Probability of a Successful 51% Attack on Catalyst tip production

Dr. Pauline Bernat, Joseph Kearney and Francesca Sage-Ling

May 16, 2019

Abstract

The scope of this paper is to estimate a) the level of confidence when producing the Catalyst ledger state update produced by a set of active worker nodes during a ledger cycle, and b) the level of security around the production of the said ledger state update, *i.e.* the probability of a 51% attack on the network with the aim to tamper with the ledger state. The level of security is illustrated for various malicious scenarii with a serie of graphs, providing guidelines for implementations.

1 Production of a ledger state update

The Catalyst ledger consensus mechanism is not based on a competitive process. Instead, the nodes in the network collaborate to collectively build the Global Ledger State Update (or GLSU) used to update the ledger state at regular time interval (or ledger cycle). At the end of a ledger cycle, new KAT tokens are injected into the system such that all the nodes that contributed to producing the correct GLSU receive a share of that reward.

A GLSU is composed of a list of transaction entries extracted from the transactions broadcast to the network during a ledger cycle. Nodes validate these transaction entries against the accounts stored on the ledger and extract the expected changes to the account balances as represented in the transaction entries, when deemed valid. A GLSU is therefore a serie of changes in account balances that once broadcast to the network allow any user to update their local copy of the ledger state. Nodes who contribute to maintaining the ledger state are called *workers* rather than miners. Indeed, workers do not solve a computationally hard problem, but instead validate the transactions created by network users and broadcast to the network and use these to build the GLSU.

A pool of workers as defined on the Catalyst network is composed of N nodes, out of which V nodes are randomly selected to be active workers for a ledger cycle, i.e. the V nodes are selected to produce the GLSU and as such are called *producers*. The remaining $N - V$ nodes are *reservists* nodes, waiting to be selected at a latter ledger cycle. As discussed later in the paper, the ratio V/N directly influences the level of security over the GLSU issued by the producers. The producers in a pool of workers build a GLSU based on the same set of transactions transferring KAT tokens from a known list of digital accounts. The random selection of producer nodes among a pool of workers as well as the relative size of the pool set and subset determine the security and confidence behind the production of GLSU, as discussed in this paper.

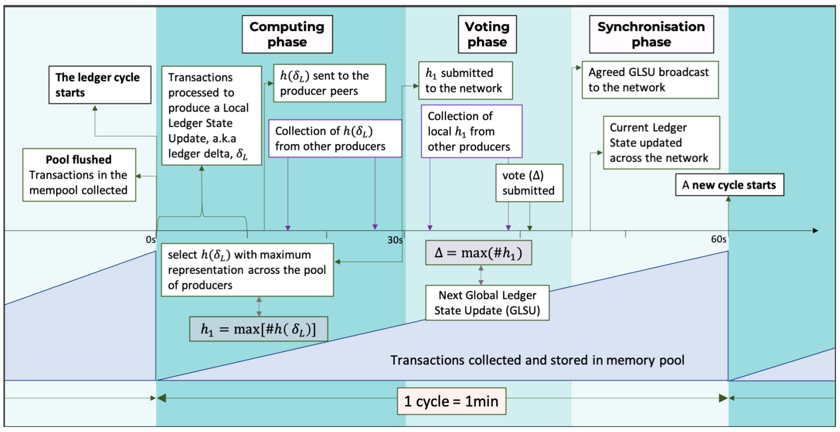

Computing phase
Voting phase
Synchronisation phase
The ledger cycle starts
Transactions processed to produce a Local Ledger State Update, a.k.a ledger delta, δ_L
$h(\delta_L)$ sent to the producer peers
Collection of $h(\delta_L)$ from other producers
h_1 submitted to the network
Collection of local h_1 from other producers
vote (Δ) submitted
Agreed GLSU broadcast to the network
Current Ledger State updated across the network
A new cycle starts
Pool flushed Transactions in the mempool collected
0s
30s
60s
select $h(\delta_L)$ with maximum representation across the pool of producers
$h_1 = \max[\#h(\delta_L)]$
$\Delta = \max(\#h_1)$
Next Global Ledger State Update (GLSU)
Transactions collected and stored in memory pool
1 cycle = 1min

Figure 1: Illustration of the different phases of a ledger cycle.

During the first phase of the ledger cycle (or *computing phase* as illustrated in Figure 1), each active worker creates a Local Ledger State Update (LLSU) δ_L and broadcasts the hash of it, $h(\delta_L)$, to the $V-1$ other producers[1].

A producer N_i collects $V_i \leq V$ hash values and creates a local histogram. Each bin in the histogram represents the number of entries m_k collected for a given hash value h_k. The bins are sorted in ascending order, *i.e.* the first bin is the one with the higher count m_1, it represents the most popular ledger state update computed by the producers and locally collected (associated hash value h_1). The histogram contains V_i entries in total, spread in a variable number of bins, as illustrated in Figure 2 in a scenario where the producer collected 4 different hash values.

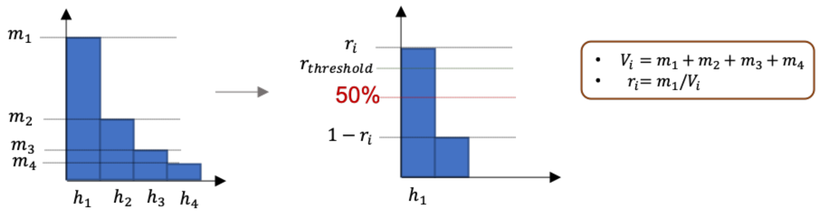


Figure 2: Illustration of the ditribution of hash values collected a producer node.

The relevant measurement for a producer to decide if the same ledger state update is found by a majority of producers is the ratio $r_i = m_1/V_i$. Although the distribution of the $V_i - m_1$ hash values in the remaining bins of the histogram are not significant to determine if h_1 should be selected by a producer as the hash of the likely correct ledger state update, this distribution could indicate malicious behaviours among the pool of workers (a second spike could for instance indicate nodes collusion or DDOS attack attempts).

In a perfect world, all the producers would compute the same hash value. In practise, $r_i \leq 1$ and a threshold r_{min} can be chosen to ensure that a sufficient number of producers produce the same LSU and that therefore a consensus can be reached over the production of the LSU during the ledger cycle. The statistical uncertainty on r_i depends on its value as well as the total number of hash values collected and should be taken into account when comparing the r_i against the agreed threshold r_{min}. In statistics, we refer to a confidence interval, also called margin error, as a mean to indicate the confidence interval around a value. A confidence level gives the level of confidence associated to a measure, *i.e.* how sure one can be of the measured value given the size of the data sample used to

[1]In Figure 1 and thorughout this paper, we ommit the discussion around the verification of the legitimacy of producers for a ledger cycle as well as the generation of the reward attributed to these producers, as this is beyond the scope of the topic discussed here.

V_i	m_i	r_i	Δr_i	$r_i - \Delta r_i$
2000	1656	82.80%	2.77%	80.03%
1800	1510	83.27%	2.85%	80.42%
1700	1331	78.29%	3.29%	75.82%
1600	1325	82.81%	3.10%	79.71%

Table 1: Example of different ratio found by a producer N_i when $V = 2000$ and $r_{min} = 80\%$.

obtain the said measure. It is expressed as a percentage and represents how confident one can be that the true ratio r lies within the confidence interval. The confidence interval on r_i is given by the formula: $r_i \pm z\sqrt{\frac{r_i(1-r_i)}{V_i}}$ where z is a score associated to the confidence level (1.96 for 95%, 3.291 for a 99.9% confidence level) and the remaining expression is the standard error of the ratio estimate.

In a perfect world, each producer would receive exactly V hash values. In practise, an producer collects V_i hash values with $V_i \leq V$. If $r_i \geq r_{min}$, the producer *should* consider $h(\delta_L)$ to be the hash a valid ledger state update. But if the number of values V_i collected by N_i is significantly lower than the expected total number of values V, the confidence interval on the ratio r_i may be too large for it to be considered a valid input in the next cycle phase. When $V_i \neq V$, it is indeed possible that $r \leq r_{min}$ but $r_i \geq r_{min}$ (where $r = m/V$). Given a confidence interval around r_i of Δr_i, a validator should therefore consider a ratio valid, given an agreed confidence level, if and only if $r_i \geq r_{min} + \Delta r_i$. As illustrated in table 1, assume that the subset of producers is composed of $V = 2000$ nodes and the agreed threshold is set to $r_{min} = 80\%$. The top row in the table considers the scenario where the producer N_i collects the maximum number of hash values ($V_i = V$) while the remaining rows shows scenarios where N_i collects less hash values ($V_i = \{1600,1700,1800\}$). In two cases, N_i computes a ratio r_i greater than r_{min}. However, in the scenario where $V_i = 1600$, it is easy to see that the statistical error on the ratio is such that the latter cannot be considered valid, as $r_i - \Delta r_i < r_{min}$. In the scenario where $V = 1700$, the producer computes a ratio r_i that does not pass the threshold r_{min}.

If $r_i \geq r_{min} + \Delta r_i$ the producer broadcasts $\Delta_i = h_1$ to the subnetwork of $V - 1$ worker nodes.

During the second phase of the ledger cycle (or *voting phase* as illustrated in Figure 1), the producer N_i collects $S_i \leq V_i \leq V$ hash values and builds a second histogram, counting the number of entries p_i for each Δ_i values collected.

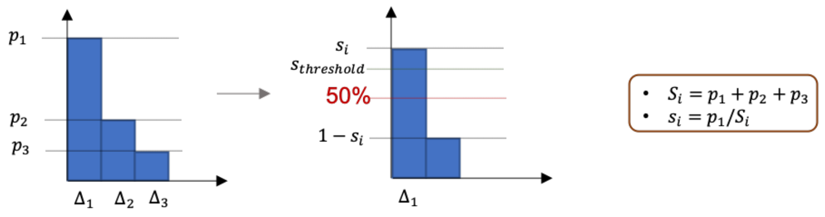


Figure 3: Illustration of the ditribution of hash values collected a producer node.

We denote $s_i = p_1/S_i$ the fraction of identical hash value of the dominant hash Δ_1 collected by a producer N_i. This is illustrated in Figure 3. A threshold $s_{min} + \Delta s_i$ can be applied on s_i similarly to when computing r_i to convince a producer that a consensus on the correct ledger state update is reached and a Global Ledger State Update (GLSU) has been found by the pool of producer nodes.

If $s_i \geq s_{min} + \Delta s_i$ the producer broadcasts $\Delta = \Delta_1$ to nodes in the network. A DFS node, after receiving q Δ values ($q_{min} \leq q \leq V$ where q_{min} is an agreed minimum number of Δ a DFS node should collect to be certain it received the correct GLSU), queries the correcting GLSU (recall that Δ is a hash value) and is able to diffuse the GLSU to the rest of the network. During the last phase of the ledger cycle (or *synchronisation phase* as illustrated in Figure 1), a user node requests the correct GLSU to a DFS node and verifies the validity of the latter by comparing the hashes of the GLSU sent by DFS nodes with the hash broadcast by active workers. The node can then synchronise its local copy of the ledger state.

A new ledger cycle begins with the random selection of a new set of V producers out of the N nodes in the pool of workers.

2 Probability of 51% attack

The probability of a successful 51% attack on a peer-to-peer network, P_A, implies that a malicious entity (or group of entities) succeeds in controling more than half the worker nodes selected to produce the ledger state update (GLSU) during a ledger cycle, giving that entity the power to tamper with the GLSU. Such probability depends on the following parameters:

- N : the total number of nodes in the pool of workers, N. We assume in the following that N is a constant number for an infinite number of ledger cycles.
- V : the subset of worker nodes selected to perform work for one ledger cycle ($V \leq N$).
- O : the number of malicious nodes in the pool of workers ($0 \leq O \leq N$). This is a total subset of malicious nodes colluding to perform an attack on the network.
- p : the number of malicious nodes in the subset V of producers. ($0 \leq p \leq V$).

An attack can be considered successful for any value $p \in [p_0, V]$ where $p_0 = V/2 + 1$ which is equivalent to $k > 50\%$. When $V \approx N$, i.e. the number of producers selected to produce the GLSU during a ledger cycle is very close to the total number of nodes in the pool of workers. The absence of a randomness element in the selection of V producers out of N nodes makes it easy to compute the probability of a successful attack on the network: $P_A \approx O/N$. A malicious entity would know exactly when an attack can successfully be performed, that is when $O > N/2$.
There are two main arguments behind having a large number of N nodes: a) to account for the fact that most nodes with sufficient ressources may want to join the pool of workers and receive tokens as reward for their contribution to the ledger state management and b) to make it increasingly costly for any malicious entity to control more than half the nodes.
When $N \gg V$, the probability of such attack can there be expressed by the discrete sum:

$$P_A = \sum_{p=p_0}^{V} P_a(p) \tag{1}$$

where $P_a(p)$ represents the probability of having p malicious nodes in the set V. When the ratio between the total number of nodes N and the number of nodes V is large (R is small), it can be expressed as follows:

$$P(p) = \frac{\overbrace{\begin{pmatrix} O \\ p \end{pmatrix}}^{\text{A}} \overbrace{\begin{pmatrix} N-O \\ V-p \end{pmatrix}}^{\text{B}}}{\underbrace{\begin{pmatrix} N \\ V \end{pmatrix}}_{\text{C}}} \tag{2}$$

A represents the number of possible combinations for choosing p nodes from O malicious nodes. B represents the number of possible combinations for choosing good (non-malicious) nodes for the remaining $N - 0$ nodes in the pool of workers. Finally, C corresponds to the number of available combinations for choosing V nodes from the pool of N nodes.

In equation 2, $P(p)$ is the probability mass function of a hypergeometric distribution over the set of parameters $\{N, O, V\}$. Note that such expression is valid for $max(0, O + V - N) \leq p \leq min(O, V)$. In practise this implies the following equation:

$$max(0, O + V - N) \leq p_0 \leq p \leq V \leq min(O, V) \tag{3}$$

which is true if a) $V \leq O$, i.e. the subset selected for validation work is smaller than the subset of malicious nodes in the validation pool and b) $p_0 > 0$ (necessarily true in this scenario) or $N > O + V/2 - 1$ (equivalent to $O < N - V/2 + 1$). When $N \gg V$ the probability of a successful attack, P_A, can therefore be estimated using the cumulative hypergeometric distribution function (CDF) for $p \in [p_0, V]$. In this paper, we provide probability estimate obtained using *scipy.stats* Python library. The graphs presented are obtained using *matplotlib.pyplot* library. Rather than computing the CDF, the probability measurements are obtained using the survival probability (SDF), which is the inverse of CDF but is known to proide more accurate results[2].

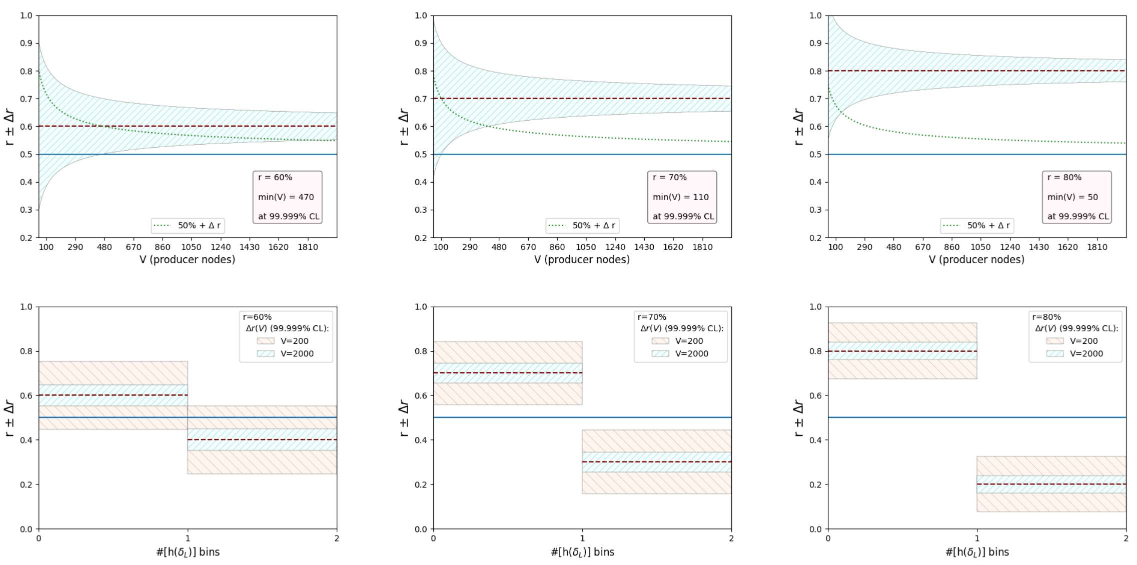
As an example, assume a rather large number of nodes in the pool of workers, $N = 10,000$ and $R = 10\%$ of nodes selected to produce the GLSU for a given cycle ($V = 1,000$). Further assume that $S = 20\%$, *e.g.* 1 every 5 nodes in the pool of workers is controlled by a malicious entity ($O = 2000$). The probability of a successful attack is calculated using the SDF of an hypergeometric distribution using these set of parameters and amounts to: $P_A = 1 - SDF(10000, 2000, 1000) \approx 10^{-9}\%$. For the same set (N, V), the probability of a successful attack reaches 0.03% for $S = 45\%$ of malicious nodes in the pool of workers. To maintain a P_A at 1 over a billion with 45% malicious nodes, V would need to increase to $V = XX$, or XX% of the total number of nodes in the pool of workers.

As discussed in section 1 for a h_1 value to be considered valid by a producer N_i during the first phase of the cycle, it must be that the ratio $r_i = m_1/V_i$ is greater than $r_{min} + \Delta r_i$. Although in theory r_{min} could be set at 50% to prove that a majority of producer nodes have successfully produced the same ledger state update, it is safer to consider a higher threshold that would prevent ambiguity, should a second spike in the histogram distribution arises, for instance in an attempt to tamper with the ledger state by a malicious entity controlling a large number of nodes. Indeed, as illustrated in Figure 4, in an extreme case where only two bins are created in the $h(\delta_L)$ histogram distribution, if the statistical error associated to the two ratio, r_1 and $r_2 = 1 - r_1$ respectively in the frist and second bin, are such that $r_1 - \Delta r_1 < r_2 + \Delta r_2$, one cannot say with certainty that a majority of nodes agrees, even if $r_1 > 50\%$. Note that in this case where the histogram only contains two bins, $\Delta r_1 = \Delta r_2$. This implies that an agreement can only really be

[2]See https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.stats.hypergeom.html for more details.

reached if $r > 50\% + \Delta_r$.

Given that Δr depends on both r and V, a value of $r_{min} = 50\% + \Delta r$ can be derived as a function of V, as illustrated in Figure 4 for various values of $r = \{60\%, 70\%, 80\%\}$ and a confidence level at 99.999%.



(a) $r_{min} = 60\%$ (b) $r_{min} = 70\%$ (c) $r_{min} = 80\%$

Figure 4: Top: $r \pm \Delta r$ as a function of V, the pool of producers. Bottom: $r \pm \Delta r$ at 99.999% confidence level, for two values of V ($\{200, 2000\}$) when two values of $h(\delta_L)$ are collected by a producer.

Figure 4.b shows that for a r_{min} set at 70% the pool of producers must comprise of at least 110 nodes in order to remove any ambiguity with a confidence level at 99.999%. Indeed, if $r_1 = 70\%$ and $V = 2000$, the second bin in the histogram would represent at best 30% of the V hash values collected by the producer, ensuring the latter that a clear majority of producers created the same GLSU since the statistical uncertainty on these two ratio would leave a significant gap between 34.3% and 65.7%. For $V = 1000$, that gap would be reduced to [36.1%,63.9%], still large enough to give enough confidence to a producer that the same GLSU was found by a clear majority of nodes. This is illustrated in Figure 4 a, b and c in the bottom graphs. These graphs demonstrate with V at 200 and 2000 and with a confidence level of 99.999% visualise the validity of our results. It can be seen in 4a that at $V = 200$ that there would be overlap, meaning that the validity of r_1 could be questioned as, with a confidence level of 99.999% we can not be certain that $r_1 > r_2$.

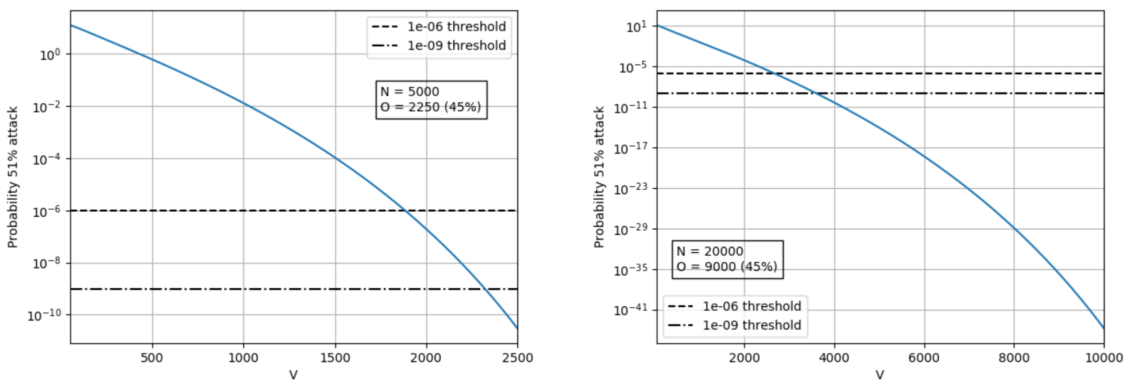
As the network scales up (N increases), it should be expected that requests to join the pool of workers increases too.This will allow the network to increase V while keeping the V/N ratio at a consistent level.

A note: as N increases it become increasingly more difficult to increase the value O sufficiently that the O/N ratio is of any significance. This is due to the cost of running enough nodes to gain a significant O/N ratio to perform a 51% attack.

The value of V should be chosen accordingly to N such that the probability of a successful attack on the network remains lower than an acceptable threshold for a given number of malicious nodes O. In the following we consider two thresholds of probability, 10^{-6} and 10^{-9}, that are equivalent to defining a set of parameters (V, N) such that the probability of performing a 51% attack is of 1 in a million and 1 in a billion, respectively.

For a ledger cycle of 1 min, this would mean that a malicious entity would on average succeed in carrying an attack roughly every 2 years for a probability at 10^{-6}.

Figure 5 demonstrates the effect V has on the likelihood of succesfully carying a 51% attack. An attacker with fewer than 50% of the total worker pool is relying on a disproportionate selection of their nodes in order to control $V/2+1$ or more nodes. Increasing V makes V more representative of N, thereby making it less likely for an assailant with less than 50% of N to be able to mound a successful attack.

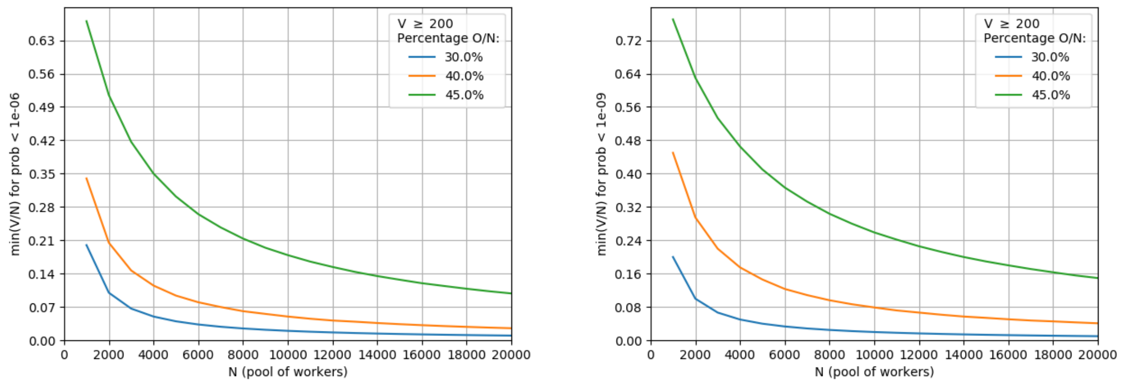


(a) $N = 5000$

(b) $N = 20000$

Figure 5: These graphs show the probability of 51% attack with regards to a variable V value. O/N ratio is 45%

We can then go further in Fig 6. These graphs describe the V/N ratio required to gain a probability of 10^{-6} and 10^{-9} for plot a and b respectively at various O/N ratio. This shows that as N increases the required V/N ratio required for the same security level decreases. This graph allows us to view a safe V/N ratio with regards to preventing the possibility of a 51% attack.

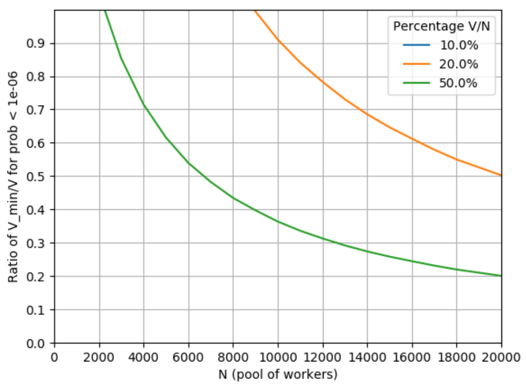


(a) Prob. Attack $= 10^{-6}$

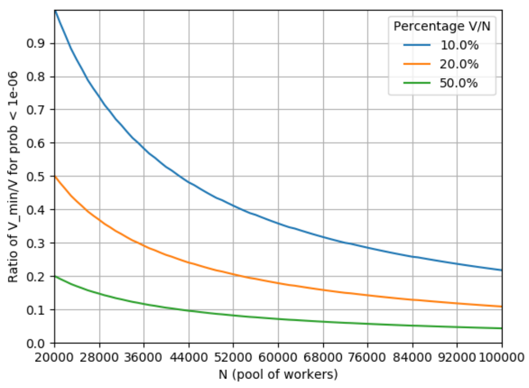
(b) Prob. Attack $= 10^{-9}$

Figure 6: This graph shows the V/N ratio required for a probability of a 51% attack of less than 10^{-6} on a validation cycle versus N. This is shown against various O/N ratio.

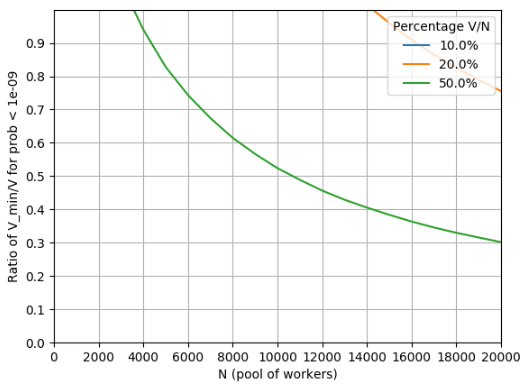
In section 1, we consider the fact that a validator may not collect exactly V ledger delta hash values during the first cycle phase. We argue that a threshold V_{min} should apply to decide whether the ratio $r_i \geq r_{min} + \Delta r_i$ produced by a worker using $V_i \leq V$ hash values should be accepted as valid when missing some data entries (*e.g.* $V_i < V$). Figure 7 demonstrates for various N values what the minimum number of V that we can afford to successfully pass a delta.



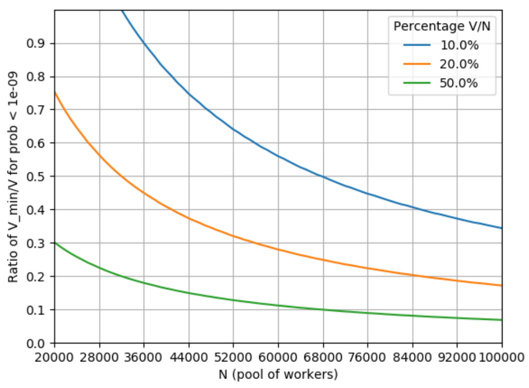
(1a) $N = 0 - 20000$



(1b) $N = 20000 - 100000$



(2a) $N = 0 - 20000$



(2b) $N = 20000 - 100000$

Figure 7: V_{min}/V ratio needed to gain a security level of 10^{-6} for 1a and 1b, and 10^{-9} for 2a and 2b, against 51% attacks, versus N over a range of V/N ratios where O/N is fixed to 45%

Table to summarise for set of value probability of attack

3 Conclusion

We have seen how to produce the ledger tip that is used to produce the global delta from the transactions processed by V. Create a histogram from the hash of the deltas received by the other validation nodes to determine the most popular change i.e. the tip. We have shown how a confidence interval/level can be derived on the choice of tip selected by the validators. This is based on the ratio of responses received. The confidence interval can be used to inform whether it can be trusted or not. We have also demonstrated how the constituents of the validation pool affect the security of the network from 51% and how N, V and O affect the resistance to 51% attack.

Through each validator checking the confidence of the tip produced, as well as analysing the risk of 51% attack we can select parameters that most attenuate the risk of malicious attack. While the scenario of a malitious actor holding 45% or more nodes on our network is highly unlikely it must be considered.