Distributed Engineered Autonomous Agents : Satoshi Fantasy

Jay Y. Berg
info@satoshifantasy.com

April 2014

1 Introduction

The Byzantine battle plan is for each division to attack simultaneously from separate locations. The exact time of the attack will be communicated from one general to another by messenger. The problem is, how does a general know when everyone received the message? How long must he wait? Using acknowledgement messages presents the same problem as the original message. Then there is a problem of trust, if even one messenger is a traitor, the will be a fork in the plan, with some generals using the bad data.

Distributed system designers are notorious for describing their systems first by its known limitations, and then by the practical workaround. The *Byzantine Generals' Problem* and its impossibility proofs are the root cause of these limitations. [1]

This paper details Satoshi Fantasy, a distributed autonomous fantasy football game named after Satoshi Nakamoto, the creator of bitcoin. Bitcoin is considered the first successful distributed autonomous system. Its main innovation is the the proof-of-work block-chain. Although bitcoin does not solve the the general *Byzantine Generals' Problem* [2]. In the context of a peer-to-peer electronic cash system, it is a"good enough" pragmatic engineered solution to the problems of trust and time. Satoshi Nakamoto has brought esoteric distributed system design problems into the mainstream consciousness, and with it a newfound public trust for logical, time-tested, distributed consensus protocols.

Overview

Satoshi Fantasy is a game that uses Fantasybits as tokens based on fantasy football points. Fantasybits are a representation of a fantasy football players skills. By using a proof-of-skill block-chain, the network is ultimately controlled by the most skilled and knowledgeable fantasy football players. Each player has a unique Fantasy Name that is tied to a cryptographic private key. Problems of trust and time, common to all distributed protocols, are solved within this context.

Driving Factors

CJ Spiller, running back for the Buffalo Bills was one of the biggest fantasy busts in 2013. After averaging 6 yards per carry in 2012, he was was being drafted in the first round of most 2013 fantasy drafts [3]. What if a fantasy expert saw this coming? He knew that Spiller was way over-valued. How can he turn this knowledge into value?

Over the past 10 years fantasy football has tripled in size [4]. In 2013, the High Stakes Fantasy Football Players Championship had over 5800 entries and over 6 million in payouts [5]. Fantasy football is played by drafting a team, and submitting your lineup each week. Some leagues allow trading players, but the high-stakes public leagues do not. There are also weekly games and a thriving professional expert service industry.

Currently if someone wanted to "sell" Spiller, he could *a*) trade him, if owned; *b*) not draft him; *c*) blog or tweet about it; and *d*) not pick him in weekly fantasy games.

Satoshi fantasy will enable buying and selling a players fantasy season production with fantasybits. Fantsybits are acquired by accurately projecting weekly fantasy point results.

2 Fantasybits

Fantasybits are created for each fantasy point scored by an NFL player. Unlike digital currency, there is no pre-distribution, distribution schedule, mining, or minting. This feature enables the use of a proof-of-skill block-chain discussed in section 5. Fantasybits are awarded to fantasynames, based on weekly projections of fantasy results. Fantasybits are fungible, and each one has a season and a real players name attached to it. **The value of fantasybits come from the fact that they are scarce, difficult to acquire, and are needed for buying and selling players.**

Projections

During each week of the fantasy season, up until kickoff, projections can be made. All that is needed to make projections is a fantasyname. This is done by signing a pointProjection event and sending it to the network. Every projection that is in the block-chain, on time, is eligible for a payout. Once a block containing the consensus results is received, a deterministic distribution algorithm is run to determine the payouts to each player.

Distribution Algorithm: (see appendix B) Let R equal actual results and p_n equal the projections made by each player. Take the difference of the projection from the results

$d(p)$. and then get the average difference $\overline{D}$.

$$d(p) = |R - p|$$

$$\overline{D} = \sum_n |R - p_n|$$

Filter out projections that are below average or are 100% or more off the mark $F(d)$.

$$F(d) = \begin{cases} 0 & \text{if } d > \overline{D} \text{ or } d > R \\ 1 & \text{otherwise} \end{cases}$$

Calculate unitpayout X, which will distribute more coins to the better predictions.

$$X = \frac{R}{\sum_n ((R - d(p_n)) \times F(d(p_n)))}$$

Finally, the award function $A(p)$ determines how many fantasy points are awarded for each projection, this is multiplied by 100 for fantasybits.

$$A(p) = X \times (R - d(p)) \times F(d(p))$$

Any points leftover, L, due to bad or no projections get distributed to the block signer .

$$L = R - \sum_n A(p_n)$$

3 Fantasy Name

Since there is no cost to making projections, the system is vulnerable to a Sybil attack. An attacker would write a program to create millions of fantasy names in an attempt to control the network. To mitigate this risk, there needs to be some kind of cost or effort involved in receiving a fantasy name. **Each player mines his own fantasy name into existence by solving a cryptographic hash.** Once mined, a nameProof event is triggered and this proof-of-work will eventually get verified by the peer network.

Score and Rank

There are four distinct ways of keeping score of fantasybits and rank of fantasy names. These values are used to determine ones ability to sign blocks see section 5.

Skill: gross total fantasybits earned; can only increase; not transferable

Data: data feed ranking; cannot be less than Skill; can be assigned to an agent.

Stake: net present fantasybits balance; transferable

Time: time-sync rankings; cannot be less than Stake, can be assigned to an agent.

Agents

Each fantasy name may assign a peer to be their agents of data and time. The skill and stake values of the assigner will be added to the data and time rankings of the agent respectively. If a fantasy name assigned herself as her own agent, she becomes a volunteer and must be willing to do the same for the entire network. By default agents are assigned as default consensus, which are decided by skill and stake consensus proofs.

If a single agent represents a majority of the network, she becomes an oracle, and a central point of control for those critical time periods, such as during live games. In fact, agents are just fantasy names appointed by the consensus of the network. Free market economics forces would choose the most capable agents.

4 State Machine

The underlying protocol changes it behavior based on its current state, there are also different block-chain and event/transaction rules, see section 5 . A transition to a different state is accomplished with signing and publishing a block. The particular proof required to sign a block, also depends on the transition context. See figure 1.

Events

A transaction that transfers fantasybits, is only one of many event types. In contrast to bitcoin, the transfer transaction is not a core feature. Point projections followed by fantasy name mining are core features in satoshi fantasy. Following is the list of events, in order of significance.

nameProof
: contains proof-of-work data, sent by new players to claim their fantasy name, signed by fantasy name.

pointProjection
: contains the playerID, week, and point projection. sent and signed by fantasy name.

dataTransition
: contains game results, draft results, player meta-data or schedule data, sent by a data agent, signed by consensus of skill.

timeTransition
: contains trading session, exchange events, or any time ordered data, sent by a time agent, signed by consensus of stake.

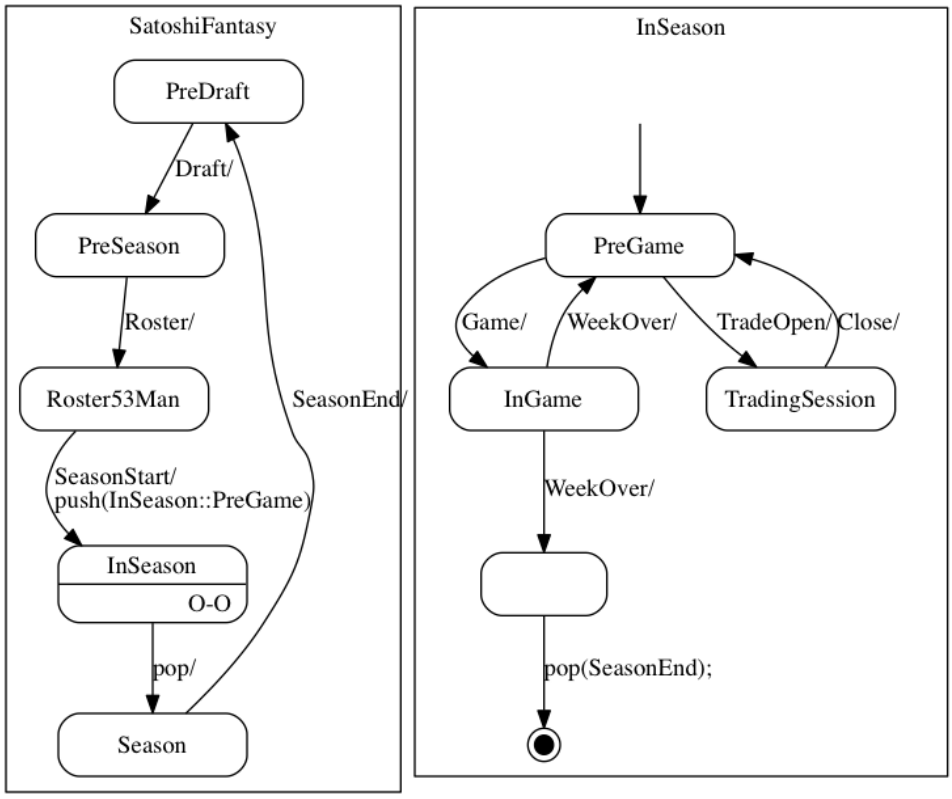

Figure 1: state machine

exchangeOrder
: contains limit order, price, quantity, playerID. first created and signed by fantasy name, then stamped and signed by time agent.

transferTransaction
: contains amount to transfer, sender and receiver fantasy names, signed by sender.

Deterministic Transactions

A state transition event can trigger multiple events and transactions.

coinbaseTransaction
: awards new fantasybits based on the distribution algorithm. triggered by a data-Transition "WeekOver" event. See section 2 and figure 1 .

exchangeExecution
: contains fills, and order status, generated by internal matching-engine (section 6) triggered by timeTransition "TradeOpen" followed by multiple exchnageOrder events.

clearingTransaction
: contains transferTransation events, generated by internal engine. triggered by a timeTransition "TradeClose" followed by multiple exchangeExecutions.

5 Block Chain

In bitcoin, a block chain is an ordered sequence of blocks, with each block containing an unordered set of transfer transactions. A miner does his proof-of-work computations based on the the previous block(s). This solves the problem of double spending, where the same bitcoin is simultaneously transferred to two different addresses. **The core event in satoshi fantasy, pointProjections, do not transfer funds, so double spending does not apply.**

The satoshi fantasy block-chain is path dependent. Current events determine when a block will be created and what kind of proofs are needed. The block-chain is really two different block-chains; 1. proof-of-skill containing nameProofs, pointProjections and dataTransitions events. 2. proof-of-stake containing timeTransitions, exchangeOrders and transferTransactions events.

Proof-of-Skill

In proof-of-stake currencies, your chance of signing a new block is proportional to your wealth. Proof-of-skill could be thought of as a proof-of-stake based on only coinbase transactions.

This block chain only requires a new block in the event of a dataTransition. This can be hours, days, or months, depending on the state; so event-chains are formed instead. nameProofs are chained to any previous generation since the last block. pointProjections are only chained together when modifying a projection, otherwise they are chained to the previous block. A dataTransition event requires a new block and and a proof-of-data consensus.

Consensus

More than 50% of skill or stake is needed for data and time transitions, respectively. This is done by recursively signing the block. Designated agents are used to sign for a player, which enables a practical implementation of the consensus protocol.

Nothing-at-Stake

Proof-of-stake block chains have a known flaw, that when presented with second block, there no cost for peers to follow both chains. This creates a bifurcation, or fork, on the chain. This is not a problem for the proof-of-skill, nameProof and pointProjection events, because it will get resolved by consensus on the next dataTransition event. However with transferTransactions the problem is still there, because each block depends on the previous. The nothing-at-stake problem creates a double spending problem as well.

One solution is to wait for the next timeTransition consensus, with stake values derived from the point of the bifurcation. However this can take months during the off-season. A second solution would be to use centralized time syncing, see section 6. Even if a fork in the stake block does occur, this would not effect the proof-of-skill blocks.

Sanity Checkpoints

Otherwise knows as developer checkpoints; this static data, gets added to the source code on a schedule. Bitcoin uses checkpoints, so in the unlikely event that a peer shows up with a complete alternate block chain, the protocol is protected [6]. Fantasybits are protected with end of season checkpoints.

6 Distributed Exchange

A pure decentralized distributed exchange is as impossible as solving the general Byzantine Generals problem. Exchange limit order books are path dependent, the time and order of events matter. In a pure decentralized distributed order book, all peers will have a different market snapshot. So, for a trader, there is no way to know the real status of her orders, or position, until a after blocked is signed.

Say there was a major capitulation move followed by a snapback in price, this would incentivize block signers to front-run, and losers to attempt forks.

Satoshi fantasy solves this with centralized time syncing. A proof-of-time consensus chooses the time syncing agent. All exchangeOrder events would need to be timestamped by the agent, who broadcasts them back to the network. The matching-engines are distributed, so each peer has a copy of the same market. This enables deterministic exchangeExecutions and clearingTransactions. Trading sessions opening and closing is accomplished with timeTransitions events and proof-of-time consensus.

7 Contact

Jay Y. Berg
info@satoshifantasy.com
(650) 560-5021
www.satoshifantsy.com

References

[1] Nancy Lynch. A hundred impossibility proofs for distributed computing. In *Proceedings of the eighth annual ACM Symposium on Principles of distributed computing*, pages 1–28. ACM, 1989.

[2] kjj. Bitcoin theory (byzantine generals and beyond). https://bitcointalk.org/index.php?topic=99631.

[3] http://www.rotoworld.com/player/nfl/5566/cj-spiller.

[4] JORDAN WEISSMANN. The insane growth of fantasy sports—in 1 graph. http://www.theatlantic.com/business/archive/2013/09/the-insane-growth-of-fantasy-sports-in-1-graph/279532/, sep 2010.

[5] http://www.myffpc.com/ffpccontent/?ffpc-history.

[6] monocolor. What are checkpoints in bitcoin code? https://bitcointalk.org/index.php?topic=194078.0.

A Scoring Rules

Passing Yards	1 point per 20 yards or .05 points per yard
Passing TD	4 points
Pass interception	-1 points
Rushing Yards	1 point per 10 yards or .1 point per yard
Rushing TD	6 points
Receiving Yards	1 point per 10 yards or .1 point per yard
Receiving TD	6 Points
Reception	1 point per reception
2-point conversion	2 points for passer, rusher, receiver
PAT kick	1 point
Field Goal	3 points for 1-30 yards, .1 point for each additional yard.
Sack	1 point
Takeaway	2 points
Defensive TD	6 points
Safety	5 points
Shutout	12 points
1-6 points allowed	8 points
7-10 points allowed	10 points

B Distribution Algorithm

```
NameValuePairs<double> DistribuePointsAvg::distribute(const Int result) ↩
    const
{
    double mean = 0;
    vector<Int> diffs;
    diffs.reserve(projections.size());

    for(const auto& pair : projections) {
        Int diff = abs(result-pair.second);
        mean+=diff;
        diffs.emplace_back(diff);
    }

    mean /= projections.size();

    Int maxdiff = min((Int)lround(mean),result);

    Int sum = accumulate(begin(diffs), end(diffs), 0,
        [maxdiff,result](const Int sum,const Int val)
```

```
19          {
20              return sum + ((val < maxdiff) ? result-val : 0);
21          });
22
23      double payout = static_cast<double>(result) / sum;
24      NameValuePairs<double> award{};
25
26      for (const auto& pair : projections) {
27          Int diff = abs(result-pair.second);
28          if ( diff < maxdiff )
29              award.emplace_back(pair.first,(result-diff)*payout);
30      }
31      return award;
32  }
```