ECC: Peer-to-Peer Electronic Cash with Trustless Network Services

Greg Griffith
October 2017
(griffith@cryptounited.io)
https://www.ecc.network

Abstract

Common network services are typically managed by a single entity with which a user must request information from using the client-server model. Single entity management can lead to heavy censorship if the entity in charge decides that it wants to restrict the flow of information to the users. We propose a solution to the issue of censorship using a peer-to-peer network in which information is publicly declared and verified by everyone on the network. Under this design, services and information are provided based on a consensus of valid events witnessed by the entire peer-to-peer node network and are only altered or added through a network-wide consensus. The state of a networks consensus is recorded in a hash-based proof-of-work or coin age based proof-of-stake unique to that service. Each service's sequence of events is validated through a different consensus network. A given node is not required to connect to all services available and will accept the longest chain when joining a service as proof of what happened while said node was disconnected from said service. Any given node is required, at a minimum, to connect to the electronic cash transaction network that operates at the core of all other service networks as their monetary exchange network.

Introduction

Since the creation of Bitcoin in 2009, it has been realized that trustless global networks can do more than replace current electronic payment systems. A peer-to-peer global network providing information and services designed around a trustless electronic cash protocol derived from Satoshi Nakamoto's Bitcoin utilizing proof-of-stake derived from Sunny King and Scott Nadal's Peercoin can be used as the foundation on which to operate trustless common network services. Using multiple consensus mechanisms built around a single peer-to-peer electronic cash system, we can expand transaction and network functionality to create a mesh network with multiple network services that are capable of replacing their centralized counterparts. The project created to complete this task has been given the name ECC.

Peer-to-Peer Electronic Cash

Before creating trustless network services, we must first define the electronic cash protocol at the core of the network that is to be used as a monetary system by every service. Our basis is the peer-to-peer electronic cash protocol Bitcoin as written by Satoshi Nakamoto [1] with the addition of the concept of proof-of-stake as written by Sunny Kind and Scott Nadal [2]. We have, however, included a few minor changes to the proof-of-stake system and proof-of-work hashing algorithm used. The proof-of-stake system now multiplies the coin age with the value of the input of the coinstake transaction to calculate a reduction value for the coinstake transactions proof-of-stake hash. This reduction is used to check if the hash meets a difficulty constraint similar to the difficulty constraint with proof-of-work mining. Only if the hash is below the target difficulty is the block considered valid. The proof-of-work hashing algorithm has been changed from SHA-256 to Scrypt.

Network Services

Having defined the core transaction network that will be used within the ECC project, we can begin to address how we can use additional consensus mechanisms in combination with the core transaction network to provide trustless common network services. All services would require their own consensus mechanism and should operate independently of one another. For our purposes, we will use a hash-based proof-of-work or coin age based proof-of-stake chain as consensus mechanism for each network service. Separate chains will allow for easier data pruning and enable a given node to only keep track data that it needs. Any node trying to perform an action on a chain that is not the transaction chain, where payment to a peer is needed to complete said action, should include a reference hash to a transaction made on the payment transaction network.

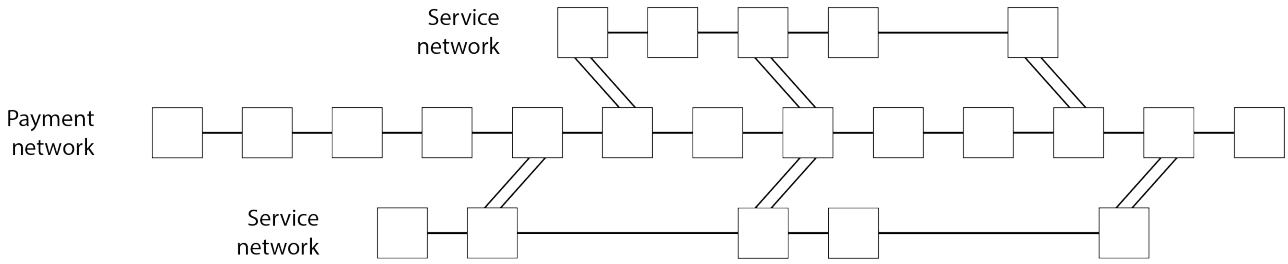

Figure 1: Visual example of multiple service networks working with payment network

Depending on the number of services a user chooses to subscribe to, the amount of storage space on disk can vary a significant amount. It is important to note that for the transaction chain, the added fields in the transaction will add no more than 256 bits to a transaction size that references a service action. Not all transactions will reference a service action so the average size increase in a transaction on the transaction network will be much less than this.

Expanding Transaction Functionality

Given that our system uses electronic transactions without relying on trust, we can prove that any additional information provided with a transaction must also have originated from the same owner as the signature used to sign the transaction. A given node can use this property to broadcast a transaction containing service data on a service network while simultaneously proving the authenticity and origin of the transaction. Transactions in service networks will use the same hashing function as the transactions on the payment network.

New fields in non-payment network transactions include:

1. Service Identifier
2. Operation Code
3. Instruction Data
4. Payment Reference Hash

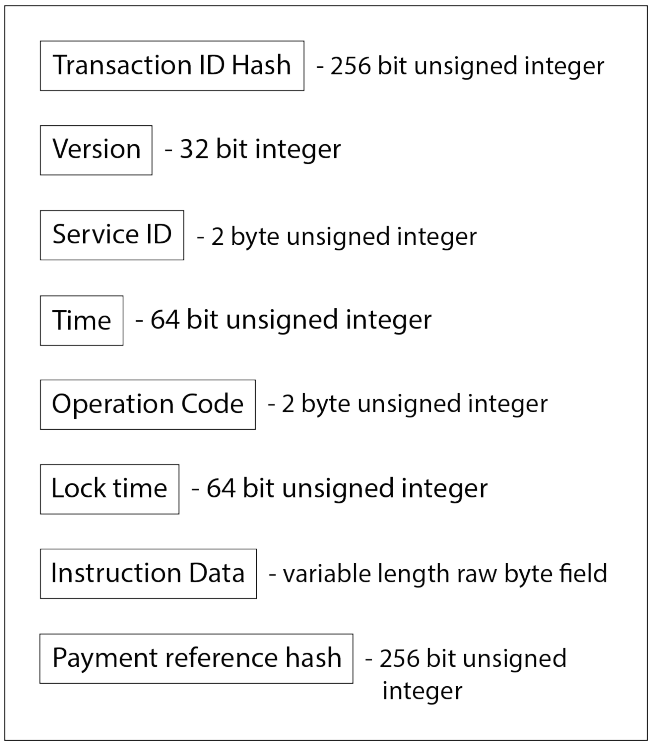

Figure 2: Service network transaction

New fields in a payment network transaction include:

5. Service Reference Hash

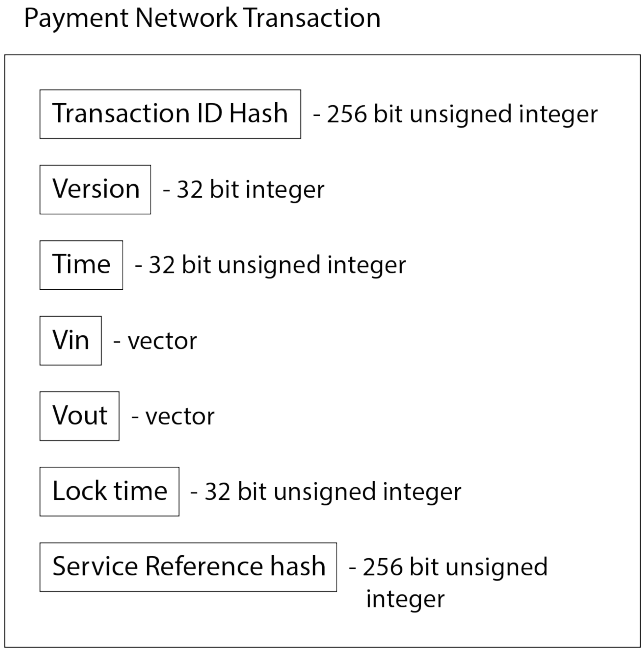

Figure 3: Payment network transaction

1. Service Identifier

The service identifier is a 2-byte unsigned field that holds the id number of the service being used. This can be expanded to a larger number of bits if more than 65,536 id numbers are needed by forking to a larger number of bytes field.

2. Operation Code

The operation code is a 2-byte unsigned field that holds the operation code of the instruction being performed. This can be expanded to a larger number of bits if more than 65,536 operation codes are needed by forking to a larger number of bytes field.

3. Instruction Data

The instruction data is an optional variable length field that contains additional data that the operation code might need. This data should be in raw byte form. The more information that is added to this field, the more expensive the operation becomes. It is best to not add more than needed.

4. Payment Reference Hash

The payment reference hash is an optional field that is used to reference a payment on the transaction network that pays for the action specified in this transaction. This hash is a 256-bit unsigned integer that should be the exact value of the transaction id in the service network that it is referencing.

5. Service Reference Hash

The service reference hash is an optional field that is used to reference a service action on a network that is not the transaction network. This signifies that this transaction should pay for that action specified in the referenced service action. This hash is a 256-bit unsigned integer that should be the exact value of the transaction id in the payment transaction network that is paying for this operation.

Cross Chain Agreements and Contracts

We cannot ensure that agreements across the different network chains are honored by the node on the receiving end of the payment if the service involves an action that is not stored on a chain. All payments for services should only be redeemable once the state change of the service being utilized has changed to include the agreed upon operation. The payment for a service must include the service transaction's id in the service reference hash field and the service transaction must include payment transaction's id in the payment reference hash field.

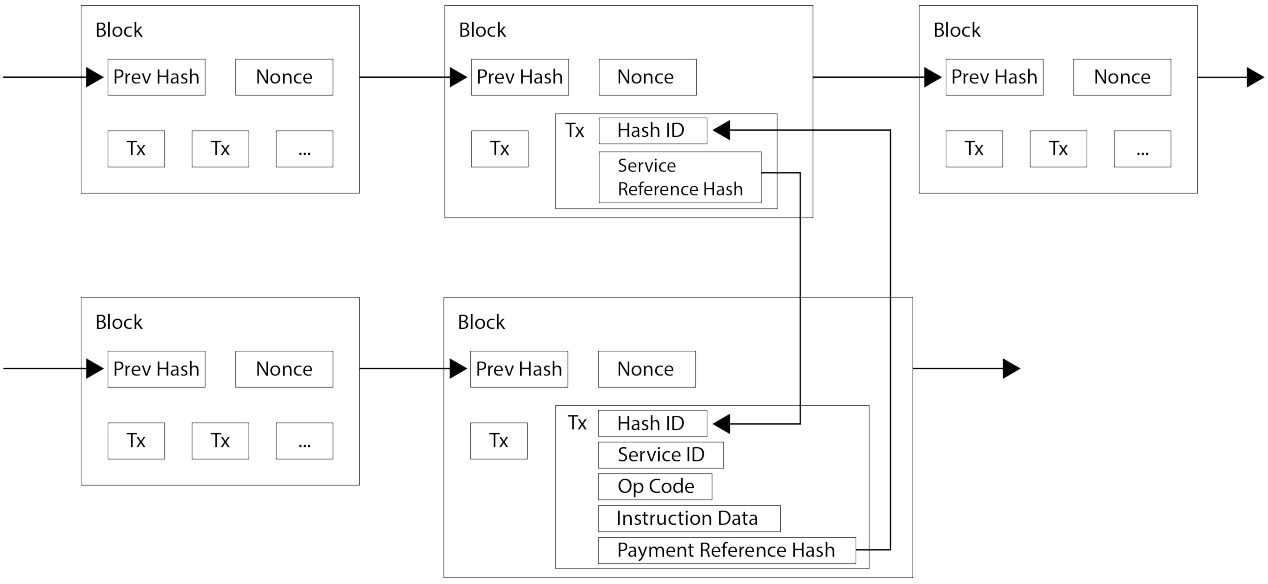

Figure 4: Payment network (top) transacting with service network (bottom)

Payments for a service should be redeemable for a set time predetermined by the sender before expiring. Once expired, the transaction can be abandoned by the sender and the containing funds returned to their original address. This expiration on payment validity can be embedded in any transactions script code to prevent redemption after the valid period of time has come and gone.

Conclusion

We have proposed a system for common network services without relying on trust. We started with a core peer-to-peer electronic cash network defined by Satoshi Nakamoto, implemented a customized proof-of-stake originally defined by Sunny King and Scott Nadal, and added additional chains to be used as consensus mechanisms for common network services. Cross chain service contracts can be completed or canceled without the need for a 3rd party or centralized regulator. Nodes can subscribe to the services they want to use to prevent storing extra data. Nodes can leave and join any service network at will, accepting the longest chain for a given service as proof of what happened while they were gone. All states on the network can be enforced with their chain as a consensus mechanism.

References

[1] Nakamoto, S. (2008, October 31). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from https://bitcoin.org/bitcoin.pdf

[2] King, S. & Nadal. S (2012, August 19). PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake. Retrieved from https://peercoin.net/assets/paper/peercoin-paper.pdf