Directed Acyclic Graphs, Data-Flow, And Distributed Ledgers

Directed Acyclic Graphs, Data-Flow, And Distributed Ledgers

Enis Olgac[1]

Wouldn't you be surprised, if you find every single issue of all overnight published newspapers in your postbox early in the morning, including your favorite one subscribed recently. And anonymous readers build a consensus, whether the issue in your hand is manipulated or not. At day one of your subscription, archives are delivered, too. The analogy is not farfetched, if you consider the published material on "Blockchain" [1], the "distributed ledger technology", the backbone of Bitcoin. In reality, Blockchain is a journal, a daybook of economic transactions, cloned on every participating node in the network. It is for sure immutable, and extensible. It is neither distributed (its copies are broadcasted as is), nor it is a "ledger". Inspired by Blockchain, other approaches like "Tangle" (the backbone of IOTA cryptocurrency [2]) even link unrelated (from data-flow point of view) transactions.

The system proposed in this paper is a value transfer system with its distributed, decentralized, immutable, and extensible ledgers. It leans on the principles of real existing transaction recording systems (double-entry bookkeeping), which make use of ledgers. These systems are distributed. They are decentralized, if the artificial requirement for trusted parties, who are privileged to assign Ids to individual accounts, are removed. They are immutable, and for sure extensible. Fraud is only possible at the cost of very heavy crime. The introduced system includes a process to prove ownership of accounts, making a central authority superfluous. Immutability is guaranteed by allowing "append" as the only operation to modify a ledger.

INTRODUCTION

Ledger is a term related to general accounting and is defined as

> "Collection of (an entire group of similar) accounts in double-entry bookkeeping. Also called book of final entry, a ledger records classified and summarized financial information from journals (the 'books of first entry') as debits and credits, and shows their current balances … In computerized systems, it consists of interlinked digital files, but fallows the same accounting principles as the manual system." *Source: http://www.businessdictionary.com/definition/ledger.html*.

furthermore double-entry bookkeeping is defined as

> "System of keeping accounting records that recognizes the dual nature (source and disposition) of every financial transaction expressed by the basic accounting equation … In this system, every transaction is entered twice in the account books; first, to record a change in the assets' side (called a 'debit') and second, to mirror that change in the equities' side (called a 'credit'). If all entries are recorded accurately, the account books will 'balance' because the total of debit entries will equal the total of credit entries." *Source: http://www.businessdictionary.com/definition/double-entry-bookkeeping.html*

Only those systems which comply with these definitions can be called "Ledger Technology".

Decentralized Ledgers

Fraud, especially attempts to "Double Spend", i.e. creating a transfer order without having enough assets to support the intended value, is unavoidable. In order to identify such improper use, Blockchain as well as

[1] MailTo: enis@digraphs.info

Tangle rely on accessibility of a local copy of their data-structure on every node. This approach has two intrinsic disadvantages:

1. Enormous increase in storage requirement, as the number of participants and transactions are increasing.
2. Keeping the copies synchronized is not guaranteed without additional effort. Very complex and error prone procedures need to be introduced in order to mitigate this disadvantage. Even after their implementation, there is a risk that the desired synchronization can never be achieved. Therefor some drastic measures like; still standing, taking a snapshot, and restarting the system are taken into consideration. The fact that every participating node has a different perception (or view) of the system, is the main reason why double-spending is possible at all. The necessity for consensus building is a consequence of this deficiency.

Nevertheless, synchronization is inevitable to keep the system well-behaved. In order to accomplish this task some participants are assumed to undertake operations which are only reserved for them. The privileged role of these make the system centralized around them. This is true for “Bitcoin” as well as for “IOTA”. The term “decentralized”, means all operations are allowed for all participants without any exceptions.

On the definition of “decentralized”

The term “decentralized” cannot be defined in general, because it depends always on the context of usage. Back to the analogy, the newspaper is decentralized with respect to reading it, but not with respect to choosing its contents. Another example is elections. Not everyone may vote due to rules and regulations, but for those who are eligible voting is decentralized. As a working definition one can assert that “decentralization” is a matter of purpose, allowed operations, and participating entities. Therefore, the context should be specified explicitly, if it is not self-explanatory.

On the definition of “distributed”

The term “distributed” in context of *distributed computing* is defined as

> “A distributed system is a model in which components located on networked computers communicate and coordinate their actions by passing messages. The components interact with each other in order to achieve a common goal … In distributed computing, a problem is divided into many tasks, each of which is solved by one or more computers.” *Source: Distributed Computing - https://en.wikipedia.org/wiki/Distributed_computing*.

Only those systems which implement “Ledger Technology” and comply with the above definition can be called “Distributed Ledger Technology”.

Value Transfer Using Distributed Ledgers

In the next sections we describe a system which uses “distributed ledgers” for value transfer. First, we define terms which are going to be used due course.

DEFINITION OF TERMS

Account

Account is a container which includes:

- An *end-balance* of the account
- An archive, a dictionary of all *transaction*s which are related to the account
- A *ledger*
- A dictionary of all *edges* which link transactions of the account to each other

The private part of a public-key schema determines the account owner and is used to sign payment-orders. The public part of the same schema is used as a generally available identifier for the account. The account is assumed to be reachable over HTTP with an address of the form "public-key @ hostname".

End-balance

End-balance is the balance of the account after the transfer is completed.

Transaction

Transaction is the manifestation of a value transfer order. The order is from sender to receiver and characterizes always a "payment". The layout of a transaction is given in Figure 1.

Elements of a Transaction

Transaction-ID

Hash value of the transaction, over Sender ID, Sender Index, Signature of Sender, Value, Receiver ID, Receiver Index, Signature of Receiver.

Sender-ID

ID (Public-key) of the sending account.

Sender-Index

Zero-based index of the transaction within sender's ledger.

Signature (~) of the Sender

Signature of the sending account.

Value

Amount to be transferred.

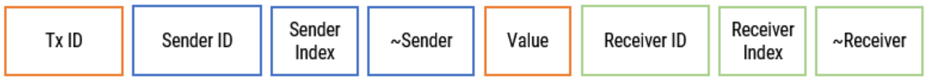


Figure 1 – Layout of a Transaction

Receiver-ID

ID (Public-key) of the receiving account.

Receiver-Index

Zero-based index of the transaction within receiver's ledger.

Signature (~) of the Receiver

Signature of the receiving account.

Ledger

Ledger is a double-linked list of edges. Only those transactions which reference the account as either a sender or a receiver are attached to the ledger. "Append" is the only operation allowed on the ledger. Other operations like "Delete", "Insert", and "Update" are considered invalid.

Edge

Edge is a tuple of two consecutive transactions related to an account, where edge-ID is the hash value over IDs of the linked transactions.

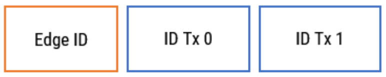


Figure 2 – Layout of an Edge

A WORLD FREE OF FRAUD

Although dreaming of a fraud free world is unrealistic, doing so can help understanding the minimum requirements of a value transfer system. Assume a *sender* transferring an amount *X* of a *value* to a *receiver*. Both parties agree that this is a legally binding transaction via a secured note and they keep a copy of the document.

Transferring a Value

1. Sender builds the transfer request:
    a. Sender-ID
    b. Sender-Index
    c. Value
    d. Receiver-ID
2. Sender signs the transaction over Sender-ID, Sender-Index, Value and Receiver-ID.
3. Sender sends the transaction to Receiver.
4. Receiver fills Receiver-Index.
5. Receiver signs the transaction over Receiver-ID, Receiver-Index, Value, and Sender-ID.
6. Receiver calculates the transaction-ID and completes the transaction.
7. Receiver sends the transaction to Sender.
8. Receiver updates its end-balance and appends the transaction to its ledger.
9. Sender updates its end-balance and appends the transaction to its ledger.

After these steps, value transfer is completed, and sender as well as receiver recorded a copy of the transaction in their ledger for auditing purposes.

FRAUD PREVENTION

Transactions defined in the previous section contain already some measures against fraud. Signing parts of a transaction guarantees the validity of its contents:

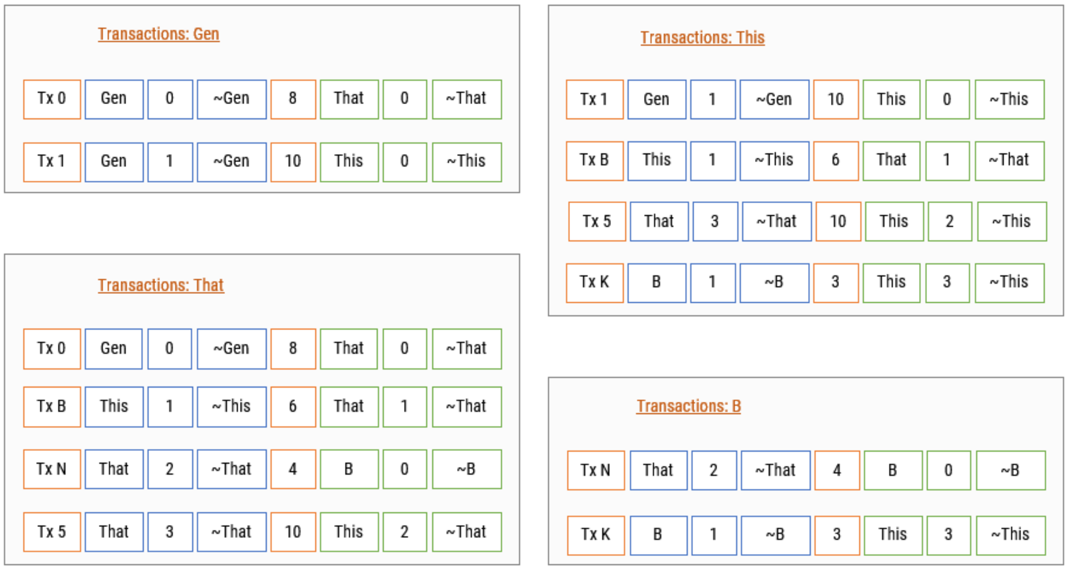
> "A valid digital signature gives a recipient reason to believe that the message was created by a known sender (authentication), that the sender cannot deny having sent the message (non-repudiation), and that the message was not altered in transit (integrity)." *Source: Digital Signature - https://en.wikipedia.org/wiki/Digital_signature*

A fraudulent account can try to cheat with uncovered spending, or not updating its end-balance. Even not updating the ledger could lead to inconsistencies. There are ways and means to detect such attempts, but it is very inefficient to suspect misconduct at all times. Transactions of a sample value transfer system with accounts "Gen", "B", "This", and "That" are shown in Figure 3. Because the transactions are well protected, i.e. none of the fields of a transaction can ever be changed without invalidating it, ledgers are the only objects of the system which are subject to fraud. A very efficient solution would be introducing a *notary*. Not a trusted third party, but a piece of trusted code which realizes several services as detailed in the extended definition of terms:

Notary

Notary is a signed code, which embodies following responsibilities:

- Checking whether the account has sufficient assets to support the intended value of transfer
- Computing the transaction-Id of the current transaction
- Creating an edge, a tuple which links the last transaction on the ledger and the transaction at hand
- Signing the edge and appending it to ledger
- Updating and signing end-balance



Transactions: Gen

Tx 0	Gen	0	~Gen	8	That	0	~That
Tx 1	Gen	1	~Gen	10	This	0	~This

Transactions: This

Tx 1	Gen	1	~Gen	10	This	0	~This
Tx B	This	1	~This	6	That	1	~That
Tx 5	That	3	~That	10	This	2	~This
Tx K	B	1	~B	3	This	3	~This

Transactions: That

Tx 0	Gen	0	~Gen	8	That	0	~That
Tx B	This	1	~This	6	That	1	~That
Tx N	That	2	~That	4	B	0	~B
Tx 5	That	3	~That	10	This	2	~That

Transactions: B

Tx N	That	2	~That	4	B	0	~B
Tx K	B	1	~B	3	This	3	~This

Figure 3 – Archive of the Sample.

Fraud Proof Transfer of Value

There are only a few steps necessary to transfer value in a well-behaved manner within the system. The transfer procedure described earlier is modified and the modifications are underlined:

1. Sender builds the transfer request:
    a. Sender-ID
    b. Sender-Index
    c. Value Receiver-ID
2. Sender signs the transaction over Sender-ID, Sender-Index, Value and Receiver-ID.
3. Sender sends the transaction to Sender-Notary.
4. Sender-Notary validates value and Sender-Index; rejects or sends the request to Receiver-Notary.
5. Receiver-Notary validates request; rejects or sends request to Receiver.
6. Receiver fills Receiver-Index.
7. Receiver signs the transaction over Sender-ID, Value, Receiver-ID, Receiver-Index
8. Receiver sends the transaction to Receiver-Notary.
9. Receiver-Notary validates Receiver-Index; rejects or calculates the transaction-ID and completes the transaction.
10. Receiver-Notary updates and signs the Receivers End-Balance.
11. Receiver-Notary creates a new edge and appends the edge to Receivers-Ledger.
12. Receiver-Notary adds the transaction to Receivers archive.
13. Receiver-Notary sends the transaction to Senders-Notary.
14. Sender-Notary updates and signs the Senders End-Balance.
15. Sender-Notary creates a new edge and appends the edge to Senders-Ledger.
16. Sender-Notary adds the transaction to Senders archive.

Network

There are no special presumptions regarding the networking environment. Any platform supporting String based messaging would be sufficient. Public-keys can be used to identify particular accounts and to build addresses of the form “public-key @ hostname”.

Ledgers

As indicated in Figure 3 transactions are stored as individual objects which do not contain any linkage information. They are stored in a key-value dictionary where the key is the transaction-ID, and the value is the transaction. It is assumed that the “Append” operation of the dictionary checks the validity of the key-value pair, i.e. the hash value of the transaction is equal to transaction-ID. The key-value pair is immutable. Changing the content of the value would change its hash, which is its key. So, the resulting key-value pair would be another unique transaction.

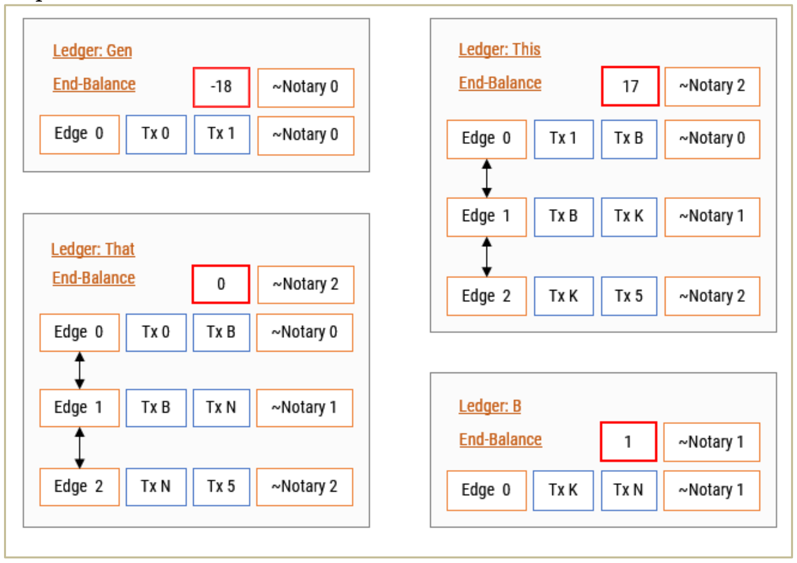


Figure 4 – Ledgers of the Sample

It should also be mentioned that “Gen” is the only account where value is generated. It can either have an unlimited negative balance or it will be initiated with a positive balance at the beginning. Figure 4 shows the ledgers constituting the sample system. Each transaction in the system appears exactly in two ledgers. This is the first principle of double-entry bookkeeping which we lean on. Each ledger is an ordered-list of transactions related to an account. As already mentioned, the links between transactions are maintained in an explicit edge-set, for which “append” is the only allowed operation. Therefore, not only the content of transactions is immutable, but also their order.

Merging Ledgers

There may be occasions where one would like to have a consolidated view of interactions between several accounts, e.g. general ledger of a company. As every transaction, and every account in the system is uniquely identifiable, different ledgers can be merged into one directed acyclic graph (dag for short). Thereby the merge operation obeys the principle “same means same”, which is crucial for the semantic of dags. The merged ledger in Figure 5 shows the subgraph induced by accounts “This” and “That”, in this case it is the complete dag modeling the system, the General Ledger.

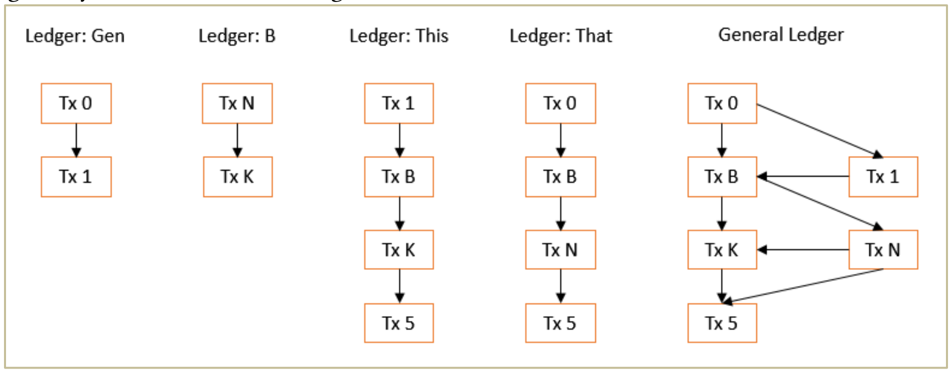


Figure 5 – General Ledger of the Sample

There are many topological orders which can be associated with the shown dag, but only one of them corresponds to the claim “as soon as possible, but as late as necessary”, which at the same time states *the topological order of dag according to its data-flow*. In order to satisfy the claim, transactions must be ordered in such a way that they succeed those they depend on as soon as possible. At the same time, they should be postponed as long as the transactions they depend on are not listed. This order is shown in Figure 6.

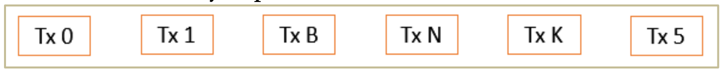


Figure 6 – Topological Order of the Transactions of the Sample

The sum of the end-balances of the accounts in the system is zero after the transactions are executed. This is the second principle of general accounting we lean on.

GENERALIZED LEDGERS

In the previous sections we discussed “value transfer” within the framework of general accounting. The scope can be extended to execute and record arbitrary transactions without modifying the process. But care is in order; any generalization should not sacrifice the two main principles of being a ledger (*an ordered list of transactions*), by no means. These principles are:

1. One ledger per account, and one account per ledger
2. Preserving the relative order of entries as they occurred

Otherwise the resulting data-structures would be anything but a ledger (see [1], [2], and [3]). Merely updating definition of some terms is sufficient to accommodate this expansion.

End-Balance replaced by ***Document***

We will allow end-balance to contain arbitrary *Strings* and change its name to *document*, reflecting this change. Document then is an up-to-date version of a *String* of characters with arbitrary content.

Value in a Transaction

Value is a *String* of characters which substantiates the semantics of a change request to the document. In the context of general accounting this could be “Add XXX to document” where document represents the total-balance of the account. In a more general context, it could be an agreement, a contract between two parties. It could be a full or partial document, an executable script or any arbitrary string. The only requirement is that it must be interpretable by notary. Notary then can take necessary actions in order to correctly update the document.

Notary

Responsibilities of the notary has to be adjusted accordingly:

- Validating value
- If valid; taking the necessary actions to update the document

CONCLUSION

We described a system to interchange arbitrary transactions between senders (initiators) and receivers. The key features of this system are:

- Peer-to-peer interaction in absence of a trusted third party (*Notary* is not an independent third party, it is just signed code)
- Preserving anonymity with proof of ownership
- Preserving the relative order of transactions as they occurred
- Keeping book of all transactions of an account, and only of that account, in a ledger unique to that account
- Ability to check whether the system is balance or not, before and after every transaction
- Immutable archive of the transactions of the account
- Immutable ledger of the account
- Unlimited horizontal growth, the number of participating accounts
- Unlimited vertical growth, extensibility of the ledger

Besides these features, subgraphs induced by any combination of accounts can be merged into one subgraph to have a complete view of their interactions (see Figure 5), while respecting relative order and data-flow of transactions. The system exhibits the behavior that for each transaction recorded in an account the same transaction is also recorded in a counter account (see Figure 3). As a result, the systemwide balance can be checked due course. The system is also flexible enough to execute arbitrary transactions. From simple amount, to complex logic realized as an executable script.
If the mission is to manage execution and recording of transactions in an accountable manner, the proposed solution should be the preferred choice.

ACKNOWLEDGEMENTS

The author would like to thank Cem Çimenbiçer for his invaluable comments and recommendations.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A Peer-To-Peer Electronic Cash System", 31 October 2008. [Online]. Available: http://nakamotoinstitute.org/bitcoin/.

[2] S. Popov, "The Tangle", 1 October 2017. [Online]. Available: https://iota.org/IOTA_Whitepaper.pdf .

[3] Xavier Boyen, Christopher Carr, Thomas Haines , "Blockchain-Free Cryptocurrencies", 2017. [Online]. Available: https://eprint.iacr.org/2016/871.pdf .