Homework4

Part 1 (DAG: GHOST & Conflux)

Read the following materials about DAG, GHOST, and Conflux:

1. GHOST protocol paper: https://eprint.iacr.org/2013/881.pdf
2. Conflux paper: https://www.usenix.org/system/files/atc20-li-chenxing.pdf

Then answer the following questions.

Now suppose a blockchain using the Bitcoin protocol runs at a very fast block generation rate, so that only 40% of the generated blocks are on the main chain on average when the blockchain runs normally without attackers. Now there is an attacker controlling 33% of the total network hash rate. Is it possible for this attacker to deterministically launch double-spending attacks? Why?

For the previous question, if the chain uses GHOST protocol instead. All the rest conditions stay the same. Is it possible for the attacker to deterministically launch double-spending attacks? Why?

One claims that by using GHOST protocol, we can now simply lower the difficulty to speed up the block generation to achieve arbitrarily high transaction throughput, because fast block generation no longer harms the security. Is this claim true or not? Why?

Why in GHAST protocol, sometimes we assign same weights to all blocks, but sometimes we assign different weights to different blocks?

In Conflux, is it possible for an attacker to create a malicious block choosing a very early block as its parent block to disrupt the transaction total order? Why?

How much hash power an attacker must control for him/her to deterministically launch double spending attacks in Conflux? Why?

Part 2

Achieving consensus in distributed systems with arbitrary (Byzantine) failure is a classical problem that has been studied by researchers for twenty years. Read the Practical Byzantine Fault Tolerant paper (http://pmg.csail.mit.edu/papers/osdi99.pdf) and answer the following questions. Note that optionally, you may also want to read Tendermint materials to see the application of Byzantine Fault Tolerant in blockchain (https://github.com/tendermint/tendermint).

Suppose we run PBFT algorithm on a cluster of 21 machines. What is the maximum number of machines we will be able to tolerant for failure simultaneously?

A node in PBFT waits for prepare/commit messages from **two thirds of nodes** (replicas) during the prepare/commit stages. What would happen if the node instead waits for prepare and commit messages from **only a simple majority** of other nodes? Would the modified PBFT be secure? If not, please explain with a counter example.

PBFT relies on the primary node to send out pre-prepare messages to drive the consensus process. What would happen if the primary node is malicious or fails? How does PBFT handle this situation?

Could PBFT skip the commit messages? Suppose nodes in PBFT commit-local directly after receiving 2f + 1 prepare messages. What could go wrong? Please explain with a counter example (Consider the case where a view change happens immediately after one or two nodes commit-locally for a request).