Inferred based Distributed Gaming and Nash Equilibrium

Ryan J. Kung
ryankung@ieee.org
Member, IEEE Blockchain Community

April 15, 2018

1 Abstract

Since the introduction of Ethereum on 2014 [3], It brought many DApps and DGames, such as CryptoKitty, Etheremon and Decentraland. Most of them were running into issues addressed to the congestion of transaction which is limitation of how Ethereum implemented. Storing on-chain data is too expensive for games needs high interaction, and may cause may problem, one of them is that gaming on distributed system may cause harder to find out a fear nash equilibrium point for numerical designing, and the nonnegligible delay of tranasction caused may limited the interactive rate of game processing.

In this paper, we studied and discussed how should a distributed game works on blockchain based distributed system. First, we discussed distributed nash equilibrium problem on certain condition, then we introduced an infered based system based on an abstraction which formalize blockchain with STMonad and Lens application, further, we discussed how to composed infered system with zero-knowledge proof for game refining.

2 Distributed Gaming

Like traditional Gaming on game theory, we defined DGaming as a series of Gaming Behaviors and Strategis which is Distributed. With Lamport's defination on 1978 [8], A Gaming is Distributed if the message transaction delay is not negligible compared to the time between event in classic gaming behavior.

2.1 Nash Equilibrium Seeking

Traditional gaming theory may easily lead us run into issues if it's distributed, such as nash equilibrium simulation or prisoner's dilemma problem.

On classic Nash Equilibrium, we seek out the nash-eq point by analyze it with action matrix, and proof that for each strategy on game (S, f), and got A strategy profile which is Nash Equilibrium. But on distributed case, the issue is that the event of strategy is not effect immediately, nonnegligible delay of transaction on system may cause N-Eq point out of expectation.

For example, we discuss a classic game as:

	L	R
U	(1,3)	(-3,0)
M	(-2,0)	(1,3)
D	(0,1)	(0,1)

With Repeated advantage solution, we can simply figure out that the nash-eq strategy is (U, L), but this game is based on a hypothesis that player i shares same knowledge about payoff and strategy space. If players are sensitive on the uncertainty of s_{-1}, they may not choose the rational strategy, like if about %1 players choose R, then D is better than U.

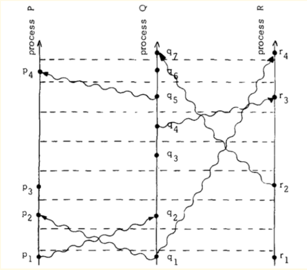

Figure 1: Lamport Timestamp

In distributed game, for example an distributed game based on a paxos based system. Addressed to the nonnegligible delay, player i may upfront know i' and action hysteretic, which may cause the nash-equilibrium become highly dependents on the distributed network. Figure 1 is showing that how events can be trigger and measured on a distributed system. Actually, on a Dgame all players may tried to mining more informations and action as later as they can, the situation may looks like Prisoners Dilemma:

	W	A
W	(1,1)	(-1, 2)
A	(2,-1)	(0,0)

On this game, players can choose from strategy space $Wait, Action$, and as what prisoner's dilemma presented to us, all players may choose W. Which entangled the gaming into a series complicated case, and break perfect information gaming in to imperfect.

Nash equilibrium seeking under communication networks is as tracing increasing attension resently, The distributed equilibrium seeking strategy(DESS) was given by [12]:

$$\dot{x} = k_i \frac{\partial f_i}{\partial x_i}(y_i), i \in \mathbb{N} \tag{1}$$

T$(\dot{x}^*, 1_N \oplus x^*)$, where x^* presents nash equilibrium. There exist Some seeking work on it with paritially coupled payoff method [11], or Communication Topologies [13] building on certain conditions and developed on the discrete-time stochastic algorithm [**?**] which the players are supposed to be capable of communicating with each other via an undirected and connected communication graph [12].

2.2 Formulation

In the game we designed, there are two tokens:

1. **Premier**: An Exchangeable ERC20 token, which can exchange with Ether and have a dynamic inflation rate $\Delta\mathbb{S}$, based on total number of player I on lamport timestamp t:

$$\Delta\mathbb{S}(t) = f \circ I(t) \tag{2}$$
$$\Delta\mathbb{S}(t) > \Delta\mathbb{S}(t') \iff I(t) < I(t') \tag{3}$$
$$\Delta\mathbb{S}(t) = \Delta\mathbb{S}(t') \iff I(t) = I(t') \tag{4}$$
$$\Delta\mathbb{S}(t) < \Delta\mathbb{S}(t') \iff I(t) > I(t') \tag{5}$$

And the total supply $\mathbb{S}$ on lamport timestamp t is:

$$\mathbb{S}(t) = \sum_{i=0}^{t} f \circ I(t) \tag{6}$$

For Premier Token, the strategy space is:

$$S_{premier} = \{B, S, H, G\} \tag{7}$$

Where:

B:= Buy Premier from Exchange
S:= Sell Premier to Exchange
H:= Hold Premier and do noting
G:= Gaming for Reward Token

2. **Reward**: An Exchangeable ERC721 token which is delegated with premier by strategy G. When players do Gaming, there is a probability P for reward getting, based on total number of player I who do game. The expected value of Reward inflation rate is:

$$E(t) = I(t) \cdot P \circ I(t) \tag{8}$$
$$P(t) > P(t') \iff I(t) < I(t') \tag{9}$$
$$P(t) = P(t') \iff I(t) = I(t') \tag{10}$$
$$P(t) < P(t') \iff I(t) > I(t') \tag{11}$$

For Premier Token, the strategy space is:

$$S_{reward} = \{B, S, H\} \tag{12}$$

Where:

B:=	Buy Reward from Exchange with Premier
S:=	Sell Reward to Exchange for Premier
H:=	Hold Reward and do noting

And the total supply $\mathbb{S}$ on lamport timestamp t is:

$$\mathbb{S}(t) \approx \sum_{i=0}^{t} E(t) \tag{13}$$

3 Infered Game

On case on time t, we have a active matrix, player 1, 2 can Gaming, Holding or Selling. Addressed to formular $(2), (3), (4)$, if 1, 2 all do gaming, they will get less payoff, and due to the inflation rate, if 1, 2 just holding, they will lose money which relative to. And if 1 is gaming, and 2 is Selling, because of the decreasing of total supply, seller will get more profit:

	G	H	S
G	(1, 1)	(2, -1)	(2,2)
H	(1, 2)	(-1, -1)	(-2, 1)
S	(2,2)	(1,0)	(0, 0)

(14)

With PAS:

	G	S
G	(1, 1)	(2,2)
S	(2,2)	(0, 0)

(15)

In this paper, we considered a game (S, f) with a set of player $\mathbb{N} = 1, 2, ..., N$, under a distributed network which has nonnegligible delay δ, all player i can well access actions of $-i$. We define a mix strategy function $\sigma = f(\delta, s)$ which is means how the information is prefect for s_i. And formulated the mix strategy function based on nonnegligible transaction as $sigma(\delta, \dot{x})$.

If δ for DGame (18) is negligible, $(\sigma(G), \sigma(S))$ should be $(\frac{1}{2}, \frac{1}{2})$, otherwise player i may known the action $-i$ does early, which may shifts the distribution to $(0, 1)$. And if most of players does same, it will fall into a zero-sum game trap. We introduce infered game for solving this problem.

3.1 STMonad and Lens

An infered Game is that for a game (S, f), all players i shares same knowledge space about S, $u_i(s)$, and s_{-i} on Lamport Timestamp τ. Unlike classic distributed system, Since some formalization of blockchain is based on the separation logic or $\pi calculis$ [2], it can easily be modeled as a time step function, or a FRP issue which can be abstract as [6], where blockchain Status can be also be abstract as $STMonad$ like MAKERDao [10]'s work:

$$time : STMonad_{time} \tag{16}$$
$$at[time]t = t \tag{17}$$

In blockchain, there is only one types of monad, the STMonad, $STMonad$ is a Monad type for status, thus except status changes, all other functions should be pure which means no side-effect, which means we can build an infered system or a $LENS$ [1] above it. A Lens should have two property, $over$, and $view$.

```
data Lens a b = Lens
    { view :: a -> b
    , over :: (b -> b) -> (a -> a)
  }
```

3.2 Consistency of Blockchain

In solidity, a pure function which means nor cost gas or cause side-effect on blokchain it can be infered based on ABI is a *Lens.view*, otherwise it's a *Lens.over* which means you have to send a transaction to modified the status of chain.

Consider the Gaming strategy G, on time t, all players i should know $I(t)$ as *Lens.view*, and calculate out the utility function of selling or mining with q tokens with cost c:

$$u(q_i) = q_i p(q) - c_i(q_i) \tag{18}$$
$$p(q_i) = f(\mathbb{S}(t)) \tag{19}$$

Like cournot duopoly model, with a two player $\{1, 2\}$ gaming, we can directly give the nash equilibrim of S is:

$$\dot{q}_1 = \dot{q}_2 = \frac{1-c}{3} \tag{20}$$

And when i trying do action, it's a *Lens.over* which overwrite the status of contract storage. Based on the Infered system, $-i$ can access S_i, so as matrix presented in 18, -1 will prefer to select S which means higher payoff. Actually according to symmetry, when i choose S, $-i$ will always choose G, so that σ should be $(\frac{1}{2}, \frac{1}{2})$ which is as same as single-process gaming.

So since classic distributed gaming may transfer the gaming from perfect information to imperfect. But if it's blockchain based, since *over* is working as eventual consistency [4] but the status of blockchain can be seems as strong consistency [**?**]. On *view* phase of gaming, when player i make decision on $STMonad\ time_0$, the information of $Monat$ is perfect. For phase *over*, players may choose to pay high price as strategy to get more payoff, since it's based on consistency information of blockchain, we can also say that the *over* phase is also perfect information gaming. So for a gaming we wished to have a fixed nash equilibrium point, it should be designed as all data in public and no private status.

3.3 Zero Knowledge Game

Such as we can formalize blockchain as a Monad (A monoid in the category of endofunctors), We can also describe abstractly the cryptology part of bitcoin as a cyclic subgroup of elliptic curve group over a primer finite field, which is also the principle of zk-SNARKs. There is an concrete implementation in our game for providing zero-knowledge proofing based on the infered system and hash algorithms, It's works like other blockchain implementation does [5]. But what further more is, we tried developed algorithms based on Abstract Algebra Type and Isomorphism or Bijection mapping.

The Abstract Algebra Type made an allowance for playing high abstract part of calculators in a more intuitionistic way. Such as when we try to implement ethereum's pubkey/privatekey algorithm, we can just simply write as [7]:

```
def pubkey(priv: CF) -> ECG:
    return ECG(G @ priv)
```

Where CF and ECG is Finite Field Cyclic Group and Elliptic Curve Group of Secp256k1. By the property of cyclic subgroup, we can easily improve the zero knowledge gaming as some endomorphism feature.

$$A + B = X \tag{21}$$
$$A + C = X \tag{22}$$
$$C \neq X \tag{23}$$

4 Concrete

We abstract the gaming activity as mining, and designed a series of numerical system, which can maintain the value of user actor by a fixed and well-designed nash equilibrium pointed.

4.1 Infered based Token

The ERC20 standard was extended with a built-in transparen exchange, which makes all gaming actor can be limited inside the blockchain or contract. And based on ERC721, we designed a reward system to keep player can get payoff if they act around the nash-eq point. For Exchangeable ERC20 and so on the Exchangeable ERC721, all matching stuffs is based on infered system, which means players can check the full depth of ticker for determining the best price they should pay for tokens.

Functions of smart contract are separated into Functor and Monad. Functors are working based on pre-infered system, and Monads are based on infered system, which is actually presented a sequence of status.

We support below ABIs:

function	type	event
checkBidTicker	Lens.view	False
checkAskTicker		
matchBid		
matchAsk		
ask	Lens.over	True
bid		
fillBid		
fillAsk		
cancelAsk		
cancelBid		

(24)

On out implementation of Gaming Token, we designed an infered based auto expended rate. The expand rate is based on $I(t)$ and *time* of blockchain, which is bounded on the nash equilibrim of what the token price should be during the game processing.

In contract PremierToken.sol, we developed a functional modifier for infered system, which works like a lazy evaluation. Above on ERC20 standard, there is two kind of balance:

1) infered balance

Infered balance is *Lens.view* which implement a measurement of the real balance should be. If a function is *Lens.view*, it will check infered balance, else it will evaluation the infer modifier and setup infered value to storage of contract.

2) real balance

Real balance is also *Lens.view* which is just returning real the balance in storage which is excluded infered value.

Standard function transfer, transferfrom and approve are all modified by Infered modifier.

4.2 Non-fungible Token

The most obviously case of nash-eq case on blockchain world is bitcoin mining [9], we can easily build an game matrix model on bitcoin mining behavior, the relationship is between market force, difficult rate, and hash rate. And in game, players can pay game token for wining reward, more token they paid, higher rewarding rate they got, which is working as a simulation of how bitcoin mining works. The Reward token is a ERC720 token, which is also mapping to IPFS hash address. Like Premier Token, Reward token also has it's expand rate which is based on $I(t)$ and t. When gaming, reward token can be immediately infered by the context like tx Hash, sender address or blocktime of blockchain Premier token paid.

For increasing value of NFT Reward Token, we designed several artwork for each logic layer of token which is storing on IPFS, which can be composed in front end of DApps for print and Zero-knowledge Transfer.

A Zero-knowledge Transfer is a transfer with zero context, which is simply developed on freeze and unfreeze strategy. When user transfer Reward Token with Zero-knowledge Transfer, the token id will be freezed on contract for a unique untraceable hashid thus receiver can redeem the hashid that has no idea where the Token is come from.

5 Conclusion

Addressed to the strong consistency property of blockchain, A game can have a certain nash equilibrium point, if the side-effects and pure-functions are in separation.

References

[1] Control.lens.tutorial.

[2] Linear types can change the blockchain!

[3] Vitalik Buterin. A next-generation smart contract and decentralized application platform, 2014.

[4] Roger Wattenhofer Christian Decker, Jochen Seidel. Bitcoin meets strong consistency.

[5] Christina Garman Eli Ben-Sasson, Alessandro Chiesa. Zerocash: Decentralized anonymous payments from bitcoin.

[6] Conal Elliott and Paul Hudak. Functional reactive animation. In *International Conference on Functional Programming*, 1997.

[7] Ryan J. Kung. Klefki is a playground for researching elliptic curve group based cryptocoins.

[8] Leslie Lamport. Time, clocks and the ordering of events in a distributed system. pages 558–565, July 1978.

[9] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system.

[10] Mikael Brockman Nikolai Mushegian, Daniel Brockman. Reference implementation of the decentralized dai stablecoin issuance system, 2 2018.

[11] L. Wang, S. Liang, and Y. Hong. Discrete-time algorithm for distributed nash equilibrium seeking of a class of aggregative games. In *2017 36th Chinese Control Conference (CCC)*, pages 11325–11330, July 2017.

[12] M. Ye and G. Hu. Distributed nash equilibrium seeking by a consensus based approach. *IEEE Transactions on Automatic Control*, 62(9):4811–4818, Sept 2017.

[13] M. Ye and G. Hu. Distributed nash equilibrium seeking in multiagent games under switching communication topologies. *IEEE Transactions on Cybernetics*, PP(99):1–10, 2017.