The Trust Chain Consensus

COTI: A Decentralised and Trust-Based Online Payment System with Mediation Service

Technical White Paper, V2.0, 1st March, 2018

Abstract

Online payment solutions in general, and cryptocurrencies in particular, lack the ability to scale in volume and speed. Several blockchain-based technologies have been created to tackle the challenges posed by providing high transaction throughput while remaining inexpensive, but these have been met with little success. Another challenge faced is the lack of trust between unknown parties, which leads to countless chargebacks and transaction cancellations. Moreover, merchants are often classified as 'high-risk' or 'low-risk' based on their association with a particular industry, rather than their demonstrated behaviour.

COTI (Currency Of The Internet) solves these challenges by using an innovative base layer protocol in the form of a directed acyclic graph-based ledger built from transactions connected through time by harnessing trust-based algorithms. Trust should be based on a combination of historical behavioural data and objective information relating to buyers and merchants. COTI takes this into consideration, calculating trust scores using a unique machine-learning algorithm. Trust is used in the Trust Chain Algorithm to validate and confirm transactions faster. Trust Chains grow as new transactions attach to two prior transactions that have similar degrees of trust. This results in an innovative consensus-based confirmation mechanism, in which every user is incentivised to have a high level of trust while engaging in trust-building behaviour due to the benefits associated with having a high level of trust (i.e. faster confirmation times).

COTI has built mechanisms to monitor, detect and defend against possible attacks, ensuring network security. An example of such a mechanism is COTI's Double Spend Prevention (DSP) Nodes. COTI also introduces novel protocols to address disputes that may arise when sending transactions, a much required feature which is not possible with other cryptocurrencies. Dispute resolution is achieved through the use of a Mediation Service. This service takes advantage of the principles of game theory to ensure a fair outcome in the case of a dispute and votes to determine which of the two disputing parties is right.

Keywords: Blockchain, COTI, Cryptocurrency, DAG, Distributed ledger, E-commerce

1 Introduction

Blockchain technologies and cryptocurrencies have proven to be successful mechanisms for managing payment transactions over the last several years. Currencies such as Bitcoin, Ethereum and many others have enjoyed exponential growth in popular interest and adoption[1], while blockchain technology has been been utilised in many applications, ranging from supply chain management [11] to decentralised health records management [17]. Indeed, many have likened cryptocurrencies to the early internet, citing its enormous potential to disrupt payment systems in the same way the internet disrupted information accessibility[14].

However, while first generation cryptocurrencies have been enormously successful, they have faced fundamental challenges that have prevented them from achieving universal adoption. Linear blockchain-based cryptocurrencies are impaired by low transaction throughput[2]; cryptocurrencies that rely on a net-

[1]For example, the number of Bitcoin transactions per day has grown from about 100 in 2009 to over 400,000 in late 2017 [2].

[2]Bitcoin delivers a maximum of 7 transactions per second [6].

work of miners to perform increasingly complex proof-of-work (PoW) computations incur prohibitively high transaction fees; and most existing cryptocurrencies are difficult to manage and subject to mass speculation. In addition, dispute resolution services, which are commonplace for credit cards and other payment platforms, are rare within the frameworks of most existing cryptocurrencies. These factors make it difficult for individuals and merchants to adopt them as a *global currency* or *digital dollar* for day-to-day transactions.

This paper introduces COTI (Currency Of The Internet), a next-generation cryptocurrency that achieves high transaction throughput and low fees, while being easy to manage and providing decentralised structures for the services users have come to expect from payment platforms, such as dispute resolution mechanisms. COTI achieves a high transaction throughput by employing a directed acyclic graph (DAG) of transactions called the *Cluster* as opposed to a blockchain. This idea is not new and has been proven to improve performance [13, 4, 15]. Typically, DAG-based cryptocurrencies[3] are intended for large numbers of low-value transactions and possibly for use between machines, such as IoT devices. Because COTI is designed to support day-to-day transactions between merchants and consumers, new algorithms have been introduced to drive the formation of the *Cluster*, and the approval of transactions. Fundamental to the new approach are *Trust Scores*, which are assigned to each user account based on its historical behaviour, and govern the approval of that account's transactions within the network and associated fees. These algorithms will be described in detail in sections 2 and 4.

In addition to the new features mentioned above, COTI introduces a Mediation Service for dispute resolution characterised by a decentralised collective of highly trusted network participants who are tasked with resolving transaction disputes. This enables the network to provide decentralised human-input services to its participants.

The Base Layer Protocol: DAG-based distributed ledger technologies show signs of being particularly adept at overcoming the scalability limitations inherent in blockchain-based payment networks. This is because while in blockchain-based networks, greater scale has undesirable effects on network usability, in DAG-based networks the reverse is generally true: greater network usage results in improved network scalability. As such, there is a positive correlation between the number of network users and the rate at which transactions are confirmed.

As a result of the positive correlation between network usage and network scalability, the DAG data structure is ideally suited for the COTI network's base layer protocol and will enable it to achieve full decentralisation without compromising on COTI's commitment to scalability, instantaneity and low-to-zero fees. Building on the foundations established by the above-mentioned initiatives, COTI is introducing as its base layer protocol an innovative DAG-based distributed ledger technology that involves the use of Trust Scores as the key mechanism by which new, unconfirmed transactions select prior transactions to validate. Furthermore, COTI's DAG-based distributed ledger technology, the Cluster, reaches faster consensus when confirming a transaction by using COTI's Trust Chain Algorithm. Eventually, the Cluster will be able to validate and confirm a maximum of 10,000 transactions per second (TPS)[4].

The Mediation Service: COTI offers a ready-to-use service a user can revert to in cases of fraud or any other dispute related to transactions settled through the COTI payment system. The Mediation Service creates a rolling reserve for each merchant to cover possible claims and a system-wide Reserve Credit Fund (RCF) to further guarantee it. Both funds are maintained in COTI's native currency, and the required size of a merchant's rolling reserve is calculated based on the merchant's Trust Score.

[3] e.g. IOTA

[4] Arguments for this can be found in Section 8.

Fees: The COTI network uses a transparent and equitable fee model. All fees are collected by the Full Nodes, which are decentralised servers run by general users in the COTI network. The COTI network receives a portion of fees collected by the Full Nodes to support infrastructural technology, such as the Double Spend Prevention Nodes (see Sections 3.2 and 11.1) and Trust Score Servers (see Section 3.3). When the network is first created, a portion of COTI's generated tokens will be set aside as a Reserve Fund to pay for all transactions until the network matures. The network fee will therefore be set at zero during the network's infancy. After this period, the fees will be minimal due to the decentralised nature of the network.

Each node charges a fee that is in part determined by the node itself. Some nodes may set a higher fee if they believe they are providing a good service; other nodes may charge less or possibly nothing. The price charged by a node for its services must be equitable, publicly available and compliant with common network rules. Network rules will define a ceiling for fees, but there will be no minimum fee.

It is possible for a merchant to run their own Full Node along with a customised wallet if they think it will provide a better experience for customers.

Glossary of Terms

Term	Meaning
Node	A specialised server run by a user for common network tasks.
Validation of a transaction	The process of checking the transaction before its attachment to the Cluster.
Source transaction	A terminal transaction of the Cluster having no inbound transactions. These transactions have gone through the validation process.
Confirmed transaction	A transaction for which the consensus algorithm has reached a defined level of total trust.
Trust Score	A user metric that is used for effective transaction processing and risk mitigation.
Attack	A malicious attempt to compromise a system's integrity.
Double-spending	A malicious attempt to execute two transactions using a duplicate account balance. This results in a negative balance and the attacker acquiring something without cost.
Escrow account	An account containing funds that can only be used when specific conditions have been met.

2 The Trust Chain Algorithm

COTI has developed a new approach for achieving consensus between transacting parties that operates on a DAG-based data structure. The Cluster is based on a completely decentralised DAG, which is not stored by any central authority. The Cluster is considered the ledger, or a record of all transactions processed by the network. It achieves scalability through its use of parallel source selection and transaction confirmations, as well as its use of COTI's Trust Scoring Mechanism.

2.1 Trust Score of transactions in the Cluster

Each transaction in the Cluster receives a Trust Score based on a combination of the sending account's historical behaviour and information about the account owner. This information is given to COTI when

the user first creates an account and is substantiated by supporting documentation. Further information on the Trust Score Algorithm can be found in section 4.

Trust Scores are the main determinant of transaction fees in COTI. Participants with high Trust Scores can expect to pay low-to-zero fees, whilst participants with low Trust Scores will need to pay higher fees. This incentivises merchants to supply the best service possible in order to increase their Trust Scores. An additional benefit of having a high Trust Score is that higher Trust Scores are confirmed faster (for more details, see Sections 6, 7 and 8).

When executing a transaction, the sender is required to validate two prior transactions in order for the new transaction to be added to the ledger. The ledger is therefore organised as a DAG (directed acyclic graph), where the vertices represent transactions and directed edges extend from each transaction to two others that it validates. A schematic of the Cluster is shown in Figure 1. Each white circle represents a transaction that has been validated by two subsequent transactions, while the darker circle represents a new, unvalidated transaction (a "source" in graph-theoretic terminology). As new transactions are added, they may validate the darker transaction.

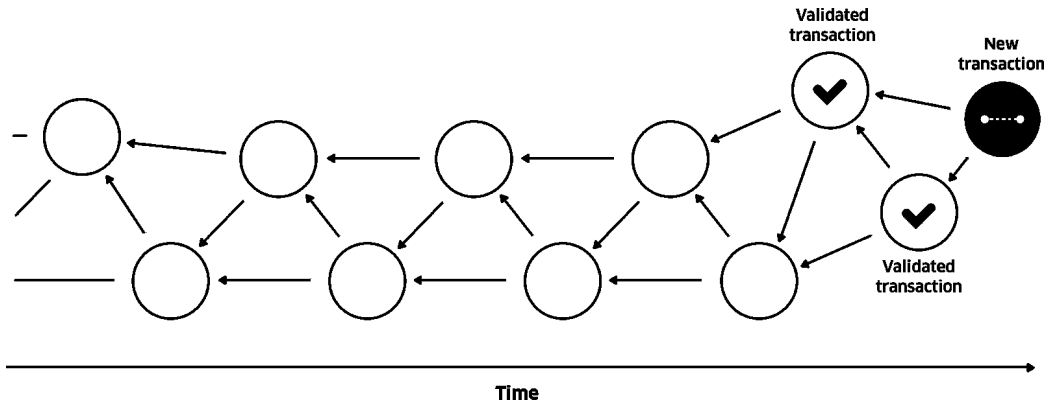


Figure 1: Cluster schematic. The source transaction (dark circle) validates two previous transactions in the Cluster.

2.2 Source Selection

The process outlined above requires each new transaction to select two prior source transactions to validate. In the COTI network, the algorithm for effectuating this choice is based on the Trust Score of each transaction (see Section 6). According to this *Source Selection Algorithm*, a source will likely choose prior transactions that are close to its current Trust Score. This results in the formation of Trust Chains, or any reverse-oriented path in the Cluster. The cumulative Trust Score of such a chain is the sum of the Trust Scores of all the transactions making up the chain.

The *Trust Chain Algorithm* makes use of the cumulative Trust Score to achieve consensus on transaction confirmation. A transaction is considered to be confirmed if it is the starting point of a Trust Chain that has a cumulative Trust Score exceeding some preset global threshold. In practice, we consider the longest Trust Chain (highest trust) starting from each transaction and compare its cumulative Trust Score to the threshold to establish if the transaction has been confirmed.

Because the Source Selection Algorithm tends to connect transactions of similar Trust Scores, Trust Chains generated by highly trusted users will mostly contain transactions with high Trust Scores. The cumulative Trust Score of such a Trust Chain will grow quickly past the threshold and achieve consensus, meaning that highly trusted users will enjoy faster confirmation times and higher transaction throughput.

Another important outcome of the Source Selection Algorithm is the soft segmentation of the Cluster based on Trust Scores. This means that the DAG is divided into nearly independent sub-DAGs consisting of users with similar Trust Scores.

The Cluster will progress through the following life stages: in the first stage, it is initiated as a new transaction; in the second, it attaches to the Cluster by validating two other transactions with similar Trust Scores; in the third, it is validated by other transactions; finally, it is confirmed and permanently added to the Cluster once the cumulative Trust Score of the heaviest path confirming it surpasses the set threshold (as illustrated in Figure 2).

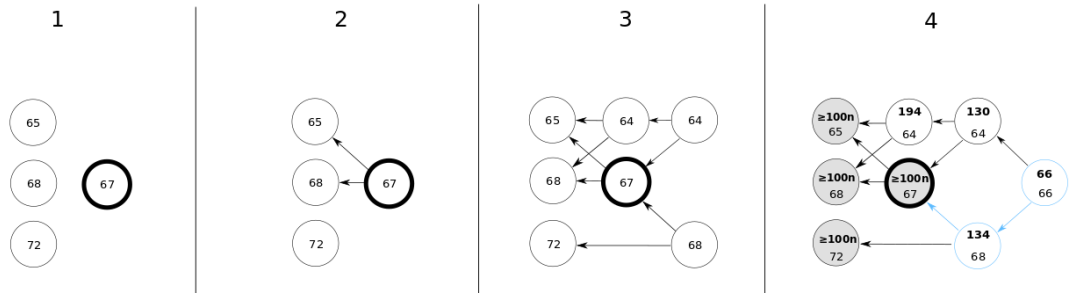


Figure 2: The lifecycle of a transaction (bold cirlce) from (1) initiation, to (2) attachment, to (3) validation, to (4) confirmation. The confirmation path is in blue, while the shaded transactions have been confirmed. For the purposes of this illustration $n = 2$.

COTI's Trust Chain Algorithm is designed in such a way that trusted users (i.e. those with a high Trust Score) will experience faster confirmation times than those who are less trustworthy. This is natural as people are more cautious when dealing with people they trust less and would like to be certain that their transactions are confirmed before accepting them. This is illustrated in Figure 3 and further explored using simulations in section 8.

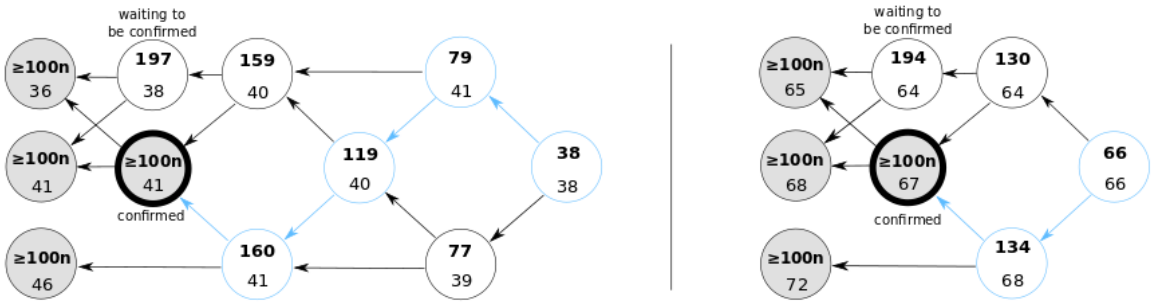


Figure 3: The different Trust Chain lengths needed to confirm moderately trusted transactions (left) and highly trusted transactions (right). Shaded transactions are those that have reached the cumulative trust threshold, while the confirmation path is in blue. Shaded transactions have been confirmed. For the purposes of this illustration $n = 2$.

3 Network Components

3.1 The Account

Each potential COTI user opens an account using a COTI wallet and is given a randomly generated private and public key. Each user has to answer several factual questions in order to create his/her Trust Score. COTI's wallets will eventually be customisable as there will be various types of roles a user can register for (see Table 1), and the detailed operation of these wallets will vary from user to user.

3.2 The Nodes

COTI provides a decentralised solution designed specifically to allow for secure and trustworthy payments. This solution relies on the distribution of responsibility in the Cluster to different types of nodes that are run by users. COTI offers the following three node types:

Full Nodes: The Full Nodes are the primary infrastructure components of the network and serve as user gateways. They receive new transactions signed by the client (wallet) app, request the Trust Score from Trust Score Servers, choose sources for new transactions to attach to, do proof-of-work and allow new transactions to attach to the Cluster.

DSP Nodes: The Double Spend Prevention Nodes keep an updated copy of the Cluster at a given point in time and are COTI's solution to one of the key problems faced by DAG-based cryptocurrencies. To run a DSP Node, a large amount of COTI will have to be deposited in a special multisig account. This amount is substantial, and may be deposited by the DSP Node operator alone, or by a group of network participants that have delegated their deposits to the node operator. DSP Node operators are paid by the COTI network. The details of DSP Node creation are further discussed in section 11.1.2.

History Nodes: The History Nodes keep the earlier parts of the Cluster after the Clusterstamp process is complete and also store the entire history of the Cluster on COTI's servers. Full account history can be retrieved from the History Nodes; however, if the History Node is unable to operate for any reason, COTI's History Servers will be used instead.

3.3 The Servers

COTI will run servers that perform important tasks for the network. The four different categories of servers are: Zero Spend Servers, Trust Score Servers, KYC/AML Database Servers, and History Servers.

Zero Spend Servers: These servers are responsible for sending zero-value transactions when any source in the Cluster has waited a long time without being validated by another transaction, or if a transaction is unable attach to a source using the Source Selection Algorithm. The activity of these servers will help to monitor the Source Selection Algorithm. If these servers experience over-activity, it may indicate that there are problems with the network or with the Trust Score Algorithm.

The Trust Score Servers: These servers are dedicated servers which calculate and store *Participant Trust Scores* according to the Trust Score Algorithm (see Algorithm 1).

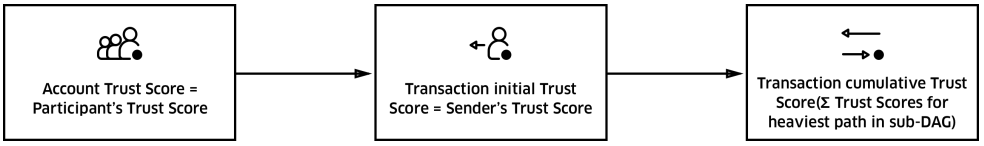
The KYC/AML Database Servers: These are the dedicated servers that process and store information for know-your-client (KYC) and anti-money-laundering (AML) procedures.

The History Servers: These are the backup servers for the History Nodes.

4 Trust Score

Trust Scores are a key feature of the COTI network and are used for effective transaction processing and risk mitigation. Trust Scores are calculated using the Trust Score Servers or the KYC/AML Database Servers.

All accounts in the payment network have a Trust Score, which may change based on particular events or statistics. An account's Trust Score is a number that is within the range [0,100]. All transactions in the payment network have a *Trust Score* that is stored in the transaction bundle. The *Trust Score* of a transaction is the sender's Trust Score when the transaction is initiated. The *Cumulative Trust Score* of transaction A is the sum of all the Trust Scores of all transactions along the *heaviest path* approving transaction A, including transaction A itself.



The *Account Trust Score* is initially determined by fields from a questionnaire filled in by the participant who owns the account and has uploaded necessary documents. After the *Account Trust Score* is

Table 1: The various participant roles

Participant type / Participant role	Person	Entity	Non-human (Device or robot)
User			
Merchant			
Financial institution			

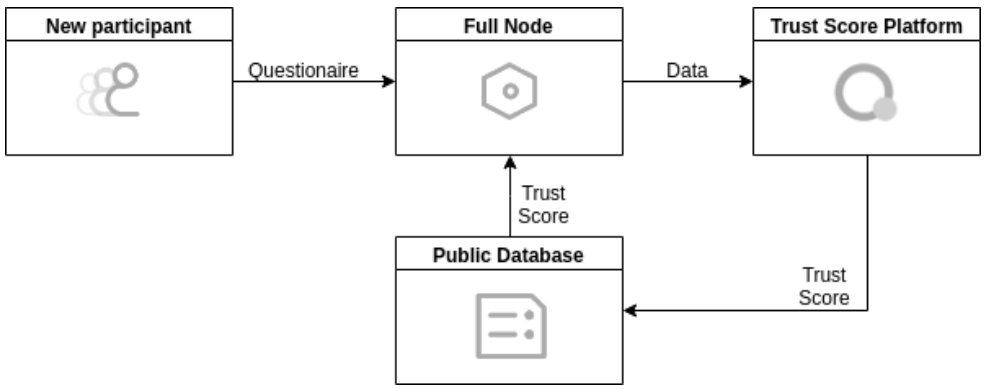
determined it is updated according to account payment statistics, participant behaviour and "big data" collected by the network.

A participant may be a *Person*, a *Business entity* or a *Nonhuman*, and may have the role of a *User*, a *Merchant* or a *Financial Institution* (these roles are summarised in Table 1). When a participant is registered as a merchant or a financial institution it enables the participant to receive payments in exchange for goods or services. Non-cooperative participants trying to sell goods or services without registering either as a merchant or as a financial institution will be blocked for network safety.

Echoing the prevailing opinion in the cryptocurrency community, new participants will not be banned from having a wallet prior to filling out questionnaires or submitting identification documentation. As a result of the absence of information regarding such participants, undocumented users will have their Trust Scores set to 1, and undocumented entities will have their Trust Scores set to 0. Participants with low Trust Scores (e.g. < 5, the small payments threshold) can send only small transaction amounts (e.g. $\leqslant 1$ COTI). Participants with Trust Score $= 0$ cannot initiate any transaction.

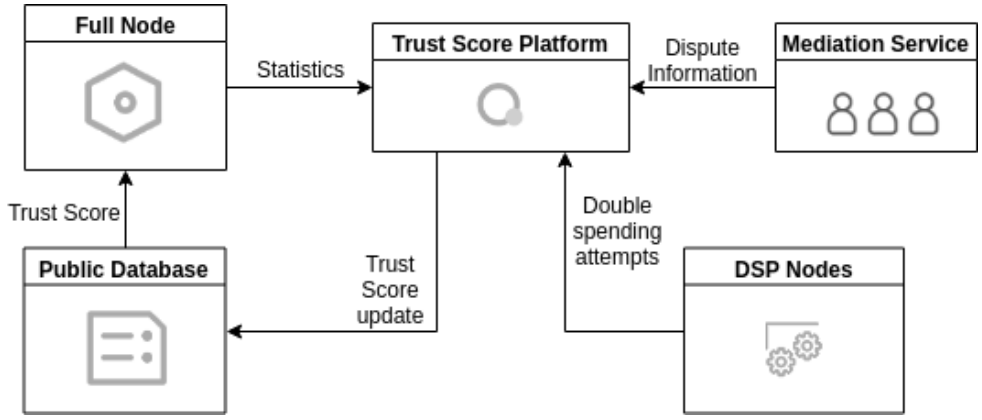
No one can participate as a merchant or as a financial institution before completing identity verification and KYC/AML procedures. In order to participate as a merchant or financial institution the participant must provide additional data on their business activities, licences (if applicable), bank references, auditing reports, etc.

Figure 4: Process illustrating how a new participant receives a Trust Score.



The Trust Score Algorithm will be designed to ensure the maximal performance of the Trust Score-based Source Selection Algorithm. Trust Score Servers periodically update participants' Trust Scores according to cumulative statistics and participant behavioural patterns. Participants' Trust Scores can also change in line with external events, such as bankruptcy news related to the merchant. The Trust Score Servers will receive reliable information on these events (for example, from authoritative sources or from users that have provided supporting documentation from official news sites). The protocol for submitting and processing such information is still in development.

Figure 5: Process for updating Trust Scores.



4.1 The Trust Score Algorithm

Prerequisites: The Trust Scores are mainly used as weights for the Source Selection Algorithm and to measure the degree to which a transaction has been confirmed. Both are essential for the operation of the network: the first affects the efficiency of the network, while the second impacts the security of the payment system.

Trust Scores are used to rank participants so the Trust Chain Algorithm can work efficiently. The criteria and parameters of Trust Score calculations will be periodically updated to ensure that the Trust Score remains efficient and equitable.

For example, the recommended level of trust (cumulative Trust Score) for the transaction to be confirmed could be 1000 until the actual recommended level is derived from theoretical modeling and computer simulations. If highly trusted participants need a Trust Chain consisting of at least 10 transactions to be confirmed, then moderately trusted participants will need a Trust Chain of at least 20 transactions. Less trusted participants, on the other hand, will be required to have a Trust Chain of at least 100 transactions to be confirmed. In this instance, the Trust Score Algorithm would assign them Trust Score values of approximately 100, 50 and 10, respectively.

COTI's current Trust Score Algorithm categorises participants into the 9 groups (baskets), illustrated in Table 2, according to the specified Trust Score ranges. These groups are subject to change according to network demands.

Trust Score Range	Group Description	Trust Score Range	Group Description
40-50	**Esquires** Fairly trusted, average transaction frequency	**80-100**	**Network backbone** Very high trust, very high transaction frequency
30-40	**Startups** Average trust, average transaction frequency	**70-80**	**Pillars of economy** High trust, very high transaction frequency
20-30	**Documented newcomers** Average trust, average transaction frequency	**60-70**	**Tigers** High trust, high transaction frequency
10-20	**Second chancers** Low trust, average transaction frequency	**50-60**	**Old merchants** High trust, average transaction frequency
0-10	**La Cour des Miracles** Low trust, low transaction frequency		

Table 2: The 9 groups specifying the Trust Score ranges. The numbers in this table only serve illustrative purposes.

It is important that COTI's Trust Score Algorithm be based on information that identifies the participant. This is necessary to encourage participants to engage in trustworthy conduct in order to improve their Trust Scores. This will also provide an effective safeguard against attacks and misuses, while encouraging participants to engage in honest conduct.

For the above reasons, COTI's Trust Score Algorithm is a combination of two approaches:

1. Rules-driven approach
2. Machine-learning (ML) approach

Algorithm 1: TrustScore Algorithm. Note that TS is an abbreviation for Trust Score.

```
TrustScore(p, t, r, Q, S)
  in: Participant ID p; Type of the participant t; Role of the participant r; Questionnaire Q;
      Payment statistics S;
  out: The Trust Score value;
  constant: TS decay rates d; minimum_TS_for_person;

  if is_new (p)
      if Q is empty then
          if t = person then
              TS ← minimum_TS_for_person;
          else
              TS ← 0;
          end if;
          return TS;
      end if
      TS_data ← empty_data
  else
      TS_data0, t0 ← Select_TS_data_TS_DDB(p); // take available TS components from the distributed
          database
      t1 ← now;
      TS_data ← apply_decay(TS_data0, t0, t1, d);
  end if

  TS_data.RD_params ← apply_rules(TS_data, r, Q, S);
  TS_data.ML_params ← apply_ML_algorithms(TS_data, r, Q, S); // use trained ML algorithms to estimate
      parameters
  TS_data.group ← define_TS_group(TS_data);
  TS ← count_TS(TS_data); // the resulting TS is counted according to group, ML_params and outcome of
      rules applied
  Store_TS_data_TS_DDB(p, TS, TS_data);
  return TS;
```

Some features cannot be used as part of an ML algorithm. For example, a user's age cannot be used to evaluate trustworthiness, as there are no statistics correlating age to trustworthiness. One possible solution for this challenge is to consider the age boundaries used in the insurance industry and use them to create an age-based rule. The outcome of such a rule can be used to define a basket, or to assist in Trust Score calculation. This rule will only be used for the initial Account Trust Score and will disappear over time as new information concerning the user's behaviour becomes available.

Algorithm 2: Algorithm for Wrongdoers

```
WrongdoersRule(p)
  in: Participant p;
  out: The minimum Trust Score group (basket) number;
  constant: time_to_oblivion t1, t2;

  MB_data_list ← Select_MB_data_TS_DDB(p); // take misbehaviour data from the distributed database
  if MB_data_list is empty then
      g ← 0; // any group is possible
  else
      if is_there_serious_misbehaviour(MB_data_list, t2) then
          g ← 9; // only worst group is possible, surely the number of the group will not be
              hardcoded constant
      else
          if is_there_small_misbehaviour(MB_data_list, t1) then
              g ← 8; // the group may not be better than group 8, surely the number of the group will
                  not be hardcoded constant
          else
              g ← 0; // any group is possible
          end if
      end if
  end if
  return g;
```

5 Anatomy of a Payment

5.1 Opening an Account

A COTI account can be opened using a simple process that generates datasets per account: the public key, the private key (both functions of a randomly chosen seed) and the initial Trust Score. Each payment that originates from account A and is deposited into account B will be encapsulated as a bundle that includes several datasets about the COTI token transaction. To ensure the transaction is balanced, the input and output sum cannot be less than 0. This is the first balance check performed to verify the entire ledger's validity.

5.2 Bundling

Each transaction is a bundle made up of several sub-transaction fields.

- **The output:** The amount to be sent to the receiver and the receiver's hash of the public key.
- **The input:** The amount to be deducted from the sender's account and the sender's public key.
- **The informative part:** The list of security features, including the BundleHash and other data.
- **The Trust Score:** The sender's Trust Score is contained in the bundle and must be in the [0, 100] range.

5.3 Hashing and Signing the Bundle

A secure hash function is used to create a hash bundle of fixed length from the created transaction bundle. The complete bundle is subsequently signed using the sender's private key and the ECDSA algorithm. From here, the data sent contains the following information: the original bundle data, the bundleHash and the signature of the bundle.

When a new transaction attaches to the Cluster, it must verify two previous transactions. This verification process, unlike the checks performed by the DSP Nodes (see Section 3.2), also reviews the signatures of the two previous transactions. To understand how this is done, it is important to note that a user's private key is associated with his or her public key.

While any person can sign the bundle hash, only the owner of the private key can sign it in such a way that the result of the verification with the public key is the same as the original bundle hash of the transaction. This is how a transaction's signature is checked against the sender's public key.

5.4 Proof-of-Work

Once the sender's node has confirmed that the two selected transactions (as determined by the Trust Chain mechanism) are valid, the sender's node performs a proof-of-work (PoW) that involves solving a cryptographic challenge. The precise PoW mechanism employed by the COTI network will be presented in future iterations of this document. The primary motivation of the PoW mechanism is to counteract spam and Sybil attacks.

A known drawback of other DAG systems is the requirement of PoW to deter spamming. This is inevitable in a zero-fee system, but in an effort to exclude potential attacks, PoW drains the batteries of low-energy devices.

In the COTI network, PoW requirements are the responsibility of nodes, which serve as gateways to the network and determine user fees. It is possible, for example, to create nodes that connect mobile devices without PoW âĂŞ as nodes do the PoW in any case, they can't spam the network.

5.5 Fees

Resources will be needed to maintain the Mediation Service and DSP Nodes. For this reason, it is necessary that at least some users pay minimal fees. The fees incurred by a transaction depend on the Trust Score of the user and the type of transaction. There are three common types of transactions:

- A transaction in which one user transfers COTI to another user. In this case, the standard fee is applied and determined by the Full Node according to the sender's Trust Score.
- A transaction in which a user transfers COTI to a merchant in exchange for goods or services. In this case, the merchant has the option to pay fees specific to his/her own account in order to make the transaction free for the ordinary user.
- A standard fee will be applied for a transaction in which a merchant transfers money to another merchant or user (for example, if the merchant wishes to refund the user for a defective good).

6 Source Selection Algorithm

The goal is to build a Cluster based on the Trust Score of the transaction sender. In the Cluster, every transaction is attached to a maximum of two other transactions with Trust Scores that are sufficiently close to its own. On the DAG, the Trust Score of the transaction sender is assigned to each transaction with a weight function ω. Let d be the upper bound for Trust Scores. This is equal to 100 according to the Trust Score Algorithm,

Any method for constructing such a DAG must be based on an algorithm that selects two transactions with some degree of randomness. For example, an algorithm which chooses two transactions b and c based only on having Trust Scores closest to the Trust Score of the source a. One can see that the use of a non-random algorithm such as the one just described, increases the probability of there being many sources that have to wait a long time before being attached by a transaction.

COTI's Source Selection Algorithm works in the following way: a new transaction a is issued by an account. $\mathscr{S}$ is the set of all the sources. The algorithm chooses the optimal threshold of $\omega(a)$. First, all the sources are partitioned with a map function $\mathcal{M} : \{1, 2, .., d\} \to \{\mathscr{T} : \mathscr{T} \subseteq \mathscr{S}\}$ such that $\mathcal{M}(i) = \{\mathscr{T} : \mathscr{T} \subseteq \mathscr{S}$ and $\omega(\mathscr{T}) = i\}$, where d is the upper bound for the Trust Score. The initial subset is $\mathscr{T}_0 = \mathcal{M}(\omega(a))$. Iterations in the algorithm generate new subsets $\mathscr{T}_i = \mathscr{T}_{i-1} \cup \mathcal{M}(\omega(a)-i) \cup \mathcal{M}(\omega(a)+i)$ until $\mathscr{T}_i$ is sufficiently populated or $i < \lceil d/8 \rceil$. Without any loss of generality, it can be said that being sufficiently populated implies that there is a constant percentage of all the sources in the source set. 10% of the population of the sources is chosen to imply sufficient population. If at any iteration $\omega(a) - i < 1$ or $\omega(a) + i > d$, then $\mathcal{M}(\omega(a) - i)$ or $\mathcal{M}(\omega(a) + i)$, respectively, is taken to be the empty set. There is one further restriction that must be applied to subset $\mathscr{T}_j$ of sources in the threshold of $\omega(a)$, namely that no transaction may be attached to a transaction with the same transaction sender. A probability function P weighting all sources s in $\mathscr{T}_j$ according to the timestamp difference between s and new transaction a is defined by the algorithm. a can then select any two sources in $\mathscr{T}_j$ with some degree of randomness, but such that the older sources will be chosen with a higher probability than the newer sources. There is zero probability of selecting sources from the same transaction sender of a. The Java Code in Algorithm 3 shows how this is performed.

Algorithm 3: Java code showing how sources are selected.

```
public SourceList selectSources(int trustScore, int minSourcePercentage, int totalSourceNum, int
    maxNeighbourhoodRadius) {

    // Start by taking the sources with the same Trust Score (clone)
    SourceList sourceList = new SourceList(sourcesByTrustScore.get(trustScore));

    // Calculate the neighbourhood radius, minimal radius is 1 (always look at neighbours)
    for(int nr=1; nr < maxNeighbourhoodRadius; nr++) {
```

```
            if(trustScore - nr >= 1)
                sourceList.add(sourcesByTrustScore.get(trustScore - nr));
            if(trustScore + nr <= MAX_SCORE )
                sourceList.add(sourcesByTrustScore.get(trustScore + nr));

            if((double)sourceList.size() / totalSourceNum > (double)minSourcePercentage / 100) {
                break;
            }
        }

        // Randomly choose source, weighted by timestamp difference
        return chooseWeightedByTimestamp(sourceList);
    }
```

Note that our algorithm can respond to changes in the flow of new transactions since it takes into account the number of sources in a transaction's threshold. In the early stage of the Cluster, there will be cases when transaction a cannot be attached to a transaction in $\mathscr{T}_j$: for example when all the source Trust Scores are accumulated too far from $\omega(a)$ or when all the sources in the selected threshold are from the same transaction sender as a. In these cases, a Zero Spend Server will create a zero-value transaction with the same Trust Score of transaction a and a will be attached to that transaction. In another scenario, if a source s has been waiting a long time to be attached by a new transaction, then a Zero Spend Server will create a transaction that can be attached to s (and with the same Trust Score as s). The waiting time before the Zero Spend Server performs these tasks will be determined by the Trust Score: high Trust Score sources will be matched faster by the Zero Spend Servers. As mentioned in Section 3.3, over-activity of Zero Spend Servers will help to identify problems in the network or in the Trust Score Algorithm.

7 Trust Consensus

Let $G = (V, E)$ be a directed acyclic graph of transactions. Assume that every transaction $v \in V$ is weighted with a weight function $\omega : V \to \mathbb{N}$ defined by $\omega(v)$, the Trust Score of the transaction sender. Let a be any transaction and d be the upper bound for the Trust Score. We say the transaction a is confirmed if

$$\max\left(\Sigma_v \omega(v) : \forall \text{ path } \mathscr{A} \text{ ended at } a \text{ and } \forall v \in \mathscr{A}\right) \geq Ld,\ L \geq 2 \tag{7.1}$$

Equation 7.1 implies that highly trusted transactions will be confirmed faster than less trusted transactions due to our Source Selection Algorithm. Note that a highly trusted transaction a is confirmed quickly because the length of the heaviest directed path is very small. Notice also that less trusted transactions need a longer path to be confirmed. The algorithm for the heaviest directed path that ends at transaction a is a linear time algorithm, namely $O(|V(F)| + |E(F)|)$ where $F \subset G$ is a directed acyclic subgraph defined by the union of all directed paths ending at transaction a. The first step is to sort F topologically. Since F is a directed acyclic graph, finding a topological sort τ is linear time (Chapter 22.4 in [5]). Let $\tau = \{v_1, v_2, ..., v_n, a\}$ be a topological sort of F. Notice that transaction a should be the last vertex at the topological sort due to the definition of F.

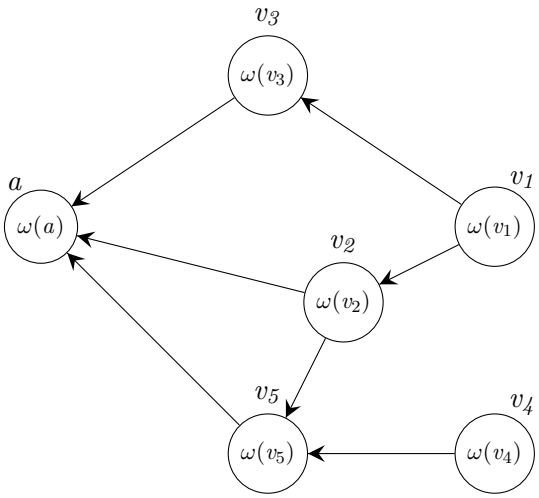


Figure 6: DAG subgraph F before topological sorting

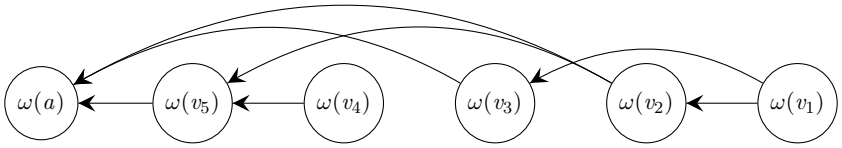


Figure 7: DAG subgraph F after topological sorting

The following dynamic programming algorithm gives the $O(|V(F)| + |E(F)|)$ time solution for the heaviest path from transaction a:

Algorithm 4: Heaviest Path Algorithm

```
Define function heaviest(v) = 0 , ∀v ∈ V(F);
Find Topological sort τ of F;
for ∀v ∈ V(F) in topological sort τ do
    Assign heaviest(v) = max(heaviest(w) + ω(v): ∀w such that (w, v) ∈ E(F));
end
return heaviest(a);
```

8 Performance Investigation

In order to better understand the performance characteristics of COTI's algorithms, we will provide a mathematical framework for making deductions about the Cluster, in addition to high level mathematical observations in a simplified context. We will then present a series of empirical investigations that make use of a full simulation of the Cluster.

8.1 Mathematical framework

For the purposes of analysis, we have made some simplified assumptions about the Cluster and the transactions taking place within it. First, we assume that all nodes take a fixed amount of time Δt to run the Source Selection Algorithm and perform proof-of-work. Second, we assume that new transactions arrive according to a Poisson process with fixed rate λ. We also assume that the transactions are all valid and the senders are distinct. All of these assumptions are not reflective of the real world, but can be locally true for stretches of time and sections of the Cluster, and are therefore useful to consider for analysis. The parameters λ and Δt will feature in the discussions below.

There are also a number of internal parameters that control how the Cluster will operate. Trust Scores can take on integer values from 1 to d, the maximum possible Trust Score. We will assume $d = 100$ in what follows. When a new transaction arrives in the Cluster, we assume it has Trust Score i, $(1 \leq i \leq d)$ with probability $P_{TS}(i)$, where $P_{TS}(1) + P_{TS}(2) + ... + P_{TS}(d) = 1$. This corresponds to the assumption that transactions of each Trust Score arrive with independent Poisson processes, wherein the process for Trust Score i has rate $\lambda P_{TS}(i)$. An additional internal parameter L appears in the Trust Chain Algorithm. This parameter determines if a transaction is confirmed when the highest trust path from that transaction to a source transaction in the Cluster has a cumulative Trust Score of at least Ld.

Finally, the Source Selection Algorithm enabless a new transaction to attach to any source with a sufficiently similar Trust Score and is controlled by a parameter ρ. If the set of available sources has size S and is sorted according to differences in Trust Scores from the new transaction, the first ρS must be available for selection, along with any others with the same Trust Score. The Source Selection Algorithm also depends on another parameter, R, which dictates the maximum absolute Trust Score difference allowed between a transaction and the transactions it approves. For the sake of analysis, we assume that $R = d = 100$.

The set of parameters $(\lambda, \Delta t, d, L, \rho, P_{TS}(\cdot))$ fully determines this simplified model of the Cluster. It is useful to visualise the transactions of the Cluster in the 2-D space described by time on the x axis and Trust Score on the y axis. The DAG structure formed by constructing a directed edge from a transaction to both of the two prior transactions that it verifies can also be visualised on this graph. One such visualization is provided in Figure 8. The spatial representation of the transactions has some useful properties. In particular, time is a reverse topological ordering of the graph by construction, so the x axis of the graph provides a valid vertex visitation order. Also, transactions are more likely to be connected if they have similar Trust Scores (y axes), especially for small values of ρ. Figure 8 illustrates that as ρ is decreases, there is less connectivity between transactions at different heights on the graph.

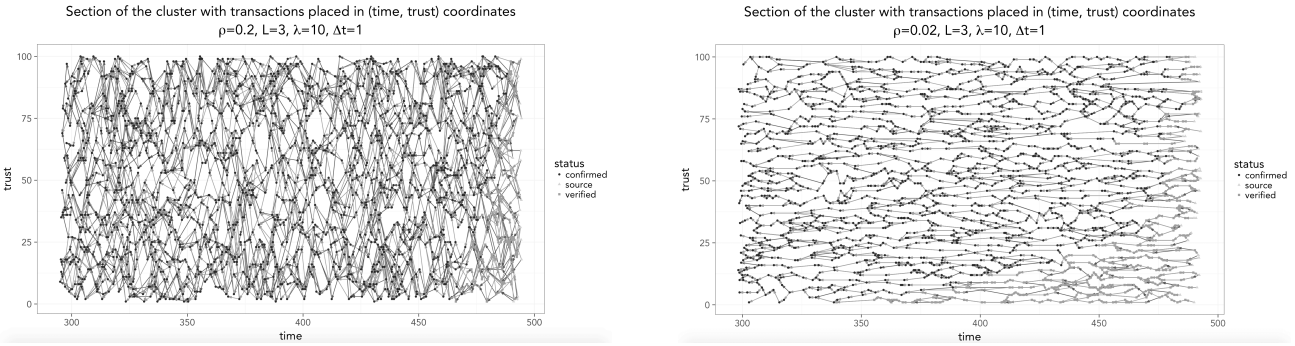


Figure 8: The Cluster in the space of Time by Trust Score. Empirically we find that the cluster becomes almost disconnected when $\rho \ll \lambda$.

8.2 Performance Analysis

DAG-based transaction systems have been analysed under the assumptions of a uniform random attachment algorithm with no Trust Score (e.g. [13]).These analyses found that the number of source transactions in the DAG should approximately approach the constant value $2\lambda\Delta t$ once the system has stabilised. A similar result can be obtained in our case for the number of sources.

Consider the case where there are S source transactions at some point in time, and let s be one of the current sources that will be selected by the new transaction that arrives. Before that new transaction publishes the selection to the network, it must perform validation and proof-of-work. This takes Δt time. During this time, s is still visible as a source transaction to all new transactions that arrive in the network. On average, there will be $\lambda\Delta t$ such transactions. Each of the new transactions will only be able to attach to s if it has a close enough Trust Score. Assuming that the Trust Score distribution for new transactions is identical to that of sources in the Cluster, the new transaction will have probability

ρ of being allowed to attach to s, and there will be on average ρS transactions available for it to connect to. Since each new transaction attaches to two sources, we can compute the probability that s is selected by a transaction as:

$$\rho\left(\frac{1}{\rho S}+\left(1-\frac{1}{\rho S}\right)\frac{1}{\rho S-1}\right)=\frac{2}{S}$$

Recalling that s has already been selected by the next new transaction that arrives (and therefore already has 1 transaction attached to it), this means that the expected number of transactions attaching to s is $N_A = 1 + 2\lambda\Delta t S^{-1}$. This is the average number of DAG edges that it takes to validate one source transaction. We also know that each new transaction adds two edges to the DAG before becoming a source transaction.

The quantity N_A therefore determines if S will grow or shrink over time. If $N_A > 2$, then each new transaction is removing less than one source on average and S will increase. Conversely if $N_A < 2$, then each new transaction is removing more than one source on average and S will decrease. However, since the number of sources S is in the denominator of this expression, $N_A = 2$ is an *attractor*: if $S > 2\lambda\Delta t$, then $1+2\lambda\Delta t S^{-1} < 2$ and the number of sources will decrease, and if $S < 2\lambda\Delta t$, then $1+2\lambda\Delta t S^{-1} > 2$ and the number of sources will increase. Therefore over time, S must approach the fixed point $2\lambda\Delta t$. In particular, for a sufficiently large amount of time, S can be assumed to be approximately constant.

We now consider the number of validated, unconfirmed transactions at time t, which we will denote by V. We will use the fact that after a sufficient amount of time, the expected in-degree of all transactions is 2 since every transaction starts out as a source transaction and will have an expected in-degree of 2 in the steady state.

At a fixed time, let $\mathscr{S}$ be the set of source transactions; $\mathscr{V}$ be the set of validated (unconfirmed) transactions; $\mathscr{V}_i$ the set of vertices in $\mathscr{V}$ with the longest reverse oriented path (in a number of transactions) to $\mathscr{S}$ of precisely i. We observe that $\mathscr{V} = \bigcup_{i=1}^{Ld} \mathscr{V}_i$ since every path length that is greater than Ld has a cumulative trust score greater than Ld, and so any vertex at the beginning of such a path is confirmed and not in $\mathscr{V}$.

Now the parents of any vertex from $\mathscr{V}_1$ are in $\mathscr{S}$. Further, each vertex in $\mathscr{S}$ contributes two out-edges to the DAG, and by the argument above each vertex in $\mathscr{V}_1$ consumes on average 2 vertices from the DAG, so we find that $\mathbf{E}[|\mathscr{V}_1|] \leq \mathbf{E}[|\mathscr{S}|] = 2\lambda\Delta t$. Similarly, the parents of vertices in $\mathscr{V}_2$ are in $\mathscr{V}_1 \cup \mathscr{S}$, and so $\mathbf{E}[|\mathscr{V}_2|] \leq 4\lambda\Delta t$. Proceeding inductively, $\mathbf{E}[|\mathscr{V}_i|] \leq 2i\lambda\Delta t$. Adding all of these together, we find that:

$$V = \mathbf{E}[|\mathscr{V}|] \leq (Ld)(Ld+1)\lambda\Delta t$$

This is not a strict bound, but a constant in time. We have therefore established that V is bounded above by a constant after a sufficient amount of time has passed.

We have established that the number of source transactions S and the number of validated, unconfirmed transactions V are both approximately constant after enough time has passed. We therefore turn our attention to the number of confirmed transactions, C. Since every transaction already attached to the Cluster is either a source transaction, a validated (unconfirmed) transaction or a confirmed transaction, the total number of attached transactions in the Cluster is $S + V + C$. Taking the expected ($\mathbf{E}$) rate of change we get:

$$d[\mathbf{E}[S+V+C]] = d[N_{transactions}] = \lambda$$

It follows that because S and V are constant in expectation after sufficient time has passed, then we must have $d[\mathbf{E}[C]] = \lambda$. In other words, the rate of confirmation of new transactions matches the arrival rate of transactions.

Empirical investigations confirm that this behaviour is indeed the case, as the confirmation rate matches the arrival rate after some initial time (see Figure 9).

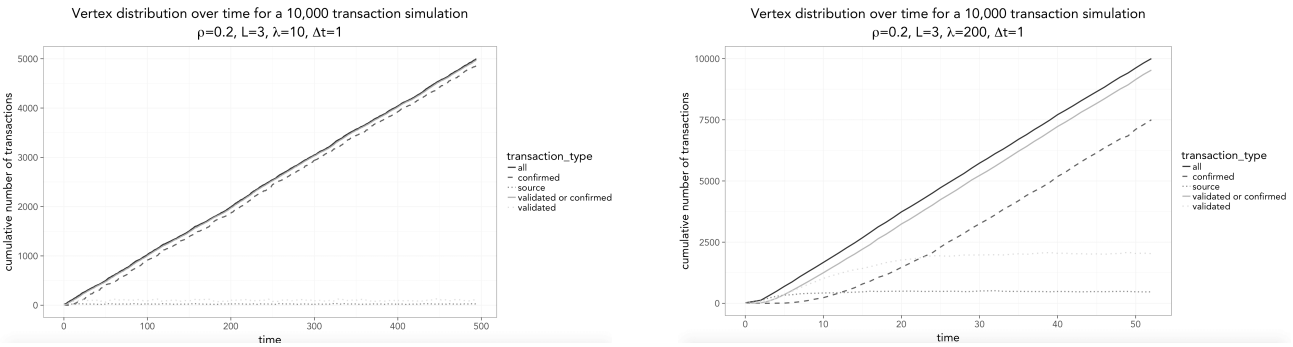


Figure 9: These figures show that after the initial phase, the system becomes stable and the rate at which transactions are confirmed equals the rate at which new transactions arrive. In particular, the number of sources and number of new (not yet validated) transactions becomes constant.

This provides conclusive evidence that the Cluster is scalable. The only theoretical limitation to the throughput of COTI is the number of transactions arriving per second. Because COTI was designed to be attractive to a large pool of merchants and consumers, we are confident that the number of transactions arriving per second will eventually exceed 10,000.

8.3 Simulations

The previous section established the throughput characteristics of the Cluster under simplified conditions. To investigate other characteristics of the Cluster and to analyze more complex scenarios, we have utilised simulations.

A simulator of the Cluster was built to facilitate further analysis, and will be made available on GitHub. While the simulator is capable of constructing complex scenarios, we have in this section restricted our attention to the limited set of assumptions outlined in the mathematical framework above.

The simulation was first used to verify that the transformation $t \to ct; \Delta t \to c\Delta t; \lambda \to \lambda/c$ is a symmetry of the Cluster. This essentially means that we can rescale our time unit from seconds to "multiples of Δt" without loss of generality. Some graphs from this initial investigation are shown in Appendix A. In our remaining analyses, we set $\Delta t = 1$.

Whilst the previous section addresses confirmation throughput, it doesn't take the amount of time a transaction has waited into account. We expect that confirmation times will decrease in line with increasing Trust Score and has been confirmed by simulation results. (Figure 10).

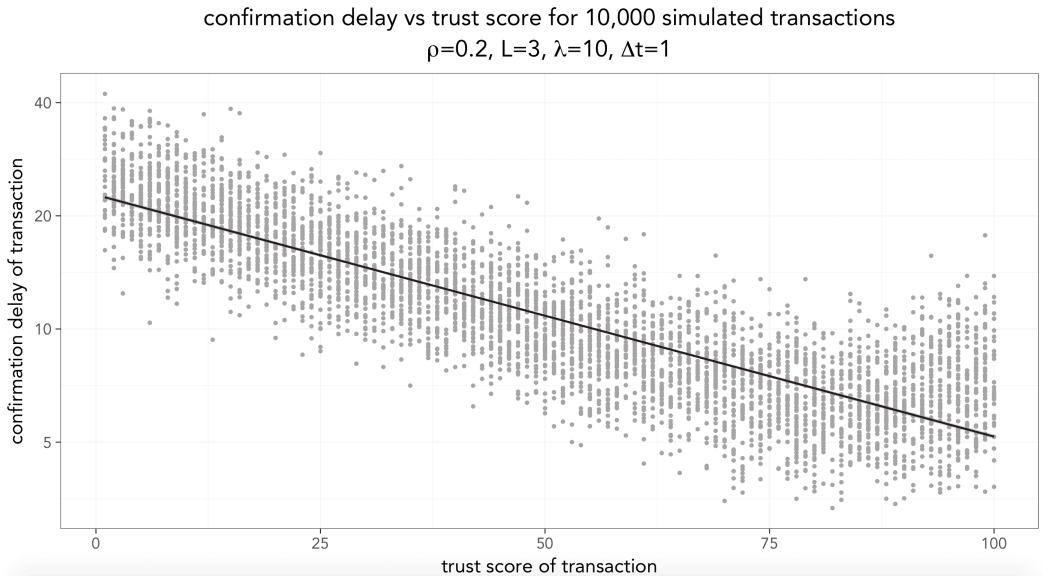


Figure 10: The confirmation delay (log-scale) decreases markedly with increasing Trust Scores, with an order of magnitude difference between Trust Score 1 and 100. The drop-off is linear almost everywhere with flattening at the end.

9 The Clusterstamp

To prevent the Cluster from growing to become unmanageable in storage size, COTI introduces the Clusterstamp. This process, however, also results in a number of other benefits for the COTI network. For example, the Clusterstamp provides a reference point that parties can revert to in dispute cases. It provides an opportunity for performing a system-wide audit to ensure that there have been no fraudulent attempts to artificially inflate account balances. This process consists of two phases:

1. The balance of each account is aggregated and attached to the relevant address and is done on the latest unconfirmed transactions in the Cluster. Each of the latest unconfirmed transactions then becomes the genesis transactions in the next generation of the Cluster.

2. An audit is performed to make sure the supply of COTI tokens contained in the Clusterstamps is equal to the total supply of COTI tokens in circulation.

The Clusterstamp process is performed automatically. Following the creation and verification of a Clusterstamp, it is stored in the network's History Nodes.

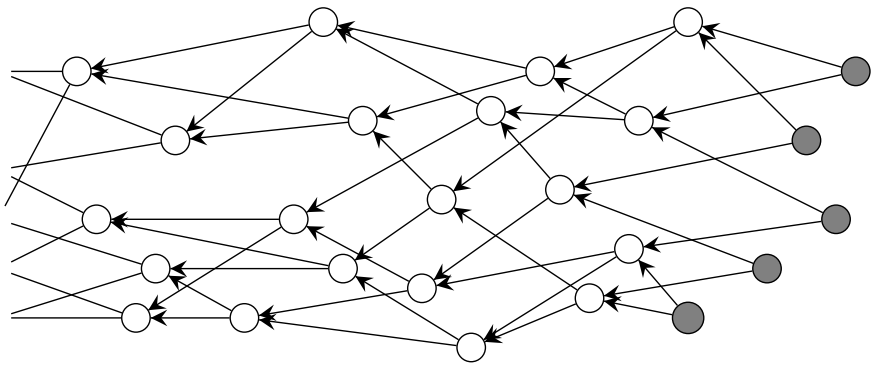


Figure 11: The Clusterstamp process captures all the information up to the time of the gray transactions. Thereafter, the next generation of the Cluster begins with the gray transactions.

10 Escrow Accounts and the Mediation Service

COTI will provide users with well designed and inexpensive multisig and escrow accounts.

The Mediation Service of the COTI platform will be designed for quick and efficient dispute settlement through an arbitration procedure. COTI provides a framework for it, a common set of rules and a jury of mediators.

Mediators are required to be reputable network participants, as such network participants may be invited to be mediators on the basis of their Trust Scores and are incentivised by the mediation fee. A business version of the COTI wallet will enable merchants to appoint an officer who represents them as a mediator without granting them access to the funds in the wallet.

Arbitration services for trade disputes cannot use escrow accounts because businesses are not interested in withholding money for general trades. For this reason, the Mediation Service may only be used to settle disputes when both counterparties have opted in. Consent to the Mediation Service is required of COTI merchants.

In the most common case of a trade dispute – in which a customer demands a chargeback from a merchant – there are two possible outcomes. If the merchant wins, no additional action is needed. If the merchant loses the case, the Mediation System transfers money from the merchant's rolling reserve to the customer's account. If the merchant's rolling reserve does not have sufficient funds, the Reserve Credit Fund will be used and the merchant will be obliged to reimburse the RCF. In the COTI Mediation System, merchants are subject to a period in which they can opt to pay voluntarily, or otherwise make an appeal.

The Mediation Service will be governed by the mediation code developed by the COTI team.

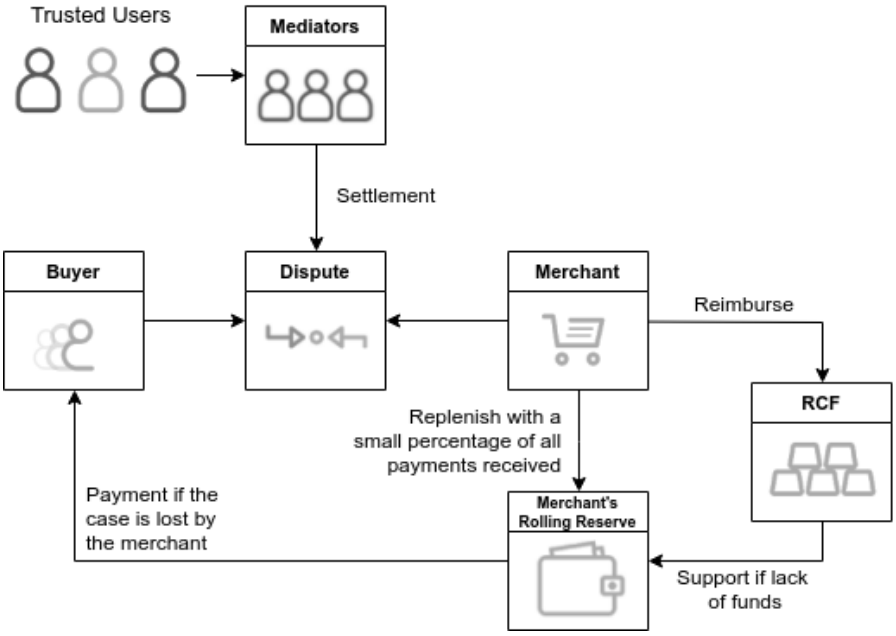


Figure 12: Mediators act as data oracles. They are incentivised to accurately validate real world information in disputes, and their consensus enforces efficient and equitable commerce.

11 Possible Attacks

11.1 Double-Spend Prevention

A payment solution cannot be open to the possibility of double-spending attacks. To mitigate this risk, COTI deploys dedicated Double Spend Prevention (DSP) Nodes. These nodes carry out additional

transaction monitoring without affecting the network.

11.1.1 The Double-spend Prevention Mechanism

To prevent double-spending , the DSP Nodes performe the following (Figure 13):

1. Keep a light version of the Cluster with pre-calculated balances for all accounts;
2. Receive a copy of any new transaction attached to the Cluster;
3. Check new transactions against a set of heuristics to detect possible double-spending attempts;
4. Check new transactions against available account balances;
5. Sign legitimate transactions;
6. Flag transactions suspected of double-spending;
7. Inform Trust Score Servers about double-spending attempts.

As the transaction verification process performed by the DSP Node is expected to be a quick procedure, only the amounts involved are checked and not the signatures of a transaction. The checks that the DSP Nodes perform are only carried out after a transaction has been attached to the Cluster. Transactions require the signature of a DSP Node before they can be considered as fully confirmed.

Any double-spending attempts detected are flagged and refused on malicious intent. Valid transactions receive signatures from DSP Nodes to be confirmed. The DSP Nodes are load- balanced to ensure that double-spending verifications are efficient.

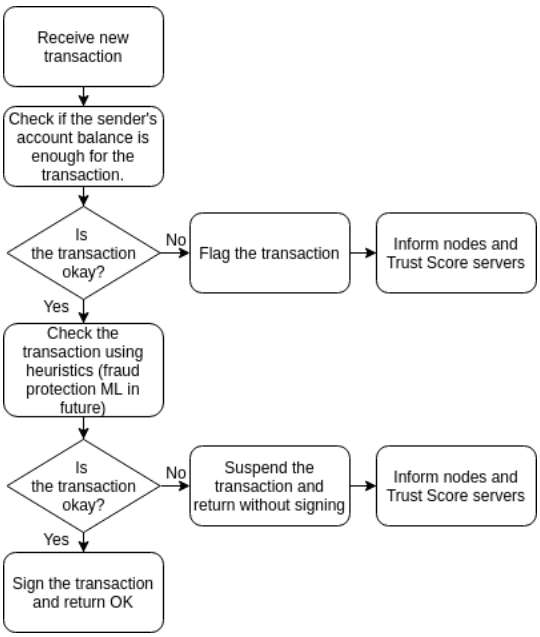


Figure 13: The verification procedure followed by DSP Nodes

11.1.2 The creation of a DSP Node

Due to the nature of the verifications required to prevent double-spending, a user who would like to run a DSP Node must meet the following requirements:

1. The DSP Node operator is required to pass compliance checks that include KYC/AMP procedures;
2. A substantial amount of COTI must be deposited in a special multisig account;
3. The performance and security of the DSP Node, including load-balancing quality, must be verified remotely.

As the amount of COTIs required to operate a DSP Node is rather large (above 10,000 COTI), a delegated proof-of-stake mechanism will be necessary. Each user's stake will be proportional to the amount of COTI he or she has contributed towards the operation of the DSP Node (Figure 14).

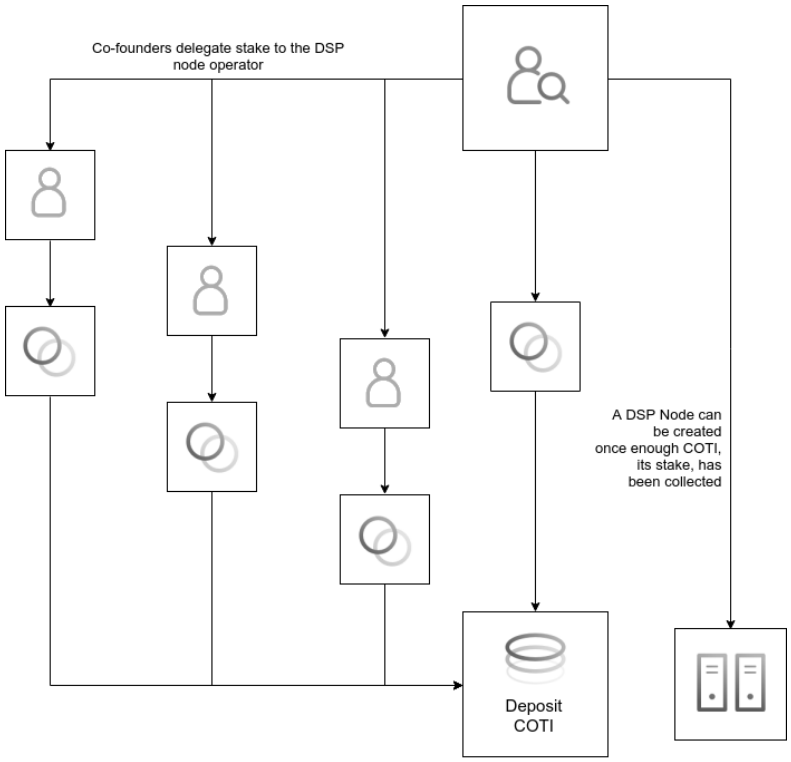


Figure 14: Procedure for DSP Node creation based on the principle of delegated proof-of-stake.

11.2 Penny-spend Attack/Transaction Flood

If an account is trusted, it will potentially pay zero transaction fees. A malicious aprty may take advantage of this by sending many microtransactions, thereby wasting node storage resources. The PoW required to validate transactions limits the number of payments an attacker can send due to the high computational resources required to launch such an attack. In the rare event that an attacker is successful at making many microtransactions, his/her account's Trust Score will decrease along with the privilege of making zero-fee transactions.

Even if an attacker with a low trust account attempts to send invalid transactions that will eventually be rejected, he/she can still try to flood the Cluster with many invalid transactions from several anonymous accounts. This attack could create a situation where all source transactions are invalid and new transactions cannot attach to the Cluster. If we assume that the Cluster has many transactions being executed (including zero-spend transactions) at a given point in time, there will be many available sources that cannot be exhausted by an attacker that has to perform a PoW for each source. To launch such an attack, the malicious party will have to have access to extensive resources that can overwhelm all participant resources, including those hosted by COTI.

11.3 Sybil Attack

An attacker can operate multiple computers, virtual machines and IP addresses to create numerous accounts with different usernames and email addresses. These accounts, known as Sybil identities, can be used in an attempt to subvert the use of trust in the network [10]. Since accounts with low trust can be created without identifying the person associated with them, an attacker with many accounts could try to create a subcluster that begins with a double-spend from one account at the beginning of their

subcluster and then proceed to validate their own transactions with other accounts by ignoring COTI's Source Selection Algorithm.

If a chain of sufficient length is created, the attacker could claim to have confirmed transactions once enough trust has been accumulated. In this situation, DSP Nodes will check if such a situation exists and will prevent any double-spending attempt. Another defense is that untrusted accounts have higher transaction fees and spending limits, which may make such an attack unfruitful.

Another possible attack can occur when a malicious party with a highly trusted account attempts to confirm transactions from another self-created account. The attacker may try to attach high trust transactions to the anonymous transactions in order to reach the confirmation threshold. This attack, however, is impossible as anonymous accounts have low Trust Scores; such transactions cannot be confirmed, as an account with a high Trust Score cannot confirm a transaction from a low trust account.

11.4 Man-in-the-middle Attack

Since packets can be inserted into communication channels by an attacker, the malicious party by attempt to take over one of the specialised nodes, such as the DSP Node. The possibility of such an attack is problematic since when a user first joins COTI they will not know if they are using the public key of an attacker or a real COTI node. To solve this problem the COTI client will have the public key of the COTI servers hard-coded in it. The COTI servers will therefore serve a similar function to a certificate authority in an SSL/TLS handshake. Once a secure connection with the COTI servers is established, the servers will then be used to get the authentic public keys of special nodes.

11.5 Malicious Node Attack

The fact that participants can purchase nodes that perform verification, namely the DSP Nodes, means that a malicious party can attempt to buy favor in the COTI network. It may seem that all an attacker needs to verify their own transactions are, at minimum, a Full Node and DSP Node.

However, since verification requires consensus among DSP Nodes, purchasing DSP Nodes is expensive, and we assume that purchasing the majority of DSP Nodes will not result in a profitable attack. Furthermore, only participants with a high Trust Score can create a DSP Node, and as soon as the other DSP Nods find it to be acting maliciously, it will lose all trust, be blocked and the owner's deposit will be seised.

11.6 Majority Attack

Due to the decentralised nature of cryptocurrencies, a single user or a small group of users with high computational resources can skew network consensus to their favor. This is known as a 51% Attack [9], wherein attackers can attempt to recompute PoW puzzles and create an alternative transaction history. This is done in the hope that their history will be duplicated by other network users and eventually become the consensus.

In the COTI network, a malicious party can attempt to recompute PoWs in a portion of the Cluster in order to reverse its previous transactions. Launching such an attack would require extensive computational power, but is theoretically possible. DSP Nodes can prevent such an attack from causing double spends, while History Nodes will prevent any attempts of deleting transactions to generate a false history record.

11.7 Protocol Compliance

If a user attaches his/her transaction to another transaction created by a user with a higher Trust Score, it can slow down the confirmation times of other transactions as the trust path might run through his/her transaction. This operation, however, is difficult to carry out in practice because transactions can only attach to sources that have similar Trust Scores (i.e. they can only attach to sources within

the same Trust Score threshold). This prevents new transactions from attaching to sources with vastly greater Trust Scores, which is necessary to make such an attack effective.

11.8 Denial of Service (DDOS) Attacks

As a result of the servers' centralised nature, an attacker could flood the Trust Score Server with requests. Such an attack implies that users will not able to get Trust Scores easily and the system might slow down, or even be incapable of processing transactions [12]. To prevent this, COTI will employ standard DDOS prevention mechanisms, such as load balancing for centralised servers. A similar attack could be launched against specialised nodes, such as the DSP Nodes because of their somewhat centralised nature. However, since some of the specialised nodes will eventually be general users, they may not have access to load balancing. COTI will therefore need to ensure that at least some of the specialised nodes that are hosted by general users will make use of robust networking technology to the extent that the network will not cease operation.

11.9 Distribution of Software Patches

Flaws have been found in many cryptocurrency implementations [18]. Patches should therefore be distributed securely and quickly to prevent nodes from being compromised. In the event of such a breach, any flaws found in the COTI client are unlikely to result in significant losses since balances and network transaction history are verified by DSP Nodes and History Nodes respectively.

12 The Future of COTI

In order to provide a completely decentralised ecosystem for online payments, COTI is exploring various alternatives to allow for decentralised governance. This governance structure will be responsible for making decisions that affect the base protocol, the future use of COTI's tokens, investments and more. This body will not only vote on these matters, but will also be responsible for executing changes. Futarchy [8] is one such type of governance that is currently being explored.

Decentralised governance could take place following a protocol update. Once the update is ready, a team of experts in the field would be tasked with creating a metric (a way of assessing the update) to determine the possible outcomes if the update is implemented. After this is determined, COTI token holders will be able to vote for the best possible outcome for the network. As such, decisions will be based on the wisdom of the crowd. The mechanism for choosing the team of experts will be determined in future iterations of the network.

To streamline COTI's future development, COTI's transaction bundles will have free space set aside where future information layers can be stored. These layers may be used by other companies that wish to run information based on the Cluster, for the transfer of other currencies and datasets across the network, or for the deployment of smart contracts.

References

[1] Peter Arntz. Blockchain technology: not just for cryptocurrency. `https://blog.malwarebytes.com/security-world/technology/2017/12/blockchain-technology-not-just-for-cryptocurrency/`, 2017.

[2] blockchain.info. Confirmed Transactions Per Day. `https://blockchain.info/charts/n-transactions?timespan=all`.

[3] John Adrian Bondy, Uppaluri Siva Ramachandra Murty, et al. *Graph theory with applications*. Elsevier Science Ltd, 1976.

[4] Anton Churyumov. Byteball: A decentralized system for storage and transfer of value. 2016.

[5] Thomas H Cormen. *Introduction to algorithms*. MIT press, 2009.

[6] Kyle Croman, Christian Decker, and Ittay Eyal Eyal. On scaling decentralized blockchains. 2016. 20^{th} international conference on Financial Cryptography and Data Security 2016.

[7] Brian S Everitt, Sabine Landau, Morven Leese, and Daniel Stahl. *Cluster Analysis*. Wiley Online Library, 2011.

[8] Robin Hanson. Shall we vote on values, but bet on beliefs? `http://mason.gmu.edu/~rhanson/futarchy2007.pdf/`, 2007.

[9] Investopedia. 51% attack. `https://www.investopedia.com/terms/1/51-attack.asp`, 2017.

[10] Brian Neil Levine, Clay Shields, and N. Boris Margolin. A survey of solutions to the sybil attack. 2006.

[11] A.P. Moller Maersk. Maersk and IBM to form joint venture applying blockchain to improve global trade and digitise supply chains. `https://www.maersk.com/press/press-release-archive/maersk-and-ibm-to-form-joint-venture`, 2018.

[12] Jelena Mirkovic and Peter L. Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *Computer Communication Review*, 34, 2004.

[13] Serguei Popov. The tangle. 2017.

[14] MIT Technology Review. Blockchain primer. `https://www.technologyreview.com/collection/blockchain-primer/`, 2018.

[15] Yary Ribero and Daniel Raissar. Dagcoin whitepaper. 2015.

[16] Sheldon M Ross. *Introduction to probability models*. Academic press, 2014.

[17] Mustafa Suleyman and Ben Laurie. Trust, confidence and verifiable data audit. `https://deepmind.com/blog/trust-confidence-verifiable-data-audit/`, 2017.

[18] Bitcoin Wiki. Common vulnerabilities and exposures. `https://en.bitcoin.it/wiki/Common_Vulnerabilities_and_Exposures`.

Appendix A Simulation Results

A Cluster simulator has been built and will be made available on GitHub. The primary purpose of the simulator is to analyze parameter and algorithm choices and to collect empirical data within a sandbox so as to optimise real world Cluster performance. The simulator can be used to analyze the impact of internal and external parameters on the performance of the Cluster. It is accompanied by a collection of data extraction and visualization tools that enable rapid scenario analysis. The core simulator is written in C++ with analysis components in R. It is able to simulate about 1000 transactions per second on a laptop computer and has been tested in simulations with up to 5,000,000 transactions.

Some selected simulation results are displayed here, while those relevant to the discussion on performance characteristics have already appeared in this document in Section 8. Below is a summary of the relevant parameters used within most of the simulations presented here:

1. Δt is the fixed amount of time for a node to run the Source Selection Algorithm and to perform proof-of-work.
2. λ is the rate of new arrivals, which are assumed to follow a Poisson process.
3. K is the number of new arrivals that arrive in the time taken to run the Source Selection Algorithm and perform proof-of-work.
4. L is the multiplier which determines the cumulative trust threshold that a Trust Chain must surpass in order to be confirmed. The cumulative trust of a Trust Chain should exceed $100L$ for the transaction to be confirmed.
5. ρ is the width of the Trust Score threshold that a transaction can confirm. This is expressed as a fraction of the total number of new transactions.

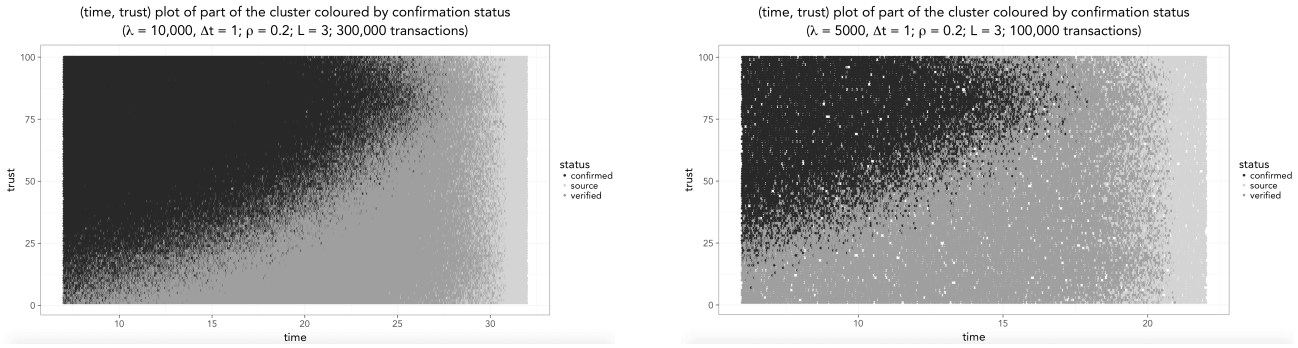


Figure 15: Waiting time as a function of Trust Score. Note that confirmed transactions are clustered around the region of high trust and low confirmation time.

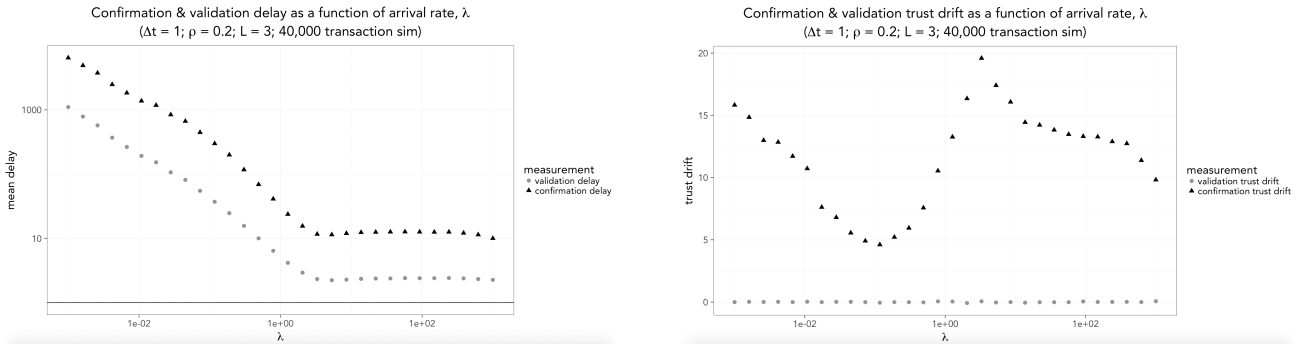


Figure 16: These figures suggest an inflection point slightly before $\lambda = 10$. Note that confirmation delay closely resembles validation delay, with the excpetion of trust drift.

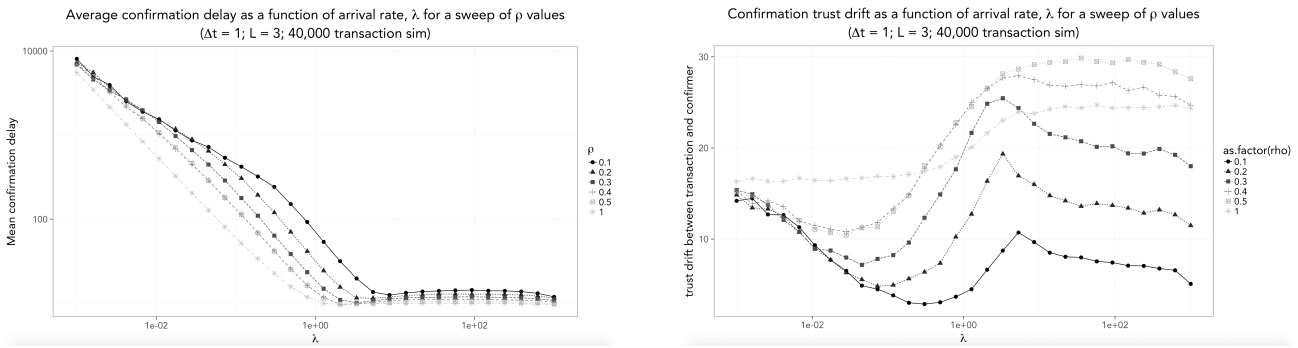


Figure 17: Note the inflection point slightly before $\lambda = 10$. This appears to be universal and is likely related to the value of λ necessary to stop the DAG from becoming disconnected as is visible in Figure 8

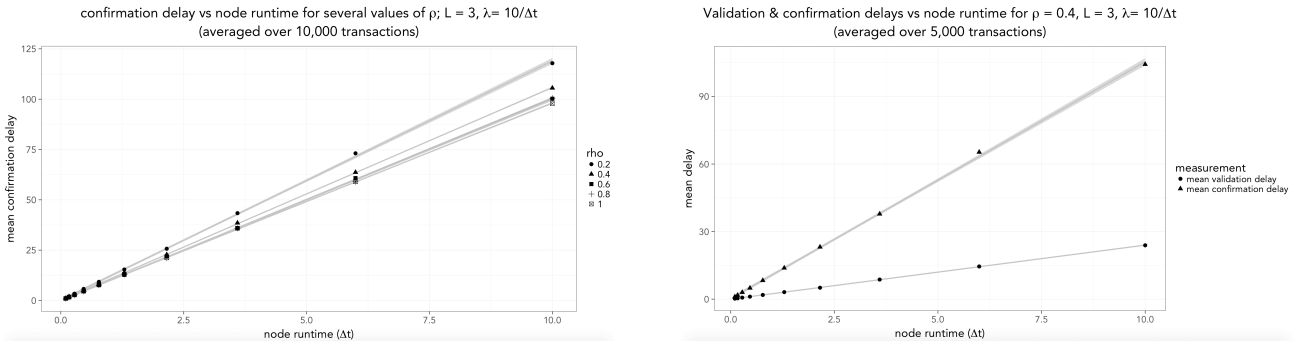


Figure 18: The figure on the left shows the confirmation time is barely affected by the size of the threshold ρ. The figure on the right shows that the mean delay in validation and confirmation times are directly proportional to the time taken to run the Source Selection Algorithm and to do the proof-of-work (Δt).

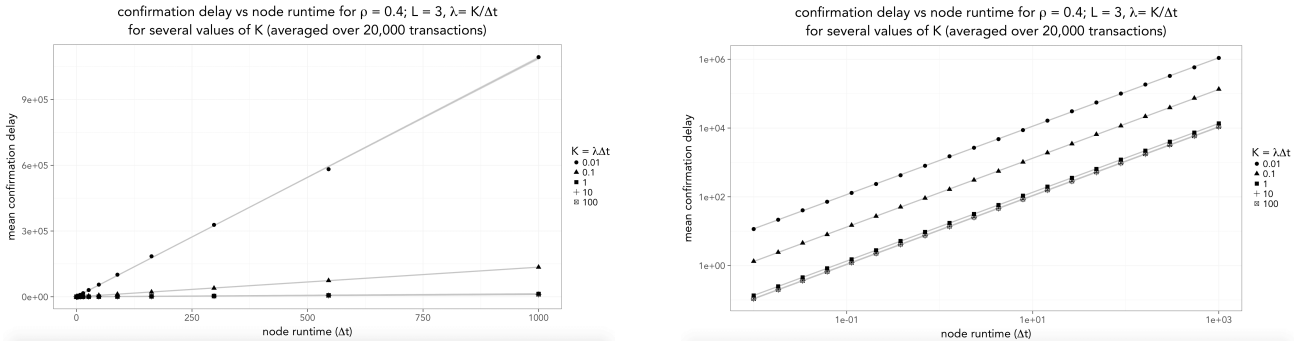


Figure 19: These figures show that the mean confirmation delay is significantly controlled by the parameter K. This means that as long as the rate of new transactions arriving is high enough, a complex proof-of-work is not detrimental to transaction throughput.

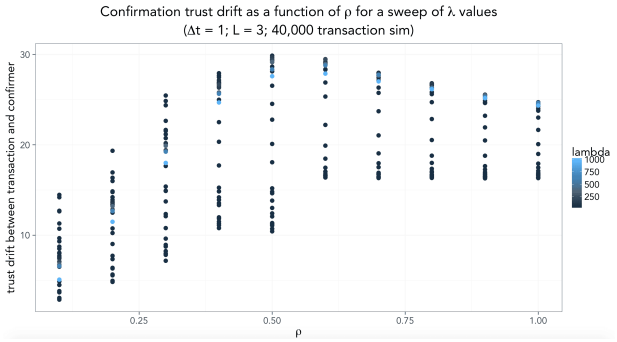


Figure 20: As can be see from this figure, the size of the threshold ρ has a definite affect on the trust drift between transaction and confirmation along the Trust Chain.