Blockchain for the Common Good: A Digital Currency for Citizen Philanthropy and Social Entrepreneurship

Shweta Jain
Department of Mathematics & Computer Science
John Jay College of Criminal Justice
sjain@jjay.cuny.edu

Rahul Simha
Department of Computer Science
George Washington University
simha@gwu.edu

***Abstract*— We present a distributed ledger application for the world of citizen philanthropy and social entrepreneurship, with stakeholder incentives designed to increase *social good* through accountability, transparency, and flexibility. In this system, called Directed Cash, individual donors specify conditions (the "Directed" part) attached to their donation or investment ("Cash"), that are then efficiently paired with interested recipients or aggregators of recipients (charities, social entrepreneurs) using distributed consensus so that the intent and pairing are open while maintaining donor anonymity. Furthermore, Directed Cash flows both ways to promote accountability and transparency: after receipt, a validation flows backwards to return to the donor so that the donor receives a report of how their donation was spent. While some elements of the system borrow from existing cryptocurrency and blockchain technologies, we propose alternative incentives for distributed consensus that are better aligned with the application and promote social good through the stakeholders. This paper describes the goals and concepts, and a system design to achieve these goals, a primary feature of which is a simple SQL-like language to enable specifying conditions, pairing, aggregation and publicly-verifiable reporting.**

I. Introduction

Citizen donors or investors concerned about the impact of their contributions face a daunting gamut of choices. For charitable donations, for example, they must either trust a well-established charity by contributing to its general fund or spend time scouring crowd-funding sites for particular projects of interest. Not only is this donor-to-cause *pairing* problem time-consuming for donors, it forces charities to spend precious resources competing for eyeballs. But most importantly, there are few opportunities for ordinary donors to specify in much detail how they want their contributions spent, and nearly impossible to verify that their donations were spent as intended. These shortcomings raise a useful question: can a marketplace of philanthropic transactions be infused with the technology to improve efficiency and create a virtuous cycle that steers stakeholders towards the greater common good?

In this paper, we argue that the following combination of technologies is well-suited to addressing the above question. First, a digital currency (called Directed Cash here) that allows donors to attach conditions to a donation such as the type and location of aid, a cap on organizational overhead, and other such expressions of donor intent. Second, a pairing algorithm to connect donations to charities and causes. Third, a distributed ledger so that the system functions without a trusted third party. Fourth, a linked structure within the distributed ledger that allows the backward flow of transactions, connecting expenditures to the original donors so that donors may receive confirmation that their stated intent was satisfied. And finally, a high-level query language called DCL (Directed Cash Language) to facilitate clear expression of donor intent and recipient action. We explain why these components help us design a system that will be effective in reaching the overall goal of directing donations to causes that matter to donors.

A. Directed Cash (DC)

The new digital currency will be linked to government backed currency through a central bank. Unlike Bitcoin, our goal is not to gain independence from the government and nor do we want the currency itself to become an investment vehicle. Instead DC is merely a transactional convenience that allows a donor to *direct* spending to favorite causes and with conditions important to them.

B. Algorithmic Pairing

With algorithmic pairing, donations are not pledged to an organization but to a cause, aligning the donors' will with the needs of potential recipients. Analogous to program trading in stock markets, algorithmic pairing has the potential to mitigate human bias and result in socially more optimal allocation of resources. It can also help address the problem of *irrational herding* [1], in which popular projects tend to attract new investors even if other deserving projects are better suited to investor intent. Just as importantly, we organically establish an open verifiable system which enables donors motivated by public recognition [2] to tout their contribution but also to stimulate others into providing "matching contributions."

C. Distributed Ledger:

Because transparency promotes good behavior amongst the stakeholders, a publicly verifiable distributed ledger without a trusted third party appears to be a promising approach, with the advantage of spreading the verification burden, and facilitating a low barrier to entry (and exit): anyone with

the software can join or leave as they please. As long as there is sufficient interest in the application, we presume there will be sufficient honest nodes to keep the application functioning with integrity. However, a proof-of-work or similar blockchain based system is unnecessary for our application, and even computationally infeasible for the types of stakeholders in our realm of interest. Instead a general consensus scheme with alternative incentives is more suitable to address the somewhat more benign threat model, as explained later. Therefore, we propose a permissioned blockchain [3] that has the following assumptions: First, anyone can send a query to the chain to perform accounting, auditing and reporting tasks. Only bonafide banks can can generate transaction IDs that accompany donation pledges and claims. Only registered charities can send DCL statements containing information about a cause, and only registered vendors can send claims with invoices for payments for goods and services, both via their tax IDs. Anyone with a bank account or credit card can send donation pledge with attached conditions. Permitted parties consisting of vendors, charities, regulators, banks and donors participate in maintaining the blockchain.

D. Directed Cash Language:

Since a working system needs a compact and precise way to specify donations as well as causes, we describe a preliminary design for a high-level SQL-like language called Directed Cash Language (DCL). The language facilitates querying, donating with conditions, automated pairing, reporting, and verification. We aim to strike a balance between the full expressibility and high vulnerability of a Turing-complete language [4] and a fixed-format predetermined list of allowable donation categories that cannot anticipate future needs, and which restricts the creativity of projects and aggregators. We propose a compromise by using a carefully constrained non-Turing complete SQL like language. For databases, SQL combines enough expressibility for simple querying with simplicity in syntax, and most importantly, can be fully validated (and also optimized for execution speed) at runtime.

A constrained language is one part of reducing system complexity (which can be maliciously gamed [5]). We mention two others. First, actual transactions involving money in our system use the standard banking system; the Directed cash is simply a token that attaches a dollar amount to be drawn from an actual bank in accordance with a donor's intent. When a donation is spent by a recipient, a donor's reference to the funds is fulfilled by a central bank that holds the money in reserve. Second, the only algorithmic component is pairing, which is achieved by a deterministic open-source algorithm. Thus, the only potential security loopholes are of two kinds: the standard ones with any existing digital payment system, and any "gaming" of the incentives for the stakeholders. We address the latter in a forthcoming section.

E. Structure of paper

We begin by describing each of the system stakeholders or roles, and their needs in Section II. Then, in Section III we outline the system, followed by an explanation of the DCL language. We present our threat model in Section IV and conclusion in Section V. We have presented related work in all of sections as we describe the relevant components.

II. Stakeholders and their Needs

We begin by identifying the stakeholders:

- *Donors* are individuals or foundations who invest in a cause, or an organization that supports their cause.
- A *recipient* is the ultimate target of a donation, typically an individual. A *project* is a fixed-duration enterprise with a specified outcome; thus, a project can be a collection of recipients along with spending goals, or a collection of physical items (such as habitat materials).
- A *vendor* is a commercial entity that provides goods and services; examples include grocery stores, landlords, social workers. Most importantly, a vendor's legally-bound receipts form the basis of validating expenditures.
- An *aggregator* brings together donors, projects, or recipients under one theme or organization. For example, a *donor aggregator* can offer to match contributions from other donors, whereas a *recipient aggregator* more closely resembles a charity that features projects and recipients. Many foundations are often a mix of recipient and donor aggregator. In our system, such organizations would list themselves in both categories to offer both functionalities.
- A *regulator* represents a third party solely interested in ensuring the system is functioning as designed; such as a government oversight agency.

A. What donors seek

Economic analyses of donor motivations demonstrate a variety of intent [2] much of it deeply personal [6], arguing for a flexible system that accommodates all types of donor motivation. What is equally clear is the desire among donors to monitor and evaluate outcomes from their contributions [7], [8], [9], to find the right opportunity, and to use public announcements for recognition or to stimulate others [2]. Thus, for our purposes, we seek a design with the following features:

- Donors should be able to easily attach directions with their contributions, such as "I want my donations to go only to food costs, through charities that have lower than 10% overhead and who operate in my state."
- Donors should receive as complete an accounting as possible of how their money was spent.
- Donors should not have to spend much time searching for projects that meet their interest, nor to check how their contributions get spent even if they have full accounting.
- Beyond the transfer of funds at the time of purchasing Directed Cash, donors should have the option of maintaining anonymity. Yet, donors who wish public recognition or to stimulate others via matching contributions should be able to do so easily.

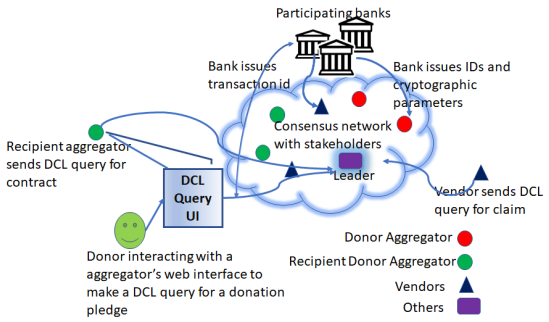

Fig. 1. Directed Cash Architecture Design

B. What aggregators seek

Aggregator's seek automation and efficiency.

- It should be easy for recipient aggregators to define projects and their attributes in ways that promote optimal pairing or large-scale aggregation.
- The system should enable effective stakeholders to stand out. Thus, attributes such as low-overhead or low-cost should be easily available as conditions.

C. Features for vendors

We assume that vendors are commercial entities ultimately interested in profit. However, we exploit the fact that indirect outcomes that lead to profit, such as advertising and reputation, are just as desirable. One of the most important system features is a reputational incentive for vendors to report on spending from recipients. This creates a virtuous cycle whereby responsive vendors are rewarded by having donated funds reach their products. Also, through the systems public verifiability, a vendor can acquire positive reputation that impacts its other customers, in much the same way that commerical entities that support social causes enhance their standing in society.

D. Other desirable system properties

The overriding goal of our proposed system is to incentivize all stakeholders to fulfill their role in making the system work, and in doing so, achieve better outcomes for all. Thus, we need mechanisms that align the common good with each stakeholder's goals to advance their own interests. Since aggregators and vendors seek recognition, we propose a points-based reward system to enable these stakeholders to rise in rankings by gaining points for participation (successfully constructing the next block) in distributed consensus. This, together with a mechanism to prevent domination by any particular stakeholder, helps us avoid the burden of proof-of-work blockchains. Finally, stakeholders should be able to make changes to the system through the consensus approach. This is particularly relevant for sanctifying nomenclature (what constitutes food) and rankings (of vendors).

III. How Directed Cash Works

We assume that donor aggregators or crowd-funding platforms will offer donors a user-friendly interface and will connect through the banking system to enable donors to both make contributions and specify their conditions in a convenient manner. Aggregators, vendors, regulators, and any member of the public may operate a *Directed Cash* server (DC-server) that participates in the computations of the system. The computational goals of the system (algorithmic pairing, validation, reporting, recognition) are achieved through the distributed computation that occurs amongst these servers. In particular, servers jointly agree through distributed consensus on the public ledger that includes anonymized donations, their pairings with recipients or projects, vendor validation, rankings, and reports sent back to donors. As mentioned earlier, this is a permissioned chain and hence all participants in the consensus process will need to be registered. Yet, participants are subject to malicious intent as well as external attack and hence protection needs to be built to counter associated security risks as described later in Section IV.

Because donors' conditions can greatly vary, there needs to be a simple way to let donors specify their conditions. To enable efficient and simple to use systems, we propose knowledge-representation systems using *ontologies* [10], that have predefined categories and hierarchies of accepted terms. We merely enable the use of any tree-structured ontology (such as those described in [11], [12]), along with a distributed mechanism to alter the ontology as needed.

A. The Directed Cash Language (DCL)

Rather than present a formal description of the language that is more suited to a tech report, we outline the language's features through examples typical for the stakeholders.

1) Identity and authentication: Because every DCL transaction (query and statement) will feature the writer's identity, and will itself have a unique ID, we first address the question of how stakeholders are identified and authenticated. The identity and authentication part of the system is not central to the contributions of the paper and thus we see any of several approaches as roughly equal, with competing advantages.

The first approach is to have the system require a single central bank or a digital payment system such as PayPal which is responsible for issuing both types of IDs: stakeholder identities as well as unique transaction (or DCL statement) serial numbers. Clearly, since our goal is to have our system tied to standard government-backed banking, it is convenient to have all stakeholders possess accounts linked to the digital payment system that generates IDs and transaction numbers. The advantages are simplicity and convenience in transaction ID generation, but its disadvantages include a single point of failure and reliance on a single source for trust. Note that computational efficiency is less of a concern because most DCL statements and queries feature a human-in-the-loop that automatically limits the rate at which transactions are likely to be generated. Thus, the maximum computational capacity needed is no worse than at any major ecommerce website.

The second approach is a slight variation: each stakeholder is affiliated to some bank or credit card account, and these are

at mutually-distrusting financial entities. These financial entities are responsible for issuing IDs. This has the advantages, again, of tying the proposed system to regular banking but, through multiple banks that audit each other, can address both issues: single point of failure and single source of trust.

In both the first and second approaches, we assume that banks will post to the distributed ledger every request for an ID. For donors that do not wish to be identified, the IDs anonymize the donor but can be "opened" via public-key cryptography in case of a legal challenge. We note that one compelling advantage of using standard banks is that they combine four useful features for the stakeholders: (1) the banks are already legally bound to operate under audit; (2) they have an incentive to be efficient since they benefit from the transaction flow; (3) the banks can serve as a policing function by blocking malicious parties from gaming the system; (4) and perhaps the most important one is that banks have the legal authority to authenticate stakeholders as individuals and as organizations based on their physical tax identifiers. This last one is critical in preventing denial-of-service or flooding attacks that create multiple fake identities.

2) Queries for donors and donor aggregators: We envision the following common scenarios for donors. Donors should have the ability to: (1) search existing proposals, projects and recipients; (2) make isolated donations with constraints and reporting requirements; (3) issue continued donations spread over time; (4) stimulate or join matched contributions; (5) rate a recipient aggregator; (6) receive reports on their spending; (7) perform validation by operating a server in the ecosystem.

Let us walk through the steps with a simple example of a donor that wishes to make a monthly donation for a year for either food or clothing. The donor first uses their account with the central digital bank to convert actual currency into a token that can be sent with the DCL statement into the public ledger. The DCL statement includes the user's ID and is carried inside a transaction message described later in Section IV. The message header contains various IDs that tie the DCL query to the originator and the bank that created the transaction IDs.

```
FROM ANON (USERID=VO8t05RV9NrZYtrT1wL4QLcjp)
TIMESTAMP 2018-02-12T20:53:00+00:00:00001 DONATE $100
MONTHLY 3-10-17 TO 3-10-18 DECIDE FCFS WHERE (SCHEMA=1.1)
AND ((CATEGORY=food) OR (CATEGORY=clothing)) REPORT ONCE
```

Here, we see that a donor who wishes to be anonymous is donating $100 per month over the period of a year, with the stated aim of allocating the funds on a first-come-first-served (FCFS) basis in the categories of food and clothing. The schema refers to the product ontology that is being applied (for which options include `schema.org` and other such ontologies [11], [10], [12]). The donor also seeks a single report at the end of spending. Next, we see that the bank issues both a timestamp that's sufficient to serialize such statements, and an encrypted token to be used as the reference so that later, when donation is paired with a charity, the charity can use the token to reference this donation. Finally, the bank also includes a signed digest to detect whether the statement itself has been tampered with. The statement is then written to the public distributed ledger, awaiting further action when a node in the system performs pairing as part of writing the next block in the blockchain.

Eventually, when algorithmic pairing occurs, such "buy orders" are paired with projects or recipients and eventually spent, with a report that aggregates over time using the token reference number so that it can be returned to the donor.

A simpler version allows a donor or a representative to query the system for existing projects that might suit their interests, as in: Query existing proposals:

```
FROM PSEUDONYM NightHawk (ID=...) FIND PROJECTS WHERE
(SCHEMA=1.1) AND (CATEGORY=food) OR (CATEGORY=clothing)
```

Since these carry no obligation, no security measures are necessary and hence are not wrapped into a transaction message. A donor may wish to change the `DECIDE` clause by seeking to match existing contributions: `DECIDE MATCH name=...` names a particular pseudonym whereas leaving the name out results in the first match that meets all the criteria. Thus, a matching contribution merely adds the requirement that another contribution has already been made of at least the same amount. And, a donor can issue a rating, as in:

```
FROM ANON (ID=...) RATE Bright Ray Charity AS 4.3/5
```

Finally, a donation statement whose conditions are not satisfied or which has expired will eventually be returned to the donor, at which time they could reconsider some of the conditions.

3) Queries from recipient aggregators: One of the most important features in enabling algorithmic pairing is to allow recipient aggregators to create a statement of intent – that is, to define a project. Here is an example:

```
FROM Bright Ray Charity (ID=...) DEFINE PROJECT Food Drive
2018 GOAL $10,000 WHERE (SCHEMA=1.1) AND (CATEGORY=food)
```

Clearly, this is a project that can be paired with the donation example from the previous section. Here is an example of issuing a call to vendors for a particular food item:

```
FROM Bright Ray Charity (ID=...) PROJECT Food Drive 2018
VENDOR RFP $1000 WHERE (SCHEMA=1.1) AND CATEGORY=food.canned.soup)
DESCRIPTION URL http://brc/fooddrive2018.html
```

Note the use of a hierarchical ontology (in Schema 1.1). Recipient aggregators may also query existing donations in the ledger, rate vendors, and are expected to report on their spending as in:

```
FROM Bright Ray Charity (ID=...) EXPENSE=$550.00 TO VENDOR
Fresh House Grocery WHERE (SCHEMA=1.1) AND PROJECT=Food
Drive 2018 AND (CATEGORY=food.canned.soup)
```

Once a vendor verifies this by posting a receipt to the ledger, such a claim can be verified. The more verified claims that a charity has, the higher its rating.

4) Queries from vendors: A vendor needs to be able to query existing calls, make bids, and rate recipient aggregators. As an example of making a bid, consider:

```
FROM Fresh House Grocery (ID=...) BID $550 TO Bright Ray
Charity WHERE (SCHEMA=1.1) AND PROJECT=Food Drive 2018 AND
(CATEGORY=food.canned.soup) DESC URL
http://fhg/search.html?prodid=98735
```

Then, it is up to a recipient to accept the bid and complete the

transaction. A completed transaction then result in the vendor posting the receipt on the ledger.

B. Distributed-Ledger and Consensus

A ledger entry is merely a DCL statement secured by standard cryptographic means into a transaction. Once a transaction containing a DCL statement is in a resolved block of the ledger it will also have a serial number so that all resolved DCL statements are uniquely ordered, as are the larger blocks in which they are contained. This is the purpose of operating a blockchain: to achieve distributed consensus by offering participation incentives and yet prevent domination by any one player.

Any registered stakeholder (with an account with a central bank) can operate a computational node in the permissioned distributed ledger system. We assume that regulators already have an incentive to become nodes. We describe in the following section an incentive mechanism for aggregators and vendors to participate.

Since DCL statements (ledger entries) flow into various nodes in the system, there needs to be a way to organize the serialization and resolution of blocks into the blockchain. Our approach is to combine a leader-based consensus mechanism (such as RAFT [13]), with incentives to spread the leadership capability and prevent any one node from dominating as leader. In particular:

- Each node in the consensus system receives ledger entries from anyone and can generate ledger entries.
- A node broadcasts its ledger entries to the current consensus leader who follows an acknowledgement based protocol to enable fault tolerance (see Section IV).
- A leader is elected as in the RAFT algorithm [13] and any secure voting process such as [14][15] can be utilized.
- Only a leader has the authority to resolve the next block through a sequence of computations to: (1) determine the potential ledger entries in the current block; (2) compute pairing if needed; (3) hash and sign the block; (4) verify P previous blocks (where P is a system parameter); (5) broadcast the current block.
- The resolution of the next block triggers a new leader election (to prevent domination). Also, the current leader is ineligible to become the next leader in the next L rounds of leader elections. Here, L is a parameter of the system.
- Leader elections are also triggered by timeouts so that no leader can deny service by merely blocking further action.
- After the next leader is elected, the previous leader finalizes the block by adding a signed ledger entry that contains the next leader's ID as well as the votes affirming the election of the leader.
- A leader's reward is based on the number of ledger entries processed (to maximize processing as many entries as possible), weighted by the age of each transactions (to prevent an entry from being left behind).

When a leader resolves a block, it is awarded a participation point. These participation points accumulate as an incentive. We propose two alternatives. One is for the central bank to charge a tiny fee for each actual transaction and to let that fee accumulate into a fund used to reward aggregators or vendors who have accumulated sufficient points. Another approach is to allow the points generated to be incorporated into the ratings of aggregators and vendors so that they compete for visibility and recognition. In order the further incentivize leaders, the reward for making a block is in proportion to the number of ledger entries in the block.

C. Algorithmic pairing

The purpose of algorithmic pairing is to find a suitable project or recipient according to the donor's criteria. A secondary goal is to ensure some degree of spread amongst multiple competing projects that satisfy the criteria. Finally, we also wish to design a system that is robust to gaming or attacks.

Our approach is to use a variation of the classic Gale-Shapley algorithm for the stable marriage problem [16]. In the most basic version of the problem, there are M type-A items to be paired with N type-B items (most often $M = N$), where each item ranks every member of the other type. Under certain conditions, a simple algorithm provides a stable "marriage." Much research has gone into variations as well as attempts to make the resulting allocation fair to both sides [16]. In our case, we can choose the donations as type A, and potential recipients or projects as type B. However, we need a variation that must account for the following:

- In our case, preferences are often binary: either a project satisfies the criteria or not. Thus, to decide amongst equivalently satisfying projects, the projects can be randomized before selecting the next one. For transparency, the method of randomization and the particular choice made would need be made public and written back into the ledger. A simple way to achieve this is to use a known algorithm (such as a Lehmer generator) where the seed uses a combination of the timestamp of the last ledger entry in the most recently resolved block and that block's hash. This way, the sequence is reasonably unpredictable yet verifiable.
- It will almost never be the case that M=N and so some items will not receive a pairing and will have to "wait." Also, it is equally probable that the current set of donations are not sufficient to meet the goal of a particular project, and thus, the difference will remain to be fulfilled. Thus, the pairing algorithm will need to go as far back as needed in the blockchain to find incompletely-satisfied projects.
- It should be possible to prioritize projects that are either very early or have received very little so that no project remains unlucky for too long or is totally unfunded.
- Finally, donations can also specify that rankings should be used, in which case, there is a natural preference order. However, to prevent a few aggregators from dominating,

some randomness can be inserted to enable spreading the wealth.

D. Reporting and donor interfaces

One of the computational tasks of a leader is to aggregate expenditure data for donors. Here, the algorithm is simple: examine posted receipts, find the originating transactions and post reports to the ledger, one for each donor whose funds were used towards a receipt. Thus, at any given time, a donor might have a list of such report entries awaiting them. Donors can retrieve reports by signing on to the system through which they originally sent the pledge.

E. Validation and challenge

A third computational task for any leader, beyond pairing and reporting, is to cross-check the entries written by other leaders. This feature is at the heart of public verifiability, so that any leader who games the system or even makes a mistake is identified, followed by appropriate corrective action.

We propose that each leader checks P randomly selected previous blocks and includes a ledger entry in the block it generates stating the block IDs that were checked. To reduce the burden on checking and to ensure all blocks are checked by a majority of leaders, we propose that any block that has been checked by sufficient number of leaders is considered "retired" and does not need to be checked anymore. What happens when an inconsistency is discovered? It is tempting to try and design an automated way to "undo" a resolved block and re-resolve the block following an inconsistency. However, this may lead to catastrophic consequences (actual expenditures) that are difficult to roll back. Instead, we propose that any inconsistency is reported in the next block through a ledger entry and it is escalated to human intervention, through a steering committee or governing board. If the penalty is severe enough (denial of participation for a year, for example), there is little incentive for legitimate entities to cheat.

IV. Security

We present our threat model and specific details that allow for building a system that is robust to the given threat.

1) An adversary might remove or change a donation
2) A donor might go back on a pledge
3) A bank might withhold funds
4) A vendor might use the same receipt to claim money from multiple donations (double claim).
5) Denial of service by the consensus leader i.e., suppressing a DCL query or activity.
6) Duplicate malicious leader sending spoofed blocks, or deploying multiple servers to win an election (Sybil attack).

In order to thwart the above threats, we define the system's transaction design and the process of maintaining consistency. We will reflect back on the threats to explain how the design removes the risks posed by the above threats.

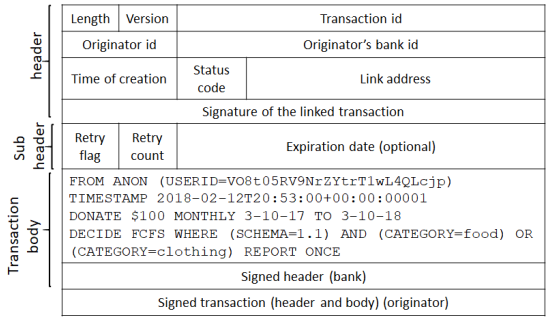

Fig. 2. Transaction structure

A. Transaction structure

We propose the following structure for a transaction in our system, shown in Figure 2. First each transaction has a header that consists of the header length field, version number, transaction id, originator id, originator's bank id, time of creation, status code, link address and the signature of the transaction that this transaction links. There is a sub-header containing retry flag, a retry count and an optional expiration date to deal with retransmission of a ledger entry when needed for fault tolerance. Following this, the transaction body consist of the contents of the DCL statement. Finally, the transaction header signed by the bank and the header and body signed by the originator are attached to the transaction. We represent a transaction with the following notation: $T_a(h_i, tid_i)$ where a is the originator id and tid_i is the i^{th} transaction id and h_i is the header associated with the transaction. The digitally signed hash of the transaction is represented as $E(K_a, H(T_a(h_i, tid_i)))$ and the header alone is signed as $E(K_b, H(h_i))$ by the originator's bank. Having this structure helps us maintain the data that is needed to thwart each one of the threats mentioned above.

B. Types of Transactions

We propose four types of transactions, Donation Pending (P), Donation Expired (E), Donation under Contract (C) and Donation Claimed (D). When a donor makes a pledge, the Donation Pending transaction is created which appears as a pending payment against the credit card or bank account. These funds remain in a pending state until the donation expires or Donation Claimed transactions are created. The link address field in the Donation Pending transaction is set to the bank's transaction authorization code and the signature is the signed hash of the authorization code. This creates a link between the digital currency and an actual bank authorization binding both the donor and the bank into honoring the pledge. The header field 'link address' in the Donation under Contract transaction is set to the transaction id in the Donation Pending transaction and its signature filed contains the signed donation pending transaction. Similarly the Donation Claimed transaction is linked with the Donation under Contract transaction and the Donation Expired transaction is linked to the Donation Pending transaction. This information ties each transaction

directly and immutably with the original donation and each transaction header. Figure 3 shows the above workflow.

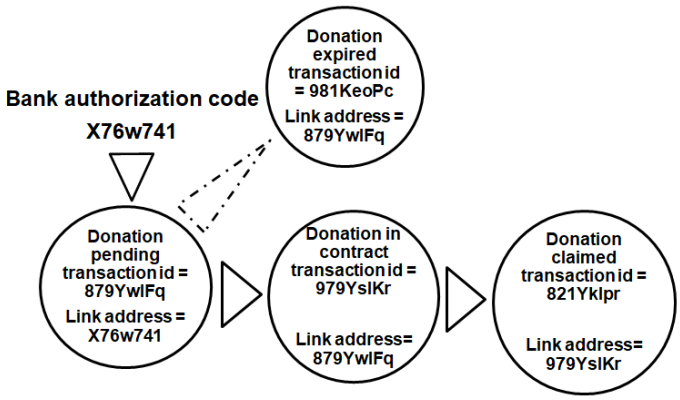

Fig. 3. Transaction life-cycle and built-in backward accounting

C. Transaction chaining and consensus

We propose a chained Merkel root based ledger structure similar to that of Bitcoin. Thus lets say a transaction $T_a(h_i, status, tid_i)$ is generated by an originator a. Then $T_a(h_i, status, tid_i)||E(K_b, H(h_i))||$ $E(K_a, H(T_a(h_i, status, tid_i)))$ is sent as a ledger entry to the consensus network. A participant q in the blockchain consensus process, sends the ledger entry to the current consensus leader. The leader constructs a block as a Merkle tree composed of all ledger entries that were received during a block generation period δ. In addition, the Merkle tree of block B_i consists of the Merkle root of B_{i-1} signed by the creator r of block $B(i-1)$. Thus $B(i-1)||E(K_r, H(B(i-1)))$ becomes the leftmost leaf in the block $B(i)$. We do not believe that a proof-of-work like blockchain is sustainable in this scenario. Therefore, as explained earlier, we use a leader-based distributed consensus algorithm with sufficient additions to ensure proper functioning in the presence of malicious participants in the system. Participants in the consensus system might be charities, aggregators and vendors who have a vested interest in maintaining an honest and verifiable chain.

D. Non-repudiation of transactions

Each transaction of type P and E is signed by the private key of the donor and the donor's bank as explained before and shown in Figure 2. The aggregator or charity signs transactions of type C and the vendor signs the Donation Claimed transaction. As in the case of donor generated transactions, their banks sign the headers of these transactions to prevent misuse and spoofing of the transaction IDs. These signatures help us achieve non-repudiation for each stakeholder. The charity and donor may still keep paper trail or electronic evidence of their physical transactions for regular audit and monitoring.

The transactions are structured as components of a linked data structure due to the inclusion of the address of the previous transaction. Thus a claim transaction can be linked to only one donation in contract transaction. If a project cannot be fulfilled by a single donation, it will need to be broken down into multiple donation contract transactions to bind with different donation pending transactions (similar to real life partial payments when multiple methods of payment are used to pay for an expensive purchase).

Due to the linked structure, an independent validator can easily create an accounting balance sheet of any donation and all claims against it to verify how a donation has been spent.

E. Defending against threats

The threat of tampering with any transaction is addressed due to inherent integrity properties of the blockchain. Each block consists of the signed Merkle root of the previous block, which makes the chain self-certifiable. The block creator signs the Merkle root of each block it creates. Thus any inconsistencies can be traced back to the participant who constructed a bad block provided the participants in the consensus system remain honest. In order to ensure that a malicious leader did not go back to change or remove ledger entries, each newly elected leader is required to check P previous blocks. The leader certifies this by producing the first ledger entry in the merkle root which contains the IDs and signed digests of the P blocks that the current leader has checked. Any block that has been checked by a majority of consensus participants during their tenures as leaders can be "retired" and hence do not need to be checked. Each Merkle root also contains the ID of the leader elected to compute the next block along with the votes that affirmed the leader. This removes the problem of spoofed blocks as the next leader can easily detect the presence of any spoofed block and the network can differentiate between genuine and spoofed blocks since the signature in the spoofed block is not expected to be that of the leader whose ID is embedded in the previous block.

The threat of donor's going back on a pledge is solved through standard banking processes as donors authorize the funds they are committing through their bank or credit-card accounts. In addition, source non-repudiation is achieved due to the donor's and their bank's digital signature accompanying a Donation Pending transaction. The Donation Pending transaction includes the ID of the bank that authorizes the payment as well as the transaction ID that the bank creates. Thus a pending donation can be traced back to the original bank that authorized the transaction and issued a transaction ID. Therefore, the bank is bound by federal regulations and must honor the commitment to release funds when the transaction enters the claimed status.

The vendor signs a Donation Claim transaction and the transaction can be traced back to the original Donation Pending transaction by following the the link address field of the transaction header. The body of the claimed transaction itself contains the receipt of the delivery of the goods signed by the charity. The recipient bank generates the transaction ID for the Donation Claimed transaction. A vendor would not be able to post fraudulent claims unless they collude with a charity. Through our openly verifiable system, such collusion

can be easily detected by someone who knows the ground truth in these transactions, which makes it easy to alert the appropriate law enforcement authority.

F. Denial of service by the consensus participants and leader

We have earlier proposed that the Raft Consensus protocol can be used as an alternative to the expensive proof-based consensus used in crypto-currencies. Leader election in Raft is subject to Sybil attack where a node makes several copies of itself to increase its odds of election. However, since the leader election can itself be based on unique IDs of candidates and voters, even if a node mounts a Sybil attack, only one of the Sybil nodes will succeed in posing as a candidate and only one vote can be casted by a voter ID. Furthermore, a leader must wait L rounds, which also mitigates the Sybil attack. Denial of service by suppressing DCL queries is another threat that is inherent to a leader-based consensus mechanism. A standard and perhaps the only way to defend against this attack is to retry a failed message. We will also follow a protocol based approach to this threat which we illustrate in the following steps. Note that we present defense against malicious leaders only:

1) A donor or stakeholder S sends a DCL query to any other consensus participant C_j.
2) C_j must generate a ledger entry and pass it to the leader L_i. L_i must acknowledge receipt of the entry with a return message that contains sufficient information and cryptographic protection so that it can be uniquely tied to the original ledger entry.
3) C_j logs the acknowledgment as proof that the query was received by L_i, waits for a block generation period and then queries the blockchain to ensure that the transaction has been entered in the block.
4) If C_j does not receive an acknowledgement, it increments the retry count field in the transaction header, sets the retry flag to true, and sends the transaction again.
5) The above step may be repeated until a maximum number of retries have failed or a successful acknowledgement is received.
6) If C_j fails to receive the acknowledgement after the maximum number of attempts, it can send the ledger entry to the next leader. At this point, retry flag is set to 1 and retry count is reset to 0.
7) If after receiving the acknowledgement, C_j does not find the transaction in the next block that was created, it would follow up by sending the acknowledgement and the transaction to the next leader. If the donor or stakeholder fails to find the DCL in the block, it can try to send the query to another consensus participant.

The above protocol steps enable senders by generating sufficient proofs that their transactions were acknowledged. They can be extended to allow the consensus participants to build a blacklist of malicious or faulty nodes in the network.

V. Discussion and Conclusions

This paper attempts to explain how blockchain technology can be adapted for the realm of charitable donations and social entrepreneurship. Instead of a proof-of-work approach, we use a well known leader based consensus scheme from distributed computing, with the addition of incentives to participate in block creation. We propose algorithmic pairing to give donors more control over how their money is used and to ease the burden of searching for best matches to their intentions. It has the potential to reduce the human bias noted by others, gives better access to funds to smaller organizations and NGOs, and reduces the outsize power currently held by foundations and big charitable organizations. Most importantly, the features of the system promote social good through incentives for transparency, accountability and participation.

Acknowledgements: Some of the ideas in this paper first appeared in a whitepaper written by the lead author [17]. Some of this work is based on the work supported by the National Science Foundation under Grant No. 1742919 and PSC-CUNY Grant No. 60814-00 48

References

[1] Inés Alegre and Melina Moleskis. Crowdfunding: A review and research agenda. 2016.
[2] Lise Vesterlund. Why do people give. *The nonprofit sector: A research handbook*, 2:168–190, 2006.
[3] Christian Cachin. Architecture of the hyperledger blockchain fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*, 2016.
[4] Pablo Lamela Seijas, Simon J Thompson, and Darryl McAdams. Scripting smart contracts for distributed ledger technology. *IACR Cryptology ePrint Archive*, 2016:1156, 2016.
[5] Quinn DuPont. Experiments in algorithmic governance: A history and ethnography of the dao, a failed decentralized autonomous organization. *Bitcoin and Beyond: Cryptocurrencies, Blockchains and Global Governance. Routledge*, 2017.
[6] R.Dietz and B.Keller. Donor loyalty study: A deep dive into donor behaviors and attitudes. *Nonprofit Times*, 2016.
[7] A.Butcher. What donors want when it comes to communication. *Nonprofit Quarterly*, September 2015.
[8] Root Cause. Informed giving: Information donors want and how nonprofits can provide it. *Root Cause, Boston, MA. http://www. rootcause. org/docs/Blog/Informed_Giving_Full_Report. pdf*, 2013.
[9] Simon McGrath. Giving donors good reason to give again. *International Journal of Nonprofit and Voluntary Sector Marketing*, 2(2):125–135, 1997.
[10] T.D. Wang, B. Parsia, and J. Hendler. A survey of the web ontology landscape. *The Semantic Web - ISWC 2006.Lecture Notes in Computer Science.*, 2006.
[11] Dan Brickley and RV Guha. Rdf schema. *W3C*, 2014.
[12] Martin Hepp. Goodrelations: An ontology for describing products and services offers on the web. In *Proceedings of the 16th International Conference on Knowledge Engineering and Knowledge Management (EKAW2008), Acitrezza, Italy*, 2008.
[13] Diego Ongaro and John K Ousterhout. In search of an understandable consensus algorithm. In *USENIX Annual Technical Conference*, pages 305–319, 2014.
[14] Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Towards secure and scalable computation in peer-to-peer networks. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 87–98. IEEE, 2006.
[15] Qi Dong and Donggang Liu. Resilient cluster leader election for wireless sensor networks. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON'09. 6th Annual IEEE Communications Society Conference on*, pages 1–9. IEEE, 2009.
[16] Kazuo Iwama and Shuichi Miyazaki. A survey of the stable marriage problem and its variants. In *Informatics Education and Research for Knowledge-Circulating Society, 2008. ICKS 2008. International Conference on*, pages 131–136. IEEE, 2008.
[17] Rahul Simha. Directed cash: A distributed ledger for charitable donations. *Whitepaper, Department of Computer Science, George Washington University*, 2016.