DOUBLE SPENDING PREVENTION IN BITCOINS NETWORK
Mandar Kadam, Praharsh Jha, Shravan Jaiswal
B.E (Computer Engineering), St. Francis Institute of Technology, Mumbai, Maharashtra, India

ABSTRACT:
The Bitcoins is a digital currency that unlike traditional currencies does not rely on a centralized authority and has complete independent network of volunteers that collectively implement a replicated ledger and verify transactions. Each verified transaction thus becomes a valid transaction between two entities involved and thus Bitcoins in transferred. As a result of various entities involved and subsequent logs that are updated by these entities after a Bitcoins transaction, the probability of double spending increases. In this paper we propose a solution with use of socket programming and threads in java, a timeout is set and the same Bitcoins hash keys (address) cannot be accessed during this timeout period. This helps prevent double spending of Bitcoins.

Keywords: Crypto-currency, Transaction Block Chain, Bitcoin Ledger, Socket Programming, Double Spending

[1] INTRODUCTION
Bitcoins is a peer-to-peer payment system introduced as open source software in 2009 by developer Satoshi Nakamoto. The digital currency created and used in the system is alternatively referred to as a virtual currency, electronic money, or a crypto-currency because cryptography is used to control its creation and transfer.

Bitcoins goes beyond the scope of cash, allowing truly global transactions, processed at same speed as local ones.

Bitcoins is slowly growing into becoming a possible alternative to the US Dollar or the Euro as more and more businesses start accepting Bitcoins for their products and services. The fact that Bitcoins is still around indicates that the underlying principles are sound. Nevertheless, there is some room for improvement.

The main problem Bitcoins sets out to so needs to reach a consensus about the balances of the accounts it tracks and which transactions are valid. Bitcoins achieves this goal with guarantees which are best described as eventual consistency: the various replicas may be temporarily inconsistent, but will eventually be synchronized to reflect a common transaction history.
[2] PROBLEM DEFINITION

Bitcoins uses multi-hop broadcast to propagate transactions information that has been requested through the network and update ledger replicas that actually helps to record a transaction that has taken place. We then use the gathered information to verify the conjecture that propagation delay in the network is the primary cause for block chain forks.

Block chain forks should be avoided as they are symptomatic for inconsistencies among the replicas in the network. The main drawback of block chain fork has been that it facilitates double spending of a particular Bitcoins.

Double spending is one of the main reasons why hackers have been able to break through the network and been able to spend Bitcoins more than actually available to them. Thus, we then propose various techniques through which double spending can be prevented.

[3] EXISTING SOLUTION

A Due to propagation delay, other transaction gets the freedom to access and manipulate current Bitcoins database.

There are several ways to improve the propagation of information in the network:

- Minimize verification
- Pipelining block propagation
- Connectivity increase

Minimize verification:
- A major contributor to the propagation delay is the time it takes a node to verify a block before announcing it to the network.
- Currently there is a block size limit of 500kB per block enforced by Bitcoins, but this is likely to be relaxed more and more as the average block size grows, so that it may include more transactions.

An initial difficulty check
- The difficulty check consists of validating the proof-of-work by hashing the received block and comparing the hash against the current target difficulty. Additionally, it checks that the block is not a duplicate of a recent block.
- The block can be relayed to the neighbors, as soon as the difficulty has been checked and before the transactions have to be verified.
- Therefore the behavior of the node could be changed to send an inv message as soon as the difficulty check is done, instead of waiting for the considerably longer transaction validation to be finished.

Pipelining block propagation:
- A further improvement can be achieved by immediately forwarding incoming inv messages to neighbors.
The goal of this is to amortize the round-trip times between the node and its neighbors by preemptively announcing the availability of a block earlier than it actually is.

The incoming getdata messages for the block are then queued until the block has been received and the above difficulty check has been performed, then the block is sent to the neighbors requesting it.

Connectivity increase:

- The most influential problem is the sheer distance between the origin of a transaction or a block and the nodes.
- To minimize the distance between any two nodes we attempted to connect to every node in the network creating a star sub-graph that is used as a central communication hub
- Speeding up the propagation of inv messages, blocks and transactions.
- We instructed our implementation to keep a connection pool of 4000 connections open. This caused it to connect to every single advertised address, as fewer than 4000 nodes were reachable at any time.
- The result is that the distance between any two nodes the hub connected to is close to 2.

![Figure 1: Message exchange after the behavior described modifications](image)

[4] PROPOSED SOLUTION

Our proposed solution:

- In our proposed solution once a user has initiated a transaction using his user name and private key, if the user again attempts to initiate a transaction using same bitcoin, transaction will be blocked.
- We have kept a particular amount of time duration in which no other transaction is allowed to access a particular database unless the first transaction is completed. Thus incase an instance of double spending arises the system by itself at initialization itself will block the login or transaction.
Socket is one end-point of two way communication channel between two programs that are running on network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent. It is like end-point of tunnel or pipe-line. The java.net package provides two classes—Socket and ServerSocket—that implement the client side of the connection and the server side of the connection, respectively.

Server: A server runs on a specific computer and has a socket that is bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request.

Client: The client knows the hostname of the machine on which the server is running and the port number on which the server is listening. To make a connection request, the client request to the server on the server's machine and port. The client also needs to identify itself to the server so it binds to a local port number that it will use during this connection. This is usually assigned by the system.

![State Chart Diagram](image-url)

*Figure: 2. State Chart Diagram*
Figure: 3. Algorithm of proposed solution
[5] RESULTS

Here, we have analyzed the concept of a propagation delay by examining the different conditions and results are as follows:

1. The user is resisted to login into his already logged in account.
2. After initializing the transaction, user is not allowed to make another transaction for next three minutes as well as the transaction being completed, this coordinates to avoid the user from double spending.
3. After making the transaction dormant user for five minutes will be logged off automatically.

![Output showing double spending is captured](image)

Figure: 4. Output showing double spending is captured

[6] LIMITATIONS

Multiple login is not allowed. User must login from single account at a single instance of time to make the transaction more secure.

[7] CONCLUSION & FUTURE SCOPE

The changes may mitigate the problem in the short term, until a scalable long term solution is found.

The paper proposed various hardware enhancements that can be useful in order to prevent information propagation delay.

We through our proposed solution were able to find a software approach through socket programming which prevented double spending. Here once a transaction is initiated and till the time the same is not updated in blocks of Bitcoins another transaction using same Bitcoins cannot be performed. Incase more than one transactions are being attempted the user trying to login is denied access.

Future work shall include using technique to enhance the network infrastructure for the hardware aspect and to use various socket programming technique in the software aspect to
restrict double spending. Using both these concepts it is possible to develop a fool-proof system which shall be able to prevent hackers from breaking through the system or causing instances of double spending. This shall thus enhance the overall performance of Bitcoins network and make it more efficient for large scale transactions.

![Client-Server interaction flowchart](image)

Figure: 5. Client-Server interaction flowchart

[8] ACKNOWLEDGMENT

We are also thankful to our guide Ms. Snehal Kulkarni for her valuable guidance and feedback which has been instrumental in its successful completion.

Further, we are also grateful to the college authorities who have provided best of infrastructure that has helped us throughout the seminar preparation.

Lastly, we thank our fellow students who have contributed in various ways for our project.

REFERENCES


