CS765-HW2

Namoju Karthikeya, 200050084
Dhvanit Beniwal, 200050035
Likith Rajesh, 200050025

In our simulated model that ends after a finite block limit is reached in the blockchain tree, there is often some level of inconsistency among the honest nodes about which chain is the LVC when the simulation ends because by design the adversaries try to split up the honest nodes by releasing competing blocks from their secret chain right when a new block is circulating.

So when counting things about "the public LVC" like total number of blocks / blocks authored by either adversary we take the median value of what is reported by all nodes in order to get rid of any outliers. Also, without loss of generality, we study from the POV of the first adversary only, who we refer to as 'the/our adversary' or 'self', and we call the second adversary 'the other adversary'.

1 Functions & Structs of interest

Please refer to the following functions or structs in the code for the implementation of critical aspects of the simulation

- `class SelfishPeer : public Peer {...}`
- `SelfishPeer::StartMining()`
- `SelfishPeer::BroadcastMinedBlkOp()`
- `SelfishPeer::ReceiveAndForwardBlkOp()`

Note about interpreting observations for high ζ

We observed, especially for larger ζ, a common theme which was that the adversaries (because of their hashing power advantage) have both amassed a long secret chain of blocks waiting to be released one by one as honest miners mine on a third chain (all originating from a single ancient block). But since our selfish miners are still modelled as singular nodes in the network, the majority will not recognise either of their versions as the LVC yet.

It can only be after the honest miners catch up with them that the adversaries are forced to release the last 2 blocks together to finally undermine the honest chain and (atleast one of them will) reap the benifits. If the honest miners don't catch up before the simulation ends, both adversaries lose as they wasted resources to collect a long secret chain that nobody saw.

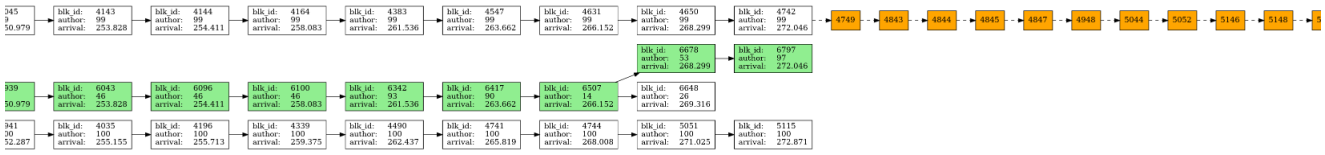


Figure 1: adversary-1 (peer 99)'s POV

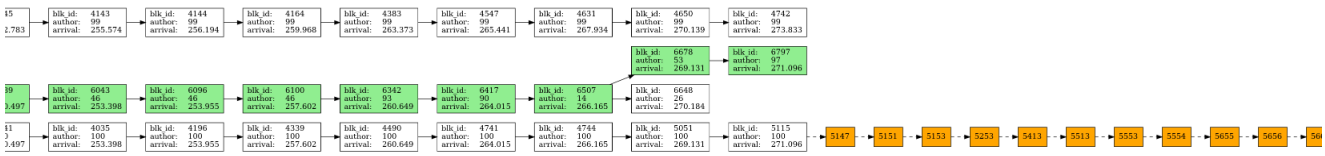


Figure 2: adversary-2 (peer 100)'s POV

Of course one obvious solution to deal with this issue would be to let the simulation run for longer instead of running more simulations. The problem is that the higher ζ gets, the longer the simulation must be ran to get the same level of statistical consitency (which is computationally intensive).

Moreover for high enough values of ζ (35% each etc), even if neither adversary on its own has a majority hashing power, one of them might have more hashing power than the honest nodes meaning that the honest nodes cannot be 'expected' (probability wise) to ever catch up no matter how long the simulation is ran.

Because of these reasons when we make observations by running many simulations instead of one long simulation, we usually see that there is a large variance in the observations whenever ζ gets large. So we decide to represent these as violin-plots (same as box-plots in matplotlib but less cluttered) instead of line plots so that it is immediately obvious how much meaning a certain mean value has.

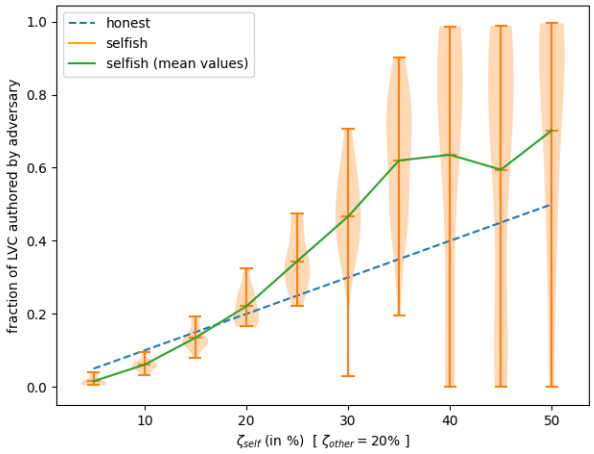
Observed values

The three quantities we will study are MPU_{adv} (out of all blocks adversary mines, how many end up in LVC), $\text{MPU}_{\text{overall}}$ (what fraction of the blockchain is the LVC ; so this is an inverse measure of amount of branching) and earnings of an adversary (out of all blocks in the LVC, how many authored by our adversary), starting with the third one.

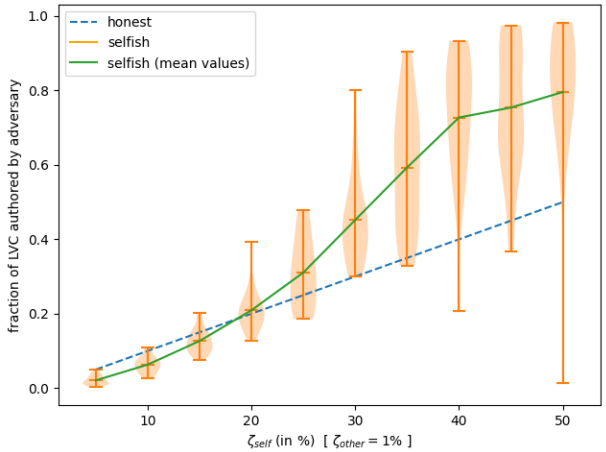
2 Effect of ζ on adversary's earnings

2.1 effect of ζ_{self}

Suppose we fix $\zeta_{\text{other}} = 20\%$ and vary ζ_{self}. For smaller values of ζ it's (qualitatively) similar to an uncontested selfish miner. There might be some negative effect from whatever competition this adversary faces from the other but it's 'constant'.



The observed 'decrease' (from the trend) in earnings for higher ζ_{self} (after 35% in the graph) is a result of two reasons: first is that our finite simulation cuts off 'earlier' than what a steady-state model would have predicted. Second is an artefact of the greedy nature of the selfish algorithm (more on that later). If we wanted to assess how big the first component is, we could look at the case when the other adversary has negligible effect (suppose same hashing power as an honest miner).



We observe that in this latter case the graph doesn't dip as low and there are fewer outliers at the bottom. This means that the negative effect on our adversary due to the existence of another adversary, even if weaker, is more pronounced if we have more hashing power:

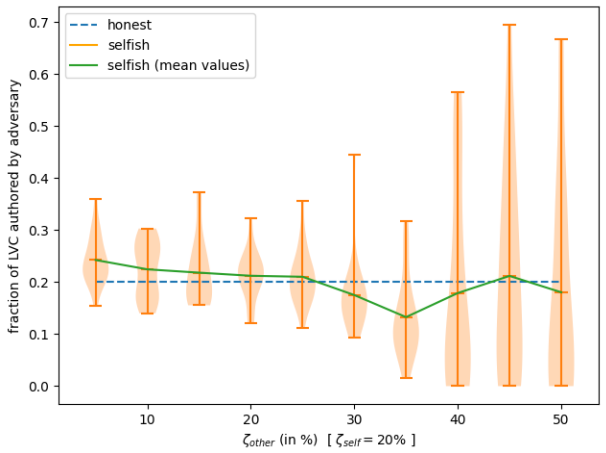
The weaker adversary is more likely to get caught up by honest miners than a stronger adversary. This would force it to release all of its secret chain and it gets to author this chain publicly. The stronger adversary has also released an equally long segment from its stash after seeing this but very few miners recognise it yet as the LVC.

By not contesting the weak adversary even when it is perfectly able of it, the stronger adversary is playing the long game where he's waiting for someone to catch up to him before he releases an even longer chain and author it publicly. But because he might not actually get to do that before the simulation ends, all he manages to accomplish was make it easier for the weaker adversary to scam the honest miners. That's where our adversary lost a share of the LVC.

This is because of the greedy design of the selfish mining algorithm. There is a large reward but there is always a prospect of making it larger at the cost of receiving it later.

2.2 effect of $\zeta_{\text{other\_adversary}}$

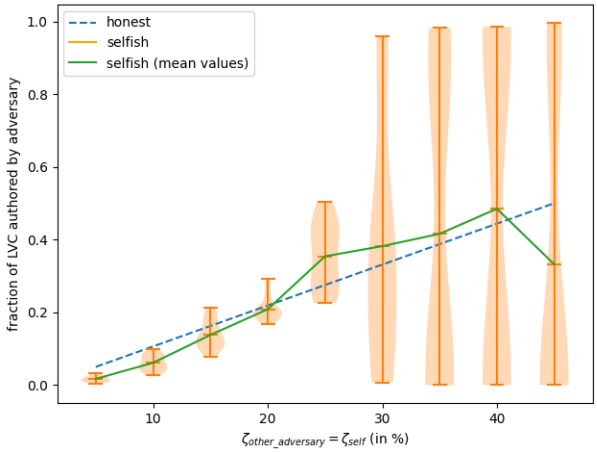
Suppose we fix $\zeta_{\text{self}} = 20\%$ and vary $\zeta_{\text{other\_adversary}}$.



We see that overall there isn't a very drastic effect of changing the other adversary's power on the earnings of our adversary except that: in general, observations have a higher variance for higher ζ; our adversary gains lesser if other adversary gets more powerful (competition); and at the right end of the graph there is as before, an opposite effect where the higher power adversary is letting the lower adversary win a little more than what we expected because it plays the long game too often and then the simulation ends.

(this is why there's quite a few outliers where the weak 20% adversary can get upto ≈ 70% ownership in the presence of a 45% adversary but in its absence there are no such extreme outliers)

It might also be interesting to note that if instead of keeping one adversary constant, we see what happens when both adversaries have the same hashing power and vary this value.

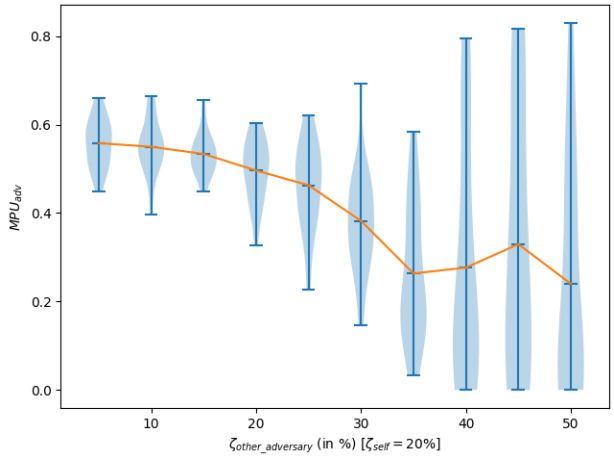


Turns out that they both wash out each other's attacks and don't gain much more than they would have if they were honest. When both have high ζ, specifically we see a transition around when ζ is comparable to combined hashing power of honest miners, the mean value of observations starts to lose significance as its as if you either win all or lose all.

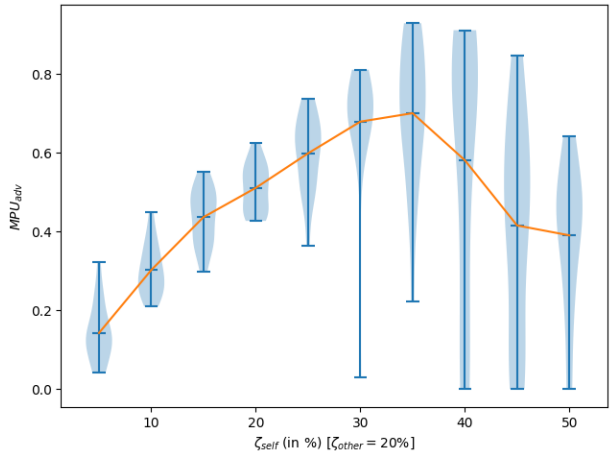
A large part of this recurring issue has to do with the fact that our selfish miners are singular nodes and do not have much influence over how much they can divide the honest miners' efforts. If they were present as different nodes in the network able to slow down honest block forwarding significantly, they wouldn't have to rely on honest miners to catch up before their greedy algorithm gives in and lets them reap benifits.

3 Effect of ζ on MPU

We observe effects similarly predictable to what we observed so far. If we increase the other adversary's hashing power, less fraction of our mined blocks will reach the LVC because of competition with him. But if the other adversary has too much hashing power, it inadvertantly lets us win a little more.



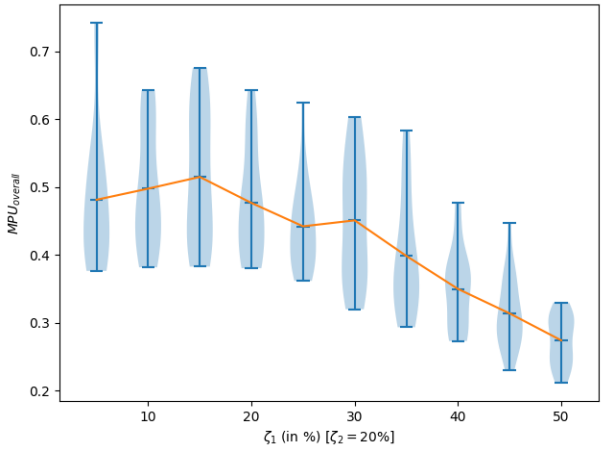
On the other hand, if we increase our hashing power, we get better 'efficiency' in terms of getting our mined blocks placed in everyone's LVC but if we have too much hashing power we mine too many blocks we never get to release (let alone place on the LVC) so the fraction goes down.



Finally, irrespective of whoever's blocks get placed on the LVC, if there's too much selfish mining them there's a lot of long branches beside the LVC. Interestingly, we have so far seen that the observations have high variance for higher values of ζ but here it's the opposite.

This is because both selfish miners consistently introduce new blocks and atmost one of them can ever get to call theirs the LVC. So regardless of who it is and how often, there's always too many branches.

On the other hand when selfish miners are not as powerful, branching due to the nature of the network's latencies show up that can vary a lot simulation to simulation.



Both these effects are exaggerated if (instead of ζ_2 constant) we vary both ζ with equal values