Analyzing Selfish Mining + Eclipse Attack in Blockchain P2P network

In a peer-to-peer (P2P) blockchain network, efficient block propagation is crucial for maintaining network synchronization. In the first part of this project (assignment-1), you implemented a basic P2P network simulator with the following characteristics:

- Nodes directly broadcast complete blocks and transactions to their peers.
- Receiving nodes:
    - Save new blocks/transactions to memory if they encounter them for the first time.
    - Forward them to immediate peers for further propagation.
    - Discard duplicate blocks/transactions.
- This results in a simple, straightforward propagation mechanism.

Required Modifications: Enhanced Propagation Protocol

In this assignment, we are introducing 2 types of nodes - honest and malicious. Honest nodes work according to the protocol, whereas Malicious nodes try to exploit the protocol for their advantage. You will modify the existing simulator to implement a two-step block propagation mechanism, which is generally used to reduce network congestion.

- **Hash-Based Broadcasting:** **5 marks**
    - Instead of broadcasting the full block, nodes will first broadcast only the hash. (actual hash of block header fields you used in assignment-1)
- **Data Request Mechanism:** **5 marks**
    - Upon receiving a hash:
        - If the node receives the hash for the first time from any peer
            - It sends a **"get"** request to retrieve the full data to the peer who sent the hash. (separate requests for separate hashes)
        - If the node has already seen the hash
            - If the full block corresponding to this hash has already been received, discard the hash.
            - If the full block corresponding to this hash has not been received yet,
                - **If no timeout exists** for this hash, the node sends a **"get" request** to the sender. (refer 3rd point below for a clear explanation about the timeout).
                - **If a timeout is already running for this hash**, the node does **not** immediately request the block again. Instead, it stores the sender's address and will send a request after the timeout expires (if needed).

- Hash/Block Propagation:
  - Malicious nodes
    - For honest blocks, these nodes propagate the hash using the eclipse attack explained below, and they won't propagate the entire honest blocks.
    - For malicious blocks, these nodes propagate hash/block as and when required according to the selfish mining attack explained below.
  - Honest nodes
    - Once they receive the hash, they wait for the entire block, and only after getting the entire block do they propagate the hash. (And they propagate the block on "get" request)

- **Timeout Mechanism:** **5 marks**
  - Implement a timeout system for data requests (The timeout time **Tt** will be a command line input parameter):
    - When a node sends a **"get"** request, it starts a timer corresponding to that hash. (Each node may have **multiple timers running simultaneously**, one per hash)
    - During the timeout period:
      - The node waits for the complete data from the requested peer.
      - In the meantime, if the same hash is received from another peer Nn, the node continues waiting – it doesn't reset the timer and doesn't send the **"get"** request to the other node Nn.
    - If the timeout expires without receiving the full block, the node can send the **"get"** request to Nn.

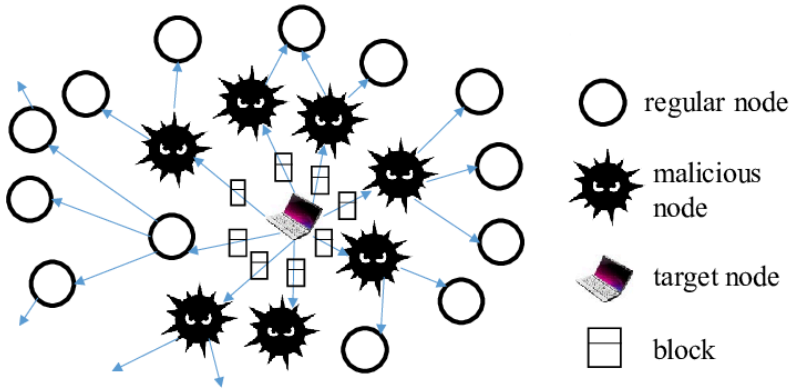
Hashing Power and Network Latency Assumptions

- All the nodes have equal hashing power.
- Network topology is the same as discussed in assignment 1.
- All the malicious nodes are fast, and all the honest nodes are slow.
- Latency is the time between which a message m is transmitted from the sender i and received by another node j. Choose the latency to be of the form ρij + |m|/cij + dij
- The propagation delay (ρij) of a link can be chosen from uniform distribution - 10ms to 500ms.
- Queuing delay (dij) is calculated the same as in the assignment 1
  - dij = 96kbits/cij
- The link speed (cij) is calculated the same as in the assignment 1
  - cij is set to 100 Mbps if both i and j are fast and to 5 Mbps if either of the nodes is slow
- Different entities sizes:
  - Block - 1MB
  - Transaction - 1KB
  - Hash size - 64B

Selfish Mining + Eclipse Attack: **55 marks - distribution below**

You have to simulate this attack. Refer to the next part of the assignment for a detailed explanation of the attack

Eclipse Attack Mechanism



A **malicious node** can exploit this timeout mechanism to delay or prevent block propagation using the following strategy: (Advanced double spend attacks are possible due to this, but that is not your concern for this assignment)

- **Attack Execution:**
  - When the malicious node receives a new honest block:
    - It broadcasts only the block hash to its connected victims (i.e., connected nodes).
    - It withholds the actual block data.
- **Attack Persistence:**
  - The malicious node sends the hash to the victim node and does not send the entire block even after getting the “get” request from the victim node.
  - If a victim node receives the same hash from an honest peer before the timeout expires, it can send a **"get"** request to the honest node after the timeout and get the block from that honest node.
- **Network Impact:**
  - Victim nodes may fail to receive the actual block data if they are not connected to any honest node.
  - Different honest nodes may have very different local blockchains.

Attack Model

Attacker Assumptions

- The percentage of malicious nodes in the network is an **input parameter**.
- There is a network where all the nodes (honest/malicious) are connected according to the instructions given in assignment-1. Also, assume that there is a separate overlay network through which **all the malicious nodes are connected** (max 3-6 malicious peers per malicious node). This assumption results in the fact that all malicious nodes can work together according to the selfish mining attack explained below. The propagation delay (ρ_{ij}) of a link in this separate overlay network can be chosen from uniform distribution: 1 ms to 10 ms.
  The remaining latencies remain the same (i.e., same link speed and queuing delay as mentioned previously).
- Malicious nodes maintain a **(nearly) consistent global view** of the blockchain by continuously sharing information among themselves, ensuring synchronization except for the small differences caused by delays in their separate overlay network, and they reliably receive all blocks created by honest nodes from the honest network.
- All malicious nodes are controlled by a single entity (known as the **"ringmaster"**, which will be chosen randomly at the start of the simulation), meaning:
  - They **combine their hashing power**, allowing the **ringmaster** to generate the next block. For example, if there are "m" malicious nodes and each malicious node contributes "x" amount of hashing power, the total hashing power of the malicious group (i.e., the power of the ringmaster) becomes m*x.

Selfish Mining + Eclipse Attack Strategy

- **Private Chain Maintenance:** **10/55**
  - When the ringmaster creates a block, it will first broadcast to only malicious nodes using the separate overlay network to maintain a hidden chain from honest nodes.
- **Strategic Block Release:** **20/55**
  - The malicious nodes broadcast the private chain to honest nodes iff: (Refer to the lecture slides on the selfish mining attack for more details of attacker strategy on private blocks propagation)
    - Length of the longest honest chain in the global view (seen by the ringmaster) = Length of attacker's private chain
    - Length of the longest honest chain in the global view (seen by the ringmaster) = Length of attacker private chain - 1
  - All the malicious nodes can start broadcasting the private chain into the original network parallelly when the ringmaster commands. They can do this because all the malicious nodes will have a private chain already using the faster overlay network. The ringmaster sends a message **(64B)** to the other malicious nodes using the overlay network saying

"broadcast private chain", and only after receiving it, the other malicious nodes broadcast the private chain into the original network.

- **Honest Chain Disruption via Eclipse Attack:** **25/55**
    - Malicious nodes strategically use the **Eclipse Attack to slow down** the **propagation of honest blocks** while ensuring that **attacker-generated blocks propagate fast** across the network when desired as per the selfish mining attack.
    - This may cause **honest nodes** to maintain **different local blockchains.**
    - Since **honest nodes are possibly working on different blocks** due to the Eclipse Attack, many forks may emerge on the honest chain.
    - This combined attack (eclipse + selfish) allows the malicious nodes to maintain an advantage over the honest nodes.

Performance Evaluation **30 marks - distribution below**

Use an appropriate visualization tool to study the **blockchain tree at a node** (suitable choices can be gnuplot, matlab, or other visualization tools). We expect you to show the blockchain tree at the ringmaster node using some visualization tool. Make sure that blocks generated by malicious nodes and blocks generated by honest nodes are shown with **different colors in the visualization**. **5/30**

You must **analyze and report the following metrics in your report**: **15/30**

- The ratio of the number of blocks generated by malicious nodes in the longest chain at the ringmaster to the total blocks in the longest chain at the ringmaster for various % malicious nodes in the network like 5,10,15, and so on with different timeout times(**Tt**).
- The ratio of the number of blocks generated by malicious nodes in the longest chain at the ringmaster to the total blocks generated by the malicious nodes for various % malicious nodes in the network like 5,10,15, and so on with different timeout times(**Tt**).
- Modify the simulator to **remove the Eclipse Attack,** i.e, malicious nodes immediately forward the honest full block to the requesting node as soon as they receive a **"get"** request for the corresponding hash.
- Recompute the above two ratios and compare:
    1. **Selfish Mining + Eclipse Attack**
    2. **Selfish Mining Alone**
- Discuss the observed differences and explain how removing the Eclipse Attack impacts the above ratios.

Also, **add the answers to the questions below**: **10/30**

- How does increasing the timeout time (Tt) affect block propagation in the presence of an eclipse attack?
- Suggest countermeasures to mitigate or weaken the above-described attack. [open-ended]

Bonus (up to 20 marks) - Implement your suggested countermeasures in the simulator.
Bonus marks will depend on the effectiveness (in terms of attack mitigation and network overhead) of your solution.