Location Verification using Latencies and Claimed Coordinates on the Blockchain

Pantea Karimi Babaahmadi

School of Computer and Communication Sciences

Decentralized and Distributed Systems lab

Internship Project

August 2019

Responsible
Prof. Bryan Ford
EPFL / DEDIS

Supervisor
Cristina Basescu
EPFL / DEDIS

Contents

1 Introduction

In many applications such as processing of transactions, speed is an important factor. In self-organizing communities without relying on a central party, the processing of transactions can be performed by a set of validators. To reach higher speed in transaction validation, a Trust-but-Verify approach can be taken. For example in paying for the daily purchase in the local supermarket with cryptocurrency, a user can rely on the local consensus and enjoy the fast processing of the transaction, but can still verify the global state to make sure at the end that the transaction was also verified in the global consensus and was not a double-spend. However, the user has a choice not to trust the local consensus and wait longer for the global consensus. In the Trus-but-Verify approach, nearby validators can start processing the transactions and provide a fast, weak, and temporary proof, while the global verification is being computed by all the validators in the system [2]. To find all the nearby validators for local consensus, each validator should know its latencies to all the other validators in the system. The primary goal of this project is to devise a scalable fault-tolerant algorithm to estimate the pairwise latencies among all the nodes of the system. The challenge is that some nodes are malicious and behave arbitrarily, including attempts to mislead others about their latencies.

2 Background

When building a system based on the latencies of the nodes of the system, one solution is to know all the pairwise latencies to make sure they all make sense together and the adversary nodes can not report malicious latencies contradicting with some of the other latencies [1]. However, this approach has some deficiencies. First, measuring all the pairwise latencies makes the system unscalable. Second, even with having all the latencies, it is difficult to detect who is a malicious node [1]. Third, Sybil Location attack is possible in this scenario [1], which enables a validator to contribute to the local consensus in multiple distinct regions and might attempt double-spend in different regions. Although the double-spend transaction will not be verified in the global consensus, the adversary might be able to enjoy local services if the users trust local consensus before the global consensus is reached. It can be shown that for small latencies, geographical distance and latency are correlated [3]. Thus, the possible solution to solve the Sybil Location attack is to demand the nodes to commit to their geographical location on a secure shared ledger, Blockchain. By doing this, all the nodes are committed to their claimed regions and can not participate in the local consensus of two regions that are far away. However, malicious nodes can claim wrong or arbitrary locations. To evaluate the correctness of their

claims, location verification methods by measuring round-trip-times can be implemented [3].

3 Models and Methods

As mentioned in section 2 , malicious nodes can show fallacious actions when reporting and measuring the round-trip-times. We assume that malicious nodes can collaborate in reporting the round-trip-times and they know each other. However, honest nodes neither know malicious nodes nor other honest nodes.

3.1 Attacks

In this section, we explore the possible attacks that could happen when we rely on only measuring the round-trip-times (RTTs).

3.1.1 Smaller RTT with the Help of Malicious Nodes

A malicious node can fake a closer round-trip-time and ask other malicious nodes to answer for it when measuring the RTT.

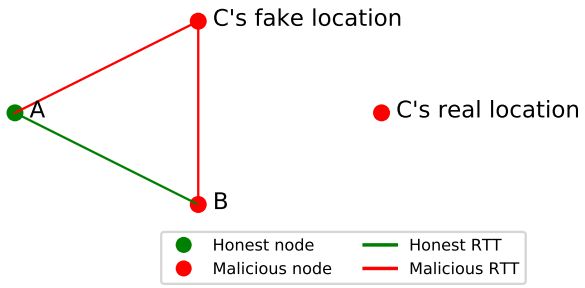

Fig. 1. Claiming smaller RTT with the help of other malicious nodes

Figure 1 shows how this attack is possible. Suppose that node B and C are malicious nodes. Node C, with the help of node B, can convince all the other nodes in the system about its claimed RTT to A. It is possible for node C to convince any honest node in the system like A, that C has an RTT of l ms, while the actual RTT is larger than l, as long as there is at least one other malicious node like B closer to A than l. Our algorithm tries to detect fake claims of malicious node C that there is no other malicious node like B to answer instead of C.

3.1.2 Sybil Location

In Figure 1 , we showed how a malicious node can pretend to be closer to other nodes with the help of another malicious node. This attack makes another attack possible which is called Sybil Location.

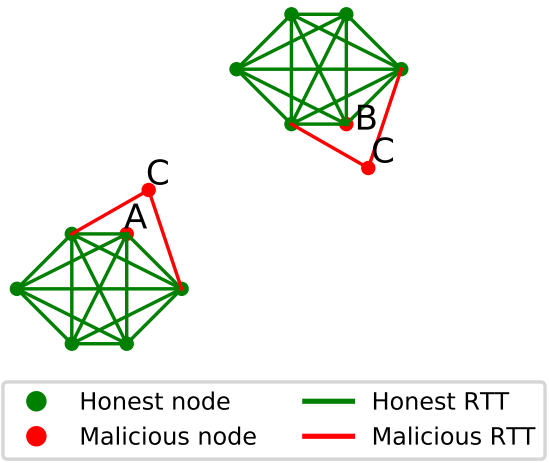

Fig. 2. Sybil Location

Figure 2 shows two distinct regions in which local nearby validators validate transactions faster to provide a weak regional verification. According to Figure 2 node C with the help of two other malicious nodes, A and B, can convince both regions that C is a part of them. In global view, C can not be at two places at the same time and will contribute to the global consensus with one vote per transaction; however, in local view, C can convince both regions that it is a part of them. This attack allows C to contribute to multiple regions and tamper with consensus protocol in multiple regions. Note that C might be successful in a double-spend attack in two distinct regions in this scenario; however, C's double-spend transactions are not verified in the global consensus. We aim to prevent all the nodes to be able to have contributions to multiple distinct regions.

3.1.3 Larger RTT by Delaying Messages

A malicious node can consiously delay RTT measurments to make itself look farther away.

Figure 3 shows malicious node C that convinces both A and B of a farther location. By doing this, node C loses its consensus power in the region containing A and B.

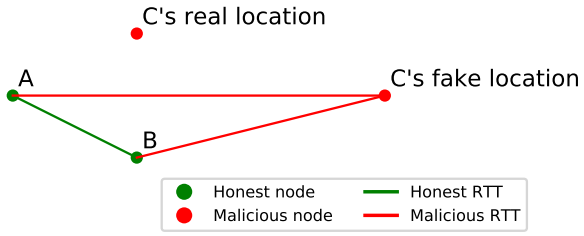

Fig. 3. Larger RTT by delaying messages

3.1.4 Eclipse Attack

A group of malicious nodes can consciously report false RTT measurements for a specific honest node and make the honest node's claims look unreliable in that malicious neighborhood.

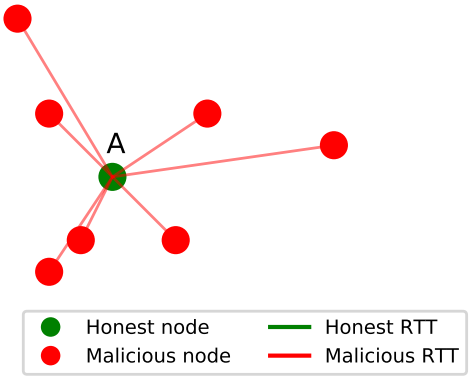

Fig. 4. Eclipse Attack

Figure 4 shows honest node A that is surrounded by malicious nodes. A's location will remain fuzzy in this malicious region because all the close nodes around A are malicious. In our model, we do not help A survive in its malicious region. It means that if A's malicious neighborhood decides that A is not within their region, A can not trust the local consensus; however, A can contribute to and trust the bigger overlapping regional or the global consensus.

3.2 Secure Measurement of Round-trip-times [1]

For the secure measurement of round-trip-times, we use the method in [1]. In this section, we briefly summarize how node A measures its latency to node B.

1. A sends B a message with its public key as an identifier, a nonce signed by A, and a timestamp A_1 signed by A [1].

2. B checks the received message from A to see if the public key matches the private key, the timestamp is not old (not a response to an old epoch), and A has not already started a new latency. If these checks are passed, B sends back to A, A's nonce signed by B's private key, a new nonce signed by B, and B's time of reception signed by B. B computes a clock skew based on its reception time and A's timestamp [1].

3. A checks that the public key matches the private key and the first nonce is the same nonce as the nonce in message 1. If these checks are passed, A computes the round-trip-time to B, using the time of reception of message 2 and the timestamp of message 1, and sends back to B the computed latency signed by A's private key and B's nonce signed by A's private key [1].

4. B performs the usual checks and checks if the clock skew has remained similar. B computes its own latency and checks to see if it is similar to the received latency from A. If these checks are passed, B sends back to A, A's signed latency with the latest timestamp all signed by B and B's signed computed latency [1].

5. A performs the usual checks and checks if the clock skew and latency have remained the same as before. If these checks are passed, A sends back to B, B's signed latency with the latest timestamp all signed by A [1].

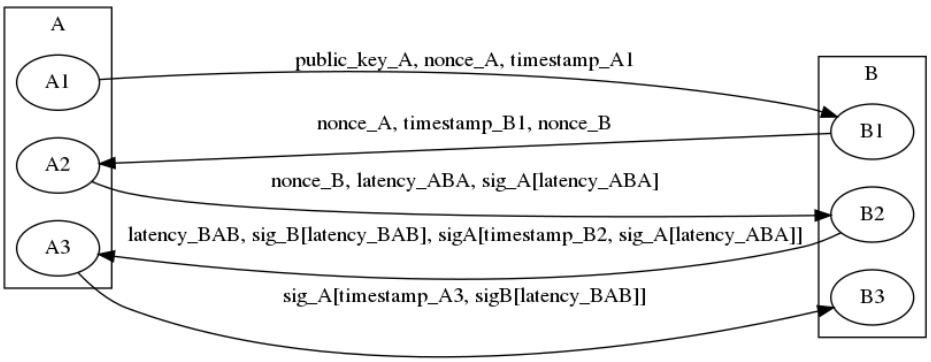

Fig. 5. Latency creation messaging protocol (all values are signed) [1]

4 Location Verification

4.1 Motivation

To solve the problem of Sybil Location, each node must commit to an initial coordinate on the Blockchain. By doing this, each node can only be in one location, thus we can prevent malicious multi-regional contributions by limiting each node to one region. Location verification is necessary for two reasons. First, honest nodes do not know other honest nodes. So, there must be verification on the claimed coordinates for honest nodes to have some certainty on the claimed coordinates of others in their regions. Second, the malicious nodes can claim faulty location and we want to prevent them from claiming random coordinates in regions where there actually is no malicious node.

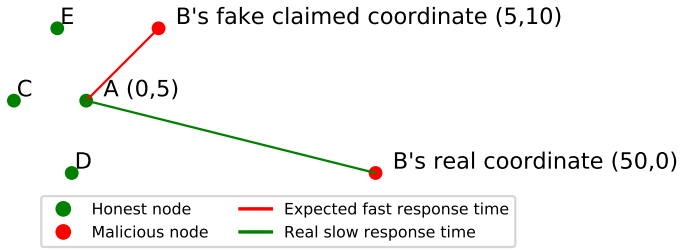

Fig. 6. Performance consideration: B slows down the consensus of the honest region.

In Figure 6 , suppose that malicious node B claims to be closer to the region including honest nodes A, C,D, and E. If no verification of the claim of B's location happens, the regional consensus of honest nodes is going to take much longer than it should, resulting in a slow consensus. Therefore, the verification of the claimed coordinates is necessary also for the performance considerations.

Based on the graph by the Center for Applied Internet Data Analysis (Caida), the round-trip-time is highly correlated to the geographical distance up to a distance [3]. In subsection 4.3 , we show how we can use this graph and more information on the relationship of honest distance and RTT [4] to help us build our model.

Figure 7 shows the quartiles of round-trip times (RTTs) versus geographical distance from the probe source, which is calculated by looking up every hop's latitude and longitude via NetAcuity and comparing with the location of the source [3].

To evaluate and verify the claimed locations, we use this correlation. For small latencies, the physical distance is correlated with the latency, thus if two nodes are close with respect to their latencies, their physical distance can

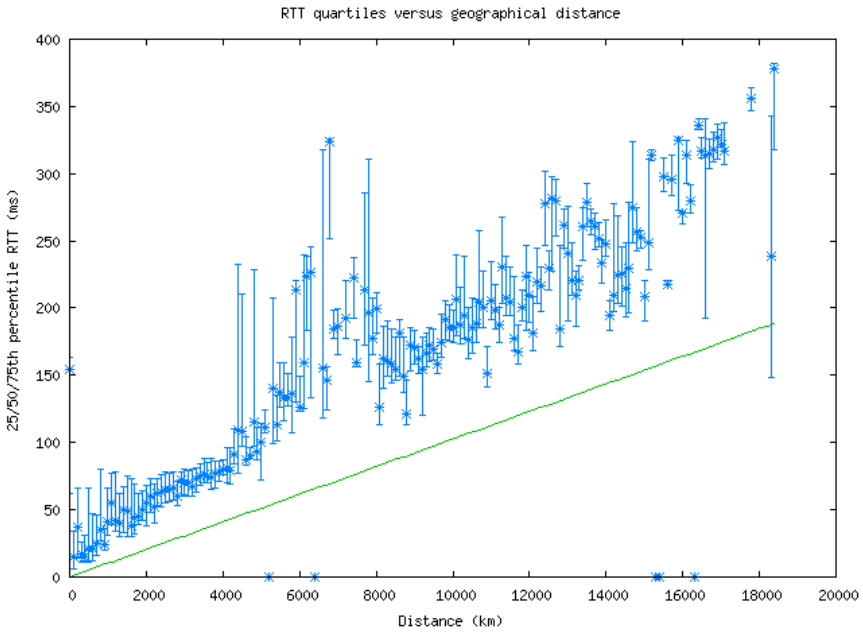

Fig. 7. RTT quartiles versus geographical distance [3]

approximate their latency and vice versa. This fact is not true for farther physical distances, but since our goal is to find close nodes with smaller latencies, we can use this correlation for small latencies. The main reason that we wanted to find the latencies was to find the nearby validators. Thus, the claiming of the locations will find the nearby validators with reasonable accuracy.

4.2 Approach

Suppose that all the nodes have submitted their claimed coordinates on the Blockchain, including node A. The system wants to verify if A's claimed coordinate is plausible in each region or not. We begin by some definitions. N is the set of all nodes and n is the number of nodes in the system. We assume that the maximum fraction of the number of malicious nodes in the whole system to n is less than $\frac{1}{3}$ (this is a given assumption). Note that $\forall A \in N : A$ has a claimed coordinate (x_A, y_A) . The circle with center node A and radius r_i is defined as $C(A,i) = \{(x,y) \mid (x-x_A)^2 + (y-y_A)^2 < {r_i}^2\}$ and we call it the neighborhood R_i of A. i is a positive integer from 1 to i_{max}, and i_{max} is the smallest integer that $C(A,i)$ includes all nodes in the system. We choose $r_i := 2^i$ (this is a given assumption); however, it is a system parameter and can be changed based on the system's preferences. The set of nodes in the neighborhood R_i of A is shown with N_i and the number of them is denoted by n_i. We call R_i a malicious neighborhood, if

there are more than $n_i\beta$ malicious nodes in R_i, meaning that there are more than $n_i\beta$ malicious nodes whose claimed coordinates are in the $C(A, i)$. We call the R_i an honest neighborhood, if there are less than $n_i\beta$ malicious nodes in R_i, meaning that there are less than $n_i\beta$ malicious nodes whose claimed coordinates are in the $C(A, i)$. We denote the number of honest nodes in N_i by h_i and the number of malicious nodes in N_i by $n_i - h_i$, in which $\frac{n_i - h_i}{n_i} < \beta$. β is a security parameter depending on what proportion of the malicious nodes the local consensus can tolerate and can be chosen as a system parameter.

Since the local consensus occurs in the local regions, it is important for us to know whether A's claimed coordinate is plausible in that region or not. Meaning that if A is pinged for the consensus, A will respond within a reasonable latency. For this reason, we try to answer the following question: Is A's claim to be in the neighborhood R_i of A plausible? Two scenarios can happen as the following.

First, R_i could be a malicious neighborhood. In this scenario, whether A is a malicious node or honest node, the consensus occurs among the malicious nodes in the neighborhood and they can decide on whatever they want. If A is a malicious node, the malicious neighborhood already knows A and A does not need to prove to the neighborhood its claimed coordinate. If A is an honest node, as in the subsubsection 3.1.4 stated, we do not try to help A to justify its coordinate in a malicious neighborhood. The malicious neighborhood already knows that A is an honest node and they can ignore A's vote in the neighborhood. The result would be for A that it will not trust the outcome of the local consensus in the malicious neighborhood. However, A can still trust the global consensus which happens later on.

Second, R_i could be an honest neighborhood. We want for an honest node to be able to prove its claimed coordinate in an honest neighborhood with a high probability and for a malicious node not to be able to prove its claimed coordinate in an honest neighborhood if there is no malicious node in that local region that based on subsubsection 3.1.1 can justify A's malicious coordinate. The latter prevents malicious nodes from claiming random coordinates.

To evaluate A's claim of being in the neighborhood R_i of A, or $C(A, i)$, A is assigned to k_i randomly chosen nodes from N_i, nodes in the R_i, by a public randomness on the Blockchain such as RandHound [5]. A is assigned to these k_i random nodes, which we call verifiers and are denoted by ${v_i}^1$, ${v_i}^2$, ..., ${v_i}^{k_i}$, to measure the secure round-trip-times as sescribed in subsection 3.2 and put the results on the Blockchain. Among the k_i verifiers assigned to A, suppose h_{k_i} nodes are honest, ${v_i}^1, {v_i}^2, ..., {v_i}^{h_{k_i}}$, and $k_i - h_{k_i}$ nodes are malicious. The claimed geographical distances of these k_i verifiers to A, based on the claimed coordinated on the Blockchain, is denoted by ${d_i}^1$, ${d_i}^2$, ..., ${d_i}^{k_i}$ respectively. We denote the outputs of the round-trip-times measurments by verifiers by ${l_i}^1$, ${l_i}^2$, ..., ${l_i}^{k_i}$ respectively. For the RTTs that

are not submitted, infinity is considered as the output of the measurement. We define the function $g(d, l)$ as the following:

$$g(d, l) = 1[l < l_d] = \begin{cases} 1, & l < l_d \\ 0, & o.w. \end{cases}$$

In section subsection 4.3 , we will show how to find l_d to derive this finction. In subsection 4.3 we will show that the $g(d, l)$ function's output is reliable with the probability of $1 - \sigma_d$. We define $score_i$ for A in $C(A, i)$ as the following:

$$score_i = \frac{\sum_{j=1}^{k_i} g({d_i}^j, {l_i}^j)}{k_i} \tag{1}$$

For an honest node in an honest neighborhood $C(A, i)$, if k_i is reasonably large, we want A's $score_i$ to be greater than a threshold c with a high probability. This verification is necessary for honest nodes because honest nodes do not know each other and they do not know how much A's claimed coordinate in that neighborhood is reliable.

$$Pr(score_i > c \mid A \; is \; honest) > \delta \tag{2}$$

$$Pr(score_i > c \mid A \; is \; honest) = Pr(\frac{\sum_{j=1}^{h_{k_i}} g({d_i}^j, {l_i}^j)}{k_i} > c \mid A \; is \; honest) \tag{3}$$

The purpose of this algorithm is to find out if node A is in the $C(A, i)$ or not. To simplify this approach, it not necessary to find all of the ${l_{d_i}}^j$s in all of $g({d_i}^j, {l_i}^j) = 1[{l_i}^j < {l_{d_i}}^j]$ for different js. The fact is that for A to be in $C(A, i)$, it is sufficiant for A to show that its distance is less than r_i to all the its verifiers in R_i. Thus, we substitute ${l_{d_i}}^j$ by l_{r_i}. If the system decides that it is essential to have more fine-grained verifications in small neighborhoods, it can change the variable r_i. We imagine the worst case scenario for the honest node A, in which all the chosen malicious verifiers assigned to A report $g(d, l) = 0$. So we simplify the approach as the following:

$$score_i = \frac{\sum_{j=1}^{k_i} g(r_i, {l_i}^j)}{k_i} = \frac{\sum_{j=1}^{k_i} 1[{l_i}^j < l_{r_i}]}{k_i} \tag{4}$$

For the neighbourhood of R_i, this reliability value is shown to be $1 - \sigma_{r_i}$ in subsection 4.3. By replacing Equation 4 in Equation 3 we get:

$$\begin{aligned} Pr(score_i > c \mid A \; is \; honest) &= Pr(\frac{\sum_{j=1}^{k_i} 1[{l_i}^j < l_{r_i}]}{k_i} > c \mid A \; is \; honest) \\ &= Pr(\frac{(1 - \sigma_{r_i}) h_{k_i}}{k_i} > c \mid A \; is \; honest) \\ &= Pr(\frac{h_{k_i}}{k_i} > \frac{c}{(1 - \sigma_{r_i})} \mid A \; is \; honest) \end{aligned} \tag{5}$$

We denote $\frac{c}{(1-\sigma_i)} = \gamma_1$ and we have:

$$\begin{aligned} Pr(score_i > c \mid A \textit{ is honest}) =& Pr(\frac{h_{k_i}}{k_i} > \gamma_1 \mid A \textit{ is honest}) \\ &= \sum_{x>k_i\gamma_1}^{k_i} \frac{\binom{h_i-1}{x}\binom{n_i-h_i}{k_i-x}}{\binom{n_i-1}{k_i}} \\ &= \sum_{x>k_i\gamma_1}^{k_i} \frac{\binom{(1-\beta)n_i-1}{x}\binom{n_i\beta}{k_i-x}}{\binom{n_i-1}{k_i}} \end{aligned} \tag{6}$$

So we should choose $k_i \leq n_i$ such that:

$$\sum_{x>k_i\gamma_1}^{k_i} \frac{\binom{(1-\beta)n_i-1}{x}\binom{n_i\beta}{k_i-x}}{\binom{n_i-1}{k_i}} > \delta \tag{7}$$

As we discussed in subsubsection 3.1.1 malicious nodes can respond for RTTs instead of each other. Thus, we try to catch the malicious nodes in an honest neighborhood that no one responds for them. This verification helps the system prevent the malicious node A from claiming random coordinates, where there actually is no other malicious node to answer instead of A. We have no assumption on the actual distribution of the malicious nodes or their claimed coordinates. For a malicious node in an honest neighborhood $C(A, i)$, where there is no malicious node to respond for A's RTTs, if k_i is reasonably large, we want A's $score_i$ to be less than c with a high probability.

$$Pr(score_i < c \mid A \textit{ is malicious } \& \textit{ No malicious node for } A \textit{ to respond}) > \delta \tag{8}$$

We imagine the best case scenario for the malicious node A, in which all the chosen malicious verifiers assigned to A report $g(d, l) = 1$. Also, A can convince each honest verifier with the probability of σ_{r_i}, since $g(d, l)$ function has a reliability threshold. For Equation 8 we have:

$$\begin{aligned} & Pr(score_i < c \mid A \textit{ is malicious } \& \textit{ No malicious node for } A \textit{ to respond}) \\ =& Pr(\frac{\sum_{j=1}^{k_i} g({d_i}^j, {l_i}^j)}{k_i} < c \mid A \textit{ is malicious}) \\ =& Pr(\frac{k_i - h_{k_i} + h_{k_i}\sigma_{r_i}}{k_i} < c \mid A \textit{ is malicious}) \\ =& Pr(\frac{h_{k_i}}{k_i} > \frac{1-c}{1-\sigma_{r_i}} \mid A \textit{ is malicious}) \end{aligned} \tag{9}$$

We denote $\frac{1-c}{1-\sigma_{r_i}}$ by γ_2. We have:

$$
\begin{aligned}
&Pr(score_i < c \mid A \text{ is malicious } \& \text{ No malicious node for } A \text{ to respond}) \\
=&Pr(\frac{h_{k_i}}{k_i} > \gamma_2 \mid A \text{ is malicious}) \\
=&\sum_{x>k_i\gamma_2}^{k_i} \frac{\binom{h_i}{x}\binom{n_i-h_i-1}{k_i-x}}{\binom{n_i-1}{k_i}} \\
=&\sum_{x>k_i\gamma_2}^{k_i} \frac{\binom{(1-\beta)n_i}{x}\binom{n_i\beta-1}{k_i-x}}{\binom{n_i-1}{k_i}}
\end{aligned} \tag{10}
$$

So we should choose k_i such that:

$$
\sum_{x>k_i\gamma_2}^{k_i} \frac{\binom{(1-\beta)n_i}{x}\binom{n_i\beta-1}{k_i-x}}{\binom{n_i-1}{k_i}} > \delta \tag{11}
$$

Combining Equation 11 and Equation 7, we find the condition that k_i should meet as the following:

$$
min\{\sum_{x>k_i\gamma_2}^{k_i} \frac{\binom{(1-\beta)n_i}{x}\binom{n_i\beta-1}{k_i-x}}{\binom{n_i-1}{k_i}}, \sum_{x>k_i\gamma}^{k_i} \frac{\binom{(1-\beta)n_i-1}{x}\binom{n_i\beta}{k_i-x}}{\binom{n_i-1}{k_i}}\} > \delta \tag{12}
$$

If there is no $k_i \leq n_i$ such that it satisfies both Equation 11 and Equation 7, we choose $k_i = n_i$.

The following graph shows the minimum needed number of verifiers for A's coordinate for the β values of $\frac{1}{3}$ and $\frac{1}{2}$ for different numbers of nodes in the neighborhood, n_i, based on Equation 12.

As you can see in Figure 8, if the local consensus is run by at most $\frac{1}{3}$ malicious nodes, the number of verifiers grows logarithmically as n_i grows. This approach has less complexity than measuring all the round-trip-times in order to find the nearby validators. Notice that the smaller r_i is, the more important it is for us to have a more reliable verification approach. Figure 7 shows that for lower r_i, honest latencies and distances are more correlated, thus the reliability value of the $g(l, r_i)$, $1 - \sigma_{r_i}$, is higher.

From Equation 9 and Equation 3 we notice that $\frac{h_{k_i}}{k_i} > max\{\gamma_1, \gamma_2\} = \zeta$. So the approximate answer, for a reasonably large n_i, would be to choose k_i such that:

$$
\sum_{x>k_i\zeta}^{k_i} \frac{\binom{(1-\beta)n_i}{x}\binom{n_i\beta}{k_i-x}}{\binom{n_i}{k_i}} > \delta \tag{13}
$$

For reasonably large n_i, we can further approximate Equation 13 by:

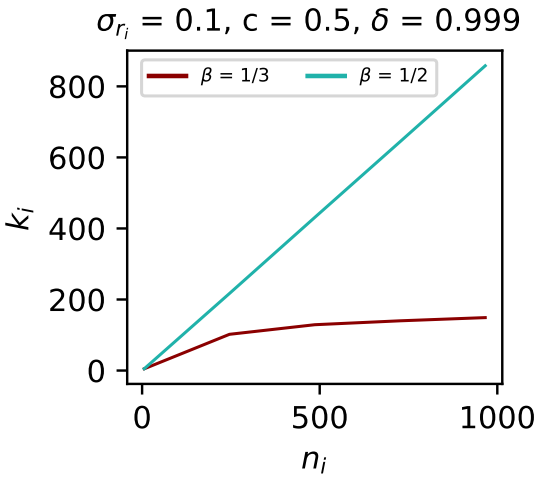

Fig. 8. Minimum number of verifiers, k_i, for different n_is

$$\sum_{x>k_i\zeta}^{k_i} \binom{k_i}{x} (1-\beta)^x (\beta)^{k_i-x} > \delta \tag{14}$$

Note that all the measurements will be put on the Blockchain. Suppose in the verification of A's claim, node A is assigned to verifier B to measure the RTT. If in the verification of B's claim B is randomly assigned to A, there is no need to measure the RTT again and it can be read from the Blockchain.

4.3 Finding $g(d, l)$ function

The joint distribution of two random variables of L, round-trip-time, and D, distance, can be derived from the Figure 7 and we denote it by $f_{D,L}$. Moreover, based on [4], the CDF of minimum end-to-end RTT to TVHosts for different ranges of linearized distances and geographic distances of paths is as Figure 9. As in Figure 9, at low values of the linearized distance, there exists a strong correlation between the delay and linearized distance for a large fraction of end-hosts especially for small values of linearized distances [4]. We expect this correlation to be much stronger as we compute the minimum over a larger number of samples [4].

Suppose that two nodes have claimed they have the distance d and they report a round-trip-time of l. For the claimed distance of d, there is a lower bound on l. However, there is no upper bound on l. This means that if we put a limit such as l_d and we expect that $l < l_d$, there is a probability that two honest nodes can not satisfy this condition, but in that case, the

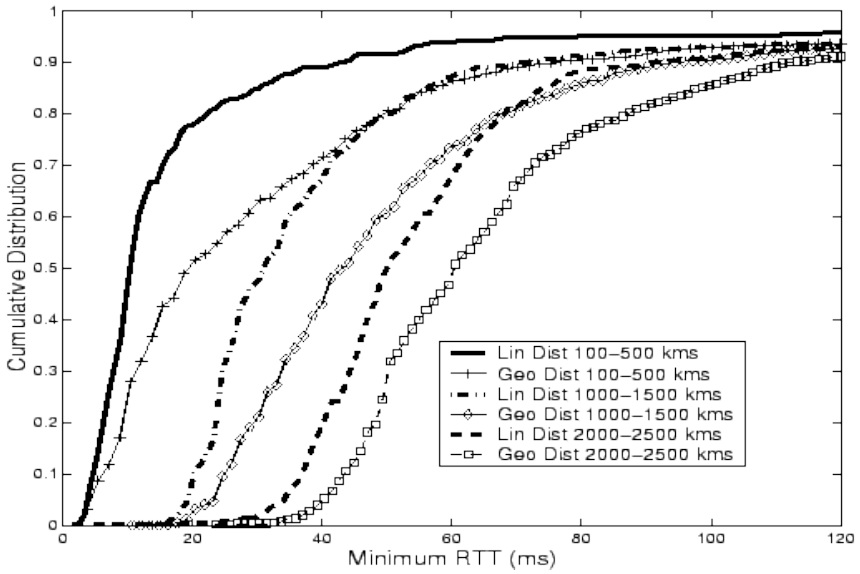

Fig. 9. CDF of minimum end-to-end RTT to TVHosts for different ranges of linearized distances and geographic distances of paths [4]

nodes actually have high latency and the local consensus, when run, takes long. Note that in our model we do not consider network-layer attacks such that malicious nodes can prevent honest nodes from measuring their round-trip-times, such as by delaying the packets.

4.3.1 Finding l_d

Suppose we want to find an upper acceptable bound on the reported l, such as l_d, when the claimed distance is d. By having the joint distribution of $f_{D,L}$, we can derive the conditional distribution of $f_{L|D}$. We can define l_d such that:

$$\begin{aligned} Pr(L > l_d \mid D = d) &< \sigma_1 \\ \int_{l_d}^{\infty} f_{L|D=d}(t)\, dt &< \sigma_1 \\ \int_{0}^{l_d} f_{L|D=d}(t)\, dt &> 1 - \sigma_1 \end{aligned} \tag{15}$$

The bigger l_d is, the lower the probability that two honest nodes find their claimed distances and latencies incompatible; however, if l_d is too large, then the probability that a malicious node can convince an honest node of its faulty distance increases. So we should consider the following to prevent malicious nodes to do that:

$$
\begin{aligned}
Pr(D > d \mid L < l_d) &< \sigma_2 \\
\int_d^{\infty} f_{D|L<l_d}(t)\, dt &< \sigma_2 \\
\int_0^{d} f_{D|L<l_d}(t)\, dt &> 1 - \sigma_2
\end{aligned}
\tag{16}
$$

So for a claimed distance of d, we should find l_d such that it satisfies both Equation 15 and Equation 16. We define reliability as the following:

$$
\begin{aligned}
Reliability(l_d) &= 1 - max\{Pr(D > d \mid L < l_d), Pr(L > l_d \mid D = d)\} \\
&= min\{1 - Pr(D > d \mid L < l_d), 1 - Pr(L > l_d \mid D = d)\} \\
&= min\{Pr(D < d \mid L < l_d), Pr(L < l_d \mid D = d)\} \\
&= min\{\int_0^{d} f_{D|L<l_d}(t)\, dt, \int_0^{l_d} f_{L|D=d}(t)\, dt\}
\end{aligned}
\tag{17}
$$

So we need to find l_d such that it maximizes the *reliability*:

$$
l_d = argmax_x(Reliability(x)) = argmax_x(min\{\int_0^{d} f_{D|L<x}(t)\, dt, \int_0^{x} f_{L|D=d}(t)\, dt\})
\tag{18}
$$

And we can define the function $g(l, d)$ as the following:

$$
g(d, l) = 1[l < l_d] = \begin{cases} 1, & l < l_d \\ 0, & o.w. \end{cases}
$$

As discussed earlier, this function has a reliability of $reliability(l_d)$ which we denote by $1 - \sigma_d$.

5 Simulation and results

We ran a simulation to see how this algorithm works. In our simulation, we set up 500 nodes. In this scenario, malicious nodes claim random coordinates on the Blockchain. The claimed coordinate of node A is tested. If A is a malicious node, its strategy is to justify its random claimed location in $C(A, i)$ when k_i random verifiers in $C(A, i)$ measure the round-trip-times to A. If the verifier is an honest node, malicious node A should be able to answer for the round-trip-time in time - either itself or with the help of another malicious node. If node A is an honest node, the malicious verifiers assigned to A to measure the RTTs all reject A's coordinate; however, the honest verifiers assigned to the honest node A, measure the RTTs truthfully.

For the simulation, we chose the values $c = 0.5$ for the *score* threshold, $\delta = 0.9$ for the security parameter, and $\beta = \frac{1}{3}$ in subsection 4.2. The $i_{max} = 7$ for the simulation and the node A has the real coordinate of (x'_A, y'_A) and a claimed coordinate of (x_A, y_A) in the space of a circle with the radius of $\frac{R_{max}}{2}$. In our simulation, the real coordinates of the nodes are randomly distributed over the space. For honest nodes, real coordinates and claimed coordinates are the same. For malicious nodes, the claimed coordinates are randomly distributed.

In our simulation, we varied m, the total number of malicious nodes, up to $\frac{n}{3}$. For a fixed m, we used the algorithm as the following. For the node A in the neighborhood of R_i of A, we use the $score_i$ and the threshold c to decide if A's claimed coordinate is plausible in $C(A, i)$ or not; meaning that, if the coordinate is plausible, we say the node A is honest in $C(A, i)$ and if it is not plausible, we say that node A is malicious in $C(A, i)$. This approach of classification has true positive - honest nodes that we classify as honest - and true negative - malicious nodes that we classify as malicious- values. For all the nodes in the system, we ran the algorithm over all of the neighborhoods of R_i around each of them for $1 \leq i \leq i_{max}$. For each i and m, we calculated true positive and true negative. The following graphs are the results of the algorithm for three of the radiuses of the neighborhoods - three values of i.

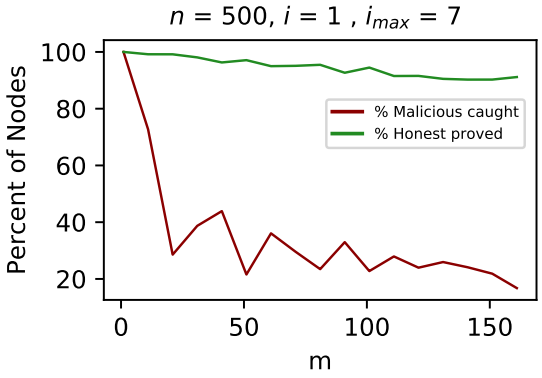

Fig. 10. True Positive and True Negative vs m for i = 1

In Figure 10, true positive is shown with the green line and is denoted by "% Honest proved" and measures the precentage of honest nodes that are classified as honest by the algorithm. In Figure 10, true negative is shown with the red line and is denoted by "% Malicious caught" and measures the precentage of malicious nodes that are classified as malicious by the algorithm. In Figure 10, i = 1 and the radius of the neighborhood is the smallest value. As you can see, the honest nodes with a high probability can prove their coordinates are correct. There will be some honest nodes in a mali-

cious neighborhood as described in subsubsection 3.1.4 that can not prove their coordinates; however, their percentage is small when m is small and increases when the number of malicious nodes increases. For smaller values of m, most malicious nodes are caught; however, as the number of malicious nodes increases, fewer malicious nodes are caught since it is more probable that there are more malicious friends to answer for the RTT measurement in any honest location.

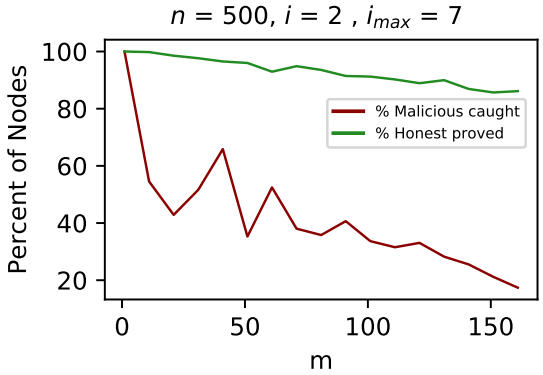

Fig. 11. True Positive and True Negative vs m for i = 2

In Figure 11, i =2 and the neighborhood is the second smallest neighborhood. As you can see, the honest nodes with a high probability can prove their coordinates are correct in their R_2. For $m < \frac{n}{4}$, more than 40% of the malicious nodes are caught when they claim random coordinates.

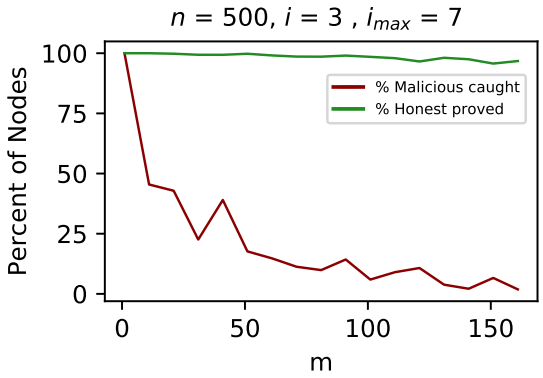

Fig. 12. True Positive and True Negative vs m for i = 3

In Figure 12, i =3 and the neighborhood is the third smallest neighborhood. As you can see, the honest nodes with a high probability can prove their coordinates are correct in their R_3. The neighborhoods are bigger in comparison with the previous graphs, and more honest nodes are in them

and it is easier for honest nodes to prove they are actually where they have claimed to be. However, there are also actually more malicious nodes in the neighborhood in terms of real coordinates and it would be easier for other malicious nodes to fake their coordinates in that neighborhood. Note that for $i_{max} = 7$, the radius of the space is $2^{i_{max}-1} = 2^6$, and the neighborhood R_3 has the radius of 2^3, one eighth of the radius of space.

As i gets closer to i_{max}, R_i covers more of the space. For a malicious node, it means that it should prove that its claimed coordinate is plausible to for example one-fourth of the space. Moreover, when i increases, with a high probability most of the verifiers will be far from the malicious nodes in the space and it would be easier for the malicious node to prove its claimed coordinate.

6 Conclusion

The described method for $\beta < \frac{1}{3}$ enjoys the lower complexity of $O(nlog(n))$ rather than the complexity of measuring all the round-trip-times which is $O(n^2)$. Furthermore, we eliminated the Sybil Location attack by asking validators to commit to their lcoordinates on the public ledger.

7 Aknowledgements

The author thanks Cristina Basescu and professor Bryan Ford, who advisored and supervised the development of this project.

References

[1] Sabrina Kall. Know-thy-neighbour: Approximate proof-of-location, DEDIS Lab semester project, EPFL. https://github.com/dedis/student_19_proof-of-loc/blob/master/report/Dedis_Semester_Project.pdf, June 2019.

[2] Trust but Verify: Blockchain applications at cloud scale. https://medium.com/@andysingleton/trust-but-verify-blockchain-applications-at-cloud-scale-33aaa1b927f9. Accessed: 2019-12-6.

[3] Rtt quartiles versus geographical distance. https://www.caida.org/projects/ark/statistics/lax-us/med_rtt_vs_dist.html. Accessed: 2019-12-6.

[4] Correlation between delay and distance. https://www.usenix.org/legacy/publications/library/proceedings/usenix02/full_papers/subramanian/subramanian_html/node21.html. Accessed: 2020-1-24.

[5] Ewa Syta, Philipp Jovanovic, Eleftherios Kokoris Kogias, Nicolas Gailly, Linus Gasser, Ismail Khoffi, Michael J. Fischer, and Bryan Ford. Scalable bias-resistant distributed randomness. Cryptology ePrint Archive, Report 2016/1067, 2016. https://eprint.iacr.org/2016/1067.