proof-of-Work
- 3. Mining Settings
- 4. Your own idea

35 marks (15 for implementation and 20 for report)

Research and reference to other Blockchain implementations highly recommended for these sections.



Threading during Proof-of-Work

- Utilise the full computational power available using:
 - Possible Solution: Multi-threading e.g. C# Threads
 - Advanced Solution: GPU implementation e.g. CUDA C
- Requirements:
 - E-Nonce (Prevent duplication of work)
 - Coordination of threads
- Reporting Suggestion:
 - You can perform a comparison study comparing mining times of single- vs. multi-threaded solution
 - Vary difficulty level to examine the point at which overheads are overcome



Adjusting the Difficulty Level in PoW

- "Dynamic" or "Adaptive" Difficulties
 - Decide your own 'block time' and implement an appropriate algorithm
- Principles and Requirements
 - Establish a target block time
 - Continuously call Mine()
 - Calculate the "Mine Rate" based on previous block timestamps
 - Adjust the difficulty based on the difference between target and actual/observed block time
- Note the current implementation adjusts difficulty by a factor of 16 with each step which you may find to be too significant – consider alternative solutions.

Adjusting the Difficulty Level in PoW

Starting off:

- Need to have revised thread management in C#
- Please refer to the notes already provided
- Can start with a moderate level of difficulty say 5 and
- Now you need to benchmark this i.e. measure how long it takes with or without threads and how many threads to satisfy a workable level of difficulty and block time
- This just means repeating the PoW using Mine() for different levels of difficulty with or without threads
- Block time = timestamp for block (n) timestamp for block (n-1)

Adjusting the Difficulty Level in Reading

- Exponential growth function
- In the **exponential growth** of f(x), the **function** doubles every time you add one to its input x. In the **exponential decay** of g(x), the **function** shrinks in half every time you add one to its input x ... e.g., Adaptive Proof of Work (APoW)
 - Establish a target block time
 - Calibration "Block-To-Block Stair-Stepping"
 - >Median block time of a batch of N recently-mined blocks
 - Exponential Decay Difficulty Adjustments

Combat attacks (Calfieb mining)

- >Adjusts the difficulty according to 3 blocktime
- > After a significant spike in hash rate and difficulty followed by a huge drop in hash rate and difficulty, each new block becomes much easier until the difficulty stabilizes according to the true size of the network

University of Reading

Mining Settings

- Selection of pending transactions for mining
- Possible Options:
 - 1. Greedy (Largest fees)
 - 2. Unpredictable (Random)
 - 3. altruistic (Oldest existing)
 - 4. Address based (Personal transactions)

This just requires

- i) Use the existing code to create many transactions with variable amounts and fees
- i) Write code to deliver that is to select the right transactions from the transaction pool in each of the 4 case
- ii) Using the Windows Form to create the 4 buttons that would select and display the right number of transactions up to a stated limit according the selection strategies 1-4 above.



Your own idea

Possible Suggestions

- 1) Implementation of a different consensus algorithm (e.g. Proof-of-Stake)
- 2) Create multiple nodes running in a local network, automating the generation of transactions.
- 3) (Advanced) Smart Contracts