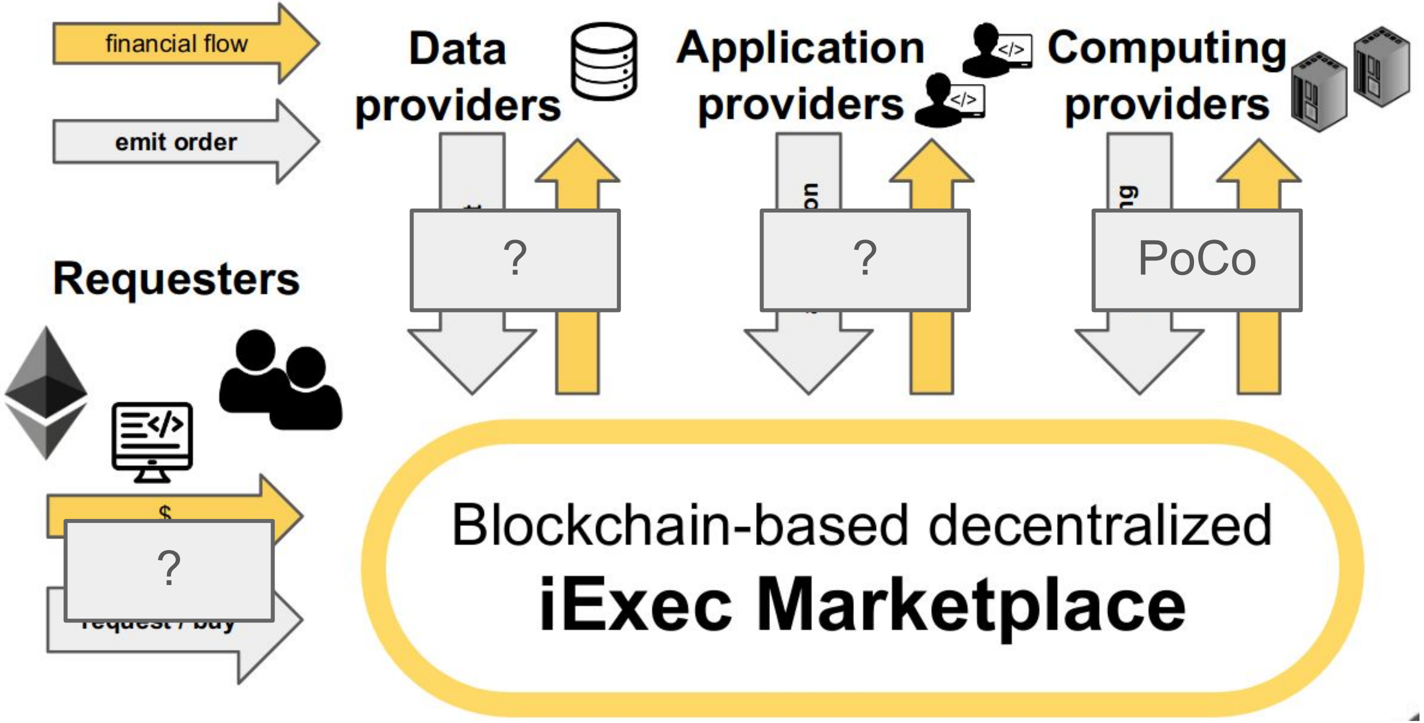
Incentivization of correct behavior in a decentralized computing marketplace

Henry MONT
Intern – DRIM

25/07/2024

Identified problem



⇒ Currently no incentive mechanisms for Requesters, Data providers and Application providers.

Can we access worker logs ?

1. I explored thegraph from bellecour.
2. I found information about tasks (We can see the difference between transactions reaching or not a consensus).
3. I did not find a way to check worker logs and errors during computation.
4. I could however retrieve task status, requester, dApp provider, dataset provider and workerpool owner. Should we try looking for pattern on failed tasks ?

Game Theory based model [4]

Requester: S

Asset \ **Computing provider**	**S**	**F**
S	($U_r-(Price_a+Price_c)$, $Price_a$, $Price_c-Cost_c$)	(0 , 0, $-(Cost_c+Slash_c)$)
F	(0, 0, $-(Cost_c+Slash_c)$)	(0, 0, $-(Cost_c+Slash_c)$)

Requester: F

Asset \ **Computing provider**	**S**	**F**
S	(0 , 0 , $-(Cost_c+Slash_c)$)	(0 , 0, $-(Cost_c+Slash_c)$)
F	(0, 0, $-(Cost_c+Slash_c)$)	(0, 0, $-(Cost_c+Slash_c)$)

Players:

- Requester
- Asset provider
- Computing provider

Strategies:

- S: Do the work correctly
- F: Do the work incorrectly

Payoffs:

- (Requester, Asset, Computing)

U_r: Utility from the result
$Slash_c$: Computing provider slash
$Price_a$: Asset price
$Price_c$: Computing price
$Cost_c$: Computing cost

Game Theory based model: example 1

Requester: S

		Computing provider	
		S	**F**
Asset	**S**	$(U_r-(Price_a+Price_c)$, $Price_a$, $Price_c-Cost_c$)	(0 , 0, $-(Cost_c+Slash_c)$)
	F	(0, 0, $-(Cost_c+Slash_c)$)	(0, 0, $-(Cost_c+Slash_c)$)

Requester: F

		Computing provider	
		S	**F**
Asset	**S**	(0 , 0 , $-(Cost_c+Slash_c)$)	(0 , 0, $-(Cost_c+Slash_c)$)
	F	(0, 0, $-(Cost_c+Slash_c)$)	(0, 0, $-(Cost_c+Slash_c)$)

Strategies:

- Requester plays S
- Asset provider plays S
- Computing provider plays S

Result:

- Task completed successfully

U_r: Utility from the result
$Slash_c$: Computing provider slash
$Price_a$: Asset price
$Price_c$: Computing price
$Cost_c$: Computing cost

Game Theory based model: example 2

Requester: S

		Computing provider	
		S	**F**
Asset	**S**	$(U_r-(Price_a+Price_c)$, $Price_a$, $Price_c-Cost_c$)	(0 , 0, $-(Cost_c+Slash_c)$)
	F	(0, 0, $-(Cost_c+Slash_c)$)	(0, 0, $-(Cost_c+Slash_c)$)

Requester: F

		Computing provider	
		S	**F**
Asset	**S**	(0 , 0 , $-(Cost_c+Slash_c)$)	(0 , 0, $-(Cost_c+Slash_c)$)
	F	(0, 0, $-(Cost_c+Slash_c)$)	(0, 0, $-(Cost_c+Slash_c)$)

Strategies:

- Requester plays S
- Asset provider plays F
- Computing provider plays S

Result:

- Task failed

U_r: Utility from the result
$Slash_c$: Computing provider slash
$Price_a$: Asset price
$Price_c$: Computing price
$Cost_c$: Computing cost

Game Theory based model: example 3

Requester: S

		Computing provider	
		S	**F**
Asset	**S**	$(U_r-(Price_a+Price_c)\ ,\ Price_a\ ,\ Price_c-Cost_c)$	$(0\ ,\ 0,\ -(Cost_c+Slash_c))$
	F	$(0,\ 0,\ -(Cost_c+Slash_c))$	$(0,\ 0,\ -(Cost_c+Slash_c))$

Requester: F

		Computing provider	
		S	**F**
Asset	**S**	$(0\ ,\ 0\ ,\ -(Cost_c+Slash_c))$	$(0\ ,\ 0,\ -(Cost_c+Slash_c))$
	F	$(0,\ 0,\ -(Cost_c+Slash_c))$	$(0,\ 0,\ -(Cost_c+Slash_c))$

Strategies:

- Requester plays F
- Asset provider plays S
- Computing provider plays F

Result:

- Task failed

U_r: Utility from the result
$Slash_c$: Computing provider slash
$Price_a$: Asset price
$Price_c$: Computing price
$Cost_c$: Computing cost

Game Theory based model

Requester: S

		Computing provider	
		S	**F**
Asset	**S**	$(U_r-(Price_a+Price_c)$, $Price_a$, $Price_c-Cost_c$)	(0 , 0, $-(Cost_c+Slash_c)$)
	F	(0, 0, $-(Cost_c+Slash_c)$)	(0, 0, $-(Cost_c+Slash_c)$)

Requester: F

		Computing provider	
		S	**F**
Asset	**S**	(0 , 0 , $-(Cost_c+Slash_c)$)	(0 , 0, $-(Cost_c+Slash_c)$)
	F	(0, 0, $-(Cost_c+Slash_c)$)	(0, 0, $-(Cost_c+Slash_c)$)

Strategies:

- S: Do the work correctly
- F: Do the work incorrectly

Results:

- Red Nash Equilibrium leads to failure but are unstable.
- Green Nash Equilibrium lead to success and is stable.

U_r: Utility from the result
$Slash_c$: Computing provider slash
$Price_a$: Asset price
$Price_c$: Computing price
$Cost_c$: Computing cost

Game Theory based model, with dynamic cost

Requester: S

Asset \ Computing provider	**S**	**F**
S	(U_r-($Price_a$+$Price_c$), $Price_a$-$Cost_a$, $Price_c$-$Cost_c$)	(0, -$Cost_a$, -$Slash_c$)
F	(0, 0, -($Cost_c$+$Slash_c$))	(0, 0, -$Slash_c$)

Requester: F

Asset \ Computing provider	**S**	**F**
S	(0 , -$Cost_a$, -($Cost_c$+$Slash_c$))	(0, -$Cost_a$, -$Slash_c$)
F	(0, 0, -($Cost_c$+$Slash_c$))	(0, 0, -$Slash_c$)

Strategies:

- S: Do the work correctly
- F: Do the work incorrectly

Results:

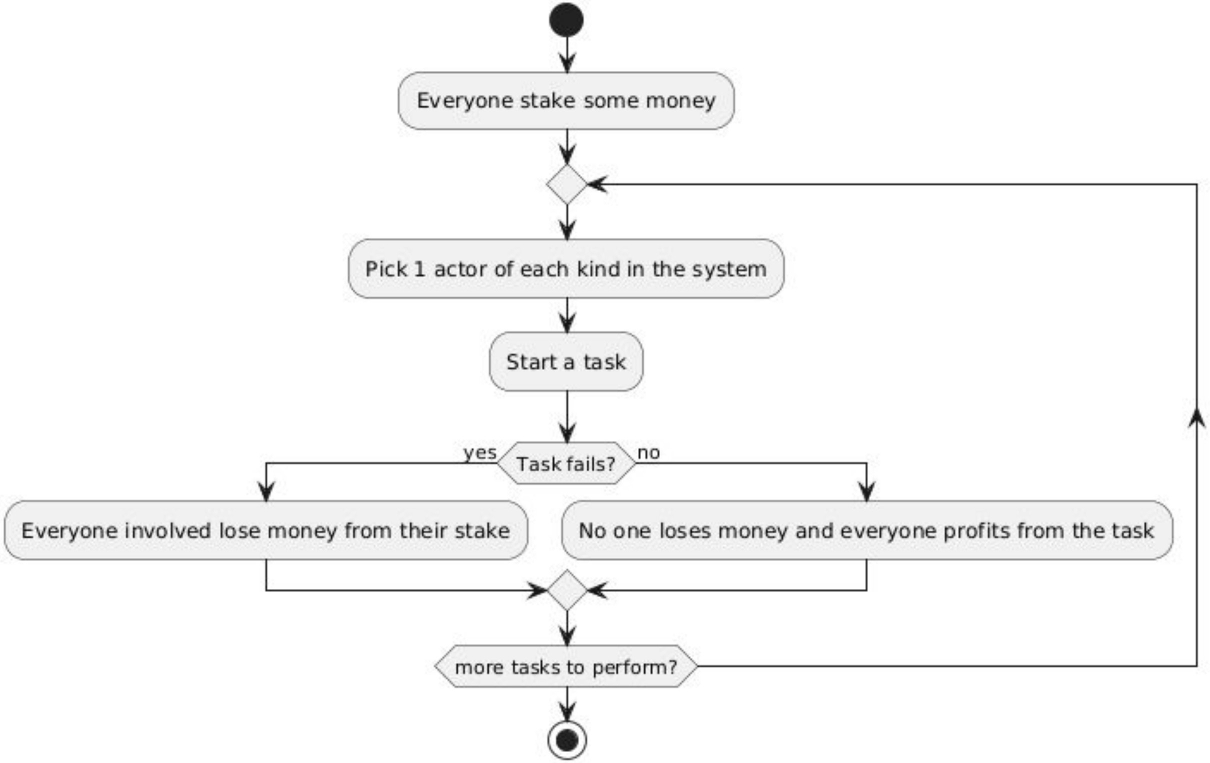
- Red Nash Equilibrium leads to failure but are unstable.
- Green Nash Equilibrium lead to success and is stable.

U_r: Utility from the result
$Slash_c$: Computing provider slash
$Price_a$: Asset price
$Price_c$: Computing price
$Cost_c$: Computing cost

Blind slashing system

Malicious actors will be involved in failed tasks more often and will lose more money.

⇒ Everyone will have an incentive to improve their behaviour. [6, 7]



Game Theory based model with blind slashing

Requester: S		Computing provider: S	Computing provider: F
Asset	S	(U_r-($Price_a$+$Price_c$), $Price_a$, $Price_c$-$Cost_c$)	(-$Slash_r$, -$Slash_a$, -($Cost_c$+$Slash_c$))
Asset	F	(-$Slash_r$, -$Slash_a$, -($Cost_c$+$Slash_c$))	(-$Slash_r$, -$Slash_a$, -($Cost_c$+$Slash_c$))

Requester: F		Computing provider: S	Computing provider: F
Asset	S	(-$Slash_r$, -$Slash_a$, -($Cost_c$+$Slash_c$))	(-$Slash_r$, -$Slash_a$, -($Cost_c$+$Slash_c$))
Asset	F	(-$Slash_r$, -$Slash_a$, -($Cost_c$+$Slash_c$))	(-$Slash_r$, -$Slash_a$, -($Cost_c$+$Slash_c$))

Results:

- Same result as before
- Incentive to switch more quickly to the stable Nash Equilibrium
- Possibility for repeatedly failing actors to get ruined

U_r: Utility from the result

$Price_a$: Asset price
$Price_c$: Computing price
$Cost_c$: Computing cost

$Slash_r$: Requester slash
$Slash_a$: Asset provider slash
$Slash_c$: Computing provider slash

Not possible to get rid of all Nash Equilibriums

1. If more than one player is failing the task, other players are blocked in a suboptimal outcome. Then changing their strategy will not change the game outcome ⇒ non desirable Nash Equilibria.
   a. Build trust between players so that they believe the other will switch strategies as agreed ⇒ reputation system encourages belief in other players right choice and coordination.
   b. A punition (slashing) mechanism align individual incentives with collective outcomes ⇒ encourage the players to coordinate on a better strategy.

Blind slashing system and ruin theory [8]

The overall task failure rate in the system can be written as:

$$F = 1 - (1 - F_u) \cdot (1 - \mathrm{mean}(F_u))^3$$

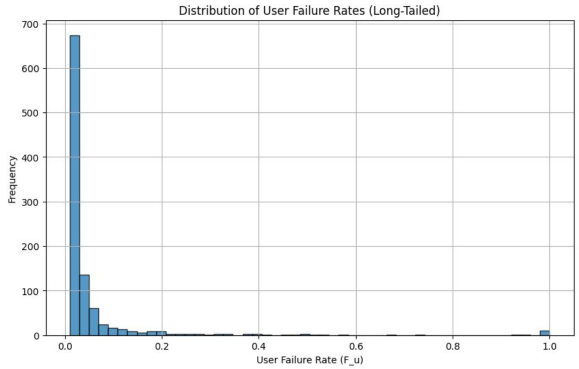
For a given participant, the expected loss per task is:

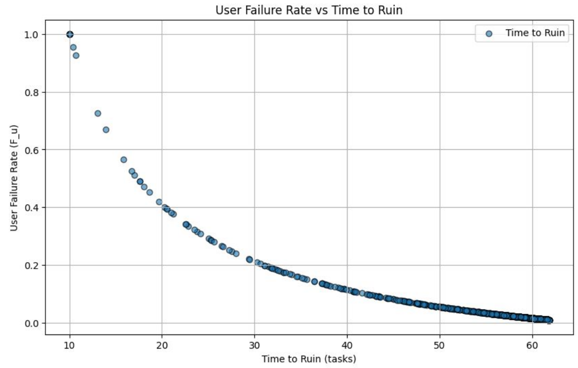
$$L = F \times P$$

So the time (number of task) to ruin that participant is:

$$T = \frac{S_0}{L} = \frac{S_0}{F \times P}$$

T: Time to ruin	F: Task failure rate
S_0: Initial stack	F_u: User failure rate
L: Loss per task	P: Penalty for failure





Blind slashing system: shortcoming

If we want users to be profitable, 2 options:

- F is null ⇒ **no room for error**.
- P is null ⇒ **no slashing mechanism**.

$$L \leq 0 \Leftrightarrow F \times P \leq 0 \Leftrightarrow \begin{cases} F \leq 0 \\ P \leq 0 \end{cases}$$

F: Task failure rate
P: Penalty for failure
L: Loss per task

Blind slashing/recovery system and ruin theory

The expected loss per task without recovery:

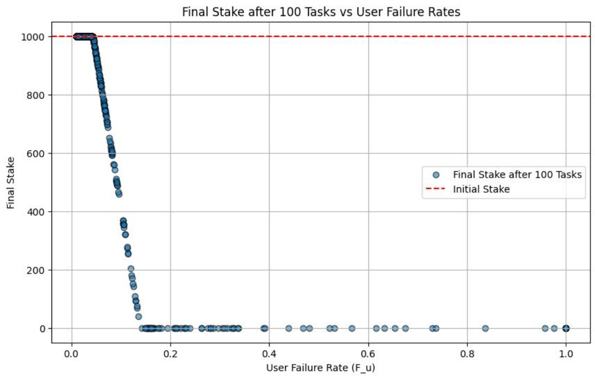
$$L = F \times P$$

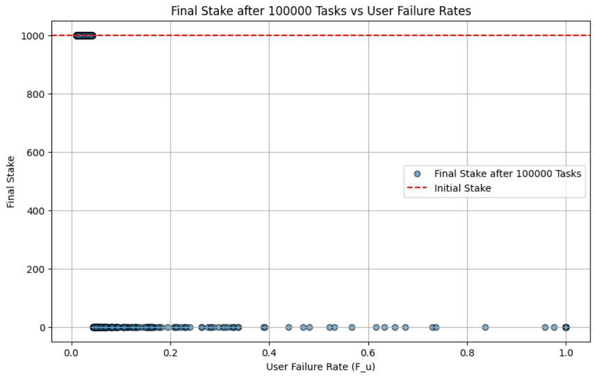
The expected recovery per task:

$$R = (1 - F) \times R$$

The net expected loss in stake per task:

$$L_{\text{net}} = L - R = F \times P - (1 - F) \times R$$





L_{net}: Net loss per task
R: Recovery for success
F: Task failure rate
P: Penalty for failure

Bibliography

[1] Ethereum Foundation. "Ethereum Whitepaper: A Next-Generation Smart Contract and Decentralized Application Platform." Ethereum, 2014. Accessed July 8, 2024. https://ethereum.org/en/whitepaper/.

[2] Fedak, Gilles, Wassim Bendella, Eduardo Alves, Haiwu He, and Mircea Moca. "Blockchain-Based Decentralized Cloud Computing Whitepaper. Version 3.0, 2017-2018." April 24, 2018. Accessed July 2, 2024. https://github.com/iExecBlockchainComputing/whitepaper/blob/master/V3/iExec-WPv3.0-English.pdf.

[3] iExec. "Proof of Contribution." iExec Documentation. Accessed July 8, 2024. https://protocol.docs.iex.ec/key-concepts/proof-of-contribution.

[4] Binmore, K. G. Game Theory : A Very Short Introduction. New York : Oxford University Press, 2007. http://archive.org/details/gametheoryverysh0000binm.

[5] Hasan, Omar, Lionel Brunie, and Elisa Bertino. “Privacy-Preserving Reputation Systems Based on Blockchain and Other Cryptographic Building Blocks: A Survey.” ACM Computing Surveys 55, no. 2 (January 18, 2022): 32:1-32:37. https://doi.org/10.1145/3490236.

[6] Goeschl, Timo, and Johannes Jarke. “Non-Strategic Punishment When Monitoring Is Costly: Experimental Evidence on Differences between Second and Third Party Behavior,” n.d.

[7] Fatas, Enrique, Antonio J. Morales, and Paloma Ubeda. “Blind Justice: An Experimental Analysis of Random Punishment in Team Production.” Journal of Economic Psychology 31, no. 3 (June 1, 2010): 358–73. https://doi.org/10.1016/j.joep.2010.01.005.

[8] Asmussen, Soren, and Hansjorg Albrecher. Ruin Probabilities (2nd Edition). World Scientific, 2010.