1 Model

We want to estimate the probability that the single value from the input range of the fuzzle was checked during fuzzle solving. We model our system with the following assumptions:

1. Communication between nodes is instant
2. Only n nodes try to find a solution for the fuzzle
3. All nodes have the same computational resources and finish execution for each input value simultaneously
4. When at least one node finds a solution, all nodes stop execution
5. Nodes don't share any information except information about the solution
6. Every node checks input values in the random order

2 Estimation

2.1 Expected value of steps to find a solution

Because of the assumption #3, we can model system's execution in rounds. In each round, every node checks one random value from the input range without repetition. Different nodes can check the same value twice since they don't share any information except solutions. The probability P_k of finding the solution on the kth round is the probability $M_{1\ldots k-1}$ of not finding the solution on all rounds before kth multiplied by the probability R_k of finding the solution during the kth round:

$$P_k = M_{1\ldots k-1} \cdot R_k. \tag{1}$$

Since all rounds are independent, the probability $M_{1\ldots k-1}$ of not finding the solution on all rounds before k is the product of probabilities m_k of not finding the solution on the round $1, 2, \ldots, k-1$:

$$M_{1\ldots k-1} = \prod_{i=1}^{i=k-1} m_i. \tag{2}$$

Let x be the size of the input range of the fuzzle. Since all nodes are independent, the probability m_i of not finding the solution on the round i is:

$$m_i = \left(\frac{x-i}{x-i+1}\right)^n. \tag{3}$$

Therefore, the probability R_k of finding the solution during the kth round is:

$$R_k = 1 - m_k = 1 - \left(\frac{x-i}{x-i+1}\right)^n. \tag{4}$$

Substitute 2, 3 and 4 into 1:

$$P_k = \left(1 - \left(\frac{x-k}{x-k+1}\right)^n\right) \prod_{i=1}^{i=k-1} \left(\frac{x-i}{x-i+1}\right)^n =$$
$$= \left(1 - \left(\frac{x-k}{x-k+1}\right)^n\right) \left(\frac{x-1}{x}\right)^n \left(\frac{x-2}{x-1}\right)^n \cdot \ldots \cdot \left(\frac{x-k+1}{x-k+2}\right)^n =$$
$$= \left(1 - \left(\frac{x-k}{x-k+1}\right)^n\right) \left(\frac{x-k+1}{x}\right)^n =$$
$$= \frac{(x-k+1)^n - (x-k)^n}{x^n}. \quad (5)$$

Therefore, using 5, the expected value of number of rounds K to find the solution can be calculated as follows:

$$\mathbb{E}[K] = \sum_{k=1}^{x} kP_k = \sum_{k=1}^{x} k\frac{(x-k+1)^n - (x-k)^n}{x^n} =$$
$$= \frac{1}{x^n} \cdot \sum_{k=1}^{x} k\left((x-k+1)^n - (x-k)^n\right) =$$
$$= \frac{1}{x^n} \cdot \left(1 \cdot \left(x^n - (x-1)^n\right) + 2 \cdot \left((x-1)^n - (x-2)^n\right) + \ldots + x\left(1^n - 0^n\right)\right) =$$
$$= \frac{1}{x^n}\left(x^n + (x-1)^n + \ldots + 1^n\right) =$$
$$= \sum_{k=0}^{x} \left(\frac{x-k}{x}\right)^n \quad (6)$$

2.2 Probability p_0 of the single value from the input range to be checked

Note that the expected value of number of rounds K equals expected number of checked values from the input range since the node, which reports the solution on the kth round, provide information on K values from the input range (remember, the node checks one value on each round). Therefore, to get the expected proportion $\mathbb{E}[l]$ of the input range that will be checked, we need to divide $\mathbb{E}[K]$ by the size of the fuzzle input range:

$$\mathbb{E}[l] = \frac{1}{x}\mathbb{E}[K] = \frac{1}{x}\sum_{k=0}^{x} \left(\frac{x-k}{x}\right)^n \quad (7)$$

Since series 6 can't be represented in elementary functions, we will investigate behaviour of 7 numerically. Figure 1 depicts the relationship between $\mathbb{E}[l]$ and the size of the fuzzle input range x for different n. We can see that the expected proportion converges to the asymptotic value with $x > 200$. As in our project input range size $x \gg 200$, we can assume that $\mathbb{E}[l]$ equals to the asymptotic value.

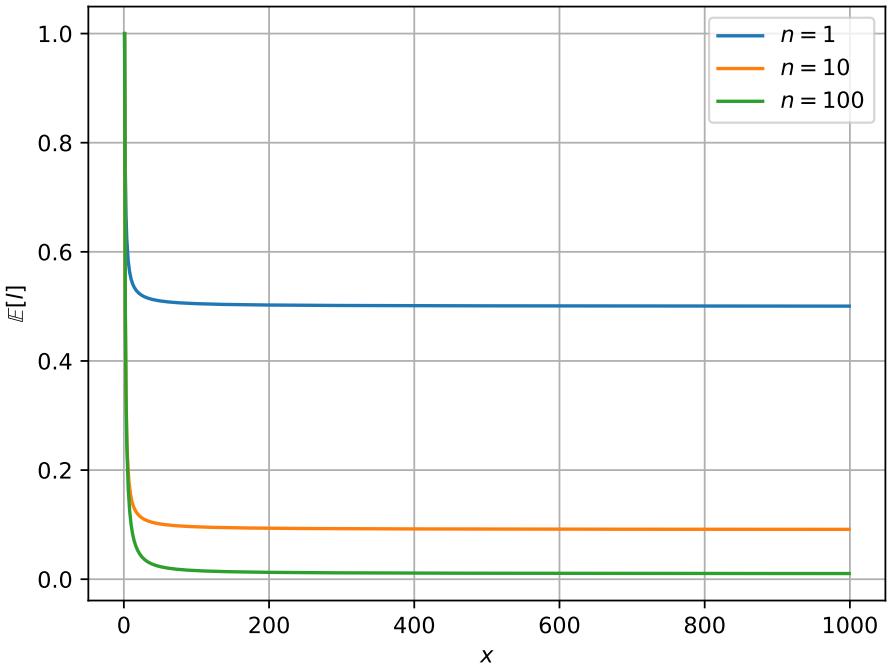

Figure 1: Behaviour of the expected proportion of the fuzzle input range that is checked during fuzzing with different sizes of the input range x and different number of nodes n working on the fuzzle.

Note that the expected proportion $\mathbb{E}[l]$ of the input range that will be checked equals to the probability p_0 of the single value from the input range to be checked because of the assumption # 6 that every node checks input values in the random order.

Figure 2 depicts the relationship between the probability p_0 of the single value from the fuzzle input range to be checked and number of nodes n working on the fuzzle. From the graph, we see that the probability decreases dramatically and is less than 0.1 for the $n > 10$.

2.3 Relationship between p_0 and x

Figure 2 shows that, with large n, the probability p_0 of the value from the input range decreases a lot, therefore, the effectiveness of fuzzing decreases. In this chapter, we provide analysis on how the proposer can achieve higher fuzzing efficiency and have probability estimation for the target input range of the proposed fuzzle. Therefore, we introduce **probabilistic fuzzing**.

We will use the following definitions:

- N — number of executors in the system.
- T — time between fuzzle blocks in the blockchain.
- t_0 — time to run the program under test with a single input value.
- t — time to solve a fuzzle.

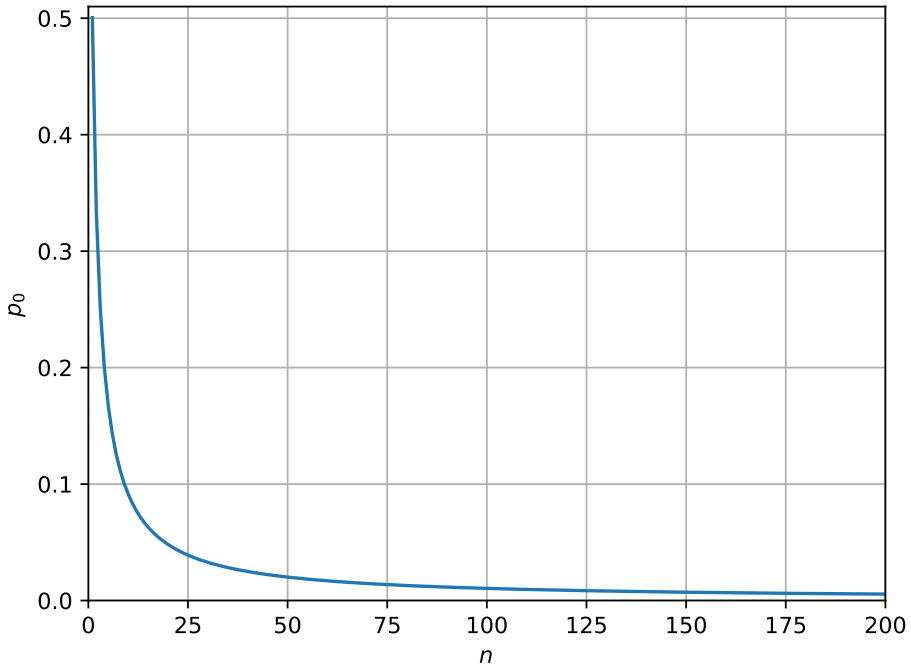

Figure 2: Relationship between probability p_0 of the value from the input range to be checked and number of nodes n working on the fuzzle.

The number of executors in the system N can be estimated as an average number for the time period since executors can join or leave the system at any time. The time T between fuzzle blocks in the blockchain can be estimated as doubled time between new blocks in the system (this is the parameter of the difficulty module). The time to run the program under test with a single input value can be estimated as an average time to execute the program with multiple random values from the input range.

As stated above, we want to find the relationship between p_0 and x, so that the proposer can adjust x, so that it achieves the target effectiveness of fuzzing. We fix the target effectiveness p_0 and derive the x.

The time to solve the target fuzzle can be calculated as follows:

$$t = p_0 \cdot x \cdot t_0. \tag{8}$$

We can also estimate t from the perspective of the number of executors. If the time to solve a fuzzle is less than the time between fuzzle blocks $t < T$, we can estimate that $n \sim N$. If $t > T$ then we can estimate that $n \sim \frac{T}{t}N$. This estimation follows from the assumption that if fuzzle takes more time to solve than the time before the next fuzzle, some executors take other fuzzle as they take it randomly. Therefore, t can be also estimated as follows:

$$t = \frac{TN}{n}. \tag{9}$$

Combining 8 and 9, we can calculate x:

$$x = \frac{TN}{n \cdot t_0 \cdot p_0}. \tag{10}$$

If we fix p_0, we can calculate n using numerical solution presented on Figure 2. Practically, we can interpolate $p_0(n) = f(n)$ using cubic spline and then find the root n of the equation $f(n) = p_0$ using Newton's method (there is at most one root from the properties of the function). Define this operation as $f^{-1}(p_0)$ Therefore, 10 can be rewritten:

$$x = \frac{TN}{f^{-1}(p_0) \cdot t_0 \cdot p_0}. \tag{11}$$

And 9 can be also rewritten:

$$t = \frac{TN}{f^{-1}(p_0)}. \tag{12}$$

2.4 Overlapping factor

The proposer could also increase the effectiveness of fuzzing by using overlapping input ranges. We define overlapping factor f as the inverse proportion of overlapping between consecutive fuzzle input ranges assuming it is constant for all overlapping ranges. Figure 3 shows an example of the overlapping factor.

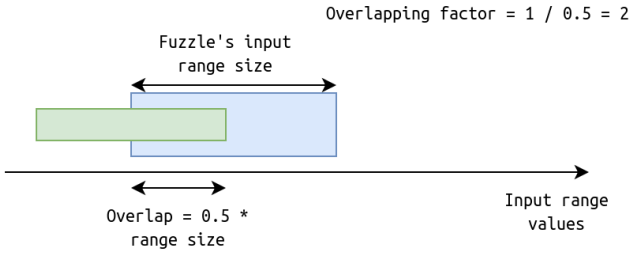

Figure 3: Example of the overlapping factor for two consecutive fuzzle input ranges.

As the probability of checking a single value from the input range is the same for all values from the range, the probability p of checking a single value from the input range with the overlapping factor f can be calculated as follows:

$$p = 1 - (1 - p_0)^f \Rightarrow p_0 = 1 - (1 - p)^{\frac{1}{f}}. \tag{13}$$

3 Conclusion

In the chapters above we presented a model to estimate effectiveness of fuzzing, adjust the fuzzle size to achieve the target effectiveness and change it using overlapping input ranges.

The main practical result is that the proposer can fix the target probability p of the value from the input range to be tested. Then the proposer can estimate the pair of parameters to achieve this probability: overlapping factor f and the size x of the fuzzle input range. The trade-off between increasing f or x is that the proposer increases costs of fuzzing by increasing f, but the proposer can't make x arbitrary large as there is a limit in the system for the solving of fuzzles.