Magnite.org

Peer to Peer Digital Currency

DRAFT

Date: November 26, 2016
Updated: December 2, 2016
Document Revision: 0.1.1
Author: Jon Taylor
Website: http://magnite.org

Magnite is a placeholder name for now. Another possible option is Safire. A final name will be chosen soon. Please send feedback if you have a preference.

Summary:

Magnite is a digital currency that runs over a distributed Peer to Peer network. Anyone can download the software, join the network, earn currency for running the software then send and receive that currency with other users.

Overview:

Magnite is a distributed digital currency. Every user can send payments to any other user with a small fee. Users are rewarded for relaying network transactions with new currency issued at a rate determined by a group preference average.

Like Bitcoin it uses blocks to confirm pending transactions and validate them for all users across the network.

Unlike Bitcoin instead of blocks being formed by the first solver of a difficulty hashing function it uses a selection function to choose one user based on the current time. Only that user can create the block for the current time period using a signed message for which they keep a public private key pair they created when they joined the network. This means that before new users can send and receive transactions they have to be granted entry into the network through a membership entry in a new block.

New currency is issued to the block creator for each generated block. The amount of currency issued is based on a network wide preference average. Accounts not used by people for commerce are discouraged through currency inflation and reward commission if they don't support the network by staying up to date.

Each user in the system can specify preferences for network behaviour. Because all users are known the averages for each setting can be used to alter core settings like inflation rates and block times.

Constraints on the the number of transactions that can be performed by the network are the propagation time for generated blocks to be sent to other nodes in the network and the amount of disk space required to store the data for each user.

If the transfer takes longer than the block time, then the block creator following can be skipped. This creates a time gap between block times allowing nodes to catch up.

Transactions older than two years are accumulated in the current year for each user so that older records can be cleared requiring disk storage for only current activity.

Key Benefits:

1) Send and receive currency with anyone over the internet even if the recipient is not running.

2) Users vote equally on the network and currency properties adjusting to future needs.

3) Currency is issued to users evenly for relaying transactions without the use of mining. Rewards are distributed evenly to all users that are able to stay up to date with relaying transactions. In mining networks rewards are unbalanced when the distribution of hashing power is uneven.

3) Decentralization and network capacity are strained mainly by current demand as old history is purged. In the long term the requirements to run the network will not increase with the entire history but only by current demand.

4) Graceful handling of excess capacity demands on the system. This means demand peaks transactions will take longer to confirm but the throughput of the network increases to compensate and catch up.

1) Send currency anywhere over the internet even to people not online.
2) Free, download it and use it for no cost. The Code is open source and collaborators are welcome.
3) Earn currency for joining the network and running the software. It only requires reasonable resources.
4) It is sustainable long term both by capacity scale and as a store of value. Older data is deleted using disk space for current use.
5) No mining. Even distribution of currency and influence.
6) Why is it important to be decentralized. So they don't go bankrupt or inflate. Use examples.
7) Why use it? You choose policy and network vote. You own it, no other single owner. It's a diversification option.
8) Privacy, can be used without registering personal identity information.

9) Easy, Design to work, take care of details and solve problems when they arise.
10) Fast, transactions, Syncing, wip
11) Reliable, beta now but core focus in
12) Integrated ecosystem. Partner with business that take currency, exchanges to trade etc...

13) Safe. Secure your balance with encryption.

Process:

A new user installs the software on their computer.

They will generate a few public private key pairs, One is a wallet key pair that allows them to transfer currency with other users and another key pair is a block creation verification set used to ensure only they can create a block when they are designated by the selection function.

They then connect to a few other users in the network to start downloading historical transaction and user data.

To participate in the network the user will submit a request to join the network. The request is sent to all other users they are connected to and the message will be passed along until it eventually reaches the current block creator.

The block creator includes new user requests in the block and broadcasts it directly to the next block creator and additionally to all other users in the network where it is relayed to all other users. Blocks are created during a set time frame and there may be a padding transit time that allows blocks to be transmitted.

Once the new user sees their inclusion in a valid block they are able to relay transactions across the network. They are also in rotation for creating new blocks and receive currency for doing so.

Each time a user generates a block they add a public key and a signed message to that block. The next time that particular user generates another block they must add to the block the private key from that past message as validation. Everyone knows if it is valid because they can test it. No one can fake it because they don't have the private key. That user will then also generate another public private key and message attaching the public key and message. When that user is designated to generate another block only that user can create it. New users will always broadcast this block key pair in their application to join the network.

Update: Rather than using a new hash each block, just use a block public signature (created on first startup) against the contents of the block. All users can verify that the block code was created with their private key.

The block previous message has to be a hash of the content with the hash of the previous block. This way no forged blocks can be added later into past sections.

To create a block the user must receive the previous block from the last user. If a block creator does not receive the previous block it in time they can't create a new block and it will be up to the next designated block creator to get the latest block and carry on. Each block creator can contact the previous block creator to check their status and request the latest block.

Over time old balances will be rebroadcast every two years to prune old transaction data.

Users that do not stay up to date and running for a period of years need to have their balances pushed forward. Any user in the network can ping other users and request their balance if they discover that a user has transactions that have not been carried forward. Users will receive a bonus for pushing other accounts up to date and then paper wallets will be reliable.

Currency Issuance:

Currency is issued to running nodes in the network for staying up to date as well as creating blocks. The new currency is issued with a variant of a transaction request called issuance. These. These issuance transactions don't have a sender and are validated by the network so that they adhere to rules about who can generate blocks and what is contained in them. Users in the network are chosen by hash function for each five or ten minute time period.

1) The user chosen to be block creator for a five or ten minute time period can collect (send themselves an issuance transaction) a bounty payment if they broadcast a valid block with transactions and user updates. If they are not running they do not receive the bounty.

In order for the block creator to publish a block get the bounty they have to broadcast the block to the next user as chosen by time hash before their time block expires. If they do not relay the data the block is not valid. This hand off process is necessary to ensure continuity so that there isn't a fork and double spending.

The blocks must be relayed directly from one user to the next to ensure that they can be linked together. Otherwise there would be an opportunity for state to change without the block creator knowing about it.

2) A set of runner up nodes to the block creator. I.e. chosen by user hash and time hash in order receive a transaction from the block creator if they are running and up to date. This effectively rewards a few random users for participating in the network to a lesser degree than the block creator. This is important because as the number of users in the network grows the rate each user becomes the block creator becomes less frequent and the distribution of new currency also.

Transactions

A user can send a portion of their balance to another user on the network by composing a network message and relaying it to other users. Once the message reaches the block creator and is included the transaction is confirmed.

A transaction includes the sender address, amount to be sent, the public key of the recipient, a sender counter value and a signed verification hash of the transaction details.

The sending user signs the combination of the sender address, recipient address, amount, and counter with their private key. They then attach this signature message to the request. Everyone can verify that the transaction message was created using the senders private key because everyone has that users public key.

Users in the network can observe validated transactions by receiving and confirming a block. Only one block confirmation is needed to ensure the transaction is final and authentic.

Users also keep track of the last transaction counter for each user. New transactions can be checked against the last counter value for that user. Any transaction with a count value less than the last user value are ignored because they are old and have been issued already. The transaction sender counter is like a transaction id and is used to prevent multiple issuance.

Transactions have the effect of debiting the sender and crediting the receiver. This is calculated by accumulating transactions in the block chain for each user. The presence of carry forward transactions moving balances up from past years is also included to validate the numbers are added correctly.

Having other nodes verify that their copy of the block chain matches other nodes can add extra validation but should be unnecessary because it is unlikely anyone can create a block hash message for a user private key they don't control.

The calculation to determine which user is responsible for creating blocks in the past depends on the inclusion of users to the network and can be used to verify that non authorized users didn't add blocks when they were not supposed to. This should not be possible even if they could add an unauthorized block or transaction without the private keys.

User Identity

Each user client node has an identity with an address key pairs, block creation key paris and voting configuration settings. A transaction ledger of transactions between other nodes. The latest transaction and balance between other users.

Settings and Voting

There is a bell curve of settings used to vote for network settings. Higher and lower on the average are counted less. These settings affect network policy like inflation rate. If a standard

deviation of users average an inflation rate of 1% then that is used to pay users in currency for validating transactions over a year.

If some users settings are 1000% inflation and they are outside of the second or third deviation their settings are weighted much lower than a vote in the first standard deviation.

Settings are stored as key value, certain keys have effect. The hash is of the key names and values. New settings added later will have defaults so that old users will adopt default settings without invalidating their history score.

Pruning and Balance forward:

All accounts transfer their balance from transactions prior to two years into the current year. Balances are calculated by the accumulation of debit and credit from transactions. Balances are carried forward using a variant of the transaction network request called balance carry forward. The client software will automatically make this request to the network. Transaction data older than four years can be deleted.

Balances for users can be validated using transaction history data as well as carry forward data.

There could be a bounty for people who identify old balances with no carry forward yet to encourage others to keep everyone up to date and not loose continuity for inactive users. This would allow for people to rely on paper wallets.

Double Spending Transactions:

The network prevents users from sending a transaction to more than one user because of the following constraints:

1) Blocks created by users in the network are validated against the historical data to ensure that the user has funds available. All users in the network receiving new blocks can validate them based on the contents such that the transfers have balance and the block creator included a content hash signature that was signed by the designated block creator.

2) Blocks are chained together so that the hash of each previous block is included in the the next block ensuring that no one can insert fake blocks at a later date.

3) Only the designated block creator can create blocks because they have a block_generation_private_key from a previous block they created or from when they signed up.

Privacy:

This system broadcasts transactions with your Keys across the network. You can create multiple account wallets to separate your identity from other wallets. Although your exposed wallet will show trades against anonymous accounts.

Very similar behaviour to Bitcoin. Extra anonymity features are not currently created. The key areas of this system are user choice on network behaviour and high transaction capacity.

Attacks and Failure Conditions:

Based on the system design and structure there are particular areas where people could attempt to manipulate the system for their advantage or disadvantage of others. Mapping out some of these areas here is useful for guarding against such attacks.

1) Prevent people from creating inactive accounts without participating or contributing to the function of the network. It takes resources by everyone on the network to store data for each user. Accounts that won't be used are discouraged by the fact that they won't receive payments if they are not up to date.

2) Adding more users means less evenly distributed currency?
If there are a million users that are up to date and blocks generated every ten minutes then on average a given user will be selected every nineteen years. Currency will also have to be issued to a set of randomly allocated users each block cycle to bring this distribution time down preferably within a few months. These payments would be of very small amounts much smaller than the block creator.

3) What happens if rouge nodes refuse to include new users? Anyone can choose the number of pending user requests to include based on resources. More data in a block means less likely to be received by the next block maker.

4) Time synchronization conflicts break communication. What happens if users have different system times then the miner selection function won't work evenly. Also timezone differences can affect what times each user sees.

6) Block creator DDOS, Because the user responsible for creating a block is know anyone can see this and initiate a network flood attack on this or upcoming users.

7) If individuals are known in the network targeted individuals could be DDOS attacked preventing them from accessing the network?

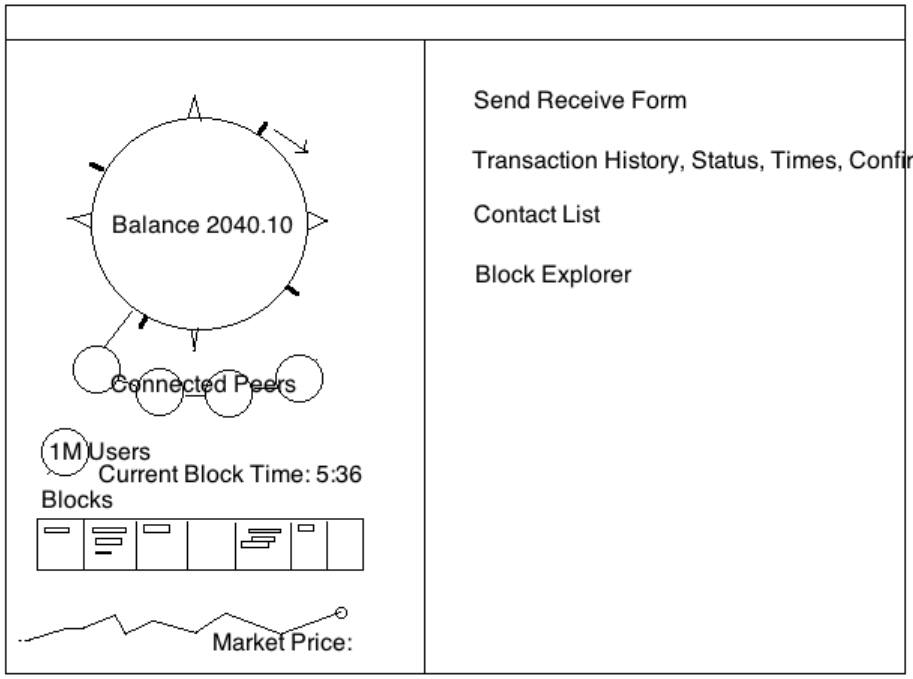
8) What if some nodes in a region of the internet are slower and can't keep up with faster areas that generate big blocks too quickly?
An option to handle this would be to have users monitor the status of other nodes in the network and relay an up to date score to the block creator. If the network is falling behind then they can reduce the size of the block.

9) A Time attack can occur if a sufficiently large number of users in the network intentionally set the system time to the wrong value. This could be used to alter for each time change who is the

block creator. This could be used to artificially set the block creator more frequently to malicious nodes that don't include transactions lowering the capacity of the network.

Desktop Client

A mockup for a desktop client has a few essential elements to allow users to manage their holdings.



The client will manage communication with other users in the network, Display a balance and network stats.

Allow transfer and receiving of currency from others.

Show transaction history.

Show contact list.

Calendar for automatic bill payments and requests.

Messenger system to communicate with other users on the network.

Block explorer to show recent clock contents and times. Heavy load needs to be managed so this is health monitoring related.

Infrastructure partners can be listed as well including merchants, exchanges, etc.

Specifications

This section describes the interfaces and data formats for each of the pieces used in a client application.

DB Schema

The following entities define the storage requirements for a Safire client.

User Identity – User key pairs and settings maybe stored in a config file not the DB.

Name
Wallet Public Key,
Wallet Private Key,
Block Creation Public Key,
Block Creation Private Key,
User IP address, - used to connect directly by the previous block creator.
Setting values: (inflation rate, number of transactions stored, initial deficit, verifications before payment, number of top transactors to discount verification fee, number of verifications before confirmed % of users, yearly wallet fee),
setting value hash,
Location,
IP address

User List -

Time Block Confirmed by network, ???
Public Key,
IP address,
Transaction_counter, // incrementing value for each sent transaction.
Confirmed Balance Cache, // Cached values are calculated from the block history.
Unconfirmed Balance Cache,
Time running Cache,
Transactions Confirmed Cache,
Transactions Failed Cache,
Last activity/alive date Cache.

Block Transactions

TODO

Transaction History -

Transaction ID,
Sender PK,
Receiver PK,
transfer amount,
verified by hash,
Date,
Confirmed,
isVerificationFee,
isIdentityFee.
- is a cached view of the complete data set.

Settings -
Name: string ,
Value: string

Function API Specification

Functions defined here are required implement a basic user node (client/server).

Create a New Key Pair. - generate public private cryptographic key. Use algo (ECDSA, rsa, ???)

Create signature – (private key, message) return signature.

Verify signature – (message, public key, signature) return boolean.

DEPRICATE Generate Transaction ID - random number. (Not considered network wide unique, just used as an index.)

Is Transaction Confirmed - Based on consensus rules.

Confirm Settings (Validate?) - check that settings names match functions. I.E. confirm settings are applied correctly.

Get Balance - Confirmed and unconfirmed based on records in transaction history DB.

Get Time Block Creator (Time) – Returns a user public key for a given time that is responsible for generating the current block. based on epoch mod 5 minutes what is the hash. ? Check other nodes too?

Validate Block - A node generating a block and other nodes validating a block made by others use this function.

1) validate that each transaction in the list sender has the balance in their account.
2) validate that the block was created with the correct block_creation_private_key signature.
3) validate that the block number is one greater than the previous.

Is Node Up To Date - Given status data determine how up to date it is within range. Also validate data is accurate.

Settings Attributes

Settings are string key value pairs and can contain any values limited in length to 512 characters. Only certain key and value pairs will have an effect depending on the client version and implemented features. The following are suggestions as examples only.

inflation rate: Factor in determining amount to be issued to block creators.
A percentage used to calculate fees. Fee paid to nodes is average rate / number of nodes in DB / number of transactions before fee.

Number of years to store transactions: (Min:3 max: -1) delete older transactions to free disk space.

initial deficit: Each new identity is assumed and verified based on a negative score to disincentive unneeded users.

Blocks verifications before payment: verifications performed before paying self.

Max Confirmation Verification Time: Time value to override verification by number of nodes in case of network congestion.

Network JSON Schema

Block – Defines a block content
```
{
 block_number: <integer>,                  // Must be one greater than the previous.
 previous_block_hash: <string>,            // Must be set to hash of previous block.
 time: <long>,                             // Epoch
 creator_user_public_key: <string>,
 Creator Block Creation Private key,
 transaction_list: { <Transaction>, ... },
 user_updates: { <User_Update >, ...},
 current_block_hash: ,
 creator_next_block_public_key: <string>,
 creator_next_block_signed_hash: <string>  //
}
```

Issuance_Transaction - Currency creation
```
{
 From_Block_id: <> ,
 To_wallet_public_key: <>
 Time: <>
}
```

Transaction -
```
{
 from PK,
 to PK,
 amount: <float>,
 transaction_id: int,
 transaction_signature: <string>
}
```

Carry_Balance_Forward - Carry balance
```
{
 fromPK,
 from Year,
 amount,
 to Year,
}
```

User_Update - (add user, change setting)

User -
```
{
 user
}
```

Network API

All network requests are sent with using a json format.

Get User List (Start Date, End Date) - connect to a user in the network and ask for a list of users, store in DB. periodically ask random nodes for list of their users to update local DB. Date ranges allow the query to be broken up into pieces once there is a large data set.

Response:

Schema:
```
{
  count: number,
  list: { { user_public_key, accepted_date },  ...}
}
```

New User Request. - Transmit new user message to network.

(name (optional),
wallet public key,
wallet public message,
ip address,
block creation public key). Also applies when settings are changed. Generate block creation public private key. and send the public key.

Delete User (From Public, From Private) – returns Bool ... Use case?.

Send Transaction (From Public, From Private, To Public, Amount) – returns delivered true or false. Network request includes the from Public key, To public Key, Amount and a sender private key signed message consisting of the from public key, to public key and amount values.

Cancel Transaction (From Public, From Private,) - Send a transaction cancel request. May not work if the transaction is accepted before the cancel request propagates. Details to follow.

Update IP address - send public key and new IP address. .

Get Node Status - return hashes and details of the latest transactions. If the dates are over two years old or the transactions in the response are invalid then this node can be seen by others an not active. used by other nodes to test if the user is active. If a node falls two years behind the one asking can make a request to claim a bonus for carrying the balance forward.

Claim Inactive Node Bounty - (requester, inactive, amount, time hash) Broadcast

Get Node Block Time Hash - Used by other nodes to confirm that a given node is on the correct time block.

Get Transaction List (Start Time, End Time) - given a time range return list of transactions on local node. Data is retured in Json frormatt.

Get Transaction List Hash - just return the hash of transactions within a time range i.e. month. Used by out of sync nodes to determine they are missing data for a time period.