BlueChain: An Innovative Way to Reward Gamers

Abstract. BlueChain is a blockchain platform, that would allow the games, to be able to have their own financial system, and while generating new blocks, it tries to reward gamers in a fair and secure way. The coins can be generated by the amount of time that the players are playing the game. Players can use the generated coin to buy game assets or trade it. We just propose a solution on how every node can reach to this consensus without involving a third authority party.

Introduction

Playing a game is always fun and Crypto is something that everybody talks about nowadays!
We have decided to combine those two and motivate people to play more and earn crypto while they are doing it. What is better than earning some money while you are playing a game?
The only question was how to make sure, a fair, transparent and secure process is conducted to earn crypto and this question is going to be answered by our solution. This was inspired by "Proof of work[1]" but with a little difference. This algorithm starts from a really difficult puzzle and the more you play, the easier it gets. So at the end when a player finishes the game, the difficulty of the puzzle to be solved is easy enough to generate a block in less than 10 minutes (default value) but the reward is probably much less. The algorithm encourages players to play more and earlier on the way, and all the coins populated by players have a limit. These coins can be used for trading or for assets in the game.

Proof of Playtime

First let's introduce some prerequisites. This blockchain can have multiple chains, each chain belongs to each game, and each game has its own parameters.

Target:
Target is a 256 bit number and the Hash of a valid block should be lower than it. Target defines the difficulty of that chain for the game. The higher the target, the lower the difficulty.

---

[1] https://bitcoin.org/bitcoin.pdf

Game codes:
These are 64 bit numbers that the game developers hide this values in the gameplay and these codes can modify the target of the chain for that game.

P(X):
This can be implemented by a smart contract on the chain easily, it's just a dictionary that maps the hash of game codes to a value which can be used to increase or decrease the difficulty of finding the right "nonce" for the next block and also it can only be modified by the game developer.

Every node should try to find a nonce and append that to the time of the transaction, in a way that the hash of the block should be less than the target of the game defined on the blockchain.
At first, the target of each game is so low and the difficulty is so high that it's really hard to find a nonce to satisfy that condition, but by playing the game, it will provide game codes, that can decrease the difficulty and over time, depending on the number of the blocks, the difficulty of the blockchain for that game will get decreased.
So for each game on blockchain, reward and difficulty is going to be defined like the following:

$$Reward = \frac{K}{2^{\lfloor \frac{CB}{T} \rfloor}}$$

$$Target = P(x).2^{\lfloor \frac{CB}{T} \rfloor}.Target$$

A target of the blockchain for a game will also be recalculated after n transaction:

$$Avg(Last_N) > (1 + Margin).TB => Target = \frac{Target}{CO}$$

$$Avg(Last_N) < (1 - Margin).TB => Target = CO.Target$$

Reward is the amount of coins generated for that game on this blockchain. A valid block should have a hash lower than the target as explained above. T is the number of blocks achieving reward until next halving happens. CB is the number of valid blocks. P(x) is a function accepting a game code and if hash of x is present, it returns a factor to multiply the target to decrease or increase the difficulty of finding the right hash.
These codes can be achieved by playing the actual game, and the game server can hide this code on its own gameplay. This function is defined by the game developer, and it's just a Map with the hashed keys of the codes and the factors to make finding new hash easier or more difficult. The code in that function is only used once, so after using an x and having a block with that, then x can not be used anymore.

Part of the block that should be broadcasted by nodes has this structure:

Previous hash (256 bit)	Nonce (32 bit)	X (64 bit)	Timestamp (64 bit)

A valid block should satisfy these rules:

- In case of blocks with same previous hash, the one with higher Target and lower timestamp will be chosen.
- $Hash(Block) < Target$
- A timestamp is accepted as valid, if it is greater than the median timestamp of previous 11 blocks, and less than the network-adjusted time + 2 hours. "Network-adjusted time" is the median of the timestamps returned by all nodes.
- Side chaining is not allowed in this blockchain so a block with a previous hash point to more than 6 previous blocks backward is invalidated.

The max supply of the coin that can be earned for each game, can be calculated by this:

$$MaxSupply = \Sigma_{CB=0}^{n} \frac{K}{2^{\lfloor \frac{CB}{T} \rfloor}} = K\Sigma_{i=0}^{n} T\frac{1}{2^i} = KT\Sigma_{i=0}^{n}\frac{1}{2^i} = KT(1+\Sigma_{i=1}^{n}\frac{1}{2^i}) \approx 2KT$$

So the variables that a game developer should decide about it are in this table:

Variable	Desc	Default
K	The first given reward value	50
T	How many transaction needed for the next halving of reward	210000
Target	The first minimum target	0xEE9BFAB4
Margin	The margin that is used for recalculating Target	0.4
Last_N	The number of last transaction that is being used for recalculating Target	2016
TB	The amount of seconds system wants rewards to get generated	600
CO	The coefficient that is being used to recalculate Target	2

The default values in the table are set in a way that a system which can produce 5KH/s, after approximately an hour, has a good chance to find the right nonce to satisfy validity conditions of the block. When gamers start playing the game and they find the clues in the game (the game codes) then by having P(x) function, the difficulty will get decreased more and more and eventually system tries to control it to have a target leading to generate a block within TB amount of time.