Decentralized Reputation System for Transaction Networks

Dept. of CIS - Senior Design 2014-2015*

Matthew Buechler
mbuech@seas.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

Manosai Eerabathini
manosaie@seas.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

Christopher Hockenbrocht
hchr@seas.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

Defu Wan
defuwan@seas.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

ABSTRACT

Peer-to-peer transaction networks such as Bitcoin have gained traction in the past few years as they are touted as an anonymous, safe, and secure way to make electronic payments with low transaction fees. Our goal is to improve these technologies by developing a reputation system that calculates reputation based on underlying network structure and allows users to look up and record histories of transaction outcomes. We believe that this will facilitate a more honest market environment. Beyond standard market transactions, developers will also be able to quickly build reputation components into their applications by utilizing our work. Our main contributions comes in two parts: a reputation algorithm called net flow convergence and a decentralized system that records transaction outcomes.

1. INTRODUCTION

There is a growing demand for more aspects of the modern Internet to be decentralized. Though Internet applications are built on top of decentralized protocols like TCP/IP and HTTP, a large portion of the application stack remains centralized. Much of the desire for more decentralized computing systems comes from concerns regarding mass surveillance over the web and limited access to information due to censorship. Only 25 years old, the Internet is amidst another radical shift in which open-source, decentralized technologies are gaining traction in many aspects of application development.

1.1 Motivation

Historically, there are many online industries where we can observe decentralized computing systems uprooting existing businesses. With the rise of Bitcoin technology, industries like online commerce have also seen decentralized alternatives emerge such as Silk Road and OpenBazaar. The success of these projects has resulted from the groundbreaking ability to transfer money online without requiring a third party to verify transactions. Bitcoin technology enables many such applications to support a variety of use cases in a decentralized manner.

Payment, e-commerce, identity, and file-sharing are just a few areas that have been touched upon with decentralized technologies. Reputation can be defined as the perception of a person's trustworthiness by others. In any interaction between two or more parties, reputation drives the ability for each party to trust one another and facilitate a successful experience. By nature, reputation systems have been centralized with application developers defining their own standard of reputation in the context of their own use case. For example, a site like Quora would come up with a proprietary algorithm to determine the reputation score of every Quora user but this reputation would not be the same as a user's reputation on a site like Yelp. The landscape of reputation systems is fragmented with every company owning a "secret sauce" that helps it make money by locking in users. Though this comes with strategic advantages for the companies, it requires that a consumer have multiple reputations stored across different services on the web. Reputation is inherently tied to identity and we believe there is a need to detach reputation from one specific company or site. Rather, consumers should be able to determine reputation in a distributed manner. We hope to build a decentralized reputation system that analyzes both the underlying transaction network structure and builds a history of transaction outcomes to eliminate the need for 3rd parties to define what constitutes good and bad reputation.

1.2 Advantages

This kind of reputation system will enable distinct advantages such as helping e-commerce businesses and Internet marketplaces develop faster by using a preexisting reputation system. Software developers will no longer need to encode their own reputation systems from scratch but rather can leverage ours to serve the same purpose. Additionally, reputation would be a global measure so that a user's reputation is not limited to one particular app or service. A decentralized reputation system will protect the consumer from being exploited by exterior forces such as governments or corporations. In general, using decentralized networks ensure the most ideal form of democracy, where the voice of the individual can be guaranteed through cryptographic security. Decentralizing the notion of reputation has not been done in the current technological landscape and there is great value in providing this ability to app developers.

2. RELATED WORK

Our work touches on a wide variety of research projects and academic papers. We studied other reputation algo-

*Advisor: Jonathan M. Smith(jms@cis.upenn.edu).

rithms to understand their strengths and weaknesses and use them as benchmarks. We built on top of decentralized application platforms. We integrated ideas from other areas of decentralized applications. One of them is identity and the OpenID project, which is an open standard and decentralized protocol that allows users to be authenticated by certain co-operating sites using a third party service[4]. In other words, it is a way for users to hold credentials without a third party having complete control over their use. Reputation is inherently linked to identity as it cannot exist without identity. When it comes to sharing data with peers in a network, Persona, a decentralized social network with user-defined privacy, is our guide[1]. Namecoin is the decentralized public registry for identities that utilizes this concept and we follow its implementation closely. This allows us to store already researched cryptographic tools, such as elliptic curve public key signatures which serve as identification tools. Attribute-based encryption is a relatively recent area of study in cryptography, which might allow such a sharing of information based on attribute or access levels and will allows users in a network to allow for fine-grained access control of their data[5]. Of course, being a somewhat new area of research, this might not be possible.

2.1 EigenTrust

EigenTrust is a reputation management algorithm that calculates reputation as a global trust value in peer to peer networks.[6]. The aim was for the algorithm to be applied in file upload networks like Gnutella so malicious users who were uploading inauthentic files could be weeded out based on their score.

The algorithm itself relies on building transitive trust by examining a peer's history of uploads. A node A trusts some peer node B so therefore node A will trust any nodes trusted by B. The algorithm works by having node A calculate a local trust value for all of its neighbors and then normalizing this value to prevent bad neighbors from colluding with their cohorts. All local trust values are then aggregated and a trust vector for the whole network is created.

This algorithm is very relevant to the premise of our network convergence algorithm. Flow of money and file uploads can be abstracted in a similar way. However, EigenTrust focuses on a reputation system that relies on user feedback, but we believe that monetary transactions can themselves provide a useful measure of reputation without any user input.

2.2 Centrality as a Proxy for Reputation Measurement

Measuring reputation using network properties and graph algorithms rather than proprietary algorithms has been attempted in a more general manner by members of the academic world in the past. Centrality measures in particular have been used to identify important vertices in graphs such as trustworthy users in a reputation system.[9]

According to graph research, eigenvector-based centrality measures have successfully been implemented to metrics like search results ranks. Specifically, in a reputation system where users can rate previous transactions with other, eigenvector, degree, betweenness, and closeness centralities. However, centrality computations are expensive, which makes them difficult to implement in a real-time transaction network with millions of nodes. What they do provide is a useful to score nodes in a network so that we can use them as a baseline comparison to our algorithm.

2.3 Bitcoin

A crucial technology for allowing peers in a network to exchange value in a trustless, electronic manner is Bitcoin[8]. Before this innovation, peers would have to use a non-electronic form of payment, cash, without a decentralized ledger that handled the accounting computation, or a centralized form of payment, credit cards, that would introduce a vital dependence on a third party and a point of failure in the decentralized system[8]. We hope to provide two ways to interact with Bitcoin–first providing an easy way for apps to interact and make transactions with other devices and second, through graph analysis, provide naive measures of reputation. Giving apps an abstracted way to interact with Bitcoin will alleviate the problems of interacting with a complicated infrastructure.

The Bitcoin transaction market is analyzed by the reputation algorithm to ascertain the issuance and verification of reputation in the network. The peer to peer computing network that will compute this reputation will be based on protocols that are extensions of the Bitcoin protocol technology.

2.4 Counterparty

Counterparty is a protocol extension built on top of the Bitcoin protocol. It allows for issuing assets and tokens as well for building decentralized exchanges and trustless betting[7]. While the processes are routed through the Bitcoin protocol, Counterparty uses an external native token issued through an initial proof-of-burn IPO that acts as a fuel for any action within the protocol. Using a native token other than Bitcoin allows the protocol more flexibility in its evolution while not detracting from the existing security benefits of the Bitcoin network in its current form.

2.5 Ethereum

Ethereum is an entirely new blockchain with different proof of work and mining security. Its main innovation is that it includes a Turing complete programming language for developing applications within the network[3]. These applications take the form of smart contracts, which are executable programs, and decentralized autonomous organizations, which are sets of executable programs that work in tandem to provide value in the economy. Ethereum includes a native token, ether, distributed in an initial IPO and then mined, that acts as "gas" necessary for executing smart contracts[3].

Our reputation system utilizes the security of the Bitcoin blockchain with the flexible of native token issuance and Turing complete smart contracts. Thus, either Ethereum or Counterparty can provide the basis for the smart contract based reputation system, and the Bitcoin transaction network can be used for reputation analysis.

3. SYSTEM MODEL

Our work is the addition to the set of Bitcoin network and suite of protocols a network overlay system for the management of distributed reputation. The primary goal of our protocol is twofold. First, two or more parties can utilize a representation of reputation among each other. Second, the system provides a computationally efficient way to recognize illegitimate behavior. Since the Bitcoin network can be

modeled as a large directed graph, we use local graph theoretic properties to model reputation and detect illegitimate behavior.

In this section, we first discuss the naive approach to solving the problem of weeding out network anomalies based on familiar graph algorithms. Following, we discuss practical obstructions to the working of this model and then we develop a model based on global and local graph properties and discuss practical ways to carry out reputation modeling and anomaly detection.

3.1 Bad Behavior and the Naive Approach

Reputation is essentially a way to judge, based on the gossip and hearsay around an entity, to determine their value, especially in transaction. Our model of reputation in the Bitcoin network is no different. We want to be able to give to any party of a transaction a measure of how well someone can be trusted in transaction. If transaction is to be based on reputation, then it is important for a party with the intent of cheating other parties to have a functional reputation in the network. In other reputation network models (e.g, eBay, Yelp, etc.), users can transact with themselves and rate themselves via other accounts to inflate their reputation in the network. However, in these networks, since there is a central authority that can moderate such things, through IP address monitoring, account monitoring, and other recorded information, the power of weeding out bad actors is a relatively simple task. In the Bitcoin network, however, much of this information will not exist in regard to a transaction.

Traditionally, reputation systems have had to be robust in design to handle many types of gaming efforts by malicious parties. Since our system is built on the Bitcoin network, it is decentralized such that users do not directly have the ability to rate or influence the score of others. Specifically, we do not encounter the challenge of trying to mediate transaction disputes between good and bad agents, issuing positive reputation as a third party.The only type of gaming we face is our presumed attack model where someone who attempts to inflate their reputation through fraudulent transactions. Essentially, that will mean that within our reputation network overlay, they will be making many transactions with other accounts they hold in order to boost the reputation of a single or a whole. To subvert this, they might attempt to trade unregistered transactions (that is, outside the reputation network overlay). However, to accommodate this, we simply search in the Bitcoin network underlying the reputation network.

First let's define a utility or scoring function for reputation

Definition A *utility* or *score* function is a function

$$S : G \times U^2 \to \mathbb{R}$$

where G is the network directed graph, and U is the space of users, such that we pass in two users. A score function is said to be *normalized* if

$$\tilde{S} : G \times U^2 \to (-1, 1)$$

The scoring function is a generalization reputation, and can be filled in by almost any function that takes in the given domain to determine reputation, however, not all scoring functions will have the same practical utility.

We combine the scoring function with a graph analysis to form the naive approach to computing the reputation between two parties. The naive idea is to find the shortest path between the two given parties and use a normalized scoring function to estimate the reputation value between these two parties. Using graph locality as the basis for rapport between parties gives a fantastic guarantee of the validity of a reputation score, *vis.*, the only way that a score will be non-zero is that there exists a path between both parties.

Equipped with a simple analytical tool and an algorithm (based on Dijkstra's shortest path), we can provide a metric that almost completely excludes bad actors from acquiring any kind of reputation from staged transactions and at the same time provide a common metric between any two given parties. The reason for this is that is simply that if a dishonest player is trying to inflate his reputation score without first having any real transactions, his network connectivity will be incredibly low or nonexistent as a result of transacting only with other accounts that he possesses (that also are assumed to have low connectivity).

Why we term this the "naive approach" is this: even though shortest path algorithm for a given graph runs in $O(|E|+|V|\log|V|)$, the Bitcoin network has two large drawbacks for this. First, edges can easily outweigh the number of nodes and multiple transactions in between two parties are expected to be very common. Secondly, the number of nodes in the Bitcoin network graph is on the order of 10^6 to , at the moment, and if our aim is to provide a means of trust so that even more users join the network, then the resulting runtime will be even more complex[10]. From a practical standpoint, even, there are considerable drawbacks. The expected look up time for any transaction is on the order of milliseconds, because of network latency and the fact that transactions are stored in a Merkel tree representation. Because of this, the sum of the necessary queries to complete the "naive" algorithm makes it impractical for use.

Using similar other graph theoretical techniques for forming reputation have similar drawbacks. Various centrality measures either take a global approach, such as eigenvector centrality in the form of a matrix, and for the given size of the graph, are impractical to compute. Other methods, such as closeness centrality like the naive method have indeterminate size in the global graph and in general are slow to compute.

3.2 Network Flow

It becomes quite obvious that we can't rely on this naive approach simply because, for our purposes, it is not efficient to compute. So, to obtain an effective score for reputation, we need to examine other graph invariants which might provide a reasonable means of a score calculation. The primary network invariant we wish to examine is network flow. We define network flow as follows:

Definition *Network outflow/inflow* is the sum complete set of edges, along with edge weight. We can define inflow for graph $G = (V, E)$, given $v(e)$ which is the weight or value of edge e, by the sum:

$$N_G(E) = \sum_{e \in E} v(e)$$

The outflow can be defined similarly. *Net inflow/outflow* can be defined by on a subset of edges, E', similarly as:

$$N_G(E') = \sum_{e \in E'} v(e)$$

and net flow is defined as the difference of net inflow with net outflow. Say that I is a set of inflow edges and O is a set of outflow edges, then, net flow is:

$$N_G(I, O) = \left(\sum_{i \in I} v(i)\right) - \left(\sum_{o \in O} v(o)\right)$$

First, note, it is not a useful distinction between inflow and outflow when examining the entire Bitcoin network graph. This is because, trivially, for some node the inflow edge will be the outflow edge of another. It is useful to note the following insight/lemma. We refer to the Bitcoin network as a "closed system" which is simply to say that the transactional currency Bitcoin cannot leave the global network.

LEMMA 3.1. *Net inflow is 0 for a closed system.*

This arises from the noted fact that inflow is the same as outflow when examining the entire graph. That is to say that I and O contain all the same edges and so the net inflow and net inflow have the same value.

Armed with these basic facts about a transactional network, we define a local property of a graph.

Definition Given a graph $G = (V, E)$ which is closed, and a total ordering of sets of edges E_i, the *net flow convergence rate* is the rate at which, for a given total ordering of inflow and outflow edges, net flow converges to the global net flow which is zero. We can state the convergence, given a total ordering of inflow and outflow edge sets I and O respectively, the total ordering being T and all edges up to the i^{th} edge being T_i by:

$$\lim_{i \to n} N_G(T_i)$$

and so the convergence rate is given by:

$$\lim_{i \to n} \frac{N_G(T_{i+1}) - N_G(T_i)}{|T_{i+1} - T_i|}$$

by looking at successive changes in rate for a given total ordering.

It is important to note that such a total ordering of sets exists, ordered by set inclusion, and can be chosen by Zorn's lemma from the set of all sets and so the convergence rate will be well defined for any given total ordering of inflow and outflow edge sets. Intuitively, however, this is quite clear.

3.3 Anomaly Detection Using Net Flow Rate Convergence

Our goal in looking at net flow rate convergence is to detect fraudulent behavior by looking at the flow of the Bitcoin in the network, at the same time as balancing the running time costs of carrying out such a procedure. The idea behind this is that locally, for any given node or subset of connected nodes, it is unlikely that there is a zero net flow. However, presumably the net flow for a strongly connected component of a graph is going to be close to zero. And likely, in the case of an adversary who is seeking to inflate his reputation through fraudulent trading activity the adversary will be contained within a strongly connected component and the net flow will be zero. The advantage is that this process can be done iteratively to any size of subgraph containing the adversary, making it relatively cheap to compute, with length of runtime only increasing the accuracy.

First we reintroduce a score function, with a different domain. We take G once again as input to the score but instead of both users in the user space U, we take certain side information T:

$$S : G \times U \times T \to \mathbb{R}$$

To formalize the procedure outlined for computing the net flow rate convergence, assuming no side information:

- Input: an set of inflow and outflow edges edges ordered by time for the adversary, or a subgraph of G containing the adversary. This set need only be as large as desired, but larger sets will give greater accuracy. This is done by doing a breadth-first search to desired depth.
- Iteratively sum the inflow and outflow edges, normalizing on time to find the net rate of flow at each step.
- Compute the rate of convergence with respect to ordered edges
- Output the rate of convergence.

We can use a practical statistical sampling of other network nodes, at run-time, to use as a control in comparison, predicated on the belief that most parties are not bad actors in the network. The statistical sampling will give a practical range of values for moderating judgement on the convergence rate. Alternatively, some convergence rate or range of rates can be assumed as the normative behavior before hand. The theoretical run-time of the algorithm is $O(n)$, where n is the number of edges chosen as the starting set to compute net flow. This will define a distribution, which then is used with the score to form a normalization factor.

In this illustrative example, we can see how net flows differ for two graphs.

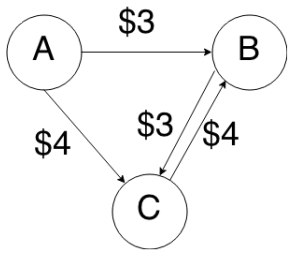

Figure 1: A bad actor and his cohorts

In Figure 1, we can see how A is a bad actor by engaging in fake transactions that only aim to boost his reputation without a transfer of money. His net flow quickly converges to zero from an initial outflow of $7.

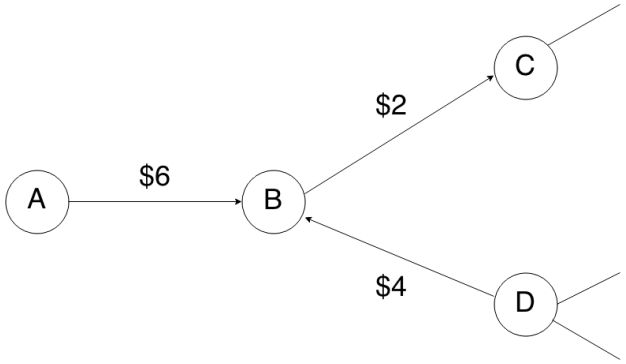

Figure 2: A good actor and his closest neighbors

In Figure 2, the good actor participates in normal market transactions where the net flow does not quickly converge to zero. It starts at $6 and has a net flow rate of -$2 after adding two more edges. We would need to look at more transactions in order to become confident of this low rate of convergence.

3.4 Other Attacks

While we chiefly address, with our model, attacks that are aimed at reputation inflation, there are other actions that will be unwanted and potentially damaging within the network. There is a second attack that we project will occur in the system, namely, that of defamation of others in the network. We introduce a different payoff model, in the game-theoretic sense, and use basic economic behavioral analysis to counteract this effect.

Specifically, the attack proceeds as follows. An adversary initiates a transaction with the verifier with an acceptable reputation score. The adversary's goal is to demarcate the reputation of the verifier by causing the transaction to result in a bad reputation marker. To counteract this, we enforce the property that transactions are never zero-sum.

First, we assume that both parties seek to minimize loss and maximize payoff. Recall that the game theoretic payoff can be described as a function:

$$A : \mathcal{A}_1 \times \mathcal{A}_2 \to R$$

where $\mathcal{A}_i$ is the action space of each party, and R is a domain of payoffs (think of it as a real interval $(-a, a)$). Here payoff will be a value that ultimately modifies the reputation score. In this game, each player has the option to play only one move, out of a space of two moves, either to fulfill the transaction or not, and we have two outcomes, call these a and $-a$. In the case both parties fulfill the transaction, the outcome will be a payoff a for both parties. In any other case (i.e., one or both parties fail to fulfill the contract) then the payoff is $-a$.

Assuming the we have rational players that will seek to maximize payoff, then it will be in the interest of both to sort out any issues while fulfilling the transaction, otherwise, both parties will take a loss. That is to say that the winning strategy for both is to fulfill the contract. In the case of our attack model, if the adversary were to repeatedly attempt to defame others, the adversary would also be taking repeated losses to his reputation–any player he cheated before would already know the outcome and any player thereafter would see the records of his transactions and hopefully be warned.

4. SYSTEM IMPLEMENTATION

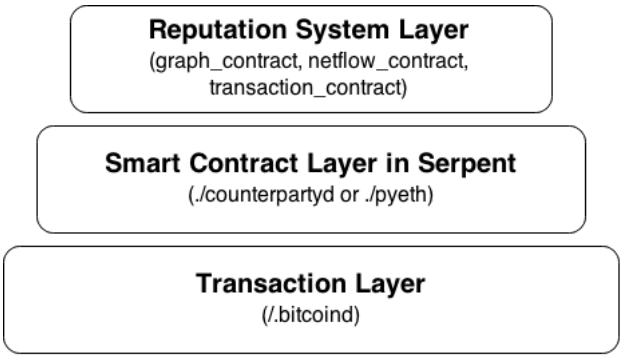

Figure 3: Major technologies utilized

The model assumptions and procedures described in Sec. 3 are implemented using smart contracts programmed in the Serpent language that can be run on the Ethereum or the Counterparty protocol. In the implementation, each account is represented by a single graph node in the Bitcoin network. Nodes, in the physical sense (which could possibly encompass more than one account), runs the system daemons: ./bitcoind, and either ./pyeth or ./counterblockd and ./counterpartyd. Node owners interact with the peer-to-peer network via commands that interact with the JSON-RPC application programming interfaces that the daemons are built with. Additionally, node owners provide valuable services for querying and obtaining information within the network.

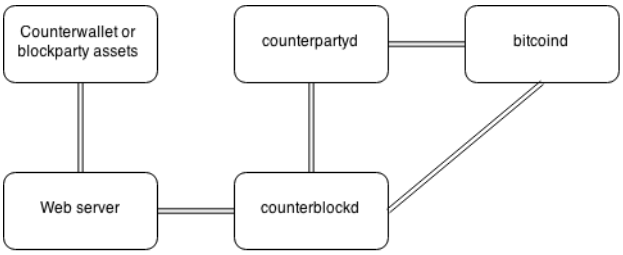

Figure 4: System daemons

4.1 Smart Contracts

The representation of reputation among a set of parties that is described in Figure 3 is implemented using smart contracts. Smart contracts provide a public, enforceable way to carry out transactions, and provide a public marker for the transaction outcome. Smart contracts are as secure as the Bitcoin blockchain itself, where the smart contract, along with its computational history, is stored. Moreover, because of the cryptographic primitives on which Bitcoin and smart contracts are built, we can guarantee unforgability and security for the parties involved.

Smart contracts, from a technical standpoint, are small programs that are compiled, registered with Counterparty the software into the Bitcoin blockchain or by the Ethereum software into the Ethereum blockchain, and then associated with a global hash value on a blockchain. They are executed according to a set of conditions defined within, for example, the default of a creditor or the non-shipment of an

item in an online transaction[3]. Smart contracts will be implemented in Serpent and can run natively in the Ethereum protocol or in a forked version of the Counterparty protocol. Serpent is a programming language that was developed with the principles of Python for creating smart contracts within the Ethereum protocol. Since our network is designed with some latency attached, we do not need to use the high-performance, low-level features available in programming languages like C++. Using a Python-based language allows us to develop quickly and intuitively.

Our practical implementation of smart contracts provides a common, prefabricated arrangement between two arbitrary parties, with standard functions to interact with the smart contract. The smart contract is, from our perspective, a standardized unit of trade–a transaction–which forms an edge in the reputation network overlay.

4.2 Reputation Scoring

We have made smart contracts the public record of transactions. On top of this, we draw from the system model, a means of computing score. We do so primarily through the implementation of calculation of net flow convergence rate and basing score on a combination of this rate and the success of the outcome of smart contracts/transactions of a party. Theoretically, however, since we provided a general notion of a scoring function and a normalized scoring function, it will also be possible for other scoring functions to be developed and implemented in its stead. Moreover, since we know, first the our score is one of many possible metrics, and second, that there will be flaws in our scoring, that by providing a method of extensibility for scoring metrics we can also secure the longevity of our reputation system on the blockchain.

More specifically, using our developed model, we outline a procedure for the computation of reputation as we will implement it:

- Compute net flow rate convergence for some ordered subset of inflow and outflow edge sets for the node in question. Let this be given as α. We use the value of α in the procedure for normalization.
- We compute an unnormalized score through comparison of reputation presented in the interactive proof phase and public records of those (and other) transactions. We do this by comparing the referenced transactions provided and the public records (optionally), and looking at the sum of the weights of the transactions.
- We use the convergence rate as a normalizing factor.

4.3 Using the Reputation System

In practice, the reputation system is composed of a series of smart contracts, or programs, run by developers who are running Bitcoin nodes. In order to run these programs, the developer must also have Counterparty or Ethereum installed on their Bitcoin node.

1. Run a Bitcoin node with Counterparty or Ethereum installed as well.

2. Engage with a counterparty for a peer-to-peer transaction

3. Run your own proprietary reputation algorithm, or use our net flow convergence smart contracts to calculate your perceived reputation of the counterparty in the transaction.

4. Based on the reputation score, decide whether to engage in the transaction.

5. If the transaction occurs, run the transaction contract to store the outcome, your numerical rating from 0 to 10 for the relative "success" of the transaction, in the Ethereum or Counterparty blockchains. By signing the smart contract with your unique digital signature, you ensure the validity of the specific entry in the distributed ledger.

6. If the transaction does not occur,i.e. due to the counterparty having a lower than expected reputation score, there is no need to run a smart contract to record any data related to the proposed transaction.

7. For future transactions, either continue to use your privately calculated reputations for counterparties or reference the history of transaction outcomes on the Ethereum or Counterparty blockchains.

4.4 Smart Contract Index

A transaction contract records the outcome of a transaction for the distributed ledger on the Etherum or Counterparty blockchains. Inputs are a transaction hash (the hash of the transaction in the Bitcoin blockchain), from (the counterparty from which the transaction originated), to (the counterparty to which the transaction was sent), outcome (the signing counterparty's rating, from 0-10, of the relative success of the transaction). Outputs are 1 (outcome successfully recorded), 0 (outcome unsuccessfully recorded).

A net flow contract runs the net flow convergence algorithm to calculate the reputations of counterparties. The inputs are the transaction data for the peer-to-peer network. The outputs are the rates of convergence of each node, the proxy for the reputation score.

A graph contract applies the net flow convergence algorithm, but can be extended to apply any reputation calculation algorithm, to the nodes in a peer-to-peer transaction network. The inputs are the transaction data for the peer-to-peer network in array format. The outputs are the rates of convergence of each node, the proxy for reputation score.

5. RESULTS

In order to determine the success of our decentralized reputation system, we have to test the accuracy and speed of our system. We first explored existing fast algorithms that may already be a good indicator for our definition of reputation, which focuses on detecting reputation manipulators with closed off proximate networks. We found some important insights but no suitable algorithm. Thus, we created net flow convergence and used Eigenvector Centrality and Degree Centrality as our two benchmarks for testing. These two algorithms are closest to the accuracy and performance that we are trying to achieve with net flow convergence. Some algorithms like Betweeness Centrality may provide more accuracy but run too slowly, which makes them completely infeasible to run in real-time for a large global transaction network. Other measures such as EigenTrust closely match our system's decentralized nature but directly utilizes user feedback and are not optimized for transactions with monetary exchanges. Although measuring the similarity between reputation algorithms may not seem like the best way to measure accuracy, it is the clearest evidence we have to show whether our algorithm works or not. It is impossible to obtain any true measure of reputation as reputation itself is inherently subjective. Rich datasets containing well

studied real reputations are also impossible to find, limiting our options.

We used an existing historical Bitcoin blockchain data set and toolkit [2] for our testing purposes. We processed the data set by grouping together addresses that provided inputs to the same transactions into singular entities. This allows us to analyze a more prototypical graph with a one-to-one relationship between nodes and entities and edges and transactions.

5.1 Strongly Connected Components

Before devising net flow convergence, we found it prudent to apply existing graph algorithms to see if they could be good indicators of reputation and provide us insights into the behavior of nodes on the blockchain. Specifically, we wanted to study strongly connected component behavior in the Bitcoin network which is defined as a maximal subgraph of a directed graph such that any given pair of vertices u and v have a directed path both from u to v and from v to u.[12] Using an iterative version of Tarjan's algorithm, we found all the strongly connected components in the Bitcoin network and examined their sizes to see if SCC behavior could be a preliminary indication of good or bad actors in the network.[11]

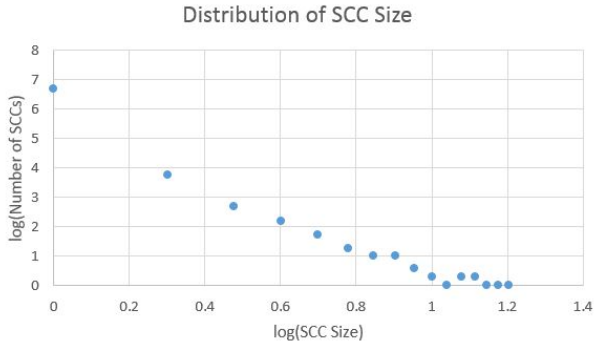

Figure 5: SCC distribution

From the above graph, we can see that the Bitcoin network has a power law distribution of strongly connected components. We observed a very low number of higher size SCCs with the majority of SCCs being in the range of size 0-3. This isn't the kind of behavior we expected to see in a transaction network as we hypothesized the flow of money would be contained in small networks of bigger sizes. Instead, the SCC output showed that most people in the Bitcoin network were transacting money one way - either sending or receiving but not doing both. This is probably due to the fact that the Bitcoin transaction network is immature and lacks liquid market flow as bitcoins are sometimes kept for speculative value. Thus, finding strongly connected components will be a poor predictor of reputation as there is not enough differentiation between users. Looking at the flow of money would provide better results.

5.2 Net Flow Convergence

The first algorithm we tested net flow convergence against is eigenvector centrality. For this measure, the centrality measure of the node is the sum of the centralities of its neighbors. Thus, big market players who transact a lot with other powerful players will have the highest reputation. In net flow convergence, these types of users will have the highest reputation as well. However, the eigenvector centrality will not differentiate users who attempt to generate fake transactions and regular users who transact with only a few people. The second algorithm is degree centrality. This is a simpler measure that calculates the fraction of nodes a degree is connected to. Well-connected users will have high reputation, but reputation can also be easily gamed by generating high numbers of fake accounts.

In order to compare the accuracy of the algorithms, we ran all 3 algorithms on the same data set and compared the reputation measures calculated. In order to do a fair comparison, we normalized each of the measures between 0-100 and removed outliers within the data set. We then ranked all the nodes by reputation and made pairwise comparisons between the 3 algorithms. We found that the similarity between eigenvector centrality and net flow convergence was around 95% when similarity is defined as difference of less than 10. The similarity between degree centrality and net flow convergence was around 82%. Thus, net flow convergence has relatively high similarity with the reputation measures and can be a good proxy for reputation.

We also found the distributions of the different algorithms. We can see how the difference in the algorithms manifest in the distributions. We believe that with each distinct group of nodes apparent in the distribution, the algorithm possesses additional power to distinguish reputation. Degree Centrality is the most basic and provides a summary view of the centrality of a node. Eigenvector Centrality adds more predictive power by stratifying those who are closely connected to others with high centrality and those who are not. Net flow convergence adds to eigenvector centrality by distinguishing actors who have high convergence rates, which we believe comprise primarily reputation manipulators and bad actors. Thus, our net flow convergence has more predictive power than other algorithms.

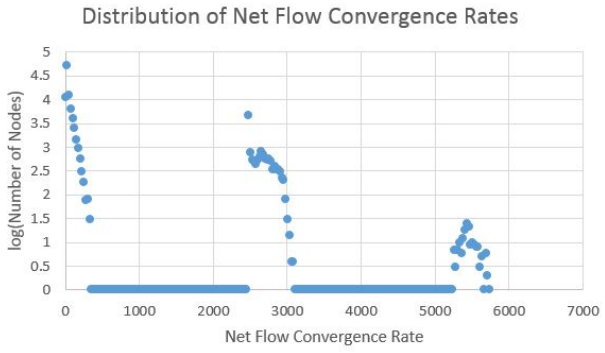

Figure 6: Net Flow Convergence Rate distribution

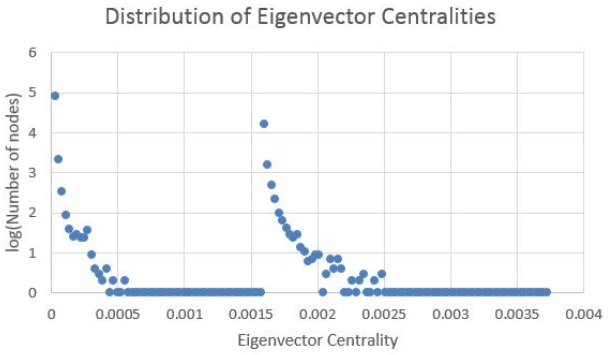

Figure 7: Eigenvector Centrality distribution

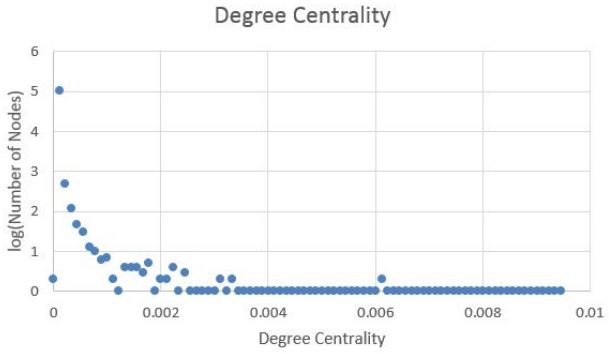

Figure 8: Degree Centrality distribution

5.3 System Performance

Beyond accuracy, we need to ensure that our decentralized reputation system runs at an acceptable performance. We analyzed the performance from a theoretical and practical comparison of runtimes of Eigenvalue Centrality, Degree Centrality, and Net flow. Below is a side-by-side comparison:

Eigenvector Centrality	$O(\|V\|^2)$ per iteration
Degree Centrality	$O(\|V\|)$
Net Flow Convergence	$O(n)$, where n is the cardinality of the flow edge set

Practically performance differences are much more apparent. When $|V|$ is on the order of the number of nodes in the Bitcoin network then solving any of the first three requires on the order of hours to compute. Net flow rate convergence on the other hand scales in the number of edges, which only requires a small portion of $|E|$, for which they are either inflow edges or outflow edges of the chosen subgraph.

6. FUTURE WORK

The work we have done has been both theoretical and at the system level. While this has laid an important foundation for continued innovation in the decentralized transaction and reputation measurement field, there is still work to be done. As it stands, our system can only be used by researchers and programmers. In order for the theoretical utility of the system to be be seen, the system needs to reach widespread use. At the very least, a majority of nodes of any one cryptocurrency network should be using the reputation system before we can verify its impact and usefulness in a practical sense. To reach this goal, an easy way to install the entire system along with its dependencies is necessary. Forking the code would also allow our system to work natively on multiple platforms instead of just Linux. Finally, we need to develop a visual user interface such that users who can not, or do not want, to program can still use the system and benefit from the values in the distributed ledgers that it provides.

In addition, there are still some unsolved problems in reputation systems that our approach did not address. Although we think that algorithms that measure network structure and annotations of specific nodes in the network can form a robust system, we do not make claims that it is the most accurate translation to real-world reputation. To verify real-world reputation, a future system could use web crawlers to link nodes in the transactional network, i.e. a cryptocurrency, with their real-world identity, or at the very least, their pseudo-identity on the Internet. This step would provide a data input that would verify the accuracy of the algorithm and ledger based system. Going along with this, some nodes in the transaction network may not only want their identities revealed to their counterparties, but also to the network at large. For example, a new website that provides a service, like exchange from cryptocurrency to fiat currency, may find that having a public identity engenders trust with potential customers. Such a system that linked public identities with nodes in the network would become much more accurate, while sacrificing the privacy of some of its users.

7. ETHICS

Though we have provided a working net flow convergence algorithm that suggests a new method of calculating reputation, we believe this is by no means an entirely universal standard. Our hope is that our algorithm will provide the baseline for calculating reputation in a transactional network where parties can rate each other. The net flow convergence algorithm will be available for use but it is up to the two transacting parties to agree upon the actual reputation algorithm they will use to assess the trustworthiness of each other. One issue that may arise after open sourcing our algorithm is that users might attempt to hold us liable for bad transactions that occur in a network even though our algorithm rated a user favorably. Thus, it is imperative that open sourcing our algorithm allows users to take our work and improve on our thought process to make reputation calculation more robust and secure.

While we are aware that making the algorithm public means certain users can game the system, we believe that in a transactional network- the cost associated with spending money will be a deterrent to a large number of malicious users. In the case that two transacting parties are not satisfied with net flow convergence as a reputation metric, they can agree upon any such standard calculation - either one that they come up with on their own or adopt from existing parties. Our hope is to provide the framework for calculating reputation in a decentralized manner using smart contracts and it is up to the users to decide which algorithm to ultimately run.

8. CONCLUSIONS

Our decentralized reputation system has two core innovations. First, we have developed a novel, decentralized algorithm for measuring reputation in a transaction network based on net flow rate convergence. Second, we have developed specialized programs, called smart contracts, that nodes running the Bitcoin and either the Ethereum or Counterparty software can use to calculate the reputation of the actors in the underlying Bitcoin transaction network and to record the outcomes of transactions between nodes and their counterparties on distributed ledgers lying on top of the Ethereum or Counterparty networks. While there are existing algorithms that calculate reputation, or centrality, based on network structure, we have combined this approach with transaction outcome values stored in a distributed ledger to build a more robust system for measuring and recording the reputations of counterparties in a transaction network. If this system can reach wide-scale adoption on top of an existing cryptocurrency network, we believe that it will substantially improve the usage of these networks by limiting the ability of bad actors to scam other nodes through malicious transactions. Even further, if companies begin to use our reputation system as a standard for measuring the trustworthiness of their users, than we will have significantly lowered the complexity of the current practice whereby companies build their own reputation systems that are not interoperable with each other and do not provide transparency about their measurement methods to their users.

9. REFERENCES

[1] Randy Baden. Persona: An online social network with user-defined privacy. http://ccr.sigcomm.org/online/files/p135.pdf.

[2] Ivan Brugere. Bitcoin transaction network dataset. http://compbio.cs.uic.edu/data/bitcoin/.

[3] Vitalik Buterin. Ethereum white paper: A next generation smart contract and decentralized application platform. https://www.ethereum.org/pdfs/EthereumWhitePaper.pdf.

[4] Dharmendra Choukse. Implementing new-age authentication techniques using openid for security automation. http://arxiv.org/ftp/arxiv/papers/1003/1003.1462.pdf.

[5] Vipul Goyal. Attribute-based encryption for fine-grained access control of encrypted data. https://eprint.iacr.org/2006/309.pdf.

[6] Sep Kamvar. Eigentrust. http://ilpubs.stanford.edu:8090/562/1/2002-56.pdf.

[7] Adam Krellenstein. The counterparty protocol. https://github.com/CounterpartyXCP/Counterparty.

[8] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf.

[9] EJ Plant. An empirical analysis of reputation effects and network. .

[10] Dorit Ron. Quantitative analysis of the full bitcoin transaction graph. http://eprint.iacr.org/2012/584.pdf.

[11] R.E. Tarjan. Depth-first search and linear graph algorithms. .

[12] Eric W. Weisstein. Strongly connected component. http://mathworld.wolfram.com/StronglyConnectedComponent.html.