P2P bitcoins trading system

Gzmask

November 16, 2011

Abstract

This is a paper of a P2P system that let users trade digital works online, kind of like freelancing. There is a basic peer-review processes that guarantee the works are being accepted fairly, and there is a work quality/honest rating system to keep bad workers out.

1 Introduction

Online trading using credit cards or paypal-like system is popular. A famous exampe is Ebay.com. All these systems are centralize, thus the trades are slow and controlled by some private parties in some degree. As bitcoins emerged, pure P2P system is getting recognized. Here I proposed a very basic and simple design for a P2P trading system.

2 Important constants(Skip this if you are reading first time)

N: Total user amount (updates weekly)
K: Total voter amount (updates three days)
O: offer of the posted Need
P: process fee of the dispute.

$$P = smaller(\frac{O}{10}, \frac{O}{K})$$

R: reward for each voter.

$$R = 2 \times \frac{P}{(K \times \frac{correct\_votes}{total\_votes})}$$

3 Models

3.1 Node

This model stores in all (N) nodes
Properties

- IP_address
- Email_address
- Email_password
- work_quality
- honest_rating
- is_voter

work quality: if ratings less than 10, shows actually numbers of ratings. i.e. 5:1:0 where 5 rates excellent, 1 rates good and 0 rates bad; if ratings more than 10, only shows percentages. i.e. 33%:33%:33% This prevents profile bias towards worker nodes with less cases done.

3.2 Need

This model stores in Log(N) nodes
Properties

- time_to_live
- offer
- accepted_proposal_address
- src_node_address

3.3 Work

This model stores in posted node
Properties

- time_to_live
- proposal
- accepted
- price
- src_node_address
- des_node_address
- solution

3.4 Dispute

This model stores in posted node
Properties

- needer_ip_address
- worker_ip_address
- vote
- needer_process_fee
- worker_process_fee
- next_voter_address

4 Actions

4.1 Post a Need

1. a node U_1 boardcasts a need N_1 in the network to $Log(N)$ nodes in its node listing table.
2. after some propagation, most nodes will receive the boardcast of N_1 and store it locally
3. after the TTL expired, each node removes N_1 permanently.

4.2 Propose a work

1. any node can propose a work for a need.
2. let U_2 be a node proposing a work W_1. U_2 send an Email to U_1, telling U_1 that W_1 is proposed at U_2.
3. U_1 gets notified that W_1 is proposed. There can be multiple works that are proposed by other nodes.
4. U_1 can accept one of the proposed work, or wait. If N_1 expires, all proposed works expire at the same time and the case is over.

4.3 Accept a work proposal

1. if U_1 accepts W_1 from U_2, U_1 needs to send bitcoins to U_2's bitcoin address, at the same time sends an network packet to U_2, notifieds that W_1 is accepted and U_2 can give a solution to W_1, or the solution can be a proof-of-work-done if it is not transferable online.
2. After U_2 submits a solution for W_1, U_1 receives a network packet notification that he can review the solution from U_2. Now U_1 can either choose to accept or reject the solution.
3. when U_1 logs off, it will not longer be able to accept work solutions. U_1 is supposed to be online until a solution is accepted.

4.4 Accept a work solution

1. If the solution is accepted, U_1 will offered a rating chance to rate U_2's work quality as excellent or good.
2. If the solution is rejected, then U_1 now needs to submit a dispute D_1. The dispute will decide if U_2's work quality is good or bad. If the dispute result is good, U_1's honest rating will be deducted.

4.5 Resolve a dispute

1. nodes can choose to peer-review dispute cases to earn honest rating. The reward for an effective vote is R, that is, when the vote agrees with the end result. A dispute is resolved only after reviewed by log(K) randomly chosen nodes.
2. when U_1 files D_1, it searches for other voter Nodes and pick one randomly, say U_3.
3. U_1 sets D_1's next_voter_address to U_3, and U_3 dupes D_1 locally.
4. U_3 reviews the case, vote, and picks another voter node randomly, until reaches $U_{log(K)}$.
5. $U_{log(K)}$ boardcasts the result.

5 Super nodes: needer nodes or voter nodes

Since needers have to be staying online to wait for solutions, thus they are most likely potential voter nodes. These nodes can be seen as super nodes that can be used to handle extra services.

6 Security

It's easy to see that the system is prone to fraud client attacks. A fraud node can post needs without a deposit and the worker node giving solution to that

need therefore is cheated. I am still figuring out a decentralize mechanism to solve this problem. For now, to solve this we need a centralize Need, Dispute, Deposit and Processing Fee hosting service. Or, simply give the Needer nodes trust while the on going work is not too costly(say, answering questions).

6.1 Account reputation directory

A potential solution to the fraud client problem is to add an P2P account work_quality directory, so that Need posting nodes will have a reputation reference similar to ebay's seller ratings/feedback system.

References