Contact person: Anwitaman DATTA (Anwitaman@ntu.edu.sg)

NOTE: You are encouraged to define your own project on any of the following broad topics: cryptocurrency network analysis, developing a "toy/proof-of-concept" application on top of a blockchain, or anything that fits the general scope of this course, for instance, doing a study similar to the papers in the reading list (please discuss with me the objective and scope of your project by email). If you do not have any idea of what to work on, you can do the following project described below. You can also use the complexity of the below project to gauge the level of effort expected in your project, to scope it accordingly.

Project report format/length guidelines are identical to project 1. Due date is 23 November 2020.

Project: RAID-6 based distributed storage system

The high level objective of the project is to build a reliable distributed storage system supporting the following basic functionalities:

1. Store and access abstract "data objects" across storage nodes using RAID-6 for fault-tolerance
2. Include mechanisms to determine failure of storage nodes
3. Carry out rebuild of lost redundancy at a replacement storage node.

The above objectives are a minimal expectation of the project, where you can just use folders as "abstract nodes" and likewise, the abstract objects could be synthetically generated data blocks. It is also okay if the minimal implementation can support only one specific configuration, e.g, 6 data stripes + 2 parities, as was the specific example of Linux RAID-6 that has been discussed in the lectures.

On top of the minimal implementation, students aspiring better grades are encouraged to consider further features. A non-exhaustive list of such features is as follows. If you have any other aspects in mind, feel free to discuss the ideas with the lecturer.

4. In the objective "1" above, it is deliberately vague as to what data objects mean. You can consider them as something like x-MB data chunks. However, actual files may be much smaller, or larger than the quantum of data which is treated as an object in the system. In a practical system, it is necessary to accommodate real files of arbitrary size, taking into account issues like RAID mapping, etc.
5. Instead of using folders to emulate nodes, extend the implementation to work across multiple (virtual) machines to realize a peer-to-peer RAIN.
6. Support mutable files, taking into account update of the content, and consistency issues.
7. Support larger set of configurations (than just 6+2, using a more full-fledged implementation).
8. Optimize the computation operations.
9. Anything else (please feel free to discuss during the course of the project)

Reporting results: Your report should be in IEEE or ACM paper format, and normally between 5-8 pages (though if you have much more information to convey, feel free to go for a longer report). The report needs to describe your system architecture, lessons learnt from the implementation process as well as setting up the experiments (you need to identify what all may be interesting to measure, depending on the rigor of your implementation), and the experiment results themselves.

You are free to choose the programming environment + language of your preference, and existing finite field operation libraries. It is preferable that you do not use an existing erasure coding library, but instead implement it yourself, unless if you are aiming for more the advanced features, in which case, please discuss your plan with the lecturer.

Note that I may request some groups to carry out a live demo, and schedule will be determined in a mutually agreeable ad-hoc manner.

Due date: 23rd November 2020.
If you are trying out some of the advanced features, and need a few extra days, please discuss with the lecturer enough in advance (no later than 20th November 2020).
Submission mechanism: Source code (well commented, and with necessary read-me documents) + report by email to Anwitaman@ntu.edu.sg