Master Thesis

Proof-of-Turn: Blockchain consensus using a round-robin procedure as one possible solution for cutting costs in mobile games

Presented to the Faculty of Economics and Business Administration of University of Duisburg-Essen

By: Braun, Dominik

Friedenstr. 105 47053, Duisburg Mat. No.: 3086262

Examiner: Prof. Dr. Stefan Eicker

Prof. Dr. Tobias Kollmann

Academic field: Wirtschaftsinformatik

Abstract

This master thesis deals with Blockchain Technology in mobile turn based peer to peer games. First, it investigates the capabilities of Blockchain Technology to be used for gaming applications. In this regard, among others, *Proof-of-Mechanisms*, *Vote-based Consensus* and several *Performance Improvements* are described. Second, several smart contracts are introduced to show the general feasibility of turn based games hosted on Blockchain Technology. More specific, *Hidden transactions*, *Randomization*, *Piles of Cards*, *Fog of War* elements, *Data allocation improvements* and other smart contracts are specified. Third, a special Proof-of-Turn consensus mechanism, based on the Blockchain Technology, is defined to enable game publishers to cut costs in the means of their provided game servers. Herein, *Byzantine Fault Tolerance*, *Peering*, the *CAP Theorem*, *Interoperability* among other characteristics are covered. Last, these measures shall additionally raise the trust level among the players in mobile turn based games.

Contents

LI	st of	Figures	IV					
Li	st of	Tables	VI					
Αl	brev	iations	VII					
1	Intr	oduction	1					
	1.1	Current Situation	2					
	1.2	Problem Statement	4					
	1.3	Research Design & Questions	5					
2	Bloc	ckchain Technology	7					
	2.1	Network types & characteristics of Blockchains	9					
	2.2	Consensus Mechanisms	12					
		2.2.1 Proof-of-Mechanisms	13					
		2.2.2 Vote-based Consensus	15					
		2.2.3 Performance Improvements	17					
		2.2.4 Comparison of Consensus Mechanisms	20					
3	Bloc	ckchain Technology in Games	22					
	3.1	Ownership, network costs, gameplay & scalability	22					
	3.2	Gaming market	26					
	3.3	Existing games using Blockchain Technology	28					
	3.4							
	3.5	Game specific smart contracts	32					
		3.5.1 Hidden transactions & Randomization	32					
		3.5.2 Piles of Cards	36					
		3.5.2.1 General draws from piles	37					
		3.5.2.2 Public pile with private draw	38					
		3.5.3 Fog of War	43					
		3.5.4 Reveal claims, trigger events & Disputes	46					
	3.6	Data allocation improvements	47					
		3.6.1 Prune bloat transactions	50					
		3.6.2 Child-chain transition	52					
		3.6.3 Meta-State Blocks	53					
	3.7	Interim Summary - Blockchain Technology in Games	56					
4	Pro	of-of-Turn	57					
	4.1	Consensus finality & Byzantine Fault Tolerance	60					

Contents

	4.2	Peering	61
		4.2.1 Pull-Mechanism	62
		4.2.2 Push-Mechanism	67
	4.3	CAP Theorem measurement	70
	4.4	Forks & transition blocks	72
	4.5	Reveal claims & trigger events	75
	4.6	Interoperability	76
	4.7	Peer-Fluctuation & adaptive turn time	77
	4.8	Further Characteristics	78
	4.9	Attack Vectors	81
	4.10	Limitations	84
	4.11	Interim Summary - Proof-of-Turn	85
5	Con	clusion & Outlook	86
5		clusion & Outlook	86
5	Con 5.1	clusion & Outlook Outlook	86 87
5 A	5.1		
	5.1	Outlook	87
	5.1 App	Outlook	87 89
	5.1 App A.1	Outlook	87 89 89
	5.1 App A.1	Outlook	87 89 89 91
	5.1 App A.1	Outlook	87 89 89 91
	5.1 App A.1	Outlook	87 89 89 91 91 93 94
	5.1 App A.1	Outlook	87 89 91 91 93 94 95
	5.1 App A.1	Outlook	87 89 89 91 91 93 94

List of Figures

2.12.22.3	Blockchain Structure (Adapted from Lin and Liao (2017))		
3.1 3.2	Collectibles store 'CryptoShapes' (Adapted from CRYPTOKITTIES.co (2021)) 23 Visibility and Scalability		
3.3	Global share of devices by gaming time (Data from LIMELIGHT NETWORKS (2020)). 27		
3.4	Global preferred game characteristics (Data from LimeLight Networks (2020))		
3.5	Sample game		
3.6	Offset revealing of a single block (As described by Kraft (2016))		
3.7	Offset revealing with game hash (From Yuen et al. (2019))		
3.8	Offset revealing with follow up		
3.9	Parties shuffling cards		
3.10	Decryption processes (Nodes A, B and C)		
3.11	Decryption processes using many nodes ($\geq D$)		
3.12	BFT shuffling cards		
3.13	Fog of war in chess - Open moves (1)		
3.14	Fog of war in chess - Open moves (2)		
3.15	Fog of war in chess - Fogged moves (1)		
3.16	Fog of war in chess - Fogged moves (2)		
3.17	General Bloat Transaction (BT) storage allocation chart		
3.18	Prune bloat transactions		
3.19	Prune procedure/Child-chain chart		
3.20	Child-chain storage allocation		
3.21	Meta-State Block process step		
3.22	Meta-State Block chart		
3.23	Storage allocation comparison		
4.1	The turn		
4.2	Layers in PoT (Adapted from Oliveira et al. (2019))		
4.3	Sample 'peering by pulling'-strategy		
4.4	Pull request procedure		
4.5	Ending turn before maturity		
4.6	Update Mechanism		
4.7	Pseudo layers of peering by pushing		
4.8	Permission Transition		
4.9	Network Partition in PoT		
4.10	Mechanisms regarding the CAP Theorem (Adapted from DE ANGELIS ET AL. (2018)) 72		
4 11	Most Progressive Chain Rule		

List of Figures	V

4.12	PoT: Trigger event mechanisms	76
4.13	Multi layer BC system	76
A.1	Share of devices by gaming time (LIMELIGHT NETWORKS 2020, p. 7)	89
A.2	Average smartphone storage space (From Wang (2021))	90
A.3	Game characteristics data (Limelight Networks 2020, p. 18)	90

List of Tables

2.1	Blockchain Network Types	10
2.2	Blockchain Blocksize	18
2.3	Proof based consensus mechanisms (Khan et al. 2020, p. 5)	20
2.4	Vote based consensus mechanisms (Khan et al. 2020, p. 5)	21
2.5	Additional consensus mechanisms	21
3.1	Sample given randomization values	36
3.2	Extended Blockchain Network Types	37
3.3	Blockchain Size	50
4.1	Consent smoothness	74
4.2	Parameters of Proof-Of-Turn (PoT)	80

Abbreviations

A Availability

BC Blockchain

BCT Blockchain Technology

BFT Byzantine Fault Tolerance

BPMN Business Process Modeling Notation

BT Bloat Transaction

C Consistency

CF Consensus Finality

CM Consensus Mechanism

DDoS Distributed Denial of Service

DNs Distribution Nodes

ETH Ethereum

GCN Garbage Collecting Node

LN Leading Node

MMORPG massively multiplayer online role-playing game

NFTs Non-fungible tokens

P Partition Tolerance

PBFT Practical Byzantine Fault Tolerance Consensus

PoA Proof-Of-Authority

PoET Proof-Of-Elapsed-Time

PoP Proof-Of-Play

PoS Proof-Of-Stake

PoT Proof-Of-Turn

Abbreviations

PoW Proof-Of-Work

Ripple Ripple Consensus

rrTs relevant (revealed) Transactions

SC Smart Contract

TCM Tendermind Consensus Model

1

Chapter

Introduction

Since the beginning of peer to peer games, the game industry emerged towards online games in order to serve a growing number of simultaneous players (NAGYGYÖRGY ET AL. 2013, p. 192). After general networking moved from small local token ring- and Ethernet-based networks (SMYTHE 1999, p. 1) towards huge online communities, gaming became a massive multiplayer online-phenomenon (WILLIAMS ET AL. (2008, p. 1); WANG ET AL. (2012, p. 1)) and publishers needed to offer servers to coordinate the high amount of players (LEE ET AL. 2008, p. 41) and to prevent cheating. Nevertheless, all peers need to trust the publisher's server, whilst the server has to carry the costs of availability and network computation. Consequently, power consumption, computation workload and likewise attached costs for networking and peering are mainly carried by 'the central participant' of the network. But sometimes the central element - e.g. a publisher - is not able to run the central server without the need to monetize its offerings. One consequence was the rise of private dedicated servers, hosted by private persons (e.g. using a peer-to-peer architecture (LEE ET AL. 2008, p. 41)), which were 'detached from the publisher's direct influence'. Additionally, the technical/mental hurdle to set up a dedicated server, whilst other more convenient games exist, as well as the single point of failure 'out of the publisher's reach' are considered reasons against a rise and large distribution of private, peer-owned dedicated servers. Nowadays, along with reduced server costs¹, many game publishers rely on in-game purchases, advertisements and data mining to target those players who are willing to pay more (Davis and Lang (2012)). Consequently, consumers "[...] are regularly exposed to predatory behaviour from the games publishers, with microtransactions and loot boxes [...]" (LANEVE AND ERSHOV 2019, p. 14). Other distributed server architecture as proposed in the documents "A Distributed Server Architecture for Massively Multiplayer Online Games" from KHAN (2006) and "Matrix: adaptive middleware for distributed multiplayer games" from Besancon et al. (2019), to the writer's knowledge, never settled in the consumer world.

On the contrary, Blockchain Technology (BCT) offers distributed computing in 'trustless environments' and offers many of those by the (game) publishers needed characteristics. Generally speaking, a BC "[...] is an innovative technology, which can have a high impact in numerous industries, such as healthcare [1], supply-chain [2], finance [3] and video games [4]. BC are append-only ledgers shared across a network of clients" (BESANCON ET AL. 2019, p. 84).

¹ Reduction of running costs due to Moore's Law (Mann (2000), 2000). Here: less power consumption or increased sessions per hardware unit.

This thesis shall bring BCT closer to practical use cases in the gaming industry. Therefore, this study may especially be of interest to the following readers:

- 1. Researchers interested in enhancing BCT and affiliated mechanisms.
- 2. Practitioners willing to understand how blockchain may affect the software (gaming) industry.
- 3. Managers and administrative staff from the software (gaming) industry looking for potential savings of server operation costs.

The remainder of the document is structured as follows:

This section will provide the problem statement, research design as well as guiding research questions in detail. In section 2, a review of BCT and included mechanisms to achieve consensus are presented. Section 3 outlines recent applications of BCT in the gaming industry as well as mechanisms to map known tabletop game mechanisms into digital contracts. In section 4, a novel Consensus Mechanism (CM) for BCT is presented and discussed in detail. Finally, in section 5 the work is concluded and directions are provided for future research.

1.1 Current Situation

First of all, data and scientific research targeting the gaming industry is scarce. Still it is known that most "[...] game projects fail to meet their financial expectations [...]" (Bethke 2003, p. 17). Nevertheless, the number of games is growing constantly (Statista.com (2021a)) and providers are forced to minimize their (running) costs (Koster (2018)). "Furthermore, small developers are finding it hard to stand out in a market dominated by giant companies that dwarf them through marketing or polish, with the average indie developer earning only around \$10 thousand a year" (Laneve and Ershov (2019, p. 14) using data from Gamesetwatch.com (2014, p. 6).

Although there are several ways to cut costs, one of them is the use of dedicated servers which are paid by single peers or clans of the game network (WIKIPEDIA (2021c)). Especially in the long run, overall costs (Weilbacher 2012, p. 13) may be reduced by these servers and help publishers to keep the game alive and establish a network of players and enthusiasts. Some publishers offer dedicated servers (VALVE (2021)) as one possible solution for externalizing expensive hardware and maintenance. Nevertheless, there are some reasons against dedicated gaming servers:

1. Giving away the source code

Behind this phrase, the publisher fears its game's code to be analyzed and cloned to provide any type of copycat product (Li et al. 2014, p. 2). Two prominent examples for this phenomenon in the game industry is the massive imitation of the mobile game 'flappy bird' (Li et al. (2014)) as well as the co-evolution of the 'battle royale mode' (Kooistra J. (2018)). Hence, these game mechanic ideas skyrocketed and (many) late adopters followed, but not because the code was given 'out of hands' in the first place. Therefore the argument can be considered obsolete as each game's general mechanics can be obtained by simply

playing. Nevertheless, the fear is not negligible in general as sometimes source code is stolen (VICE.COM (2021)).

2. Update laziness

Games progress, therefore both 'security'- as well as 'game-mechanics'-updates shall comply with publishers' and gamers' expectations. But dedicated server providers are detached from the publishers' direct influence. The providers act by intrinsic motivation, ranging from *control* over *fame* up to *monetization*. Hence, publishers as principals and providers as agents follow different incentives - which presents a *Principal-agent problem* as generally described by STIGLITZ (1989).

3. Nerdy peers

As a server needs to work 24/7 to serve whenever needed, there need to be maintainers (Lowell et al. 2004, p. 211). These maintainers are not supposed to be casual gamers, but peers which offer expertise with web-servers, scripting and so forth (Teamfortress.com (2021)). On the one hand, without these type of persons, the offer of dedicated servers is worthless and consequently cheap 'scalability and accessibility' can not be achieved. On the other hand, a game publisher has to consider an additional group of (silent) stakeholders.

4. Shadow server

A *shadow server* is considered a server which runs on one players device, without the player's knowledge/consent. This is not recommended as players could 'force quit' the game or shut down their system (unintended). Consequently the (dedicated) shadow server would end to serve and remaining players are kicked out of their games which presumably leads to a bad consumer experience.

5. Lost peer management & data collection

In the assumption that dedicated servers are generally detached from the publisher's reach, metrics and other data get lost although gamers stay within the game's ecosystem. Still, publishers have a "Need for Metrics" (Palmer 2002, p. 152) to find flaws in their games. Thus, dedicated servers are seen as a misleading approach.

6. Performance

Bad performance of dedicated servers leads to bad consumer experience (WEILBACHER 2012, p. 10). Nevertheless, the reasons for bad performance are manifold, from slow server devices not suitable to host their (large scale) games up to slow broadband connection (WEILBACHER 2012, p. 10).

7. Integrity & Cheating

The combination of nerdy peers and the possibility to modify game content (e.g. within a 'modding community') could lead to cheating behavior and consequent changed game characteristics (Morris (2003)). Although there are basic possibilities to check the integrity of server instances (Agosta and Crosby (2003); Deswarte et al. (2004)), the publishers or other players can't rely on the system to be somehow silently manipulated.

Still, the number of persons 'willing and able' to provide dedicated servers is seen low compared to the number of gamers playing in the manifold games. To the writers knowledge, despite the enormous amount of mobile (Android) apps offering server-based network multiplayer modes (ITCH.IO (2021)), there are only some mobile games, such as 'Among Us' and 'Minecraft' offering dedicated servers. Those games have been designed for desktop primarily and have only thereafter been ported to mobile operating systems (Here: Android & iOS). Compared to the 'Hard-core' gamers, mobile and 'casual' gamers can be considered less likely to set up additional systems to boost their game sessions due to reduced general involvement (PRUGSAMATZ ET AL. 2010, p. 388). As small, independent developers might not be able to afford additional running servers, there is a (growing) demand for distributed computing in the mobile gaming ecosystem. At this point the interest in the present work originates from the insights into game development and knowledge about BCT. During the last years, BCT emerged and offers the possibility to establish distributed networks to share data without a central authority. For now, "[...] an analysis of the different projects, [...]" has shown "[...] that the main problems blockchain games face are related to scalability and transaction speed" (LANEVE AND ERSHOV 2019, p. 47). Nevertheless, if there is a possibility to tweak a Proof-/Voting-Based Consensus Model (KHAN ET AL. 2020, p. 3) to fit the need of the gaming industry, it might keep cooperative niche games alive or kick start some community driven game development projects (Laneve and Ershov 2019, p. 14).

1.2 Problem Statement

The biggest issue seen for game publishers is the shortage of money, both during production and operation (Koster (2018)). Sometimes, when games do not settle in the market, publisher servers need to be shut down (ROCKPAPERSHOTGUN.COM (2021)). This beholds true for all genres as network- and cross selling effects play a key role for the survival of online (multiplayer) games (Rong et al. 2018, p. 45).

Once game servers are shut down, players can only play offline, if at all. Additionally, games are established in the perception of long term revenue. Hence, once an investment decision for a game idea is made, long availability is a key feature. Only an accessible game can continue to yield revenue. Last, due to tight budget plans during a game's creation process, additional effort for dedicated servers is mostly not met as an active community has not yet existed. Consequently, many game publishers are in a problematic situation between *bankruptcy* and *squeezing their consumers* for income generation Sotamaa and Svelch (2021). Once a game is established, besides fixing bugs and implementing new features, running servers are a key driver of 'keep alive costs'. The question arises whether there is any backend technology to further minimize some of the games running server costs. The solution has to be:

- 1. **Distributed**, to relieve the central running server from some/many workloads.
- 2. Trusted, preventing malicious behavior and cheating.
- 3. Cross-plattform applicable to take advantage of gamers using different operating systems.

- 4. Scalable & Fast, at least for some designated game niche use cases.
- 5. Leightweight in terms of preventing heavy computation on single devices.

On the first look, BCT offers some of these characteristics. Nevertheless, as "[...] it stands, blockchain technology does not seem applicable for the design of the most popular game genres such as first-person shooters or real-time strategy[...]" (Serada et al. 2020, p. 3). Still, "[...] several attempts have been made in this direction (e.g., EOS Knights, HyperDragons, and Epic Dragons), and many more are likely to follow [...]" (Serada et al. 2020, p. 3).

Therefore this document will examine BCT thoroughly to find a suitable application area.

1.3 Research Design & Questions

The main research question which shall be answered with this thesis is:

"Can BCT be used to reduce publisher's server costs whilst providing (mobile) players a suitable gaming experience?"

To guide the further procedure, no explicit framework is used. As a starting point the recent literature has been reviewed to reflect the state of research about BCT and related CMs. The results of the grounding research and the CMs performance characteristics are given in the chapter Blockchain Technology. Additionally, as BCT is still pretty young regarding the field of information systems - starting from Nakamoto (2009) in 2009 - there is only scarce literature to be found in the intersection with the gaming industry, yet. Still, some applications and use cases of BCT in games could be found. "The industry has started its exploration on this topic by integrating traditional games with blockchain systems" (MIN AND CAI 2019, p. 1). Further more, MIN AND CAI (2019, p. 1) state that blockchain games have already become an important component of decentralized applications and have held a considerable market capitalization. Nevertheless, these applications are barely suitable for game play related issues.

As BCT builds on trust and agreed rules, the mechanism called Smart Contract (SC) has to be mentioned. SCs are rulesets which can be used to establish agreements between participants of BCT networks. As there seem to be reasons, which prevent even slow paced games to be conducted on BCT, the following guiding question has been raised:

"Which kind of SCs need to be established to cover typical in-game mechanics?"

Both, existing games using BCT as well as game specific SCs are given in chapter Blockchain in Games. Moreover, during the search for suitable CMs for games hosted on BCT, it appeared that games with realtime-mechanics, such as first person shooters and massively multiplayer online role-playing game (MMORPG), can be considered as not suitable for BCT backbones (Serada et al. 2020, p. 19). Therefore, fast paced games became out of scope as distributed computation lacks performance regarding speed most of the time (Serada et al. 2020, p. 3). Thus this present work concentrates on slow paced games, such as asynchronous 'turn based (strategy)'-games (Bergsma and Spronck 2008, p. 1) and raises a second guiding question:

"What is the best fitting BCT CM to cover an asynchronous game play scenario?"

To address this research gap, the different Proof-/Voting-Based Consensus Models are evaluated and a novel CM (Chapter: Proof-of-Turn) is proposed. PoT is a special merge of existing protocols to fit the need of turn based games and remains a theoretical proof of concept for a (new) Proof-/Voting-Based CM. Consequently the research hypothesis is:

"The PoT CM leads to a reduction of server running costs for game publishers."

Additionally the following working assumptions are used:

- 1. Throughout a game a majority of peers is constantly online.
- 2. The network offers enough transmission performance (throughput/bandwidth).
- 3. Every peer has the needed storage available.
- 4. Depending on the 'real world'-scenario, the publisher's servers help on special occasions (e.g. a trusted timestamp).
- 5. Symmetric and asymmetric encryption is secure.

Due to the intersection of this research between *business administration* and *computer science*, the document falls into the research area of *economics of information systems*.

Last, in other contexts gamers, players, peers and nodes are sometimes used as synonyms, but in the following they need to be distinguished:

- 1. **Gamers** are persons, who play a game.
- 2. **Players**, which are considered digital in-game characters, are controlled by *gamers*.
- 3. **Nodes** are hardware systems, which is part of a BCT-(gaming-)network.
- 4. Peers are persons, who can be gamers but predominantly own or maintain a node.

Upcoming the core principles of BCT are described.

<u>Chapter</u>

Blockchain Technology

Although the general pattern of linked data, primarily using the terms '(One-Way) Hash Chain' or 'Merkle (Hash) Tree' (Hu et al. (2005)), was known before the therm BCT, the rise of Bitcoin, enabled by Nakamoto's core paper in 2009 presenting the fundamental Proof-Of-Work (PoW) protocol, pushed BCT into public attention. In this thesis, BC is from now on considered to be a (special) instance of the abstract term BCT. Additionally, BCT is classified as a database technology. More specific, BCT "[...] is a form of distributed ledger technology, deployed on a peer-to-peer network where all data are replicated, shared, and synchronously spread across multiple peers" (Butijn et al. 2020, p. 13). Nowadays many different Blockchain network approaches exist, serving different (special) needs. Before further details of the different BCT approaches are presented, a short recapitulation is given, why all this effort is important.

Especially for cryptocurrencies, but also within all the other use cases (e.g.: Serada et al. (2020)), fraud in distributed trust-less environments shall be prevented (Xu et al. 2019, p. 1). Within cryptocurrencies, which are build upon BCT, fraud is primarily known as **double spending** of money (Nakamoto 2009, p. 1). This is why Nakamoto (2009)'s core paper was a breakthrough, as it offers "[...] a solution to the double-spending problem using a peer-to-peer network" (Nakamoto 2009, p. 1). Generally, the "[...] blockchain protocol was designed to maintain a permanent traceable record between two parties that is transparent and open to public scrutiny without the need of a middleman to authenticate transactions" (Gainsbury and Blaszczynski 2017, p. 483). Hence BCT is not only capable of dealing with money - it can also administer any kind of digital assets.

Consequently, in the context of online games, fraud can be equated with cheating - the illegal transactions which raise the probabilities to win a game. Later on the space between providing needed information to keep the network alive and the need for hiding information, not yet allowed to be revealed, will become small (see section Hidden transactions & Randomization). Especially under theses given circumstances, the prevention of double spending has to be ensured. To understand the mechanics and techniques to prevent fraud better, this chapter offers core characteristics and an overview on recent mechanisms. Six of them are briefly described:

1. Chain Structure

Data is stored in a linked chain, consisting of 'blocks of data'. The blocks are linked by fingerprints (hashes) of previous blocks and the write/update-process is cryptographically secured as shown in figure 2.1 (alike Lin and Liao (2017, p. 654)). "The exception to this rule

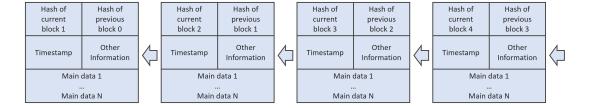


Figure 2.1: Blockchain Structure (Adapted from LIN AND LIAO (2017))

is the **Genesis Block**, which is the first block in the chain and has a predetermined hash" (OLIVEIRA ET AL. 2019, p. 181). The *genesis block* will be of further importance in section Data allocation improvements of chapter Proof-of-Turn,

2. Decentralization, Distribution & Disintermediation

"The blockchain is designed for distributing and synchronizing the data across multiple networks" (Sharma et al. 2020, p. 6). "In traditional centralized transaction systems each transaction needs to be validated by a (trusted) third party (e.g., a bank). The decentralized workings of BCT enables the direct transfers of digital assets between two counter parties without this third party leading to direct disintermediation" (Butijn et al. 2020, p. 17). Although BCs may be computed by only one peer, the benefit of BCT stems from the network and the distribution of data itself, reducing the single point of failure (Bodkhe et al. 2020, p. 79764). Finally all nodes strive to have the whole BC stored to prevent fraud and to keep persistence with the other nodes (Bodkhe et al. 2020, p. 79766).

3. Persistence & Immutability

A BC "[...] is a permanent record of transactions. Once a block is added, it cannot be altered. This creates trust in the transaction record" (Sultan K. Et al. 2018, p. 53). Generally data can only be invalidated by reference. Nevertheless, data can not be deleted entirely (Sharma et al. 2020, p. 88). Hence published data is immutable and distributed transparently throughout the network (Butijn et al. 2020, p. 60-61).

4. Transparency

Moreover, until the last node is shut down, the BC's data is immutable published and all peers have the full data set (Sharma et al. 2020, p. 88). As long as data in blocks is not encrypted to some peers, the BC is transparent and all peers possess the same collective knowledge (Gainsbury and Blaszczynski 2017, p. 483) Additionally, as (new) data is always signed by a peer, information is traceable and transactions can be verified by all peers (Sharma et al. 2020, p. 88).

5. Consensus Driven

Although there are special types, generally all nodes have the right to write blocks, if they follow the given ruleset (DIB ET AL. 2018, p. 54). The given ruleset for adding data is called the CM and differs regarding the BC's use case (Sharma et al. 2020, p. 88).

6. Mining

The action of adding a new block to a BC "is called mining and the nodes executing the calculations are referred to as miners in the Bitcoin nomenclature" (BUTIJN ET AL. 2020, p. 4). The name stems from the reward - any type of tokens - granted by many CMs.

Before the usage of Blockchain in Games is introduced in the next chapter, this chapter presents different *network types and characteristics of BCs* as well as commonly used CMs.

2.1 Network types & characteristics of Blockchains

In this section, accessibility of BC networks will be described before the layer structure for pushing data is explained. Last, characteristics of Blocks, Transactions and Smart Contracts are given.

1. Accessibility of BC networks

The following definitions deal with the accessibility of BCs. Note that the following characteristics regarding accessibility do not influence the allocated storage space on the network's nodes - every node stores the full BC. Generally it is distinguished between an **access policy** in the *private/hybrid/public*-scheme and a **validation policy** in in the *permissionless/permissioned*-scheme (Daniel and Guida 2019, p. 2). The networks can be categorized as follows (Table 2.1):

- a) "In **public permissionless networks**, consensus mechanisms are required to be very strict due to the lack of trust between the participants. This is justified by the main feature of a public network, the equality between nodes. Once a network participant, the user receives a pair of cryptographic keys to sign and perform transactions. Any node can be a miner and participate in the consensus mechanism. The problems associated with public blockchains are the fees that must be paid to encourage users to participate in the network and to mine blocks; the concern with the network scalability; and the time interval to confirm the transactions. Such problems are related to the fact that public permissionless networks are collaborative environments and, therefore, depend on the benign behavior of nodes. Besides, for sensitive data networks, the fact that all information is available to everyone represents a challenge to data privacy. Due to the characteristics of public permissionless networks, costly consensus mechanisms are required" (OLIVEIRA ET AL. 2019, p. 182). In this manner, BC "[...] systems like Bitcoin and Ethereum are called permissionless, i.e. any node on the Internet can join and become a miner" (DE ANGELIS ET AL. 2018, p. 1).
- b) To cure the needed computation power of BC systems in *public permissionless networks*, "[...] **public permissioned networks** were developed to implement less costly consensus mechanisms on public networks. The difference between the public permissionless networks and the permissioned ones is the different roles that can be played by nodes in the network. In public permissioned networks, the node only participates of the network after proper verification of its identity" (OLIVEIRA ET AL. 2019, p. 182).

	Permissionless	Permissioned
Public	a)	b)
Private	c)	d)
Hybrid	e)	f)

Table 2.1: Blockchain Network Types

- c) "The private permissionless networks differ from public networks because they restrict the entry of participants. The private network is usually governed by a single institution or a set of institutions, which determine who are the nodes allowed to participate in the network. Participant nodes have equal functions and carry the same importance. Once authorized to participate in the network, the node can generate transactions and blocks, and participate in consensus" (OLIVEIRA ET AL. 2019, p. 182).
- d) "The **private permissioned networks** only allow some nodes to participate in the consensus process, and only a subset of these nodes can generate the next block" (OLIVEIRA ET AL. 2019, p. 182).

If Sultan K. et al. (2018)'s definitions are considered as well, accessibility behaves within a grey scale between *public*, *private* and *hybrid*.

Public BCs ".. have no single owner; are visible by anyone; their consensus process is open to all to participate in; and they are full decentralized. Bitcoin is an example of a public blockchain" (Sultan K. et al. 2018, p. 53).

Private BCs ".. use privileges to control who can read from and write to the blockchain. Consensus algorithms and mining usually aren't required as a single entity has ownership and controls block creation" (Sultan K. Et al. 2018, p. 53).

Hybrid BCs ".. also known as consortium, these blockchains are public only to a privileged group. The consensus process is controlled by known, privileged servers using a set of rules agreed to by all parties. Copies of the blockchain are only distributed among entitled participants; the network is therefore only partly decentralized" (Sultan K. Et al. 2018, p. 53). Concluding, hybrid BCs are operating in a public network, controlled by all nodes and hence do not fit into the bipolar private, public scheme. Hybrid entries are added as well:

- e) **Hybrid permissionless** BCs operate in a public network, but only an associated node can grant access (Sultan K. et al. 2018, p. 53). No additional consensus has to take place (Sultan K. et al. 2018, p. 53). This does not differ much from a *public permissionless* BC except the first entry barrier.
- f) **Hybrid permissioned** BCs, consequently, operate in a public network as well, but access needs to be granted by a majority of nodes (regarding a consensus protocol) (SULTAN K. ET AL. 2018, p. 53).

Within both hybrid types, again, the nodes are distributed and not controlled by any higher entity.

From here on it is important to keep in mind that a round-robin consensus model is aimed for. The round-robin "[...] consensus model is usually used in *permissioned* Blockchain. It allows nodes to take turn one by one for generating blocks" (Khan et al. 2020, p. 3). Although the following PoT mechanism (Section: Proof-Of-Turn) could also be used for *public permissionless* BC networks, subsequent the focus will be on private as well as hybrid permissioned BCs.

2. Layer structure

BCT generally consists of four layers (OLIVEIRA ET AL. 2019, p. 181), which pass data from the publishing (pushing) source node towards the global distribution throughout the network (Figure 2.2). First, in the Transaction Layer, new, local data from transactions is signed and given to the validation layer (OLIVEIRA ET AL. 2019, p. 181). Second, in the Validation Layer, each of the transactions are validated to the respective predefined rules (OLIVEIRA ET AL. 2019, p. 181) - for example, defined by a SC. As many/all nodes need to comply with the given data, the transactions are here (already) distributed among the nodes (OLIVEIRA ET AL. 2019, p. 181). Third, in the Block Generation Layer, many transactions are combined and summed up in one block (OLIVEIRA ET AL. 2019, p. 181). This block is then passed into any CM (OLIVEIRA ET AL. 2019, p. 181). This mechanism will distinguish the next added block and has to prevent hostile behavior of nodes (e.g. publishing a block against the consensus rules) (DIB ET AL. 2018, p. 54). It can be seen as the gatekeeper for the data to be distributed. Please note that the wrapping of transactions into one block is primarily used to reduce computational effort (KIM ET AL. 2018, p. 1204), as some CMs, like Bitcoin's PoW, are computation hungry (LIU ET AL. 2018, p. 11008). Still, every transaction may also be appended individually to the BC, which is only recommended along computationally lightweight CMs. Different emendations regarding this manner are given in the following

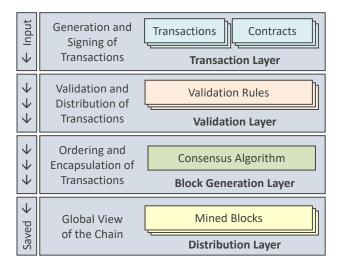


Figure 2.2: OLIVEIRA ET AL.'s layers of BC consensus

subsection Performance improvements. Last, in the *Distribution Layer*, all added blocks are 'finally' stored and the CM 'ensures' that data, once written here, cannot be deleted (OLIVEIRA ET AL. 2019, p. 182).

Finally, the "[...] data structure of a blockchain, whether public or consortium, corresponds to a linked list of blocks containing transactions also referred to as the 'ledger'" (DIB ET AL. 2018, p. 52).

3. Blocks, Transactions and Smart Contracts

Next to the metadata the linked blocks, building the backbone structure of BCT, consist of data, as shown in figure 2.1. More specifically all blocks can be seen as new, updated states of a BC database (Danzi et al. 52018, p. 1). Hence they are transactions, sometimes without direct impact or constraints to other nodes, sometimes even with mandatory consultation (Danzi et al. 52018, p. 1). Moreover, a transaction can be "[...] an exchange of assets that is managed under the entity service's rules" (Sultan K. et al. 2018, p. 51). "These rules also form the basis for smart contracts. A smart contract is a set of logic rules in the form of a coded script which can be embedded into the blockchain to govern a transaction" (Sultan K. et al. 2018, p. 51). The specific rules of SCs and their validation are, understandably, implementation dependent and out of scope. Nevertheless, some game related meta-types of SCs will be covered in the subsection Game specific smart contracts of the upcoming chapter Blockchain in Games.

For now the skeleton of BCT is known. Thus, CMs are now presented in further detail.

2.2 Consensus Mechanisms

A major driver of progress in BCT research lies within new CMs, which unlock new fields of application. As stated before, the CMs are the gatekeepers for new data within each BC. A CM can be grounded on some sort of **proof**, which has to be delivered by the pushing node (Nguyen G.-T. and Kim 2018, p. 106). Further, a CM can be grounded on **votes** as well (Nguyen G.-T. and Kim 2018, p. 106). Generally, nodes who participate in the CM need any incentive to meet the effort (Catalini and Gans 2016, p. 2). Before some popular CMs and their benefit system are described and discussed, two characteristics, Byzantine Fault Tolerance (BFT) and Consensus Finality (CF) need to be addressed:

BFT describes the share of nodes within a system, which are allowed to become hostile/unavailable before the system is vulnerable to break down, freeze or becomes prone for fraud (Gramoli 2017, p. 1).

CF is a phrase to describe the *indelible finality* of written data (DE ANGELIS ET AL. 2018, p. 3). Depending on the chosen CM, data in the distribution layer (Figure 2.2, Layer 4) are 'only supposed to be final' (DE ANGELIS ET AL. 2018, p. 3). The low probability of *non-finality-data* (e.g. in PoW) makes data only implicitly 'consensus final' (DE ANGELIS ET AL. 2018, p. 3). Nevertheless, other CMs - e.g. based on voting - provide data with (directly) secured final states (DE ANGELIS ET AL. 2018, p. 3). From a more formal perspective, DE ANGELIS ET AL. (2018) uses VUKOLIĆ (2016)'s

comparison on PoW with BFT-like approaches "[...] introducing the distinguishing property of consensus finality: the impossibility of reaching consensus without fully distributed agreement. In blockchain's jargon, it amounts to the impossibility of having forks. As expected, PoW does not enjoy *consensus finality* (as forks can happen), while all BFT-like approaches does (all parties reach agreement before consensus)" (DE ANGELIS ET AL. 2018, p. 3).

With this knowledge, CMs by **proof** and CMs by **votes** are given. The reader has to be reminded that the list is limited on purpose due to manifold CMs.

2.2.1 Proof-of-Mechanisms

There are several Proof-of-Mechanisms which differ in their characteristics. To get an overview important/general mechanisms are introduced:

1. Proof-of-Work

The mechanism, which is supposed to be the most popular one, is the PoW mechanism. It is used in the Bitcoin network and was introduced by Satoshi Nакамото (2009). Here, nodes have to solve mathematical problems to generate new blocks (BUTIJN ET AL. 2020, p. 4). Pieces of the underlying cryptocurrency are used as incentives for finding the next suitable block (NAKAMOTO (2009, p. 8); CATALINI AND GANS (2016, p. 2)). Hence the predominant motivation for the mining nodes to keep the network alive is supposed to be extrinsic. Additionally, the Bitcoin implementation adapts its energy consumption along the rising computation power within the network to generate a steady output of one block every ten minutes (MA ET AL. 2020, p. 2254). Generally an "[...] attacker would need the majority of the network's computational power to rewrite history and calculate new blocks" (DEMI ET AL. 2021, p. 3-4). Still, "[...] each node needs to keep the history of all the transactions made in the network, so the storage space keeps increasing, and the number of transactions that can be processed by the network is quite limited, around 7 transactions per second [...]" (BESANCON ET AL. 2019, p. 81). Nevertheless Bitcoin scales along its stakeholders, who are not mining, pretty well - there is no limit in participants (Nакамото 2009, p. 8). However, PoW fails to reduce power consumption along an increasing mining network (SALEH 2020, p. 1-2).

2. Proof-of-Stake

Proof-Of-Stake (PoS) was introduced to overcome the energy waste issue of PoW (Chaudhry and Yousaf (2018, p. 56); Narayanan (2018)). Whilst everyone is allowed to mine in the PoW approach, the PoS protects the network's stakeholders - those nodes with intrinsic motivation to keep the chain's integrity - from external nodes (Chaudhry and Yousaf 2018, p. 56). The PoS CM"[...] saves more energy as compared to the PoW model. It is an energy efficient variant of PoW. In PoS, miners have to showcase and declare the ownership on a certain amount of currency. It is presumed that people with more cryptocurrency or coins would not attack the blockchain network" (Khan et al. 2020, p. 3). Nevertheless, as nodes always seek to gain rewards, PoS is running the risk of a Nothing-at-Stake problem, wherein "a validator will always update the ledger whenever given the opportunity even if the update necessarily perpetuates disagreement" (Saleh 2020, p. 2). The reduced power

consumption stems from the reduced computation power (less mining nodes) within the network (Chaudhry and Yousaf 2018, p. 56).

3. Proof-of-Authority

Third, the Proof-Of-Authority (PoA) approach, also known as *Proof-of-Identity*, is presented: If a node wants to publish transactions, the node has to prove its "[...] identity and should be verifiable in the blockchain network. Basically, the publishing nodes are putting their **identity and reputation** for being a publishing node. Publishing node's reputation is directly linked to the publishing node's behavior. Any malicious activity by publishing [...] can **damage the reputation** of the node in the Blockchain network. Node reputation would increase if it acts in a manner that Blockchain users agreed with. Nodes with less reputation are **less likely to get the chance to publish a new block**. [...] This model is preferred in permissioned blockchain for it requires a lot of trust on the nodes" (Khan et al. 2020, p. 3).

4. Proof-of-Elapsed-Time

For this mechanism special hardware (Software Guard Extensions - Intel SGX) is needed to create a Trusted Execution Environment called TEE, which works on the basis of exact timing (DIB ET AL. 2018, p. 55). The Proof-Of-Elapsed-Time (PoET) Consensus Model "[...] selects random leader via election protocol. [...] A leader is selected to add the next block to the Blockchain in this model. [...] Every miner asks for the running code within the TEE for a waiting time and the miner with the lowest waittime becomes a leader node. TEE function can prevent tampering by any internal or external source. The only drawback in this consensus model is that it requires special hardware implementation" (Khan et al. 2020, p. 3-4).

5. Proof-of-Play

Fifth, in the Proof-Of-Play (PoP) approach the interaction of a player behind a node is evaluated. "In the current PoP design, the evaluation is adjusted according to the player's ability. This provides a fair chance for everyone to pass through the evaluation stage with enough effort of their ability" (YUEN ET AL. 2019, p. 23). Finally, the chosen node writes all parallel published data into one block (YUEN ET AL. 2019, p. 23).

Many of the presented Proof-of-Mechanisms were enhanced using several names, such as: Delegated Proof-of-Stake, Proof-of-Activity (PoP/PoS-hybrid), Proof-of-Achievement (Komiya and Nakajima (2019)), Proof-of-Burn, Proof-of-Capacity (or Proof-of-Storage), Proof-of-Excellence (King and Nadal 2012, p. 5), Proof-of-Existence, Proof-of-Importance, Proof-of-SequentialWork, Proof-of-Validation (PoV), Proof-of-Vote as well as Proof-of-WorkOrKnowledge and Proof-of-ZeroKnowledge.

To dig deeper into most of these algorithms, Butijn et al. (2020, p. 5) and de Angelis et al. (2018, p. 3) are recommended as starting points. Nevertheless, these enhancements are considered to be out of scope due to their specificity.

Until now the permission to write a block was either dependent on available computation power

(PoW), share of stake (PoS), reputation within the network (PoA), exact timing (PoET) or some kind of special effort (PoP).

In Proof-of-Mechanisms much independence among the writing nodes is given - nodes are never asked to consent (explicitly) (Nakamoto 2009, p. 8). Therefore competing versions of the BC, known as **forks**, occur (Butijn et al. 2020, p. 60). "A user only needs to keep a copy of the block headers of the longest proof-of-work chain, which he can get by querying network nodes until he's convinced he has the longest chain, and obtain the Merkle branch linking the transaction to the block it's timestamped in" (Nakamoto 2009, p. 5). A miner "[...] follows the **longest-chain rule** if she always chooses the last block of one of the longest chains" (Ewerhart 2020, p. 2). This behavior, formally known as the *Longest Chain Rule*, was first described by Courtois (2014). "Note that the *longest-chain rule* is a class of strategies, rather than a single strategy" (Ewerhart 2020, p. 2). The *Longest Chain Rule* is the reason why blocks in some Proof-of-Mechanism BCs may never reach 100% CF (DE ANGELIS ET AL. 2018, p. 3).

"Because of the possibility of forks, there is no such thing as absolute reliability of the data retrieved from the blockchain. It is decreasingly high toward the most recent blocks data, as one only get the version of the ledger stored on a node at a given time, so that a blockchain-specific time-dependent reliability weight has to be determined" (DIB ET AL. 2018, p. 62). Some Proof-of-Mechanisms have found their way around this flaw, such as PoET and PoP. Nevertheless, on vote based (/consortium) BCs, "[...] the use of adapted consensus algorithm allows for "block finality": once a block has been validated, it remains on the main chain and forks are not allowed" (DIB ET AL. 2018, p. 54).

2.2.2 Vote-based Consensus

In contrast to *Proof-of-Mechanisms*, **Vote-based-Mechanisms** - also known as **consortium** mechanisms - only accept blocks which already reached consensus (DE ANGELIS ET AL. 2018, p. 3). The accepted blocks reach a 100% CF directly after approval (DE ANGELIS ET AL. 2018, p. 3). "The consensus is coordinated by the distributed nodes controlled by consortium partners which will come to a decentralized arbitration by voting. The key idea is to establish different security identity for network participants, so that the submission and verification of the blocks are decided by the agencies' voting in the league without the depending on a third-party intermediary or uncontrollable public awareness. Compared with the fully decentralized consensus—Proof of Work (PoW), vote-based consensus has controllable security, convergence reliability, only one block confirmation to achieve the transaction finality, and low-delay transaction verification time" (Kejiao et al. 2017, p. 466).

Again there are multiple solutions:

1. Practical Byzantine Fault Tolerance Consensus Model

Malicious "[...] attacks, operator mistakes, and software errors are common causes of failure and they can cause faulty nodes to exhibit arbitrary behavior, that is, Byzantine faults" (Castro and Liskov 2002, p. 399). The Practical Byzantine Fault Tolerance Consensus (PBFT)

is designed to prevent Byzantine faults. "It is mostly used in permissioned Blockchain such as in Hyperledger as it can manage up to 1/3 malicious byzantine replicas. In practical PBFT, the next block is accepted in a round process. There are certain processes to follow in every round for choosing a primary node. The practical PBFT model's process is classified into 3 well-defined phases: preprepared, prepared, and commit. For changing the state, the node should have at least 2/3 votes from all nodes. It will ensure that all nodes are recognizable and know with each and every node. [...] In PBFT each node needs to inquire about other nodes" (Khan et al. 2020, p. 4).

2. Ripple Consensus Model

This model will further be called **Ripple**. Every "[...] node needs to create a unique node list (UNL). All Ripple Consensus (Ripple) nodes are part of UNL and are considered as reliable nodes for all other nodes to rely on them. No node are to go against UNL. Ripple network encourages all nodes to communicate with various nodes available in that UNL to achieve consensus in the network. Every node in the UNL should make 40% of overlap with all other nodes. In the Ripple network, consensus could be achieved in a couple of rounds where all nodes understand the responsibility for assembling the transaction with the state of the candidate set in a "a well-known data structure" and transmit its candidate sets to nodes in the current UNL. Nodes are responsible for validating the transaction, vote for a specific transaction and transmit the votes to the network based on the results of the collective votes. Every node filters out its candidate set and transaction receiving the highest votes are transferred to coming round of consensus. Right after achieving the 80% votes of candidates set from every node available in the UNL then that particular candidate set becomes the ledger in Ripple term. Another round of consensus would start with the new and pending transaction which could not get accepted. Consensus in the whole network can only be achieved after every sub-network achieves consensus" (KHAN ET AL. 2020, p. 4).

3. Tendermind Consensus Model

Khan et al. (2020, p. 4) summarizes the Tendermind Consensus Model (TCM) from Kwon (2014) as follows: TCM "[...] is another variant of PBFT model which mostly works with permissionless Blockchain. In this model, clients have the privilege of directly creating a new transaction to the nodes. The clients in this model utilize the gossip protocol to broadcast the transaction for the validating nodes. Within the Brach broadcast pattern, the progression would be done with the external validating situation so validating nodes are required to collect the transaction by gossip right before it gives a right to include the transaction in the block.

The significant difference between Tendermind and PBFT is that it continuously rotates the leader node right. Tendermind takes PBFT view change mechanism into the common case pattern. Here this shows as it is waiting to timeout, and validating nodes wait for the leader node to convey the first message in the Bracha broadcast pattern. which is more relevant to the view change the timer in PBFT. As the timer expires validating nodes vote to a nil block and validating nodes take part n the Bracha Broadcast message pattern on a

continuous basis. The Tendermind was suffering from many issues and the obvious one was livelock, presenting locking and unlocking votes by validating nodes."

The last two approaches which are covered are enhancements of above given CMs: *MultiChain* and *Child-chains*.

MultiChain "[...] enforces a round robin schedule, in which the permitted miners must create blocks in rotation in order to generate a valid blockchain" (Greenspan 2015, p. 7). Hereby the round robin is implicitly enforced regarding a value called *mining diversity* (Greenspan 2015, p. 7). "A value of 1 ensures that every permitted miner is included in the rotation, whereas 0 represents no restriction at all. In general, higher values are safer, but a value too close to 1 can cause the blockchain to freeze up [...]" (Greenspan 2015, p. 7) due to inactive miners - the network stalls. If a network split occurs, "[...] the fork with the longer chain will be adopted as the global consensus. The diversity threshold ensures that the longer blockchain will belong to the island containing the majority of permitted miners, since the other island's chain will quickly freeze up" (Greenspan 2015, p. 8). *MultiChain* is only an example for a general round robin approach.

Child-chains can be used with every BCT and presuppose parallel BC threads. Further, *side-chains* assume that there are transactions which necessarily belong on the main tread (primary chain) and others which do not need to be stored on the primary chain (KIM ET AL. 2018, p. 1206). Therefore the ladder can reduce the load on the *primary chain*. Here transactions which are written to the *primary chain* are called *on-chain* and those written to any *sidechains*, *off-chain*. "In a traditional on-chain transaction, each transfer would have to be validated by all peers in the network before the transfer is labelled as complete, which keeps it very slow. In contrast, in an off-chain transfer, not all peers need to wait for the transfer to be verified before it is labelled as successful or complete" (Sharma et al. 2020, p. 15).

Concluding, *MultiChain* seeks a reduction of needed overall computation power and *child-chains* reduce the reconciliation effort. Last, as *child-chains* are not tied to a special CM, they are covered in the following subsection Performance improvements. A comparison of the given CMs follows thereafter.

2.2.3 Performance Improvements

Next to the general functionalities to achieve consensus, there are some scalability solutions, which can be applied to multiple upper level CMs. On this topic, KIM ET AL. (2018, p. 1204) offer five categories of improvement-methods they found in their survey. Whilst On-chain, Off-chain and Child-chain improvements operate within a main-chain's network (only), Side-chain and Inter-chain try to connect BCs out of different networks. The key facts are given briefly:

1. "The **on-chain** solution refers to methods that increase scalability by modifying only elements within a blockchain" (KIM ET AL. 2018, p. 1204). One prominent example would be to increase the number of transactions within one block until it is considered a 'Big block' (KIM ET AL. 2018, p. 1204).

In the following specific case, the "[...] Bitcoin Core protocol limits blocks to 1 MB in size. Each block contains at most some 4,000 transactions. Blocks are added to the blockchain on

average every 10 minutes, therefore the transaction rate is limited to some 7 transactions per second" (Goebel and Krzesinski 2017, p. 1). If a block was allowed to be bigger than the traditional 1 MB of Bitcoin Core, it would enable more transactions to be processed in the same amount of time (Table 2.2: 'Big block' 1 & 2), as solving the PoW's mathematical mining problem is barely constraint to the actual block size (Zhang et al. (2018)).

- 2. "The off-chain solution is to improve the scalability by processing the transactions at outside the blockchain. This is also called a state-channel solution, because it maintains the state of the main-chain and applies the last state that has been processed in the other channel" (KIM ET AL. 2018, p. 1205). In other words, in "[...] a traditional on-chain transaction, each transfer would have to be validated by all peers in the network before the transfer is labelled as complete, which keeps it very slow. In contrast, in an off-chain transfer, not all peers need to wait for the transfer to be verified before it is labelled as successful or complete" (Sharma et al. 2020, p. 15). 'The Bitcoin Lightning Network' (Poon and DRYJA (2016)), Sprites (MILLER ET AL. (2017)) and game channels (KRAFT (2016)) fit into this category. All three papers split transactions from the main chain, starting with an introducing interaction just to merge the results of the sidechain branch back into the main chain later on. In figure 2.3 temporary off-chains can be seen. Whilst the main-chain is managed by all nodes of the network, the off-chain state channels are only governed by the (two) affected nodes. Generally, Poon and Dryja (2016) can be seen as the founders of this idea and MILLER ET AL. (2017)'s research aims to further improve the speed. As those two are located in the cryptocurrency business only, they are out of scope. Regarding BCT in games, LANEVE AND ERSHOV (2019) (p. 27) state that this "[...] approach gives the developers all the flexibility they need with proven infrastructures while decentralising only the parts that need it". Although Kraft (2016) is using BCT based on a cryptocurrency in both on-chain as well as off-chain transactions, he is located within online gaming business. Therefore KRAFT (2016)'s research will become important later on.
- 3. "The **child-chain** solution has a parent-child structure, processes the transactions in the child-chain, and records the results in the parent-chain" (KIM ET AL. 2018, p. 1206). On the one hand, this can be used for further improvements on cryptocurrency transactions (KIM ET AL. (2018, p. 1206)), on the other hand, special transactions can be stored on the child-chain, which are only transferred to the parent-chain on fulfilled smart contracts (see Data allocation improvements in chapter Blockchain in Games). Consequently, in figure 2.3 a sample permanent *child-chain* is 'every once in a while' used to push certain data into

	Transaction / second	Transactions / block	Block size
Bitcoin Core	7	4.000	1 MB
Big block (1)	70	40.000	10 MB
Big block (2)	700	400.000	100 MB

Table 2.2: Blockchain Blocksize

the main-chain (dark arrows).

- 4. "The **side-chain** approach is to exchange assets of different blockchains with each other. And their goal is to bring the function of another blockchain into the current blockchain" (KIM ET AL. 2018, p. 1205). In the first place, two BCs with different CMs are combined to obtain the advantages of both approaches, such as using smart contracts of Ethereum together with Bitcoin's cryptocurrency (KIM ET AL. (2018), p. 1205-1206). Here, prominent example which is mentioned more often in the literature is BACK ET AL. (2014)'s paper about 'Pegged Sidechains'.
 - In figure 2.3 the side-chain approach is used, if one of the layers needs a divergent CM to serve special service characteristics. In section Interoperability in chapter Proof-of-Turn the PoT CM serves as as an off-chain/side-chain solution for a PoW BC (e.g. for an ecosystem's in-game currency), whilst the PoT CM is served by an off-chain/side-chain PBFT BC to enable straight forward and lightweight voting (Figure 4.13).
- 5. "The **inter-chain** method is a way to enable communication among the various blockchain" (KIM ET AL. 2018, p. 1206). Inter-chain is very similar to the side-chain approach and tries to make e.g. Litecoin and Bitcoin interoperable (KIM ET AL. 2018, p. 1206). Hence, the inter-chain method can be seen as the infrastructure technology for implementing the side-chain (KIM ET AL. 2018, p. 1206).

Further advantages and disadvantages, here out of scope, can be taken from KIM ET AL. (2018) (p. 1206, Table 1., A comparative analysis table of the scalability issues). Concluding, if possible, a novel CM shall offer the possibility to be interoperable, as Besancon et al. (2019, p. 81) states that the need for interoperability "[...] can be found at multiple levels: a) between different BC, b) between different projects running on the same BC, and c) between BC and other technologies used to create decentralized applications."

Later on, during the design of PoT, **on-chain** improvements in regards of number of blocks written within a given time slot, the **off-chain** method regarding interoperability, **child-chains** to counterweight in-game mechanics as well as **side-chain** solutions for mixing CM characteristics will be of interest.

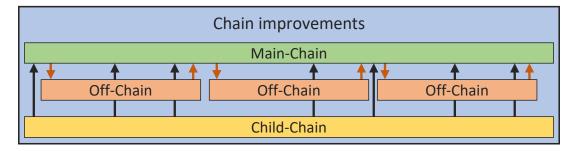


Figure 2.3: Off-Chain & Childchain

2.2.4 Comparison of Consensus Mechanisms

To sum up the given CM overview, Table 2.3 (Proof based CMs) and Table 2.4 (Vote based CMs) show the mechanism's characteristics regarding *write permissions* and *BFT*. Values are taken from Khan et al. (2020, p. 5), which match values from DIB et al. (2018, p. 55).

Although values in Table 2.5 (Additional CMs) are only anticipated, they are given to complete the list. Table 2.3, 2.4 and 2.5 show the different approaches, which can be used to model a game specific CM. All values, to the author's knowledge, are given for plain CMs without enhancements from the previous section performance improvements.

Nevertheless, the "[...] existing proof-of-based models seem to be supporting open participation but they are not suitable for the real-time applications where immediate transaction finality and high transaction rates are the main requirement" (Khan et al. 2020, p. 6). Finally, as all theses mechanisms are solving storage related issues, the 'CAP theorem' as described by Brewer (2012), has to be mentioned as well. In a distributed data store, such as BCT, out of the three following properties Consistency (C), Availability (A) and Partition Tolerance (P) only two can be ensured (DE ANGELIS et al. 2018, p. 6). Thus either C+A, C+P or A+P can fully be met (DE ANGELIS et al. 2018, p. 6). Additionally, "[...] an Internet-deployed permissioned blockchain has to tolerate these adverse situations: (i) periods where the network behaves asynchronously; (ii) a (bounded) number of Byzantine authorities aiming at hampering availability and consistency" (DE ANGELIS et al. 2018, p. 7). Consequently from (i), there is no way around P and a trade off between C and A has to be found (DE ANGELIS et al. 2018, p. 7). The CAP theorem will further be covered in chapter Proof-of-Turn, section CAP Theorem measurement.

Additionally, from the given information it can be concluded that *forks are expensive*. Consequently, designing a CM without forks or *diminished computation in vain* along forks is strived for to increase energy efficiency and to reduce network traffic as well as inconsistencies. Still, to meet the needs of speed/throughput the interoperability with **off-chain**, **side-chain** and **child-chain** improvements is beneficial.

Revising the research question, "What is the best fitting BCT CM to cover an asynchronous game play scenario?", the answer is not clear, yet. If there was a novel CM design, the lowest possible consultation in regards of the given scenario is aimed for. This solution does not necessarily need

Proof based	Write	BFT
CMs	permission	
PoW	Heavy computation	<25 %
PoS	Stakeholder	<50 %
PoA	Authority	<51 %
MultiChain	Round Robin	_
PoET	Time	_

Table 2.3: Proof based consensus mechanisms (Khan et al. 2020, p. 5)

Vote based	Write	BFT
CMs	permission	
PBFT	Vote rounds	<33.3 %
Ripple	Leader rotation	<20 %
TCM	Leader rotation	<33.3 %

Table 2.4: Vote based consensus mechanisms (KHAN ET AL. 2020, p. 5)

Proof based	Write	BFT
CMs	permission	
PoP	Effort in game	-
PoT	Round Robin, turn (time)	Implementation
	and special rules	dependent

Table 2.5: Additional consensus mechanisms

an attached cryptocurrency.

In chapter Proof-of-Turn the following characteristics will become important:

- 1. Round Robin, in regards of reducing the permissioned writing node down to one using a fixed succession.
- 2. Time, not as strict as in PoET, but not negligible.
- 3. *BFT*, to an (at least) reasonable extend of <33.3 %, or more (implementation dependent).
- 4. *CF*, both in the fashion of short-termed forks as well as possible long-termed rewrites of the blockchain.
- 5. *PoP*, in regards of the final scope of the PoT CM for games.
- 6. Off-chain, as stated in Kraft (2016)'s paper using the phrase Game Channel.
- 7. Child-chain(s)/Side-chain(s), in the manner of organizing different types of transactions.
- 8. *Interoperability*, as an additional desirable characteristic.

Still, before getting into the details of the PoT CM in chapter Proof-of-Turn, the next chapter Blockchain in Games will list characteristics BCT needs to fulfill in general to serve turn based games.

Chapter

Blockchain Technology in Games

Since Nakamoto (2009)'s paper, many different CMs were invented, as given in section Consensus Mechanisms. Currently the use cases of BCT in games can be grouped by concepts of 'real-world ownership', 'digital tokens which represent game assets', the 'reduction of upkeep costs' as well as 'enhanced engagement from the user-base' (Laneve and Ershov 2019, p. 15). In other words, MIN ET AL. (2019, p. 1) state that BCT in games is used for Rule Transparency, Asset Ownership, Assets Reusability, User-Generated Content as well as Current scalability. Therefore, using ".. games to grasp the functionalities of cryptocurrencies and the blockchain technologies that make them possible seems like an obvious choice" (Serada et al. 2020, p. 2), especially as beneficial synergies between the needs of gaming networks and promises of BCT can be found. This intersection seems to be situated around slow paced asynchronous games, as BCT is basically not suited for fast applications (SERADA ET AL. 2020, p. 19). Therefore the following chapter wants to answer the first guiding question: "Which kind of SCs need to be established to cover typical in-game mechanics?". Hence, first recent solutions are analyzed and grouped by ownership, gameplay and network costs. Second, the gaming market is examined briefly and some sample games are presented. This shows blind spots of BCT game play and respective (asynchronous) in-game mechanics. Third, a problem area is defined to increase tangibility of problems in distributed games. Last some solutions to these problems are shown, custom tailored for distributed game play. Afterwards, assuming that BCT is suitable for (asynchronous) games, the following chapter,

Proof-of-Turn, will provide a CM which aims to deliver the features described at the end of the last chapter, Blockchain Technology.

3.1 Ownership, network costs, gameplay & scalability

For ownership models, the general idea is derived from Bitcoin which administers a cryptocurrency. Besides or instead of a cryptocurrency, digital assets are regulated by the game's BC (PFEIFFER ET AL. 2020, p. 2). This idea is called Collectibles (Collection Games) and, as it is not too circuitous, it was one of the first mentioned type of games hosted on BCT. "The ownership enables the game assets to be independent of specific game operators, which allow the players to retain their digital properties and in-game relationships, even after the game stops its operation" (MIN ET AL. 2019, p. 1). A prominent example is CryptoKitties (PFEIFFER ET AL. 2020, p. 2). Next to CryptoKitties many other cryptogames were developed (e.g., Cryptopunks, Decentraland,

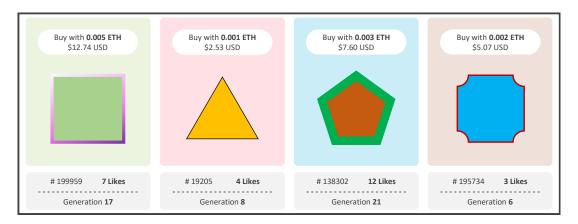


Figure 3.1: Collectibles store 'CryptoShapes' (Adapted from CRYPTOKITTIES.CO (2021))

MyCryptoHeroes, HyperDragons, Gods Unchained, Etheremon, Blockchain Cuties, NeoWorld, and Axie Infinity) (SERADA ET AL. 2020, p. 2), which fall into this category. A sample store to buy assets from these *Collectibles* might look like figure 3.1. Digital assets can be bought using some sort of payment channel (here the cryptocurrency: Ethereum (ETH)). Sometimes Collectibles offer the opportunity to breed new assets out of the existing ones (SERADA ET AL. 2020, p. 2). During breeding properties (here: border thickness, colors, edges, size etc.) of two shapes are recombined. To increase asset diversity, sometimes mutations happen and new characteristics (e.g. color gradients, figure 3.1: rectangle) occur, which makes the single asset special. As a bread shape can be considered a child-generation of the two parent shapes, breeding generations are given (compare CRYPTOKITTIES.CO (2021)). The breeding may be rather seen as part of the Collectibles-game than part of the asset ownership itself (PFEIFFER ET AL. 2020, p. 2). Further, one advantage of Collectibles is, that the "[...] possibility of asset trades across different games and blockchain platforms stimulates the players to better engage in the game economy" (MIN ET AL. 2019, p. 1). Regarding reusability, game "[...] developers can leverage blockchain to design an ecosystem that allows players to reuse their characters and virtual items across different games. To this end, newly launched games can directly inherit game assets from the existing ones" (MIN ET AL. 2019, p. 1).

Moreover, User-Generated Contents "[...] in traditional games are restricted in the specific game, thus, belongs to the game operator. In contrast, these contents can be preserved by the players, thus, has the potential to be shared among multiple games. This benefit will, in turn, encourage the players to participate in the construction of new content" (MIN ET AL. 2019, p. 1).

These connected games are called 'Layer two games' as they use Non-fungible tokens (NFTs) "[...] from other games to fuel their gameplay. Players could only access these games by using items they had acquired from other sources" (Laneve and Ershov 2019, p. 26). Depending on the used cryptocurrencies or tokens, "[...] game operators may benefit from the value increase of the tokens they issued" (MIN ET AL. 2019, p. 1).

Network costs — No game has been found which uses BCT only for the sake of reducing upkeep costs of servers, yet. In this regard, Serada et al. (2020, p. 3) states: "Oftenmost cryptogames are neither developed by game companies nor is their design shaped by typical business models or assumptions such as profit making."

Gameplay — Here, the literature emphasizes the value of *rule transparency* as an anti-cheating mechanism and BCT's potential in games (DIB ET AL. (2018, p. 57); MIN ET AL. (2019, p. 1)). BCT "[...] addresses one of the biggest issues with direct peer-to-peer transactions online; that is, a lack of trust. Historically, a lack of trust has been cited as one of the greatest disadvantages of online gambling [...]" (GAINSBURY AND BLASZCZYNSKI 2017, p. 483). "Due to the transparent characteristic of blockchain data, players or third-party organizations can audit the smart contract based games rules, which was hidden in the centralized server in traditional games. The transparent game rules will enhance the trustworthy of the game operation" (MIN ET AL. 2019, p. 1).

Yet, except the theoretical PoP paper from Yuen et al. (2019), the literature only shows one single example of BCT in games to effectively replace the *central server network approach* (Wu et al. (2020)) especially if cryptocurrency ownership models are *excluded*. Wu et al. (2020, p. 4559-4560) came to the same conclusion: "To the best of our knowledge, this is the first blockchain-related game demostration that achieves a real-time serverless gaming system with an anti-cheating mechanism." Still no professionally produced game could be found, which already left its BETA-status or can be considered more than a proof of concept, such as *Huntercoin* (Kraft (2016)).

Current scalability — Additionally, scalability of BCT has to be put in contrast: "As it stands, blockchain technology does not seem applicable for the design of the most popular game genres such as first-person shooters or real-time strategy, although several attempts have been made in this direction [...]" (Serada et al. 2020, p. 3). This statement derives from BCT focusing on consistency and partition tolerance regarding the CAP-Theorem instead of high availability (DE ANGELIS ET AL. (2018)). Additionally cryptocurrencies like Ethereum and Bitcoin show that "read availability of blockchains is typically high [...]" whilst "[...] write availability — for transaction management — is actually low" (Weber et al. 2017, p. 64).

Furthermore, in a centralized server context, all players (p) send their information to the server - 'one on one' relationships for the players and 'one to p' for the server, sum: 2p. In CMs wherein all nodes are allowed to write, the number of relationships (r) becomes squared $(r=p^2)$. Consequently, consultation and throughput increases tremendously with growing networks. CMs which reduce the writing nodes, help to cure this scalability issue.

Additionally, a major constraint is the visibility of every player. It has to be determined who receives necessary (movement) information and who does not. This could either be done to hide information or to reduce network throughput. This information is especially restricted by objects blocking the visibility as shown in the 2D-figure 3.2 (Playground of RED BLOB GAMES (2020) was used). In figure 3.2, a player (circle, middle) is shown with its range of visibility (lighted area). The visibility is blocked by rectangular objects, which cast shadows. Yet alone in figure 3.2 (A) there are no constraints. But assuming other players around (Figure 3.2, B) one can tell that distance is not the only measure for other players to be displayed - a 'Line of Sight' has to exist.

Leaving the display to honest nodes is not applicable here as fraud is a major concern in this document. Adding cryptography and signing information only to specific players to prevent fraud, slows down the network even more. Consequently, whilst a fast paced four player game might be possible using the right CM, 40 simultaneous players as in *Star wars Battlefront* (Wikipedia (2021e)) or even 64 as in *Battlefield* (Wikipedia (2021a)) are far out of reach (Serada et al. 2020, p. 3). Last, in distributed environments each player wants to keep its position secret as long as possible to prevent cheating. Still the position of both players has to be published to calculate the possibility of a 'line of sight'. This brings the anti-cheat mechanism at odds. Hence, BCT is clearly not a solution for high traffic (shooter) games and due BCTs characteristic to be "barely scalable" Serada et al. (2020, p. 19), MMORPGs with multiple simultaneous actions are out of scope as well.

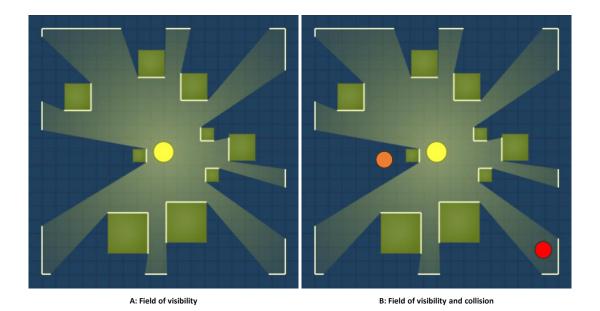


Figure 3.2: Visibility and Scalability

3.2 Gaming market

Until now the focus of the document was primarily on the technical side. This section sheds the light on the recent gaming market - especially on the gamers. The following data is taken from a market report of LIMELIGHT NETWORKS (2020).

First it has to be determined which kind of device is suitable for games based on BCT. The given data (Limelight Networks 2020, p. 7) attest mobile phones a superior position for gaming globally (Figure 3.3). This claim is backed by the data of each observed country (Appendix - Figure A.1). If a publisher wants to tackle a big market, mobile phones shall be considered before targeting personal computers and gaming consoles. Additionally, tablets are also part of the mobile market as they are run with the same operating systems (STATISTA.COM (2021b)) and offer the same distribution channels. Consequently, slow paced (turn based) games which can be played on mobile phones or cross platform offer a market to be hosted on BCT. Although data allocation improvements may be established in a game, BCT still needs storage space (Besancon et al. 2019, p. 81) as data can not just be pulled on demand from a central server. Still, as a Counterpoint Research-article from Wang (2021) stresses: "Average smartphone nand flash capacity crossed the 100GB threshold in 2020" (Appendix: Figure A.2). This trend enables gaming with smartphones using BCT. The other devices, computers and gaming consoles, from figure 3.3 are supposed to offer more or at least similar amounts of storage capacity.

Second, game characteristics shall be mentioned which are preferred by gamers. Data out of the report (LIMELIGHT NETWORKS 2020, p. 18) offers five categories of preferred games (Appendix: Figure A.3). The average of each category is shown in Figure 3.4. The categories 'Interesting storyline' as well as 'simple gameplay' are out of scope as they have no influence on the game's backbone (e.g. BCT). On the contrary 'Fast performance', quickly loading and speedy interactions, 'offline activity', the ability "to play the game while disconnected from the internet") (LIMELIGHT NETWORKS 2020, p. 18) as well as the 'mulitplayer' option are of importance regarding BCT. Here, BCT enables offline time slots (NAKAMOTO 2009, p. 8) in multiplayer games without the need for a central server. Depending on the application - as stated before - fast performance is dependent on the chosen CM.

Third, "[...] fair play is essential to any game [...]" (YAN AND RANDELL 2009, p. 44) and thus cheating (playing against agreed rules) harms gameplay. Hence (HAEBERLEN ET AL. 2010, p. 2) cites from McGraw et al. (2010): "Cheating in online games is an important problem that affects game players and game operators alike". Many centralized games suffer in their game experience from this type of hostile behavior. Therefore some "games try to prevent modding (e.g., multiplayer games to avoid cheating)" (Lee et al. 2020, p. 2493). Others install anti-cheating systems like PunkBuster, Warden or Valve Anti-Cheat (HAEBERLEN et al. 2010, p. 8). "In addition to privacy concerns, this approach has led to an arms race between cheaters and game maintainers, in which the former constantly release new cheats or variations of existing ones, and the latter must struggle to keep their databases up to date" (HAEBERLEN et al. 2010, p. 8). In the end anti-cheat measures commonly annoy gamers and reduce gaming system's performance. With these problems in mind, BCT promotes itself, despite its lack of speed and the multiple data allocation, to be used in slow paced online (multiplayer) games.

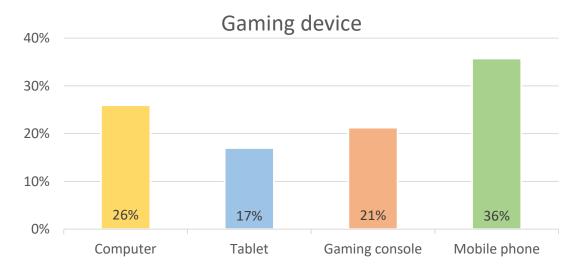


Figure 3.3: Global share of devices by gaming time (Data from LIMELIGHT NETWORKS (2020))

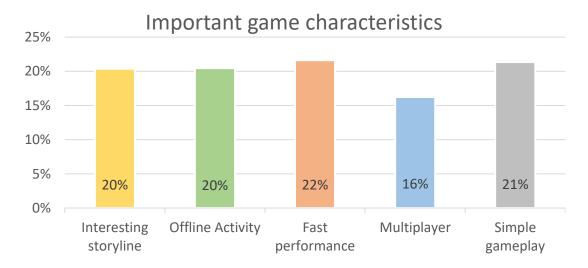


Figure 3.4: Global preferred game characteristics (Data from Limelight Networks (2020))

3.3 Existing games using Blockchain Technology

With the knowledge of the previously given four categories Ownership, network costs, gameplay & scalability, some already existing games, using BCT are described:

- 1. As described before *CryptoKitties* is a game based on collectibles wherein digital assets (here: virtual cats) can be collected, bred, bought and sold (SERADA ET AL. 2020, p. 2). CryptoKitties itself was born "[...] from a Hackathon idea and released officially in November 2017" (LANEVE AND ERSHOV 2019, p. 25). According to Laneve and Ershov (2019, p. 15)'s as well as Serada ET AL. (2020, p. 18)'s research the Ethereum network was slowed down by CryptoKitties significantly during that time. "One of the pioneering features of Cyptokitties was the introduction of the ERC-721 protocol for NFTs, which rendered each kitten unique instead of being another form of cryptocurrency" (LANEVE AND ERSHOV 2019, p. 25). The general case of reusability as described by MIN AND CAI (2019) before is reflected by PFEIFFER ET AL. (2020, p. 16): "Applications of The KittyVerse are not only developed by the Cryptokitties producer but also by third developer studies and this shows the special potential of virtual objects/assets on Blockchain basis. The possession is with the player and the asset can be used for other games or applications." Thus CryptoKitties faced many imitators such as Cryptopunks, Decentraland, MyCryptoHeroes, HyperDragons, Gods Unchained, Etheremon, Blockchain Cuties, NeoWorld, and Axie Infinity (SERADA ET AL. 2020, p. 3). Further information about CryptoKitties can be found in Laneve and Ershov (2019)'s paper (p. 25-26). Staying at ownership models, Forgotten Artifacts (LOSTRELICS.IO (2021)) is mentioned because it uses blockchain to prove ownership of assets, but stays a role play game in the first place (LANEVE AND ERSHOV 2019, p. 29). Forgotten Artifacts is described in more detail by LANEVE AND Ershov (2019) (p. 30-34).
- 2. DECENTRALAND.ORG (2021) is a game which combines collectible traits with crafting mechanics (alike 'Minecraft'). It "proves ownership of virtual land hosted on a decentralised content distribution system" (Laneve and Ershov 2019, p. 29). More information on Decentraland can be found in Laneve and Ershov (2019)'s paper (p. 38-41).
- 3. Although it can only barely be listed as part of a games list, Socios.com (2021) is mentioned because it connects realworld assets (e.g. voting rights) with digital assets (tokens) stored on a BC. The digital tokens provide voting rights, which makes *Socios* a voting platform (Laneve and Ershov 2019, p. 29). Again, further details can be found in Laneve and Ershov (2019)'s paper (p. 35-37).
 - Generally, as "[...] of April 2019, based on open data sources, the number of cryptogames was estimated to be over 650, excluding gambling games" (SERADA ET AL. 2020, p. 3). Within this pile of games there are even games containing terms of useage, which "[...] are completely contrary to the spirit of the blockchain. Players' ownership of virtual properties cannot be guaranteed." (MIN ET AL. 2019, p. 2).
- 4. A completely different approach was taken by *Huntercoin* using the cryptocurrency CHI from the XAYA blockchain framework (XAYA.IO (2021)). "*Huntercoin* started as a 1-year

experiment in 2014 to test how well a blockchain network could handle thousands of transactions happening in real-time, but due to an explosion in popularity development continued [...]" (LANEVE AND ERSHOV 2019, p. 24). Huntercoin uses the PoW algorithm (KRAFT 2016, p. 85). "When a miner finishes a block, he gets 10% of the block reward in CHI, the other 90% gets added to a pot that is then distributed to developers so they can reward to players" (Laneve and Ershov 2019, p. 44). Additionally, for "[...] a fee paid in huntercoins, users can create hunters (corresponding roughly to player accounts) in this game world. This allows for human mining: Parts of the block rewards are not paid to the proof-of-work miner but instead placed inside the game world, where hunters can pick them up and bank them to their on-chain address. This is not straight-forward, however, and requires skill since other hunters can fight for and steal the coins until they are secured. This is intended to give humans a chance to "mine" huntercoins by playing the game" (KRAFT 2016, p. 85). A downside of this approach, used by Huntercoin and Motocoin, is given in the PoP core paper: The "[...] act of play in both Huntercoin and Motocoin [...] becomes incentivedriven due to the blockchain, making the game progress lack entertainment" (YUEN ET AL. 2019, p. 21). Moreover, KRAFT (2016, p. 84) being the developer of huntercoin states that this approach leads to "[...] large growth of the blockchain and heavy resource requirements". Last, as playing the game offers the opportunity to earn CHI, Huntercoin sufferd from an army of bots mining ingame until certain countermeasures were introduced (KRAFT 2016, p. 88-89).

5. Throughout this list, *Taurion* seems like the most interesting game regarding the scope of this thesis. Although still in development, it focuses more on a throughout gameplay using BCT instead of asset monetization as it "uses a custom blockchain to host the game world and its interactions" (Laneve and Ershov 2019, p. 29). Taurion, as Huntercoin, claims to use the XAYA blockchain framework. "To handle the speed of transactions and scalability needed to host massively multiplayer online games, the XAYA team introduced three mechanisms: Atomic Transactions, Game Channels and Ephemeral Timestamps" (Laneve and Ershov 2019, p. 42). "Game Channels for Turn-Based Interactions" are described by (Kraft 2016, p. 90-92) in more details. Game channels can be seen as off-chain games, branching off the main game to reenter later on (Kraft 2016, p. 91). They "[...] interact with the blockchain only for part of their functionality" (Laneve and Ershov 2019, p. 27). Recently, *Taurion* is on hold as the developers concentrate on *Soccer Manager Elite*, which primarily fulfills the ownership case (yet again).

Finally, revising all the mentioned use cases, BCT is not (yet) used for cutting costs on infrastructure for the game publishers, which is one main goal of this document. Still, BCT "[...] could be cost-effective, removing the centralized authority's need to monitor and regulate transactions and interactions between different members" (Sharma et al. 2020, p. 5).

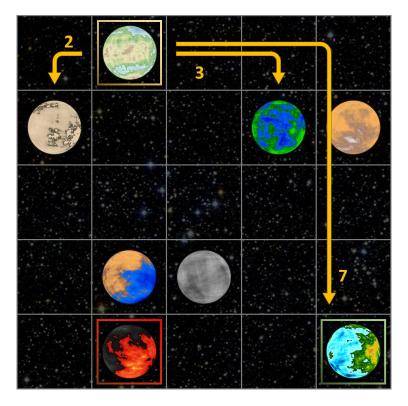


Figure 3.5: Sample game

3.4 Problem area

Once BCT is chosen for cost cutting and used as a trust building backend, games in scope need to be chosen and SCs need to be defined.

The PoP-mechanism already provides a good basis for games and is considered as a basic benchmark throughout this chapter. Hence it is assumed that the BCT, which is used, meets speed criteria in non critical scenarios (≤ 1 second delay).

Consequently, in-game mechanics which rely on fast reactions, such as first person shooter games or based on real-time strategy are not supposed to be in the scope as updates might not reach the opponents in time and thus prevent good game play experience.

Nevertheless, there are game genres, which are in the scope such as card-, quiz-, collaborative-as well as (slow) strategy games. First, *Hearthstone* (PLAYHEARTHSTONE.COM (2021)) and *Magic: The Gathering Arena* (MAGIC.WIZARDS.COM (2021)) are prominent examples for (online) card games. Second, the German mobile app *Quizduell* (QUIZDUELL (2021)) can be seen as an example for simple question-answer-games. Last, *Pen- & Paper* role-play games and strategy games which allow more than 1 second response time e.g. through separated turns such as *play-by-email* video games (WIKIPEDIA (2021b)) are beneficial.

Next, a hypothetical game is introduced, Reach for the Stars (RftS) (see figure 3.5), to become

less generic and abstract in the subsequent chapters. After a short introduction, significant game mechanics are extracted and described theoretically. RftS is a 2D, turn-based multiplayer game, where players try to colonize all planets in a sample galaxy - alike the game *Risk* (HASBRO (2021)). At the beginning, each player starts with only one planet. Each planet has both, a fixed and a variable amount of production of space ships. For the variable amount, *randomization* is needed. From a controlled planet, fleets can be sent to colonize/conquer other planets. Fleets need fixed game-rounds, based on grid distance to reach planets (Figure 3.5, arrows with grid distance). Fleet movements are not visible for the other players. Hence the game lives from the nature of *hidden transactions* and consequently originating *fog of war* (BY (2011); HAGELBÄCK AND JOHANSSON (2008); SETEAR (1989)). Fleets can be called back to the origin planet (once), but can not change the direction otherwise (*Follow-Up hidden transactions*). For the sake of reduced complexity colonization as well as battles are computed by linear algorithms and here not further of importance. Last, there are two piles for each player to *draw cards* from. Both decks offer equal cards, but one is a *shared deck* for all players and the other is a *private deck*. The information given on the cards is not important.

From this RftS-scenario, we can extract the need for randomization, hidden transactions including Follow-Up hidden transactions, fog of war as well as mechanics for private and shared decks of cards. In this regard, time constraints on local machines (e.g. game: M.U.L.E. from 1983, WIKIPEDIA (2021d)), as they are not seen to be confirmable, are out of scope. Last, as storage space on low end devices endagers gamers to keep the game, possible considerations to reduce storage space shall be taken into account. For these transactions and movements SCs have to be established which can be used in a BCT context.

3.5 Game specific smart contracts

Although there are innumerable designs of SCs, games offer some common patterns. Some of these patterns like *hidden transactions*, *fog of war*, *pile of cards* or *randomization* were just mentioned and are explained together with *reveal claims triggers* and *disputes* subsequently:

3.5.1 Hidden transactions & Randomization

If a game incorporates transactions which shall not be visible to other players immediately, a suitable SC is needed. The *receiving players* have a justifiable desire to obtain a proof for the *publishing player's* interaction. Still the *publishing player* wants to sustain the information, contained in the transactions, unpublished as long as possible/needed. Hence there is a need for the information to be published 'veiled', which becomes 'unveiled' later on. There are two approaches for this course of action, which from now on will be called *offset revealing*.

First, the *publishing player* uses **encryption** to veil the information and pushes it into the BC network. By the time the needed information has to be revealed, the password of the encryption is published. As all nodes do not trust each other, they recalculate the transaction with the newly published key. **Second**, only the hash of the transaction data **game hash** could be published (KRAFT 2016, p. 94). Both solutions offer benefits and downsides.

On the one hand, a game hash is lightweight ($\sim 0.032~{\rm KB})^2$ and abstracts from the actual storage size of the transaction. On the other hand, encryption allows to pass the decryption key to single elected players instead of the whole network. If this measure is needed, game hashes might be the inferior solution. Hence, decryption to a part of the network becomes relatively easy/cheap in the case of encryption. Still, encrypted data might offer implications on the content (e.g. transaction size) or the game requires data to be written unencrypted to the BC later on. Without using storage restoring procedures (see chapter: Data allocation improvements), encryption might lead to unnecessary allocated storage. Additionally, if there are only a few movement options, which can be iterated using 'brute force', salt data (Morris and Thompson 1979, p. 597) should be considered. However, the BC itself would look like figure 3.6 which aligns the blocks according to their publication time from old (left) to new (right). In figure 3.6 there are blocks 'before m-1', 'after p+1' and 'in between the revelation m+1 to p-1'. Block m equals either the encrypted or hashed data, whilst block p equals the encryption key or the data described by the hash. Block p reveals the information contained in block m to the network.

This procedure can be used for semi-simultaneous turns as well (KRAFT 2016, p. 94). Here 'sim-

² The SHA256-algorithm generates fixed size 256-bit (32-byte) hashes (RACHMAWATI ET AL. 2017, p. 7).



Figure 3.6: Offset revealing of a single block (As described by KRAFT (2016))

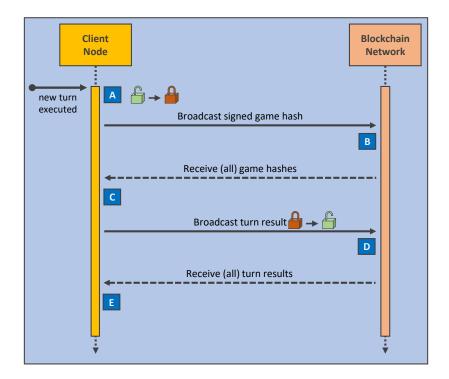


Figure 3.7: Offset revealing with game hash (From YUEN ET AL. (2019))

ultaneous' does only refer to the game play - depending on the CM, the network can both be sequential (e.g.: PoP) or semi-parallel (e.g.: PoW).

In YUEN ET AL. (2019, p. 22)'s paper, which introduces PoP, a procedure for *simultaneous hidden turns* is given alike figure 3.7. In this case, multiple players have to publish any kind of information *simultaneously*. The recent transaction's *game hash* will be encrypted (Figure 3.7, A) and sent (Figure 3.7, B). Comprehensibly, none of the players wants to reveal the information before the other has not published his information in the BC. Hence it is waited until all involved nodes have published their *game hash* (Figure 3.7, C). As soon as all information is gathered, the information is released (Figure 3.7, D). Finally, all players could make their moves simultaneously and check the other peer's results (Figure 3.7, E).

Consequently, synchronous and asynchronous as well as solo, paired and grouped hidden transactions can be served with this type of SCs.

Hidden follow-up movements

Likewise to figure 3.6, in a game a following transaction might occur, which implies the existence of the base transaction (Figure 3.8). This type of transactions for BCT has not yet been described in the literature.

To stay within the context of RftS a fleet, sent to any planet, is called back before its arrival at the intended location. This movement is only allowed given the side constraint that it has to return

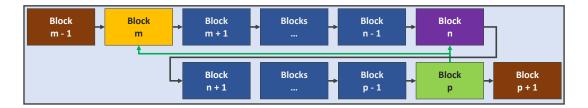


Figure 3.8: Offset revealing with follow up

to its planet of departure. Additionally, such a maneuver can only be performed once within a fleet-sending procedure.

As long as the implementation dependent *maximum time to hide information* is not exceeded, *hidden follow-up movements* are possible. On a more abstract level, in figure 3.8, **m** is the base transaction enhanced by the follow-up transaction **n**. Both **m** and **n** are revealed by block **p**. In figure 3.8 the hidden follow-up movement is consequently finished before block **p+1** is published. If not intentionally, in a poor design of upper level software, the node may be enforced to publish block **m** before the fleet has returned to the planet it has originated from. Hence block **m** needs to be published by e.g. any *timeout rule*. The timeout implies (assumption) that the fleet would have had to reach the destination planet already. Consequently block **m** represents an inconsistency. Therefore, at least a statement has to be published that the fleet's callback has occurred somewhere in between. Now:

- 1. The range of rounds for the revisit of the fleet at its origin country to occur becomes limited.
- 2. If there is only one block left which could represent the callback, the hidden follow-up movement is revealed by implication. The number of rounds between sending and calling back offers all players to calculate the round for the fleet to return.
- 3. If the *timeout rule* is set very low, block **n** has to be published as well before the fleet has finally returned.

As the implications of *hidden follow-up movements* are known, further details remain implementation dependent.

Randomization

Randomization in a distributed network needs *hidden transactions* as well as *trigger events* (section Reveal claims, trigger events & Disputes). To compute (pseudo) random values a common procedure is to take a *random number generator* and give it some sort of *seed*. Mostly something alike the *system clock time* or requested *random mouse movement* etc. is used. Nevertheless, if one node **P** needs a random number, other players might not comply with **P**'s local generated number - a lack of trust into **P**'s seed exists. Chatterjee et al. (2019) address this phenomenon in detail and describes four approaches to solve the problem. Herein Chatterjee et al. (2019) discourage from three of these approaches:

- 1. The last block's hash can be seen as a random value, therefore it can generally be used as a seed (Chatterjee et al. 2019, p. 403). Still, using a value from "[...] the current block as a seed [...]" (Chatterjee et al. 2019, p. 403) is seen not desired as the previous block may be tempered by a party which has interest into a 'fitting' random value (Chatterjee et al. 2019, p. 406). Harkanson et al. (2020)'s approach targeting 'Texas Hold'em Poker' can be put in this category.
- 2. Alternatively an external provider could be used who acts as an oracle for the network (Chatterjee et al. 2019, p. 403). This solution is out of scope as it requires a trusted external source.
- 3. Chatterjee et al. (2019)'s own algorithm may be used. In this case a BC environment has to be used which enforces to pay fees (Chatterjee et al. 2019, p. 403). This solution is encouraged by Chatterjee et al. (2019, p. 403), but the fees in form of cryptocurrency tokens are seen as a flaw is this document's context.
- 4. Last, the "anyone can submit randomly-generated numbers" (Chatterjee et al. 2019, p. 403) approach. Although Chatterjee et al. (2019, p. 406) discourage from conducting this approach, it is given in more detail as one of Chatterjee et al. (2019, p. 406)'s grounding assumptions can be changed.
 - But first, the process: For the sake of simplicity a simultaneous hidden turn is assumed after a trigger call from **P** (Figure 3.7). Node **P** offers a *game hash* which represents a certain number for the seed (Figure 3.7 & table 3.1: **B**). Subsequent **P** awaits other nodes to add their desired values as well (Figure 3.7 & table 3.1: **C**).

Depending on the implementation, **P** waits until at least one answer arrives or 'a certain time is passed' (Chatterjee et al. 2019, p. 406). On the one hand, the 'at least one answer'-solution may stall the network - if no answer is given. On the other hand, the 'time based'-solution may lead to attempts to choose a time slot which gives no answer and node **P** picks the most desired seed alone, which breaks the intention of randomization.

Once the values are given or the time slot has exceeded, the values are revealed (Figure 3.7 & table 3.1: $\bf D$) (Chatterjee et al. 2019, p. 406). After all given values are revealed (Table 3.1: $\bf X$, $\bf Y$ & $\bf Z$ in $\bf D$) or time has exceeded (Table 3.1: α in $\bf D$), $\bf P$ calculates the seed out of the given values (Table 3.1: $\bf E$). For the sake of tangibility a simple $\it sum$ is used here. Last, $\bf P$ receives a random number out of a predefined $\it static\ random\ generator$ (Here: Random of (1, 522)). During the randomization procedure, once a (hidden) seed is given, no adjustment is possible. Additionally, none of the participating nodes ($\bf P$, $\bf Q$, $\bf R$, $\bf S$ & $\bf T$) can adjust their value to that of the others as the one-way hash functions prevent altering in between (Chatterjee et al. 2019, p. 406). Finally, the randomization's process outcome is dependent on every involved node and 'non participating'-nodes cannot veto (afterwards) because they are guilty in the case of 'not participating'.

Chatterjee et al. (2019, p. 406) sees a flaw in this approach as the time slot of revealing each seed (Table 3.1: **D**) offers the possibility to alter the sum. If node **S** had calculated the sum and came to the conclusion that revealing α would lead to a less desirable random

value, **S** would gain an advantage from revealing late. This results in two possible shots for each node Chatterjee et al. (2019, p. 406) except the demanding node (this one has to be revealed anyways) and a race to become the last emitting node. This scenario only beholds true if each node reveals directly to the whole network. On the contrary any other node (e.g.: **Z**) can add a transaction containing its encrypted value. The transaction containing **Z**'s encrypted value is now part of the BC. Assuming that there is one node **P** who asked for a random number, **Z** sends the key only to **P**. Only **P** can therefore obtain **Z**'s value during the revelation time slot. After the revelation time slot is over, either **P** or **Z** reveal the key which offers the seed to the whole network. Of course this additional step slows down the answer.

Against Chatterjee et al. (2019, p. 406)'s claim that the process "[...] provides no incentives to the participants to submit random numbers [...]", incentives are seen here in the upper level game itself. Therefore same results from constantly given static values is inapplicable here as well (Chatterjee et al. 2019, p. 406).

It has to be kept in mind that random number generators need to fit the purpose and are likewise encryption, a key attack vector. Although the thoughts are backed by Chatterjee et al. (2019)'s paper, the solution is considered intuitive and supposed for short time frames to cover recalculation risks etc. It is not error prone by definition. Although further details regarding random number generators are not in the scope of this document.

3.5.2 Piles of Cards

Dealing with cards, such as on a player's hand, in a pile or currently being drawn can be difficult regarding security in a BCT context. Any type of the subsequent *card draw scenarios in BCT* cannot be found in the literature at the present time.

The easiest part is distinguishing between a secret hand, just visible to the owner and an open hand, visible to all players. Moreover decks/piles need to be shuffled or drawing has to be grounded on randomization. Consequently, drawing cards from decks introduces specific obstacles. The different cases are shown in table 3.2.

On the one hand it is distinguished here between **private** and **public** piles in the means of accessibility. On the other hand draws differ in regards of **open**, **private** and **hidden** transactions

Node\State	Start (B)	Answers (C)	Reveal (D)	Calculation (E)
P	X	_	X = 412	_
Q	_	Y	Y = 369	_
R	_	_	_	_
S	_	α	$\alpha = ???$	_
Т	_	Z	Z = 741	_
Seed	_	_	_	$\sum_{X,Y,Z} = 1,522$

Table 3.1: Sample given randomization values

Pile\Draw	Open	Private	Hidden
Private	1.	2.	3.
Public	4.	5.	6.

Table 3.2: Extended Blockchain Network Types

in the means of visibility/openness to other players.

In **open** transactions a card is shown directly to all players - no restrictions in regards of accessibility exists. In contrast, **hidden** transactions are only known to the acting player. Other players have no indication that a card has been drawn at all - still the acting player receives a card with the attached information. In between *open* and *hidden* there are **private** transactions, which show other players that a card has been taken from the pile, but the card's information is only revealed to the drawing player.

3.5.2.1 General draws from piles

The different types of draws work as follows:

- 1. A **private pile** with **public draw** is supposed to be the least obstacle. The cards in the pile are supposed to be known by everyone and only the card to be drawn has to be determined. A random number helps to choose the card to be drawn (by index). Finally, the card is shown to all players.
- 2. More complicated is a **private pile** with **private draw**. Still it is manageable as the drawn card can be proven by *offset revelation*. Here the effective pile is only known to the drawing player in regards of order (shuffled cards). Again, a random number helps to choose the card to be drawn. The draw has to be execute as a hidden transaction for later verification. Still the validity of every played card may only be verifiable at the end of the game. The latter is e.g. depending on the fact whether the cards in the pile are generally known to other players or not.
- 3. A private pile with hidden draws needs another mechanism. Already a call for randomization, the process to receive random values, implies to draw a card. Still, without this call a draw is not supposed to be possible. Contrariwise the absence of a call for randomization implies that no card can be drawn. Therefore faked draw transactions are the consequence. But even faked draw transactions have to be documented on the BC to fulfill the promised transparency and trust. This aspect will be covered in more detail in the section Data allocation improvements.
- 4. A **public pile** with **public draw** is supposed to be as easy as a *private pile* with *public draw* (1.). The only difference is that all players can access the pile to draw from it the pile is not explicitly reserved for one specific player.

- 5. A **public pile** with **private draw** is supposed to be the most complex situation as it requires the highest level of collaboration.
 - Therefore, it is moved to the following section (Public pile with private draw).
- 6. Last, a **public pile** with **hidden draw** offers a special mutually exclusive case. As this type prevents to inform other nodes that a card was drawn, not even the drawn index from a multiple times shuffled and encrypted pile is allowed to be published. Consequently, a node in a subsequent turn may draw an already taken card from the pile. Hence a race condition (Netzer R. H. B. AND MILLER B. P. 1992, p. 75) in the form of Nakamoto (2009)'s **double spending problem** occurs. To the current knowledge, this problem connot be solved without external help, such as Helper Nodes. These are discussed in further detail in the section Further Characteristics of the Proof-Of-Turn approach.

3.5.2.2 Public pile with private draw

The key issue of this type is that only the drawing player is allowed to receive the card's information. Nevertheless, the other players are not allowed to draw the same card from the pile thereafter. Hence, they need to know which card's placeholder (e.g. index) was taken. Naturally, the index is not allowed to reveal the cards information/identity. A mechanism has to be implemented, which hides the cards information, but prevents overlapping transactions (double spending) from the pile. To untangle the challenge, the single card's information and the card's destination in the pile need to be separated to provide a suitable placeholder mechanism. This can be done by shuffling and encrypting the cards. Still, the 'shuffling and encrypting'-party has full knowledge about both the previous as well as the subsequent state of the pile. Therefore several parties have to be involved.

During the process at least two rounds of 'shuffling and encryption' are needed and at least three parties have to participate. Basically, with every additional party, another round of 'shuffling and encryption' has to be conducted. The first round of encryption has to be executed using asymmetric encryption. The asymmetric encryption has to be split on two (distinct) parties, one encrypting and another offering the decryption keys. Last, a third party is needed to prevent data exposure. The process is given in more details in Figure (3.9) and the steps and their possible pitfalls are described hereafter.

First **Party A** creates asymmetric encryption key pairs for each card. *Party* **A** is not allowed to encrypt the cards. If *Party* **A** would encrypt the cards, it would become the single gatekeeper and knew every drawn card. Hence **Party B** shuffles the unencrypted pile and then encrypts each card using the (public) keys received from *Party* **A**. Without *Party* **B**'s shuffling the encryption would be useless as the *default pile* is known. The resulting pile, consisting of encrypted cards complemented with their key's hash, is therefore (only) given to *Party* **C**. At this moment *Party* **A** is not allowed to be involved, as it could decrypt all cards to receive the pile's change of order. To prevent *Party* **A** from decrypting, **Party C** shuffles to diminish *Party* **B**'s knowledge (Indexes) and encrypts each card again to prohibit *Party* **A**'s access. Now none of the parties is capable to access information from the pile without consultation of all other parties.

More precisely, after the process: **Party A** does not have access to the pile unless *Party C* grants permission and hands over a card which is only encrypted with one of *Party A*'s public keys. **Party B** knows the first pile's change of order (from origin), but does neither know *Party C*'s shuffle nor *Party A*'s private keys. Last, **Party C** neither has *Party A*'s private keys, nor *Party B*'s change of order to the source pile. Consequently, the resulting **twice** *shuffled and encrypted* pile can then be written to the BC. During the game the pile's cards are drawn numbers (Indexes). Each party knows which card has been drawn from *Party C*'s published pile as the drawn indexes are known. This solves the 'double spending' challenge.

To the knowledge of the author, there is no possibility to decrease the number of parties below three. Nevertheless, an infinite number of parties can be added between *Party C* and the publication of the pile, represented in figure 3.9 by **Party D**. Each added party reshuffles the pile and encrypts it with its own keys, just as *Party D*. Except the mandatory asymmetric encryption used by *Party A* and *Party B*, all other encryption is free to use either a symmetric or asymmetric encryption procedure. Naturally, with each additional party, the reconstruction becomes more intricate. Generally only the shuffling is needed until *Party B*'s encrypted card can be combined with *Party A*'s key to retrace the final information. But in a hostile environment, each party insists on full safety/transparency. Hence, on each step towards decryption, the drawing party insists on proof of correctness. Thus, for not getting tricked, the key on every stage is claimed and validated. If the key does not work properly (e.g. wrong data was transmitted), the sending party has to be blamed directly, as it might try to betray (the network). Additionally, the party which is asked to provide the card's information claims valid input data to ensure that particularly this card

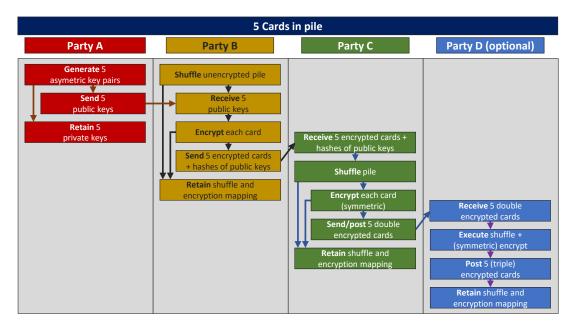


Figure 3.9: Parties shuffling cards

is chosen and has to be revealed (especially for Party A's private keys). Hence, the *encrypted information* from the previous party, which equals to one of the asked party's produced cards *plus* its *corresponding index* have to be delivered. Only if **both values** are supplied, representing a valid claim to receive information, data is provided.

In figure 3.10 the decryption processes for the *parties A*, *B* and *C* are shown. Given the case that there are more than three parties, such as another *Party D* and beyond, the adapted process is shown in figure 3.11.

Note that there are pitfalls if wrong information is transmitted or a party, which has to offer specific information, is not available. This shows a major problem of this solution and stresses why recently it was not talked about *network nodes*, but rather about **parties**. The case of unavailability of any node, for whatever reason, freezes the network. If the network freezes as described in the Multichain approach, there is no possibility to recover unless the affected node reconnects to the network. Thus a game designer has to choose between a full mistrust scenario and cooperation of nodes. In a three players game - as mentioned before the minimum amount of players - each node has to become one of the three parties. But if there are more nodes participating in the process, a party may consist of multiple nodes as well. The latter decreases trust and reduces security. Still, if multiple nodes are capable to provide needed information responsiveness of the network rises. This is especially desirable because the decryption as well as the shuffle trace back can only be performed in one specific order. A review of the BFT problem to ensure fraud resilience has to be conducted. Fraud resilience follows the graphs shown in figure 3.12³. Assuming **full mistrust** (Figure 3.12: δ), starting at a minimum of three nodes, all nodes would need to turn hostile to decrypt information. Hence, no node can trick another or the whole network. Nevertheless, it is the most vulnerable type in regards of a frozen network state. Increasing the number of nodes able to provide the information leads to the graphs ϵ and θ . Dividing all nodes into groups of two (Figure 3.12: ϵ) implies that each group needs to contain at least one hostile node to enable fraud. Hence at least 50.0% of nodes have to become hostile to break the system. In the best case only one group with fully honest players is able to keep the system fraudless. An even higher redundancy, dividing all nodes into groups of three (Figure 3.12: θ), leads to at least 33.3% fraud resilience. Raising redundancy even higher as of fixed three, four or five groups containing many nodes (Figure 3.12: α , β & γ) reduces fraud resilience dramatically. Depending on the desired resilience level, δ , ϵ and θ are recommended. Nevertheless, in particularly huge games with dropping out players a low number of groups might be superior. Here it has to be reminded of Laneve and Ershov (2019) who state that BCT "[...] is reliant on the game's popularity to keep it running, making it a risky choice for developers" (p. 27). The final decision remains implementation dependent.

Figure 3.12 plot's Python code is given in the appendix.

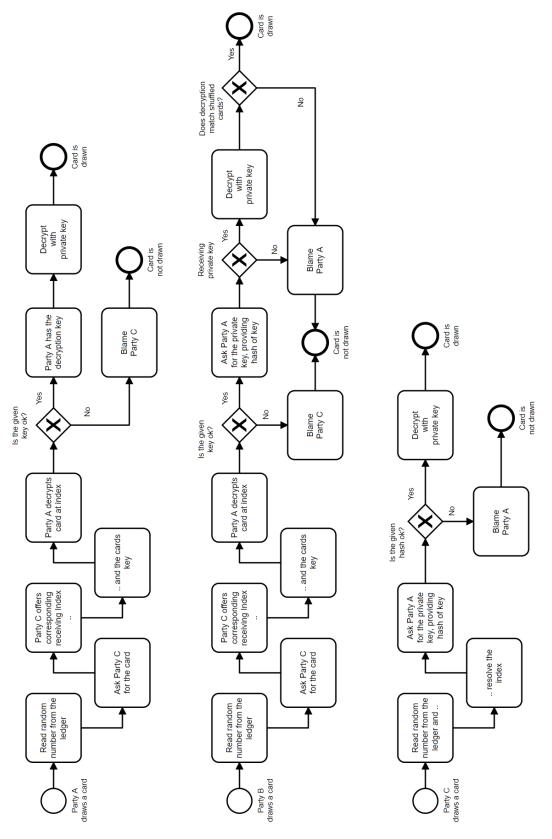


Figure 3.10: Decryption processes (Nodes A, B and C)

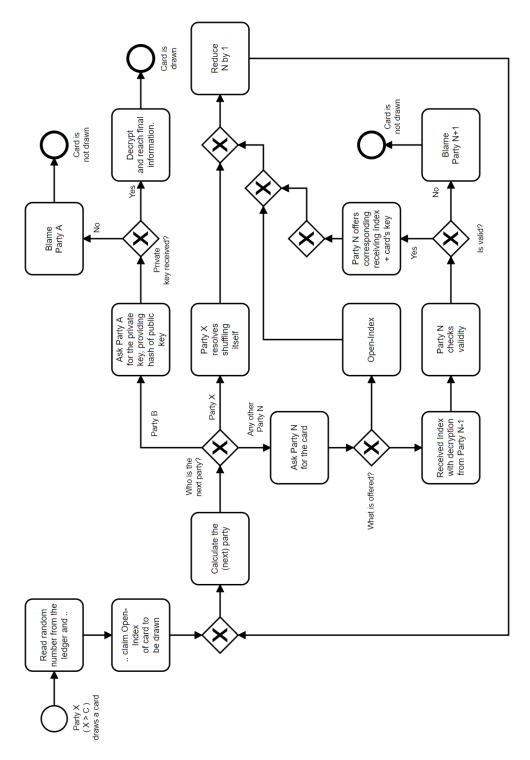


Figure 3.11: Decryption processes using many nodes ($\geq D$)

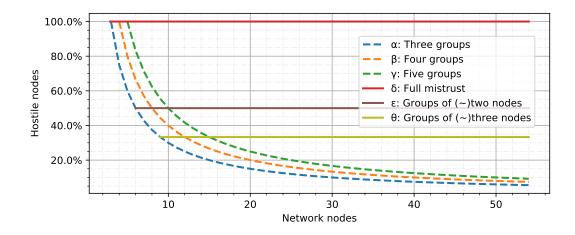


Figure 3.12: BFT shuffling cards

3.5.3 Fog of War

Comparable SCs might be needed, if a game includes *fog of war* elements, which deal with uncertainty of combat situations (Setear (1989)). Again, there was no literature to be found which covered *fog of war scenarios in BCT*.

Four general uncertainties are assumed: 'enemy's intentions', 'natural environment', behavior of 'friendly forces' and 'underlying laws of war' (Setear 1989, p. 3-4). The last case does not apply in this scenario as it is diminished by defined SCs and game rules. Nevertheless, in many (realtime) strategy games especially the first two listed uncertainties are used. In gaming environments with a central server, which is aware of every move, this does not offer special challenges as the server can always provide all needed values. Back in a distributed environment, a challenge similar to the previous card draw scenario occurs.

First, the easy case: The game is designed in a way that every possible move can only be performed on fully revealed instances (Figure 3.13, A). Hence, each move can be calculated by the executing node (Figure 3.13, B). The information for the next move will be provided during the next network round. All other nodes provide needed data (lifting the fog) for the next round on the affected fields (Figure 3.14, C: Squares and dots). On those fields, which are out of the new area of visibility, fog rises again (Figure 3.14, D).

Second, the tricky case: Just as games offer the possibility to inspect another player's hand of cards, in figure 3.15 (E), all pawns in the game can only see the (directly) adjected tiles - regardless their movement possibilities. Hence, performing the same move as seen in figure 3.13 to figure 3.14, the player with black pawns does not know whether the queen will hit any enemy's pawn (Figure 3.15, E to F). The information has to be provided by the other players (see Reveal Claims below). Consultation is needed to complete the move. Implications:

1. All this is generally possible using *hidden transactions*.

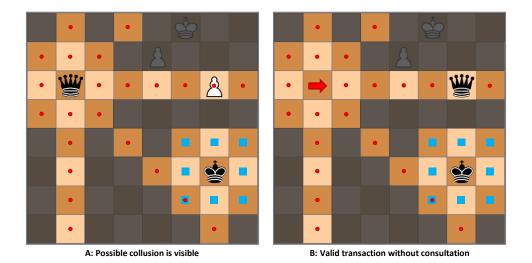


Figure 3.13: Fog of war in chess - Open moves (1)

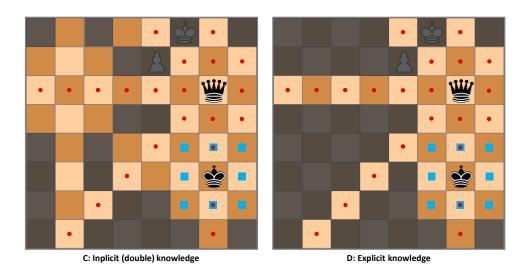
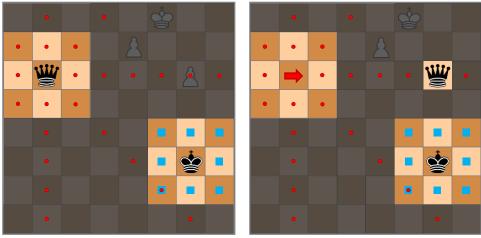


Figure 3.14: Fog of war in chess - Open moves (2)



E: Possible collusion is not visible

F: Transaction needs consultation

Figure 3.15: Fog of war in chess - Fogged moves (1)

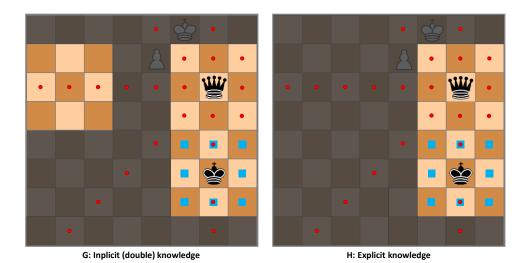


Figure 3.16: Fog of war in chess - Fogged moves (2)

- 2. Due to the rules of chess **implicit knowledge** is granted see figure 3.14, C as well as figure 3.16, G which is not applicable to every game.
- 3. The implication, that the player in black color has to offer all other players the (new) location of his queen, makes the whole 'fog of war'-situation **pointless** at least for chess.

Consequently, the underlying mechanisms two (*implicit knowledge*) and three (*pointlessness*) need to be considered throughout during a game's development process.

Point two and three implicate that fog of war is generally not possible in distributed environments conducting BCT, but that is not completely true. Chess has many constraints, such as only one move per turn can be performed, which diminishes the reappearance of fog. Additionally, chess offers binary type information, such as the true/false answer to the question: 'Is the black queen located on field B4?'.

Third, the case in between: Other games may offer to hide information better, especially if information is given in a numeric type and no *exact amount* has to be delivered. Less cryptically, assume a SC exists in RftS, which says:

- A) After his turn a player has to publish the number of ships (X) stored on each of his planets.
- **B)** The real number (**X**) does not need to be published directly it can be included in a hidden transaction.
- **C)** The publicly accessible number has to be within the range of +-**X**%.
- **D)** The deviation has to be calculated from a randomization process. The value extracted from the randomization may stay hidden until it has to be revealed.

Concluding, setting the number of ships to 100 *constantly* and **X** to 30, the published values vary between 70 and 130 ships. This could be recalculated long term using the mean. Still, with extending variables such as the *production of ships* (influenced by randomization) and changing amounts due to *incoming/outgoing fleets* etc., this SC, based on numeric values, fits the guidelines of *fog of war* (Setear 1989, p. 3-4). Finally, the real number of ships can be fogged in distributed environments like BCT.

3.5.4 Reveal claims, trigger events & Disputes

Reveal claims and trigger events - Some games offer the possibility to inspect another player's hand of cards, lift the fog on specific values of the game or in case of RftS claim a planet's (real) number of stored ships. Living in the assumption of multiple players (≥ 3) the information has to be provided for one player (only), a subgroup of players or to all players. In the first two cases, the information is only encrypted to the *legitimately claiming* node(s). Still, the information transfer needs to be backed by a SC. Moreover, within a game spontaneous transactions/triggers could exist. These may either be 'throw in transactions' or previously placed SC which are just waiting to be triggered. Anyway, these transactions follow sequential order and can therefore be conducted using BCT. Moreover this has certain implications on the CM and will therefore be covered in chapter Proof-Of-Turn.

Disputes - Although SCs and the chosen BCT CM shall prevent disputes, sometimes they are inevitable. Hence, a general pattern to resolve disputes is given: **First**, the dispute may be solved

within the network. This could be conducted by a vote - the majority (>50%) wins. This is probably a good solution for disputes among a little group within the whole game network, as stakes in the game may influence the vote in an unfair manner. **Second**, if possible, hand the dispute to any upper level instance as shown by KRAFT (2016, p. 92). **Third**, the case which shall be prevented if possible: Stop the game(-channel) and prevent an unjustified win for any involved party. The possibilities a game offers regarding the three cases determine the options after a dispute:

- 1. Waiting / freezing the game until an absent player/node is answering. During this time the unavailable player may be kicked by vote (see section Peer-Fluctuation & adaptive turn time in chapter Proof-of-Turn,).
- 2. Impose any additional upper level penalty against the node e.g. by the publisher (Chatterjee et al. 2019, p. 407).
- 3. Stop the game generally. This is only recommended if players can be measured by points already.
- 4. Conduct any kind of recover strategy, which is implementation dependent. This might not be possible e.g. if cards had been dealt with the previous presented shuffling mechanism, as both the associated reverse shuffle order and encryption keys are lost.

3.6 Data allocation improvements

In the literature *data allocation improvements* could only be found before data was written to the BC - e.g. game hash vs. encrypted data (KRAFT 2016, p. 94) but not altering the BC thereafter. The general idea of removing data from a BC after blocks became CF harms the characteristic of *immutability of data* (BUTIJN ET AL. (2020, p. 17-18); DIB ET AL. (2018, p. 57); SHARMA ET AL. (2020, p. 21)). Nevertheless, to meet the demand of reduced data consumption (See section Gaming market), ways to reduce data allocation whilst keeping a shared consistency need to be evaluated. If there was no central server from the publisher, not only the information in a transaction can be withhold (*hidden transactions*). Already the knowledge about a transaction can be a non-desirable factor. To make this more tangible, in the aforementioned game, RftS, a player makes a move, sending a fleet to another planet. This fleet is stored as an encrypted block. Any opponent can now assume, that a potential fleet is on its way and is consequently able to calculate round by round whether the fleet may reach the one or the other controlled planet. As a player has to proof his turn later on with *offset revealing*, publishing the move (somehow) to the blockchain is unavoidable.

Nevertheless, moves can be hidden in noise data using dummy data, here referred as a BT. No matter if a player takes a move or not, an additional (random) number of BTs is added. BTs can be be recognized as such right after their revelation. All BTs need to be revealed to claim the win (even if its the second place). The enforced revelation prevents **double spending** of a planet's ships into several fleets. More details on *double spending* in the section Attack Vectors. Additionally, game mechanics define the number of rounds until BTs have to be revealed (e.g. 10

rounds after the last non-BTs is revealed). Consequently, the number of transactions can not be calculated from the number of published blocks during a turn. Therefore the number of moves can be hidden using BTs. Nevertheless, there is a **downside of BTs** as well, **data storage**. In long lasting games with many moves and players, the number of BTs will probably derive to a problem.

The following, storage size calculation is highly assumptive and strongly dependent on the application's use case. Nevertheless, general traits of different approaches shall be shown. Therefore, presumptive values are given:

If most BTs consumed ~ 0.01 MB and in contrast most **relevant transactions** ~ 0.1 MB, BTs could be distinguished from *relevant transactions* easily by size. Hence, *relevant transactions* as well as BTs need to allocate a comparable amount of storage space. This increases data allocation by the factor 'BTs per *relevant transaction*'. Alternatively, game hashes could be used decreasing each transaction down to (~ 0.032 KB). This is always recommendable if the average transaction's size exceeds the *game hashes* fixed amount.

During the following equations a $\mathrm{BT}(l)$ is a large blob of bloat data augmenting a *hidden transactions* whilst a $\mathrm{BT}(s)$ is only augmenting a game hash. If encryption is used, the data allocation can be calculated with *hidden transactions* (h), and the 'factor of BTs per hidden transaction' $(f(\mathrm{BT}(l)))$. Additionally, revelations (r) of the encryption key (k) to different nodes and the network have to be taken into account.

Encryption:
$$h + f(BT(l)) + r * k$$

On the contrary, game hashes (H(h)) can be used for hidden transactions.

Game hashes (i):
$$H(h) + f(BT(s)) + r * k$$

If a *hidden transactions* (game hash) has to be revealed to any other node r the whole network, it has to be written to the BC.

Game hashes (ii):
$$H(h) + f(BT(s)) + r * k + h$$

Now, the two versions are compared:

$$h + f(BT(l)) + r * k$$
 vs. $H(h) + f(BT(s)) + r * k + h$

For the sake of simplicity, 'h' as well as 'r*k' can be shorten. Given the application's use case dependent values of $\mathrm{BT}(l)$ and $\mathrm{BT}(s)$, the break even can now be calculated by equating the two formulas and solving for f.

$$f(BT(l)) = H(h) + f(BT(s))$$

This will show the use case's best suited solution. Presumably, *game hashes* are superior as they promise strong abstraction - especially of big files.

The following plot figures omit the number of published blocks by turn and/or round. Instead, the types of transactions are relevant. It is distinguished between *relevant transactions* and BT. *Hidden*

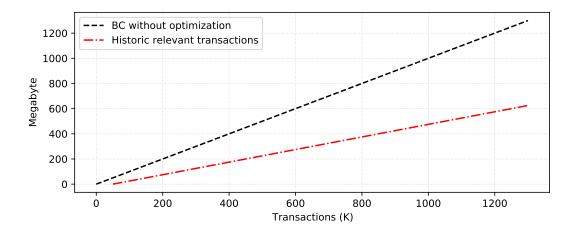


Figure 3.17: General BT storage allocation chart

transactions are a subcategory of relevant transactions. Hence, relevant transactions can have the status **hidden**, (recently) revealed and **historic**. The last category does not change the game state anymore and could be deleted (depending on the use case). The plots assume either a long term or heavily written data due to BTs. A worst case scenario is assumed as a 50:50 ratio between relevant transactions and BTs (no matter if game hashes are used or not) are shown.

First of course each BC starts with a genesis block at 'zero'. The displaced start of the two graphs (Figure 3.17⁴) reflects the *offset revelation*, as unrevealed transactions can not be distinguished upon being *historically relevant* or *deletable*. Accordingly to the predefined *ratio*, the upper $\sim 50\%$ of the BC's size (Figure 3.17, 'BC without optimization') accounts for BTs (Figure 3.17, Area between the lines). Nevertheless, only the 'Historic relevant Transactions' are needed long term (Figure 3.17, Lower line).

Estimating an average block to be of ~0.001 MB and an active BC to consist of $\sim100,000$ transactions, it would consume ~100 MB of storage space (Table 3.3, Row 1). Consequently, 600,000 transactions *0.001 MB =600 MB (see Figure 3.17). Considering that there is not only one game being played simultaneously, the needed storage space raises linearly (Table 3.3, Row 2). Although a single block consumes little storage space, massively generated BT raise the allocated storage space tremendously. Importantly, each entire BC is supposed to be stored on all network nodes. Generally, BCT is built upon networks with well equipped nodes regarding hardware. In contrast, (casual) gamers might want to join using devices which do not offer much storage space, such as smartphones or handheld consoles. Here a storage size consuming BCs could be limited by (other) software restrictions. In the worst case it may lead to a removal of the application/game by the gamer or even by any higher authority, such as popular app stores 5 .

Consequently, a mechanism to reduce long termed storage allocation is needed. Still, the thought

Figure 3.17 plot's Python code is given in the appendix.

⁵ E.g.: Apple 'App Store', Google 'Play Store', Epic Games Store and the Steam Store

	1 Transaction	100,000 Transactions
1 BC game	0.001 MB	100 MB
10 BC games	0.01 MB	1 GB

Table 3.3: Blockchain Size

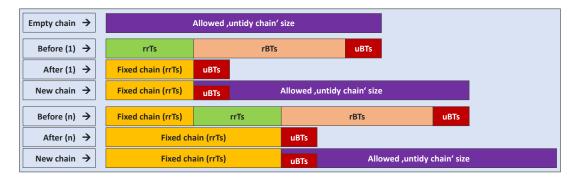


Figure 3.18: Prune bloat transactions

of orderly deleting data within BCT is far from the usual approach, such that BCT is thought to prevent removal of any content in general. Nevertheless, some approaches exists which make "[...] it possible to re-write or compress the content of any number of blocks in decentralized services [...]" (Ateniese et al. 2017, p. 111). These approaches are not considered in the scope of this thesis as they (only) *compress* but do not *delete* content. Additionally, most BCs prohibit deleting single transactions passively, as they are encapsulated in one block with other still needed transactions. But in this special case of utilizing BTs, the following three mechanisms could be identified to reduce allocated storage space: Prune bloat transactions, Child-chain transition (alike Kraft (2016)'s game channels) and Meta-State Blocks.

3.6.1 Prune bloat transactions

Once a critical chain size is met, the chain could be recalculated using a **prune procedure**. As *bloat transactions* could not be found in the literature, the deletion of these transactions is missing as well.

The chain before the recalculation consisting of relevant (revealed) Transactions (rrTs) (Figure 3.18), **revealed BTs** (Figure 3.18, rBTs), *unrevealed BTs* (Figure 3.18, uBTs) as well as other negligible meta transactions is called the **untidy chain** (Figure 3.18, Untidy chain).

The BC's network wants to get rid of meaningless allocated storage space and decides by vote to 'garbage collect' *revealed BTs*. One node has to conduct the *prune procedure* - probably the node which broke a given *untidy chain* size limit by adding transactions. The node calculating the prune is called the Garbage Collecting Node (GCN).

As rrTs, revealed BTs and unrevealed BTs are not separated or sorted within the untidy chain, the GCN generates a new chain from scratch except the **genesis block**. During the recompilation,

the *untidy chain* is parsed by the GCN and all *revealed BTs* are deleted (Figure 3.18, Before(1) \rightarrow After(1)). This is only recommended in lightweight CMs, as the GCN needs to find suitable hashes for new blocks easily. Once the process is finished, another 'consensus vote' approves the new chain.

It has to be reminded that every network node exhibits the following behavior:

- 1. Network nodes do not trust each other. The new chain, which was generated by the GCN is checked/recalculated by every other network node.
- 2. All network nodes have a desire to reduce their allocated storage space and look forward to a reduced chain as long as the new chain complies with **one**.

Consequently, if the GCN fails to generate a valid new chain, the network tries to calculate a new chain delivered by another GCN. in this case, the first GCN does not only have to calculate the first version, it also has to verify the second version. Therefore every GCN wants to avoid this (overhead workload) loose-loose situation which leads to honest recalculation and trust within the network.

The rrTs are transitioned into a *fixed chain* which will not be changed anymore (Figure 3.18, After (1): Fixed chain). Transactions in the *fixed chain* are unencrypted and do not store any additional bloat/coverage data. Subsequent calculated *prune procedures* will only focus on the 'new' *untidy chain* (Figure 3.18, New chain: untidy chain). Additionally, the *untidy chain* size starts after the last rrTs and directly contains all transactions which are not yet revealed.

All rrTs which are transitioned into the fixed chain during this *prune procedure* may be written into a single block and signed by the GCN, the transcribed encapsulated transactions are still signed by their emitting nodes. Hence, the integrity of the chain is sustained. Depending on the share of garbage data (BTs/relevant transaction), the chain can be boiled down. At the end of the *prune procedure*, a marker is set which defines the start of the *untidy chain*. The next *prune procedure* will start from that block (Figure 3.18, Before(n)).

Although the permanent growth of the BC along 'Historic relevant Blocks' (Figure 3.19⁶, lower line) can not be prevented using prune procedures (Figure 3.19, Serrated line), the overall storage allocation can be dropped significantly.

If not triggered regularly, the prune approach is a computation heavy bulk operation. It is computation heavy in the means of two factors: **First**, generating the new chain as well as checking the integrity and **second**, in regards of network traffic to pass the whole new chain. During a prune is calculated the game may be paused as described in section Adaptive turn time of the Proof-Of-Turn. Last, finding an optimal *prune procedure* frequency (allocation size) is not part of this document. For a more continuous approach to reduce allocated storage space, **Child-chain transitions** may be used.

⁶ Figure 3.19 plot's Python code is given in the appendix.

3.6.2 Child-chain transition

The idea of **child-chain transition** derives from KRAFT (2016)'s *game channels*. Still, although *game channels* conduct off-chain transactions using sidechains, their intention is to speed up the main chain's throughput. To the writers knowledge these *game channels* are kept long term by the consulting parties. Therefore, *game channels* are not used for storage allocation improvements. There was no further literature found regarding **child-chain transitions** for storage allocation improvements.

A downside of *prune procedures* is the aforementioned peak of computation and network traffic on one single node. A continuous **child-chain transition** from *child-chains* (see Performance Improvements) to the *main-chain* may be a reasonable answer to this challenge. Here, a game uses an undefined number of *child-chains* (Figure 3.20), each limited to e.g. 1 MB ($\sim 1000 \text{ transactions}$, following Table 3.3).

Whilst the **main-chain** stores all rrTs, all unrevealed transactions are stored in *child-chains*. Once the storage size of a *child-chain* is exceeded, all newly published blocks will be stored on the next *child-chain* (Figure 3.20, $n \rightarrow n+1$). The newest *child-chains* may therefore consist of *unrevealed BTs* only (Figure 3.20, n+1). Still the filled *child-chains* are temporarily kept (Figure 3.20, n+1) and n).

By the time a transaction is revealed, the revelation key will be put on the dedicated *child-chain* - next to the unrevealed transaction - and the unencrypted residue is stored on the *main-chain*. Hence the *main-chain* is unencrypted throughout.

By the time all transactions on a *child-chain* are revealed or obsolete, the *child-chain* can be deleted (Figure 3.18, 0). The deletion is conducted by each node individually. Hence a node - if desired - could keep the data as well. Concluding, bloat/coverage data, which has initially been added to rrTs becomes obsolete and the key emitting (/revealing) node itself can sign the blocks on the *main-chain* as well. In this scenario *child-chains* with small storage sizes offer faster deletion as it

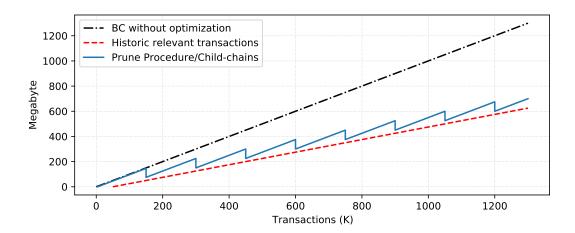


Figure 3.19: Prune procedure/Child-chain chart

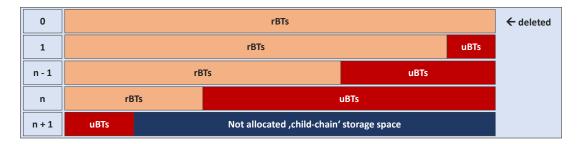


Figure 3.20: Child-chain storage allocation

has to be waited until the last transaction is revealed. Nevertheless, although small storage sizes lead to smooth overall storage allocation, more filled *child-chains* have to be stored until they can be deleted.

The overall storage size equals to the one from *prune procedures* (Figure 3.19, Serrated line). In this regards, frequent *prune procedures* and small *child-chains* equal each other as seldom *prune procedures* and big *child-chains* do. Whilst *child-chains* offer an comparably equal distribution of computation and network traffic, *prune procedures* could be designed to be conducted on nodes offering strong computation power only.

Still, for *child-chains* a lightweight CM is needed throughout. Finding an optimal *child-chain* size is not part of this document.

3.6.3 Meta-State Blocks

Both, the *prune procedure*- as well as the *child-chain*-approach offer a full (game) history. Nevertheless, if storage space is limited, the network could decide to define any fully revealed state before the first *unrevealed* (*bloat*) *transaction* - to be the new **genesis** (*block*). Again there was no literature found which changes the genesis block to rewrite a BC's history for data allocation improvements.

A new **meta-state block**, at the beginning of the chain, represents the game's state at the *genesis* (Figure 3.21, *After*: Meta-State Block). The *meta-state block* summarizes all until then emitted transactions, defining a 'new default start'. Old obsolete transactions are dropped, obfuscating

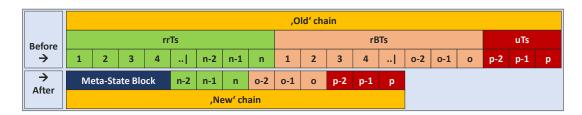


Figure 3.21: Meta-State Block process step

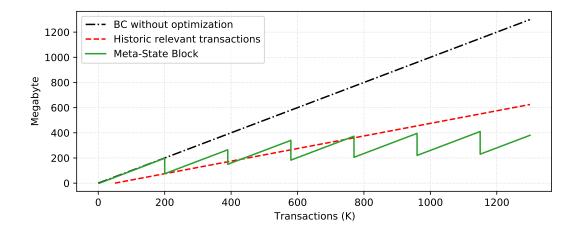


Figure 3.22: Meta-State Block chart

'historical relevant' data (Figure 3.22⁷, Meta-State Block). Except historically relevant values, the data in the *meta-state block* does not differ from the network's knowledge 'at the starting point' (Figure 3.21, *Before*: '...|' \rightarrow *After*: Meta-State Block). Therefore, it is encouraged not to use the last unrevealed (bloat) transaction, but to leave some historic data in between (Figure 3.21, *After*: 'n-2' \rightarrow 'n', 'o-2' \rightarrow 'o' and 'p-2' \rightarrow 'p').

Although it is discouraged from, the equivalent in a cryptocurrency context would be a *meta-state block* at the beginning of (e.g.) the last elapsed year - the wallets amounts remain the same, but transactions before the *meta-state block* can not be retrieved anymore. Still, each node is free to either *keep* or *delete* the until then generated (old) chain. Nevertheless, here it is assumed that the described nodes drop the old chain and receive released allocated storage space in exchange (Figure 3.22, Meta-State Block).

Consequently, in contrast to *prune procedures* and *child-chains*, all previous *meta-state blocks* containing movement information, randomization agreements and other transactions are deleted. The *meta-state block* is always the genesis. Agreeing to the trade-off, the network looses its history before the genesis round, but retrieves storage space (Figure 3.22, Meta-State Block). The baseline after the historic data was deleted was set to sample $\sim 200 \mathrm{MB}$ long term (Figure 3.22).

In figure 3.22 it is assumed, that the application allocates an equal amount of storage in the long run, e.g. $\geq 1.000k$ transactions. Until this level is reached, the *meta-state block*-approach is (here) not especially advantageous. Given a (smaller) larger transactions size, this metric changes. This method is well suited for applications, wherein history is considered obsolete in general. Still all SCs need to remain consistent for the upper level software. Although the *meta-state block*-approach is mentioned independently, it can be combined with both, the *prune procedure*-as well as the *child-chain*-approach.

Figure 3.22 plot's Python code is given in the appendix.

Finally, figure 3.23^8 shows the procedures together in one plot in larger scale (2.000k transactions). All procedures combine the characteristic that cheating is not possible, because only those blocks which are already verified by the network are deleted. Of course, systems which contain 'above-average storage space' are able to keep historic data (longer). Nevertheless, superior systems do not have any advantage regarding the quality of the stored information or benefits regarding upper level game play.

Figure 3.23 plot's Python code is given in the appendix.

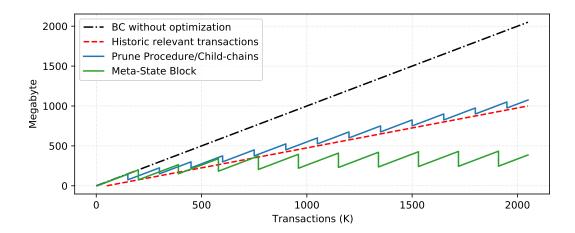


Figure 3.23: Storage allocation comparison

3.7 Interim Summary - Blockchain Technology in Games

Before presenting the PoT CM, a short summary of the so far identified results is given. This chapter shows that BCT is suitable for certain games (e.g. *gambling* and *collectibles*) and is in some regards already used to enable new types of games (here: *collectibles*). Still, publishers do not yet use BCT to reduce network costs but rather try to leverage cryptocurrencies and assets backed by NFTs to boost their revenue. Further, these applications proof that recent CMs are already able to host slow games generally.

Additionally, a problem space was defined to identify general, primarily board game related, SCs. Thereafter, Hidden transactions & Randomization, Piles of Cards, Fog of War, as well as Reveal claims, trigger events & Disputes were analyzed. Last, possible Data allocation improvements, new to BCT, were presented.

All these SCs were given to answer the guiding question:

"Which kind of SCs need to be established to cover typical in-game mechanics?"

In this regards, one key factor has to be mentioned again: A game can only be won by offering a full disclosure of all hidden content. Without this final revelation it has to be assumed that fraud is being covered and consequently the affected node is not allowed to win.

Nevertheless, the section covering the Gaming market emphasizes the importance of well performing games on mobile devices such as smartphones. Looking at the recent storage capacity of smartphones, BCT games developed for mobile devices probably demand the just described Data allocation improvements to reach reasonable customer bases. But the aforementioned CMs do not offer the mandatory characteristic to grant lightweight writing permission freely to specific nodes, whilst keeping an adjustable BFT - especially in the *proof-of* category.

Therefore, a CM offering these characteristics to enable games on mobile operating systems, such as Google's Android and Apple's iOS is seen crucial for BCT to evolve further applications and use cases. Additionally it would prevent the need for *nerdy peer* setting up dedicated servers and circumvent the *shadow server* problem. Last, the comparably easy option of providing updates to the gamer's devices via the app stores reduces the publishers constraints to gamers not willing to update their dedicated servers to recent changes.

Consequently, next the Proof-of-Turn algorithm is presented.

4

Chapter

Proof-of-Turn

In most of the mentioned CMs extrinsic incentives, like tokens of cryptocurrencies (Butijn et al. (2020, p. 4); Khan et al. (2020, p. 3) Kraft (2016, p. 85); Nakamoto (2009, p. 4)), are given. The PoT approach assumes intrinsic motivation instead, in this regard comparable to CMs based on voting. Still, the upper level software is assumed to offer enough incentives. Nevertheless, mechanisms need to be established, which prevent nodes to behave hostile or malicious.

PoT is guided by the business case of bringing *turn based games* onto BC infrastructure. On a high level of abstraction, PoT reminds of the round robin nature of 'Token-Ring Local-Area Networks' (Bux (1989); Ergen et al. (2004)). Although sticking to the phrase '*turn*', this chapter abstracts from the gaming context. Still, the *turn*, as emphasized in its name, is the key factor of PoT and therefore described first (Figure 4.1). In the most basic approach, each node awaits its turn, which is a predefined time slot in the round robin procedure (Figure 4.1, (a): **Green turn**). Derived from the gaming context, on the one hand, a turn may last only ten seconds. On the other hand it is assumed that a turn will commonly last several hours or days. By the time a node has its turn, it is called the Leading Node (LN).

Moreover, the turns are assumed to enable **transition times** between the turns (Figure 4.1, (b): **Yellow T**). The transition times are supposed to be set to a constant value of *five* seconds, no matter how long the turn. It is essential for transition times to last long enough to prevent network related race conditions (Netzer R. H. B. and Miller B. P. 1992, p. 75). Consequently the last LN and its successor do not assume to be the LN at the same time, preventing mutually exclusive forks.

In a more advanced case, the LN and its successor both sign an arranged handover-block to

Proof-of-Turn – The turn					
(a)		Turn			
(b)	Т	Turn			
(c) H	Т	Turn			
(d) H	Т	Turn V H			
(e) H	Т	Turn V F H			
Time ——		→			

Figure 4.1: The turn

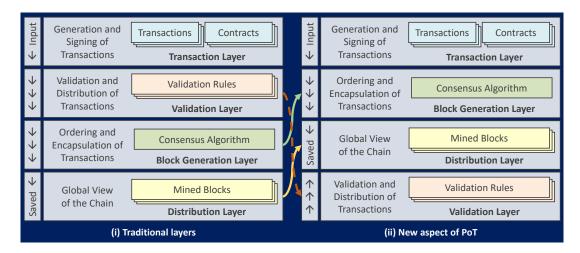


Figure 4.2: Layers in PoT (Adapted from OLIVEIRA ET AL. (2019))

seal the transition (Figure 4.1, (c): **Orange H**). This raises the security for the LN's successor to write to the latest branch(/fork) whilst ensuring the CF for the LN's written data.

Additionally, once finished writing all data, the LN may undertake a **network vote** (Figure 4.1, (d): **Gray V**). The vote asks upon the compliance of all in this turn published blocks. Each node which received the vote call, raises the possibility for CF as the possibility for a fork, leaving out the LN's data, is diminished. Additionally, each positive reply increases the possibility for posted data to reach *effective CF* (Section: Consensus finality and Byzantine Fault Tolerance) even before the LN's writing permission has ended. If some data does not pass the vote, the LN has the opportunity to revise the posted data until the turn's time slot is over. A revision is supposed to only append blocks, setting things right. Forks are prevented.

Once the LN considers its turn to be finished, a **finalizing block** may be created, which terminates the turn even before the designated time slot has passed (Figure 4.1, (e): **Red F**). Consequently, a *finalizing block* is the 'soft version' of a 'hard and explicit' handover enforced by the end of the turn's time slot.

During its turn, the LN is free to post all desired data. In the first place the amount of data is only limited by the turn time and the LN's physical computation capacity. Therefore, the turn's time slot has to be designed appropriately to ensure that all data can be written. The latter is dependent on the upper level use case. The bare minimum for data/blocks to be accepted is complying with BC's base structure (Figure 2.1) combined with the LN's (valid) signature.

At this point, the four layers of BCT (Figure 2.2) are used in PoT as well. Depending on the attitude, on the one hand, one can argue that the validation is only conducted by the LN and then directly pushed into the distribution layer. The argument for this theory is the immediate CF of each signed block. The block is then part of the chain and immutable. Following the definition, the CM ends with the distribution (Figure 4.2: (i)). On the other hand, PoT requires the data to be investigated after being distributed. The data can not be deleted, but may be invalidated thereafter. Consequently the validation layer is set last (Figure 4.2: (ii)). It is assumed that given a proper

rule set, the invalidation will never occur. Nevertheless, the nature of *hidden transactions* may offer constraints to any previously published data (e.g. a node is offline during its turn (benign fault) and cannot reveal critical data) and therefore raise the possibility of invalidation.

Regarding performance, nodes in other CMs as PoW have to try multiple times to mine a new block. Comparably, proposing a new block in PoT is cheap, as the LN's fingerprint, derived from private-public key encryption, on a block is sufficient for proposing the next block into the BC. Hence the runtime of each block's creation in PoT is assumed to be O(1).

As PoT does not tie the LNs block creation on mining success, on-chain improvements like 'Big block' are obsolete. In the first place, it is not distinguished between a **single block** containing 'all in the turn created data' from proposing **multiple blocks**. Here, upper level rules may enforce to propose multiple blocks. The latter, stems from randomization calls, which have to be finalized in the network, before the LN is allowed to continue its turn.

PoT is fully asynchronous. Therefore, in the most basic approach, PoT does not allow other nodes to interfere a turn anyhow. There is no repository for other node's transactions which have to be put into a block by the LN. Nevertheless, to enable randomization, disputes and likewise transactions, which demand consultation, either a fast single purpose round (Section: Reveal claims & trigger events) has to be conducted or any side-chain solution (Section: Interoperability) may be used.

Of course, any time each node may send a direct message to the LN to raise a dispute. The LN is supposed to follow *legit disputes* to prevent double work, invalidation, or (worst case) being sentenced with a handicap or any harder punishment by the whole network based on upper level rules. Still, the LN may decide to follow the validated (but not yet voted upon) dispute or ignore it. The latter may lead to an invalidation of the affected data and a possible reset of multiple turns. As the LN has an intrinsic incentive for progress, it will most likely give in to a dispute call and raise a network vote upon the issue.

The PoT concept is resembling to multiple, already existing approaches:

- 1. **PoA** assumes only a few nodes of the network to operate as writing/publishing nodes. The two clients *Aura* and *Clique* from the general PoA are pretty similar to PoT as they both elect one primary writing node, which is changing after a defined time slot (DE ANGELIS ET AL. 2018, p. 2). Whilst *Clique* allows multiple nodes to write simultaneously granting priority for the blocks written by the LN, *Aura* and PoT assume exactly one node to be allowed to write/publish data. But Aura votes for every emitted block directly, which leads to instant CF, but slows down the system itself. Voting upon newly published blocks is optional in PoT and generally tied to finalizing a turn. Additionally, PoT's LN is generally not assumed to write data originating from other nodes.
- 2. **PoP** and PoT are both primarily located in the gaming context.
- 3. **Multichain** uses a loose round-robin procedure, whilst PoT enforces a strict round-robin sequence. *Multichain* prevents nodes to write until a defined percentage of other nodes has written any block (Greenspan 2015, p. 7-8). If all the nodes allowed to propose a block remain silent, *Multichain* freezes. The possibility freeze is not applicable in PoT due to strict

(maximum) time slots for each node's turn. Hence, the network has to wait until the next LN's successor becomes the LN, but the network does not stall completely.

- 4. **PoET** uses exact timing to determine the next writing/publishing node. Although PoT uses transition times to prevent the need for exact timing hardware, timing plays an important role passing the right for writing/publishing data to the BC.
- 5. **Vote based consensus** decides on each block or writing node separately. The common ground with PoT arises in special occasions e.g. disputes which leaves all nodes to vote to either accept or invalidate a previous block or transaction.

Due to its stiff progression of LNs, PoT is considered to be used in *private* or *public* environments, operated in permissioned or hybrid manner (Table: 2.1). If nodes joined and left the network in high fluctuation (e.g. in a permissionless environment), PoT would stall from absent nodes and may break its intended performance characteristics.

In the following, mechanisms and characteristics of PoT are described:

- 1. CF and BFT are discussed.
- 2. General peering of nodes is addressed.
- 3. The classification into the CAP Theorem is given.
- 4. Passing turns and handling of possible forks are covered using transition blocks.
- 5. Interoperability with other BC CMs is described.
- 6. Measures regarding peer-fluctuation are presented.
- 7. Implications of trigger events are shown.
- 8. Further characteristics are given.
- 9. Possible attack vectors are described.
- 10. Limitations of PoT are addressed.

4.1 Consensus finality & Byzantine Fault Tolerance

Each LN is supposed to validate its blocks following the given SCs. As stated before the blocks, written to the BC by the LN, meet CF instantly - no consultation is needed. The price to be paid for this fast-forward solution, is the possibility of a block to be invalidated thereafter. Here it has to be differentiated between *legal enhancements* (legally **added** to the BC) and *legal transactions* (legal **content** of the transactions). In PoT *legal enhancements* of the blockchain structure (appending any block) meet CF instantly. The capsuled *transactions*, may be invalidated sometime later. The latter does not happen in the most BCT algorithms like PoW, as each transaction has to be open to

everyone in the network to pass the validation. Unfortunately, the nature of game play sometimes prevents the direct detection of fraudulent (hidden) transactions. In a worst case scenario the node itself - due to other hidden transactions - does not even know that its own transaction is invalid. The latter has to be prevented by upper level rules. Once a fraud **can be detected** by another node, a 'non action' being the LN, equals a consent. Hence, after data has passed $\geq 50\%$ of the nodes without intervention - no matter obviously fraudulent or not - the data can be seen as *effectively consensus final*. The $\geq 50\%$ refer to active nodes. After all, this drawback - CF vs. effective CF - has to be accepted. At the latest, when the writing node becomes the LN again, it can be sure the published and revealed data is accepted without dispute. Consequently, the scenario of a *computational super power* recalculating the chain - as possible in PoW (Nakamoto 2009, p. 8) - is not applicable in PoT. PoT does not follow the Longest Chain Rule.

Additionally, voting to kick nodes out of the network which behave *obviously hostile* is possible. For both, voting for 'invalidation of any block' and 'kicking due to hostile behavior', a vote process has to be conducted. The threshold, used in such a vote, is supposed to be set to $\geq 50\%$, but can be shifted as needed. It is important that the following is known: **this threshold marks the BFT for the whole algorithm**. Consequently, the BFT of PoT remains implementation dependent as stated in table 2.5.

Game play related BFT, as dependent on card draw mechanics, are explicitly not part of the PoT-specific BFT performance. Last, to prevent any node flooding the BC, it has to be considered whether a limit of written blocks/data is set for each turn. Generally a limit is not recommended, as first bloat blocks may exceed the limits and second disputes votes and following inflicted handicap upon hostile nodes can easily be conducted in a retrospection.

4.2 Peering

Data needs to be distributed through procedures and processes in the network. Although the following section is about networks and peering, specific details of the TCP/IP- as well as the OSI-model (FOROUZAN (2003)) are not discussed.

As the CM manages new transactions decentralized, passing them trough the layers (Figure 4.2), the whole network and its state is not (always) updated on every attached node until a block can be considered CF. Those transactions are previously, encapsulated in mined blocks (Butijn et al. (2020); Khan et al. (2020); Nakamoto (2009)), pushed towards the ledger from each node within the network. In the most CMs each node can always push a block into the network legally (Khan et al. (2020)). Hence, the (active) ledger is represented by many nodes simultaneously. On the contrary, a pull(only)-mechanism to fetch updates would generate an enormous amount of network traffic as every node has to pull from every other node constantly. Given the number of nodes, n, the number of pull requests for updates would scale by $f(n) = n^2$. In a binary world, using a push-mechanism for new data is evidently the right choice as the next block's origin node can be any network node. Nevertheless, popular clients as Ethereum, Hyperledger Fabric and Corda mix both mechanisms (Daniel and Guida 2019, p. 6). Still, literally, a node has to push hard to fight its data into the ledger.

In contrast, in PoT, the (active) ledger is only represented by one node. Only the LN is allowed to push new transactions. Thus, push- and pull-mechanisms are generally possible (as well) and described in detail. Nevertheless, before digging deeper, it has to be distinguished between two scenarios, **consultationless** and **consultationfull**. On the one hand, a **consultationless** network would be a shared storage wherein data is pushed only for others to be seen. No node has a desire to change other node's data. There is no reason for disputes or spontaneous interactions - such as triggers or throw in transactions. On the other hand, a **consultationfull** network is assumed to utilize randomization calls, votes, disputes as well as spontaneous interactions multiple times during a full (round robin) network round.

Moreover, the nodes are supposed to be loosely coupled, as players of games may quit (the game) or shut down their client. Additionally, high mobility of nodes may hinder static network addresses and connectivity. Hence, a broadcast in dedicated (static) sub networks is not applicable.

Last, permanent direct, linked connections are subsequently excluded due to possibly high numbers of nodes. Therefore, more than five nodes are expected.

The literature describes peering mechanisms of BCT only on seldom occasion referring to single transactions (Daniel and Guida 2019, p. 4). Consequently push-mechanisms from an emitting node into the (whole) network are supposed to be common, whilst pull-mechanisms exceeding SCs could not be found in the literature. Consequently the literature lacks an overall picture using 'only a pull-Mechanism' or 'only a push-mechanism'.

Given these assumptions, a sample pull-mechanism and a sample push-mechanism are described:

4.2.1 Pull-Mechanism

As stated, the decentralized nature of BCT prevents frequently 'full-network' pull-mechanisms generally. Nevertheless, in the special case of PoT, the LN circles, alternating from node to node, due to the round robin procedure. Hence, the source for the latest update is defined by time. Consequently, nodes may pull from the latest LN directly or from its predecessor to prevent interventions of the LN's turn. Hence, the number of requests is tremendously reduced.

The following description is therefore set in a **consultationless** use case. The nodes are lazy and try to pull seldom to *reduce network traffic*. Hence, each node is just waiting for its own turn. A node missing its own turn is considered a worst case scenario. Therefore, if a node expects to have its turn soon, it becomes more active. A sample pull strategy is given in figure 4.3 and different node's behaviors are described in the following.

On the y axis, the pull strategy in 'requests per time slot' is given. For the sake of tangibility, first, 60 minutes is set as a turn's time slot (t=60min). Second, the network consist of 'n=50' nodes. Third, the network is not stiff, as 'ending a turn prior to its maturity' is possible.

This pull strategy is designed to be constraint to the node's distance to its next turn. Along the round robin alternation, the node will have its turn again and again in the multiple of n. But the number of turns until the next slot of being the LN has to be within 'x < n'9. Predecessors of the LN are not directly reset - hence 'x > 0' is possible - and consequently a *sample overflow* is set to

Depending on the base index, ' $x \le n$ ' is applicable as well

ten (n * 1/5 = 10). Hence, most network nodes (n * 4/5 = 40) assume to have a negative index. The range of x can be described as:

$$-n*(4/5) \le x \le n*1/5 = -40 \le x \le 10.$$

The used index overflow is implementation dependent.

Effectively, assuming from a time measurement, that nine turns need to be conducted until becoming the LN, offers the input 'x = -9' on the x axis. Following figure 4.3's graph, 'x = -9' leads to *two* pull requests per turn (time slot) or likewise *one* pull request (p) each 30 minutes.

$$p = t/f(x) = 60min/f(-9) = 60min/2 = 30min$$

Given these assumptions, the following cases (A to E) are distinguished:

1. Node A is half way through the round robin

$$x = -(n/2) = -(50/2) = -25$$

and looking forward to its next turn, which is more than 15 turns away (Figure 4.3: x < -15'). The sample pull strategy function returns y = 0.5' for values below x = -15'. One could say that node **A** is conducting in a 'network deep sleep', as it pulls only every second turn:

$$p = t/f(x) = 60min/f(-25) = 60min/0.5 = 120min$$

Nodes do not ask node **A** for updates, as they prefer to pull from nodes, which are assumed to just have finished their turn (e.g.: $1 \le x \le 10$), likewise node **E**. Obviously, node **A** does not assume to have its next turn soon. Even if some peers finished their turns early, a seldom pull request is sufficient to not miss the time slot for becoming the LN. Hence, an

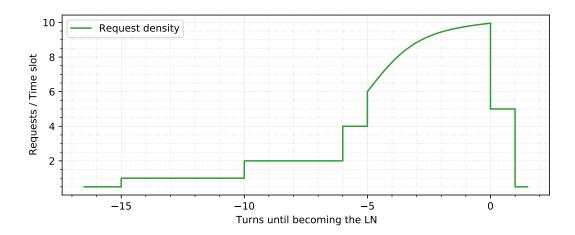


Figure 4.3: Sample 'peering by pulling'-strategy

update is requested after *two time slots have passed* (Figure 4.3: x < -15). The request follows the logic of the Business Process Modeling Notation (BPMN) diagram (Figure 4.4). As turns can be ended prior to its maturity, node **A** only has a scale guess on the recent LN. Consequently, node **A** tries to calculate the LN heuristically, assuming no prior prematurely turnover. Hence node **A** requests its updates from the node, which is assumed to have its turn right now (here: node **D**). If node **D** is the LN, it is free to either directly provide the update or refer to an already updated node (e.g. node **C** or node **E**). The latter shall prevent node **D** from being bothered during its turn.

In the **best case**, although targeting the LN, node **A** does not consult node **D**, but node **E** instead. Although node **E** is not the LN anymore, it has just finished its turn and replies the request offering the latest version of the BC. In the **worst case**, no answer is given at all, as node **D** is offline. Node **A** therefore asks node **D**'s predecessor (node **D**+1) and node **D**'s successor (node **D**-1), one after another. Except the LN all nodes are assumed to be willing to provide updates immediately as shared data is predisposed to reveal flaws and advances progress. Node **A**'s requests are conducted in both directions until an appropriate answer is received. A pull request from itself is obviously skipped. In the **cases in between**, neither node **D** nor node **D**-1 are targeted by the pull. Although data is received, the new information implies that the round robin procedure was (way) faster than expected. From the received update, node **A** can extract the timestamp from the last legal transition block. With this information it can recalculate the heuristic. Using this knowledge, node **A** may restart the process directly or wait until its next request trigger.

2. **Node B** is approaching its turn and assumes to be eighth in line (x = -8). Consequently, the sample pull strategy function (Figure 4.3) causes node **B** to perform requests each 30 minutes:

$$p = t/f(x) = 60min/f(-8) = 60min/2 = 30min$$

In regards of a full round, node $\bf B$ assumes to have its next turn somehow soon. Additionally, node $\bf B$ is not consulted by other nodes and utilizes the same procedure as node $\bf A$ (Figure 4.4).

- 3. **Node C** is almost at its turn and knows to be first in line (x=-1). Whilst the pull requests increased during the last time slots continuously (Figure 4.3: See $'-5 \le x < 0'$), node **C** tries to establish a stable, linked connection to the LN. Once successful, node **C** does not need to repeatedly request for updates. Node **C** is only consulted by other nodes in exceptional cases e.g. other nodes do not reach the LN (node **D**) or its predecessor (node **E**). Node **C** awaits to become the the LN.
- 4. **Node D** is the LN and in the middle of its turn's time slot (x=0). During the transition, node **D** received the latest version of the BC from its predecessor, node **E**. Consequently node **D** represents the latest version of the ledger and has no need to pull from any other node. Therefore the pull procedure is skipped by the first junction of the pull process (Figure 4.4). Still, a suitable value has to be found which balances between triggering the

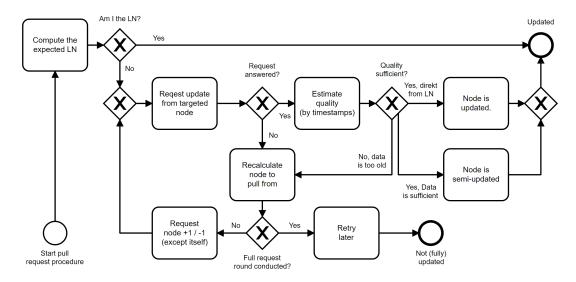


Figure 4.4: Pull request procedure

pull procedure too often and loosing connection to the network after the turn. All **four equations** (I to IV) represent the same intention: The LN starts pulling from other nodes after its own turn.

Whilst formula (I) does not allow to be calculated (Condition: ${}^{\prime}f(0)=0$), resulting not triggering the pull process anymore, formula (II.a), using a value nearby zero (${}^{\prime}f(x)=0.\overline{0}1$) increases the pull timer tremendously as shown formula (II.b). Until the next pull procedure is triggered after one year, many turns will have been missed already. Both, (I) and (IIa & II.b) disconnect node D from the network permanently and are therefore dismissed.

Not applicable:
 (I):
$$p = t/f(x) = 60min/f(0) = 60min/0 = NaN =>$$
 no pull and
 (II.a): $p = t/f(x) = 60min/f(0) = 60min/0.\overline{0}1 = p \rightarrow +\infty =>$ no pull - likewise -
 (II.b): $p = t/f(x) = 60min/f(0) = 60min/0.0001 = 60,000min > 1$ year

Therefore f'(0) = 5 was used in figure 4.3, which can be seen in formula (III). Here, node \mathbf{D} triggers the pull procedure each 12 minutes until the first junction of the BPMN diagram (Figure 4.4) is answered 'No'.

Applied: (III):
$$p = t/f(x) = 60min/f(5) = 60min/5 = 12min$$

Instead, a ratio could be used to send node **D** into a 'network deep sleep' as shown in formula (**IV**). The equation changes and uses a quarter round (n*(1/4)), constraint to the number of nodes. This leads node **D** to pull after 12.5h:

(IV):
$$p = t * (n * 1/4) = 60min * (50 * 0.25) = 60min * 12.5 = 12.5h$$

While node **D** is not pulling from other nodes, a stable, linked connection with the successor node **C** is established. Frequently node **D** is asked whether it is the recent LN. If node **D** had ended its turn already, it would provide the update. The other way around, node **D** may reference to a likewise updated node, such as node **E** or node **C**. Once all data is pushed, node **D** may end its turn early. Of course node **D** is allowed to decide whether it wants to remain the LN until the end of its turn's time slot or pass the turn to its successor before. An upper level application may incentive node **D** to pass the turn as fast as possible or prevent prior finalization of its turn. A reason against passing the turn ahead of schedule is that the LN's successor is not linked, yet. A variable turn time may allow the LN to even prolongate its turn's time slot given certain conditions. The changing starting times with every prior-finished turn may allow to last a LN's turn until the stiff's end (Figure 4.5). As the node might assumed to have the start of its turn later on. Either first, the node can not be blamed, second the rules stipulate a strict variable plan or third rules enforce online activity and aligned turn times.

5. **Node** E has just ended its turn. The node D has taken the lead and node E will pull again when the trigger (formula III or IV) is activated. Node E will reset its index, once it reaches the *sample overflow* (x=10). Consequently, for it's own updates, node E raises pull to the bare minimum and falls into a 'deep sleep', alike node A. Still, the next time slots will consist of many pull requests, which have to provide the latest data.

Concluding, aiming to optimize network traffic, a pull strategy is supposed to be the most light-weight scenario as nodes are allowed to fall into a 'deep sleep'. Still the pull strategy function has to be adjusted to the networks needs. Here both, step function elements (Figure 4.3: '-5>x' and $'x\geq 0'$) and continuous function elements (Figure 4.3: $'-5\geq x<0'$) may be used.

Obviously, the drawback of consultationless peering is the loss of short termed answers on behalves of the whole network. Here, to fix the latter issue, a full network round until a decision is made, would be needed. A special round for consultation, alike in the subsequent section Reveal claims & trigger events would breaking the stiff time slots and is therefore not suitable in consultationless environments. Implications of turns 'ending a turn prior to maturity' will be discussed shortly. Anyhow, stability can be risen using more linked connections or with a higher density of pull requests.

Still, regarding the pull request density on the LN, in larger networks this might lead unintentionally, to several internal, flawed by design, *network/transport-level flooding attacks* (ZARGAR ET AL. 2013, p. 2046), also known as Distributed Denial of Service (DDoS). Of course there might be possibilities to reduce the network load on the current LN, but updates would still trickle slower throughout the network than push based mechanisms, due to the asynchronous nature of pull timings. An issue which has to be covered generally using 'ending a turn prior to maturity', is the handling of nodes which take their turn 'too late, but in time'. Assuming nodes to stay in a permanent deep sleep until their next turn, only pulling once per round, may be sufficient in stiff environments (Figure 4.5: A, Eight stiff turns, green). But using early turnover (Figure 4.5: B), the turns one to six might have been ended that fast, that turn seven did not receive the update in time. Node six has to wait until node seven believes to have its turn again. Here, the waiting time



Figure 4.5: Ending turn before maturity

(Figure 4.5: B, Red) would be long enough to conduct two full turns. A upper level implementation would need to decide either to skip turn seven or to wait until the regular turn slot has passed. The latter prevents the round robin to proceed faster. Nevertheless, a fluid turn time demands a higher density of pull requests.

4.2.2 Push-Mechanism

Still in a binary world, in contrast to pull-mechanisms, a push based procedure offers several benefits. Again no literature regarding (full network) push-mechanism could be found. Nevertheless, the loose nature of *public permissionless* networks (see table 2.1), e.g. Bitcoin, does not fit the subsequently described, custom tailored push-mechanism for PoT.

In PoT, the LN only needs to communicate whenever it is willing to do so. Consequently, the LN is not flooded with several update requests as seen the described pull-mechanism. Depending on the network size (n), in small networks the LN may send its data to all peers itself (Figure 4.6: A) , using a direct push. In large networks (e.g.: n>50), a mechanism may be established to pipe the updates through Distribution Nodes (DNs) to reduce the workload for the LN (Figure 4.6: B). In this regards, this example uses a '10x' scale. Proceeding, instead of sending an update packet to every node, only the next *nine* nodes (here, indexes: '0 < x < 10') receive a direct update from the LN. The LN is its own source and has *nine* DNs. The DNs build a partly-master structure and lead a sub net each (Figure 4.6: B, DNs $1 \to$ nodes 10 to 20, etc.). An update may consist of any kind of new information (e.g. a new block). Except the LN, each node has one source-node and up to ten DNs. The index in this mechanism is counting down towards the LN. The DNs (d) of each node are calculated from the node's recent index (x). The calculation follows the function:

$$d = x * 10 + y$$

As each node has several DNs, y defines the number of DNs (here: $0 \le y \le 9$). Hence, the node on index two updates

$$d = x * 10 + y = 20 + y = 20 \le d \le 29$$

the nodes from 20 to 29. Consequently, the node on index *four* updates the nodes 40 to 49 and the node on index 55 updates the nodes 550 to 559.

Even if a node has poor connectivity, this source-drain mechanism reduces the network traffic being the LN tremendously from n down to *nine*. A drawback of this DNs variation are offline nodes, which have to serve other DNs. If e.g. node six is offline, the nodes to be consulted by six, 60 to 69, and their DNs will miss the latest turn's update. Nevertheless, the round-robin

alternation leads to changing update paths with every turn.

During the next alternation, the offline node *six* becomes the index *five* and hence, now fails to update nodes 50 to 59, whilst 60 to 69 are consulted by the new node, which just moved on index *six*. Again, this stiff and predefined solution is not possible in traditional BC solutions. Changing peers and undefined source nodes prevent this kind of peering in other CMs.

The so far explained push mechanism can be called vertical-push, as it establishes pseudo layers (Figure 4.7). Here the LN represents the (1st) source layer (Figure 4.7: Node 0) and it's DNs are contextualized on the 2nd layer (Figure 4.7: Nodes one to nine). The push mechanism follows a traditional tree structure (Figure 4.7: Black arrows) and offers the corresponding h = log10(x) to calculate the number of hops needed to transport an update from the LN to any index x. Additionally, further sample enhancements such as horizontal pipes, Jump pipes and full-push are described.

1. Horizonal pipes:

A *horizontal pipe* pushes information on the same layer from one node to another, e.g. from node 21 to the nodes 11 and 31 (Figure 4.7: Brown arrows). If the nodes 11 and 21 have not already received, the designated upper level node(s) may be asked to confirm the horizontal update. Alternatively, the neighboring nodes may be asked as well.

2. Jump pipes:

Jump pipes skipping a level may occur, if any source node does not reach a distribution node. Then, e.g. node *zero* is not able to reach node *two* and therefore pushes the update directly to the nodes 20 to 29.

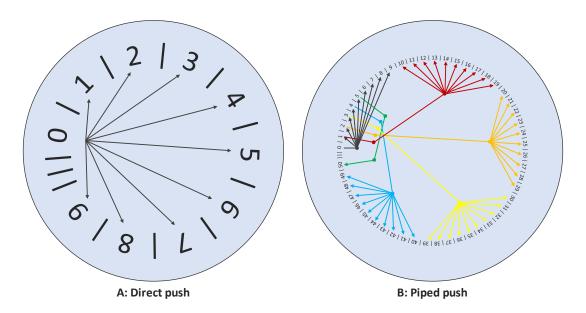


Figure 4.6: Update Mechanism

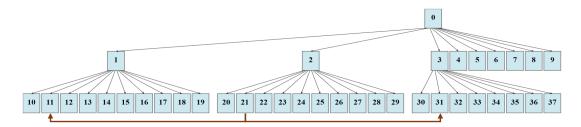


Figure 4.7: Pseudo layers of peering by pushing

3. Full push:

As stated before, pushing all updates to the whole network might be expensive for the LN, especially in large networks. Nevertheless, in contrast to loosely coupled *public permission-less* networks, in PoT the LN is expected to knows all the network's nodes. Therefore, some circumstances may benefit a 'full push'-solution (compare *Aura* and *Clique* in DE ANGELIS ET AL. (2018)), such as transition blocks or any block which demands immediate consultation. Especially, the handover-/finalizing block is a crucial block for the LN, as it marks the end of all in this turn written data. If this block is delivered, it sends all missed data to any receiving node. Hence, unattended and dropped behind nodes receive a fresh input. Additionally, once the LN has finished its turn, it is not occupied anymore. Therefore the node is able to distribute the data without performance loss (during its turn) throughout the network.

Further mechanisms to fix *update prevention*, caused by offline nodes, remains implementation dependent.

Concluding, the sample **pull-mechanism** offers to skip updates (e.g. pulling only every second turn) and therefore reduces network traffic throughout. Obviously, if an increased pulling is not conducted, this comes to the cost of outdated data. The latter is bad for spontaneous events, such as randomization and trigger events. The other way around, much consultation in vain is happening as pulls are conducted without receiving new data. An implementation dependent trade-off has to be found.

The key benefit of **push-mechanisms** is the ability to provide updates fast whilst reducing the traffic for requests. Especially randomization and trigger events, as far as needed, demand for throughout high availability of every node and therefore favor push based peering solutions. Nevertheless, loosing the network is supposed to be one of the major threats for BCT systems. Therefore, pull mechanisms are an essential part of the network, especially if nodes loose connectivity. Finally, a push-mechanism appears to be superior to a pull-mechanism, nevertheless it was shown that PoT can generally be operated by a pull-mechanism as well.

Additionally, the two mechanisms are not mutually exclusive and may be combined as needed. Thus, a mix of both is generally recommended. Hence, it is implementation dependent when the network is assumed to be lost e.g. if there was no update from the last three writing nodes. Here it should be aware that BCT solutions assume that nodes have an extrinsic (e.g. PoW) or intrinsic (PoT) desire to stay online. Consequently, it is assumed that the peers next to the leading nodes

behave in a way offering high availability. Still, it has to be assumed that mobile clients only offer a reduced bandwidth. This behavior is unlikely for fat clients, traditionally linked with mining in other CMs. Nevertheless, if nodes are represented by consumer systems, mobile clients are assumed to have an increased online availability compared to desktop systems, because mobile clients are turned of or cut from the network (flight-mode) only on seldom occasion. Smartphones, being usually switched on during mobility, strengthens this assumption. In this regard, if not addressed by lower level peering, mobile clients might change their network address more often. If this case is applicable, a higher interconnection may be needed. After all, it is recommended to first design the upper level application to assemble the needed characteristics of PoT. With these characteristics the effective features regarding consultationfull and consultationless interconnection and consequent peering can be decided upon.

As the general information flow is supposed to be solved without a superior ledger, other questions raises regarding CF, Byzantine Fault Tolerance and so forth.

4.3 CAP Theorem measurement

The CAP theorem (see: Brewer (2012)) is one of the most important models to describe distributed database systems. "The CAP theorem asserts that any networked shared-data system can have only two of three desirable properties. However, by explicitly *handling partitions*, designers can optimize *consistency* and *availability*, thereby achieving some trade-off of all three" (Brewer 2012, p. 23). DE ANGELIS ET AL. (2018) compare in their paper two PoA implementations, *Aura* and *Clique* with PBFT regarding the CAP theorem.

They claim that **consistency** is achieved, when forks are avoided (DE ANGELIS ET AL. 2018, p. 6). Consequently, consistency in the context of BCT is represented by instant consensus finality (DE ANGELIS ET AL. 2018, p. 6). *Aura* fails in this matter as glitches might occur during the transition from one LN to another, resulting in *no consistency* (DE ANGELIS ET AL. 2018, p. 7). In detail, the time frames to write blocks in *Aura* are shown by *node* E and *node* E (Figure 4.8: Aura). *Node* E is the successor of *node* E. Due to exact timing, *node* E could propose a block within the time slot E (Figure 4.8: Aura), wherein any node of the network may already assume *node* E to be

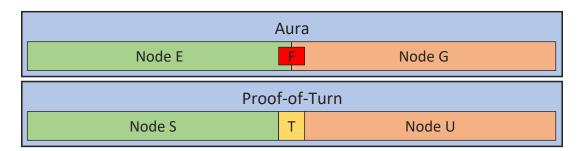


Figure 4.8: Permission Transition

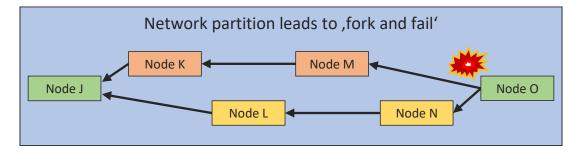


Figure 4.9: Network Partition in PoT

the LN. The *Aura* client fails and the conflict is not solved. *Clique* does not meet *consistency* as well, as several nodes are allowed to write simultaneously. Solving this issue using the Ethereum GHOST protocol leads to *eventual consistency* (DE ANGELIS ET AL. 2018, p. 8). In this regards, PoT's *consistency* can be considered high because of the before mentioned *transition times* (Figure 4.8: Proof-of-Turn, T). First, a *transition time* is only needed if *node S* uses the full turn time. *Node U* has to assume that the writing permission will not be received before 'turn time plus transition time' is over. Second, *transition time* prevents the network from assuming different leaders due to *time skews*. Therefore the duration of each *transition time* has to be chosen large enough to compensate inaccuracies of the client's time protocols. After all, PoT does not need as precise timing as *Aura* or PoET.

Viewing from another angle, a BC "[...] is available if transactions submitted by clients are served and eventually committed, i.e. permanently added to the chain" (DE ANGELIS ET AL. 2018, p. 7). Following DE ANGELIS ET AL. (2018)'s example of the PBFT mechanism, PBFT stalls sometimes giving up availability to achieve consistency (DE ANGELIS ET AL. 2018, p. 8). The stall occurs, when the network can not agree on a LN or a transaction proposed by any node is not added to the BC for a long time. Assuming PoT as a 'networked shared-data system' it prevents writing by every client simultaneously, sacrificing availability. If the LN, by chance, has no network connection and is therefore not available, PoT stalls (temporarily). Data from the LN can not be written to the BC by other nodes, as other nodes can not sign the blocks accurately. Arguing that the other nodes are not willing/allowed to write would break the 'networked shared-data system'-assumption and makes the CAP Theorem classification obsolete. It would result in a circulating centralized server. Last, when "[...] a **network partition** occurs, authorities are divided into disjoint groups in such a way that nodes in different groups cannot communicate each other" (DE ANGELIS ET AL. 2018, p. 7). This scenario is shown in figure 4.9, where the network is split into two groups, orange (Node K & node M) and yellow (Node L & node N). The two groups do not see each other and assume that the other side is 'just absent'. Now, node O two distinct branches of the BC. If these branches represent different game states, they may be mutally exclusive. As game states depend on each other, e.g. in RftS, node M unknowingly sends a fleet from a planet which has been conquered by node L just before. The resolution of such forks can only be solved within an upper layer and might demand a reset of the BC to any previous block/state. In the worst case an upper level game is broken and has to be stopped. The resolution of forks by the game layer remains implementation

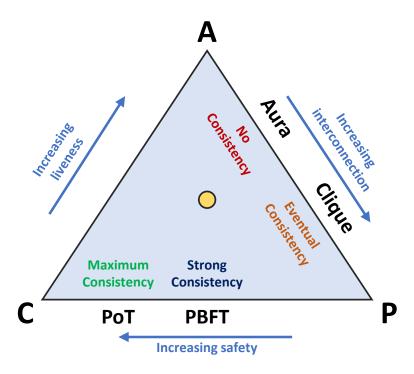


Figure 4.10: Mechanisms regarding the CAP Theorem (Adapted from DE ANGELIS ET AL. (2018))

dependent. Concluding, regarding the CAP Theorem, PoT fails partition tolerance. Along these thoughts, DE ANGELIS ET AL. (2018, p. 7-8) categorize the PoA implementations *Aura* and *Clique* as well as PBFT into the CAP triangle. If PoT was categorized as a *circulating centralized server*, as all nodes except the LN are considered to be silent throughout, it would be positioned in the middle of the triangle (Figure 4.10: Yellow circle). If PoT was categorized with *eventual consistency*, as a pull mechanism does not always provide the newest data, it would be in the same area as *Clique*. If PoT was categorized as handling *network partition* well, as the upper level use case was only a dumb storage solution without constraints within pushed data, it would be similar to PBFT. Nevertheless, following the argumentation, PoT is added, as shown in figure 4.10.

4.4 Forks & transition blocks

In any BCT certain problems occur, such as forks (Butijn et al. (2020, p. 60); DIB et al. (2018, p. 54)). A fork takes place if there are two mutually exclusive continuations of the approved BC (Yuen et al. 2019, p. 22). The network has to decide which trail is going to be followed (Courtois (2014); Ewerhart (2020)) or has to start again from the last legitimate (non exclusive) block (Finlow-Bates K. 2017, p. 3-4). It has to be mentioned that this section is not about *soft-/hard forks* on the BC's CM's protocoll (Finlow-Bates K. 2017, p. 4-5).

While most Proof-of-Mechanisms solve this issue using the Longest Chain Rule (COURTOIS (2014);

NAKAMOTO (2009)), PoT insists on following the rules regarding timing. The blocks which can only be written by the LN have to be pushed within the given time frame of the turn (see figure 4.8: PoT). Hence, the race condition is not dependent on computation power, but on a valid identity regarding time. Having the turn grants authority to write. At this point, PoT could be classified as a special type of PoA. If a fork is detected, there are basically **four choices**:

First, the branches are soft exclusive, as the upper level software allows to merge the branches without dispute (Ranchal-Pedrosa and Gramoli (2020, p. 7-8); Wang et al. (2018, p. 4-5)). The data from one of the two branches is encapsulated by the recent LN, signed and appended on the other branch. Once merged, data from both branches are CF on the main chain and the LN continues with its own data. **Second**, the branches are *mutually exclusive*. Nevertheless, encapsulation takes place to regain a consistent main chain. Next, a dispute has to be raised to decide upon CF and invalidation of the questionable parts. This triggers a network vote on which (on-chain) fork has to be continued. Herein might be decided on a reset, which resolves in a replay of some turns. Alternatively, third, a node (or transition block) is chosen to be continued from, terminating in a (soft) loss of some turns which will have to be conducted again. This behavior equals PoW's Longest Chain Rule (EWERHART 2020, p. 2) as some transactions/blocks were computed in vain (FINLOW-BATES K. 2017, p. 3-4). Fourth, the worst case for the LN originates from a failed vote - now the LN has to decide on its own which branch to continue. The easiest way for the LN would be to just choose one branch (randomly, internally). The drawback is that lacking consultation prevents consent and endangers the CF of the LN's data. Likewise, the decision can be made using a randomization call - which is the most interactive version, raising participation. The other possibility is to choose the chain which implies the highest loyalty e.g. the most turns. It is important to mention that the branch offering the most turns is not necessarily the one with the most blocks. Consequently, the Longest Chain Rule is not applicable here. Nevertheless, the chain which represents the most turns, is the one which offers the highest resilience against network votes (e.g. regarding invalidation). The more stakeholders can be assumed for a branch, the higher the possibility for a positive network vote. Concluding, the chain, continued by the LN, is the one representing the most progress regarding turns (Figure 4.11). After all, to give this scheme a name, PoT uses a 'Most Progressive Chain Rule' instead of the Longest Chain Rule. Here, it is likely that a loyalty based approach is superior to a randomization based approach and randomization is only chosen, if the forks offer the same loyalty factor. The node could also use

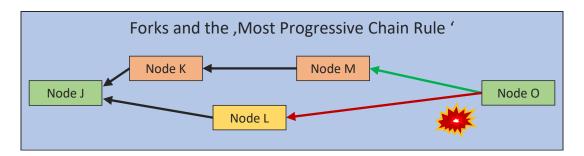


Figure 4.11: Most Progressive Chain Rule

Consent	Round Robin	BFT	Effective	
threshold	smoothness		CF	
Low (≤ 33.3 %)	High	Low	Fast	
Medium ($\sim 50~\%$)	Medium	Medium	Medium	
High ($\geq 66.6~\%$)	Low	High	Slow	

Table 4.1: Consent smoothness

the values from the failed network vote to influence its decision (slightly).

Last, here is a **fifth** negligible solution: As creating new blocks in the PoT CM is cheap, a node could follow several branches. This solution creates forks even if none has occurred in the first place. Each node would fork from the last some transition blocks, resulting in turns which have to be calculated several times with changing preconditions. Therefore the latter option is not considered as an option in the first place. The effective arrangement is implementation dependent. Of course, the validity of a transaction is checked by every other node. If a block can neither be checked or rejected, as for hidden transactions, it is accepted. Still it does not change the game state, yet. If one node does not want to accept an open block, voting determines the validity of the written block - *silence infers consent*. Nevertheless, the probability of a block to be accepted raises with every silent consent or approved check and is finally accepted if it is approved by a defined threshold of nodes. Of course, silence could infer refusal as well, but first PoT assumes online peers and second, if too many peers were offline, the network would stall constantly. Hence a *silence infers refusal*-approach is strongly discouraged from. The threshold, for a vote to pass, can be defined as 'a few nodes' or 'the whole network'. In table 2.5 this was marked as *implementation dependent*.

The chosen value affects the BFT, the CF as well as the smoothness of the BC as shown in table 4.1. Luckily a high consent threshold does not lead to a freeze, such as in MultiChain BCs because of the *silence infers consent*-principle. Even if there is no interaction, the Round Robin procedure proceeds. Consequently, latest after one full round a block reaches effective-CF. By then it can not be invalidated anymore. The CF is therefore both faster approached and can be realized with a higher threshold than e.g. PoW, as PoW suffers from a small residue probability to be forked by any computation superpower (DEMI ET AL. 2021, p. 4).

Blocks which are not supposed to be in discussion are transition blocks, those blocks which pass the turn from one LN to the next. These transition blocks may enable *adaptive turn time* (See section Further Characteristics) and ensure consistency (figure 4.8: PoT). Therefore, transition blocks can be used for *fast update verification*. If a node stays offline for longer, it may ask for an update providing the last known block. But due to forks or Data allocation improvements that block is not existing anymore. Instead the requesting node could provide its last transition blocks, which is not allowed to be deleted. The last known transition block from the requesting node will then become the start block of the delivered update.

4.5 Reveal claims & trigger events

These spontaneous transactions were already mentioned at the end of chapter Blockchain in Games. Although it was stated that general mechanism can be conducted using BCT, certain examples were left out. Therefore, three different approaches are presented hereafter (Figure 4.12). Following it is assumed that the turn is on node α .

Claims and triggers are active transactions conducted by an intervening node (e.g. β), breaking the regular writing window, to change the outcome of player α 's (sub-)move. Due to the generally asynchronous nature of RftS this is not recommended, but still possible within the PoT mechanism. Possible solutions are given in figure 4.12 (A-C) and range from 'piping a signed block' (A) over conducting a 'trigger round' (B), up to 'defined detours' (C). In case A signed blocks are piped via node α to the BC. Case B describes a fast paced network round wherein only triggers can be pushed legally. In case C the writing permission is only actively given to claiming nodes, who write the triggers themselves before handing the writing permission back to node α . Whilst reveal claims are straight forward and any answer has to be given by the targeting node, moves which **might** trigger, offer certain challenges regarding figure 4.12:

- 1. A trigger can only be performed by β, γ and δ if both, **possibility** (any kind of resource) and **will** (to interfere) are given.
- 2. In case A (figure 4.12) nodes β, γ and δ have to believe that node α received the piped transaction in time and is not actively ignored. The same applies to case C. As the LN is generally allowed to ignore other nodes claims, interoperability may fix this issue (Section: Interoperability).
- 3. Depending on the rules, triggers are not performed in line, players wait for others to execute triggers before they use their own resources. Hence, triggers are 'not in line'. For case B (figure 4.12) this would lead to several rounds of consultation. Consequently, if not automatically performed, the manual interaction will slow down the game and/or annoy players due to repeated consultation.
- 4. Additionally, depending on the numbers of players, case B lacks especially from a long freeze-like waiting time until all nodes have performed their answer regarding willingness to fulfill a trigger call. However, answering automatically, if the resource to pull a trigger is not available, could unwillingly offer vulnerabilities (e.g.: This node is not able to perform any defensive action).

Bullet point two could be handled by a child-chain storing raised triggers by β , γ and δ including their timestamp. Due to the random character of calls to pull a trigger, this chain would need to rotate fast with PoT or has to use another type of consensus mechanism, e.g. out of the proof based CMs.

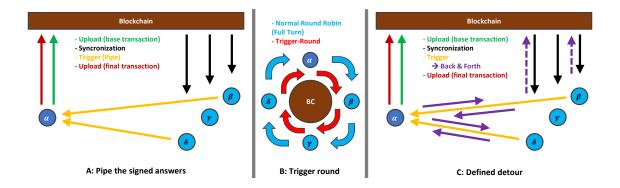


Figure 4.12: PoT: Trigger event mechanisms

4.6 Interoperability

Except from Besancon et al. (2019) and Kraft (2016) there were no papers to be found regarding interoperability of BCT for games. Nevertheless, some implications are given: Those in the previous chapter mentioned side-chains refer to the before mentioned on-chain, off-chain, child-chain, side-chain and inter-chain methods (KIM et al. (2018)), see section Performance Improvements. These methods shall now be inspected for internal and external use regarding interoperability. If not mentioned explicitly, there is no special use case. Primarily it is focused on multiple layers next to the main-chain.

First, internally: If a game wants to augment direct immediate responses in an asynchronous network, preemptive actions have to be taken. This would include some spare encryption keys or other conducted computation which include network consultation. It has to be aware that allowing such measures might break the integrity of a game. For example, computing a randomization in advance lets the player know the chances before actively deciding to take the randomly impacted decision. Nevertheless, those pre-computed blocks which need consultation, but are not secure to be conducted, shall not be written to the main-chain - just as bloat blocks. Not conducted turns can be invalidated with a revocation or left open (unrevealed) - just as bloat blocks. Only

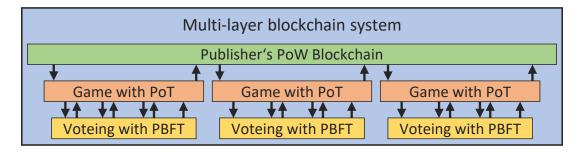


Figure 4.13: Multi layer BC system

those, which are finally used are written to the main chain. Once all blocks in the side-chain are revoked or written to the main-chain, the side-chain can be deleted. Moreover, an effective implementation of PoT has to decide whether transactions stored in the main-chain only have to be CF or at least effectively CF (use case dependent). The latter would call for a child-chain with CF blocks/transactions, which is referred to by the main-chain. The CF blocks/transactions in the child-chain are referenced in the main-chain earliest after the defined BFT threshold has been passed (=> effective CF). As revoking of CF blocks/transactions is supposed to be seldom, the transcription/reference can be done by the emitting node, after the round has passed. If used, these mechanisms lead to an internal two-layer structure of PoT.

Second, externally: This part focuses on PoT being an off-chain or side-chain to any other CM or PoT using any other CMs as off-chain or side-chain solutions. Hence it has to be distinguished between PoT being below or above the other CM. For an improved understanding, figure 4.13 shows PoT in both situations - on the one hand being a transaction channel child-chain for a PoW parent-chain and on the other hand being a parent-chain for a child chain with PBFT functionality (e.g. for voting or trigger events etc.).

The below part was already shown by KRAFT (2016), who uses PoP in Huntercoin for game-channels outside a main-chain using another proof-based CM. The other way around would be to fix the issue of availability shown in by the CAP theorem. Here, any child-chain offering a simultaneous proposing of blocks would be suitable. Consequently, dispute resolution by votes and other messages which are of simultaneous nature may be covered with those child-chains, whilst the actual game play is maintained by PoT. A child-chain for votes and disputes could prevent LNs from ignoring unwanted messages and enable simple trigger-event resolution without circumlocutory logic (Section Reveal claims & trigger events). Last, if needed, a publishers server may be established as a part of any of these chains - the publisher's incentive to do so is consciously left open.

4.7 Peer-Fluctuation & adaptive turn time

During the existence of a BC, there are multiple reasons for nodes to leave or join the BC's network. Whilst joining is generally considered an active move, leaving can both be an active operation as well as a passive coincidence.

Join operations can be maintained as any node within the network proposes a new node. Either the network decides whether the proposed node will be added, or a trusted entity (e.g. a publisher's server) may add a node to the network $(n \to n+1)$. The network's (/game's) state predetermines where the node is added regarding the round robin state. The new node's index is chosen by the upper level software as it may be inserted at the end of the round or within the next some turns. In a self organized network using the pull mechanism, without a publisher's server, it may take long to add a new node if no trigger event mechanism is used. Nevertheless, to prevent two nodes from assuming to be the LN, it is discouraged from replacing the recent LN as well as inserting the new node as a successor of the recent LN.

Leave operations can be conducted by publishing all data the network needs to keep the upper

level software running. This is especially needed in a game's card draw-scenario and likewise situations. The leaving node is cut out of the round robin and each round progresses faster $(n \to n-1)$. It has to be determined whether a node is allowed to rejoin. Here, if private keys are shared during the leave operation, a rejoin is only possible using a new identity. Both, join and leave operations are active operations.

Dropping out by (passive) coincidence is possible as well. For nodes leaving without a trace like software/hardware errors or any longer network connection outage, the upper level software has to be designed in a resilient way to cover such coincidence (e.g. shared keys for shuffling cards, figure 3.12). If it fails to cover these cases, the BC could be hold hostage by any absent node. As the Round-Robin procedure continues, there have to be mechanisms which lead both, the network and the node to regain the connection. First, if all nodes are considered fluid parts of a larger network, a static server may help to keep trace as a lost node may ask the static server upon the other nodes of the network. Alternatively, second, if the lost node is the LN, the BC's network could sacrifice the static amount of time of the LN's turn in favor of a likely re-connection. This action can be seen as the counterpart of 'ending a turn prior to its maturity'. Both actions are seen as measures of adaptive turn time. The granted amount of time (from minutes to days) as well as punishments after the lost node's re-connection remains implementation dependent. To find a suitable measure Laneve and Ershov (2019, p. 27)'s claim of BCT to be a "risky choice for developers" comes back to mind. In a gaming context, the other players would have to wait longer during the prolongation. Still, for in-game experience it is important to reduce waiting times as waiting "[...] is frustrating, demoralizing, agonizing, aggravating, annoying, time consuming and incredibly expensive." Buffa (1976) as cited in (Maister 1984, p. 1). Consequently chopping down waiting time is considered a key feature. To solve this deadlock, BCT in general seems inappropriate.

Nevertheless, following the idea of adaptive turn time, measures such as a *night switch* or a *vacation break* could be implemented as well. For Pause-by-Consensus votes, some players might not be able to answer manually, as they do not suppose to have their turn. This is especially true for long termed games with multiple players. Therefore opt-in, opt-out or predefined answers are recommended. It has to be aware that a malicious node could pause the game repeatedly if too many nodes give automated positive answers for each prolongation.

4.8 Further Characteristics

This chapter covers further mechanics which can/have to be used to enable certain game characteristics.

1. Helper Nodes

Although most game mechanics can be guaranteed, using BCT only, some mechanisms need intervention of external resources/nodes.

In a basic scenario there are no such mechanisms which need strict timing to circumvent race conditions as transition times (figure 4.8: Proof-of-Turn) prevent race conditions from delayed network messages. However, if transition times are diminished or left out to speed

up, additional external resources (figure 4.13: Upper BC) may be needed to solve race conditions. Still, each additional consultation results in a slower algorithm. It is encouraged to just set the transition time appropriately.

Nevertheless, such helping nodes shortcut cumbersome solutions such as randomization and (hidden) card draw. A helping node might even be a publishers (central) server. Consequently, the publishers server does not host games entirely, but keeps being available for certain services. Especially for randomization this may help to meet tight time frames instead of waiting for slow (inactive) peers. In this case 'tight' means ≤ 5 seconds to maintain a reasonable user experience.

Additionally, if no lower level chain (figure 4.13: Lower BC) for intervening messages is provided, a central server may be used as a referee in any dispute. Any LN would therefore not be able to deny transactions/messages with for the LN disagreeable content. However, this is not seen as a suitable solution, a structure as shown in figure 4.13) should be sufficient for this case and is therefore recommended.

2. Shared node resources and cooperative gaming

The distributed nature of PoT enables each player to use several devices to act as one node (Harkanson et al. (2020)). if multiple participants offer the same authorization and authentication parameters, they can be seen as one node (Karafiloski and Mishev (2017)). Although this feature could be enabled, it has to be considered how the clients are supposed to behave. Here, contradicting data has to be avoided as, only one of the devices shall behave as a LN. If its the player's turn, consequently only the used device shall be consulted to pull from. Nevertheless, BCT enables cooperative gaming of several persons as one player in the game (Harkanson et al. (2020)). The players have to be aware that once granted access rights (private keys) can not be reversed. Nevertheless, this feature could enable to play cross platform with many devices (e.g. Computer, Smartphone and Tablet).

3. Game session interaction

Imagining parallel games of RftS with likewise parameters this could allow (given certain parameters) to send a fleet from game A to another game B. Here, following figure 4.13, both games might be initiated on the upper level and played on distinct PoT BCs. The sending transaction would include to join the other game's network and become a node in both round robin circles. The games do not necessarily need to be in sync and the newly added node (/player) becomes a node following the peer-fluctuation rules of the specific implementation. Here, an upper level BC as seen in figure 4.13 or a helper node for the transition is believed to be necessary.

4. Reference Implementation

There are manifold implementation dependent characteristics for a network based on PoT. PoT has to fit the purpose of the business case and has to be set up accordingly. The described business case is not part of this document and hence a reference implementation does not fall into place. Therefore it is desisted from conducting a reference implementation. Nevertheless, table 4.2 lists parameters to be considered:

Parameter	Value
Nodes	Open (1 to ∞)
Turn time	Open (10 seconds to several years)
Maximum block size	1 to ∞ transactions
Transition times	1 to 10 seconds
Finalizing blocks	Boolean
Handover blocks	Boolean
Peering	Push- vs. Pull- vs. Mixed-mechanism
External resource	None vs. Server vs. Upper level CM
Sub-Chain	Boolean (+ chosen CM)
Votes (/& Disuptes)	Boolean (+ define rules)
Randomization	Boolean (+ define rules)
Adding nodes	External resource vs. Self maintained
New node's index	Start, Next, End, etc.
Adaptive turn time	Boolean (+ define rules)
Multiple devices	Boolean (+ define rules)

Table 4.2: Parameters of PoT

4.9 Attack Vectors

Now, some general attacks on BCT are described. Consequently, attacks against popular CMs are described and the applicability on PoT is given. Additionally, some attacks special to PoT are described as well.

An attack shall be considered any hostile behavior which might impact negatively on single nodes or the network itself. It has to be distinguished between *inside* and *outside* attacks. Here it is asked whether the saboteur is part of the network or not. Moreover it has to be distinguished between tricking the upper level software and destroying the BC's network.

1. Byzantine Fault Tolerance

As this is one of the most discussed topics (DIB ET AL. (2018); KHAN ET AL. (2020); GRAMOLI (2017)) on the PoW approach, it has to be mentioned here. As PoW is designed to be used in a public permissionless (Table 2.1) environment wherein all nodes are able to enter and leave freely (Nakamoto 2009, p. 8), an attacker with computational superpower (likewise the majority of CPU power) could take over the network (Nakamoto (2009, p. 8); Demi et al. (2021, p. 4)).

The nature of PoT is completely different here, first PoT is supposed to be used in private/public hybrid/permissioned networks. Stil,l proposing new nodes is possible. Second, instead of CPU power, the number of controlled nodes is the key for fraud in PoT. The threshold which marks the BFT for PoT as described in section 'Consensus finality and Byzantine Fault Tolerance' marks the number of nodes needed to become the networks superpower. Once archived to control a share of network nodes above the threshold, lets the attacker vote (and kick) in any wished direction. The network has to maintain that such majorities are prevented. Still, a BC network 'dies' when one all nodes have left. Any added node is a threat to the voting power of the other nodes. Hence, the upper level software has to define rules for joining nodes.

2. Benign faults

Next to the BFT, benign faults are a source of inequality and injustice. Here "[...] leader misbehaviours can be caused by benign faults (e.g., network asynchrony, software crash) [...]" (DE ANGELIS ET AL. 2018, p. 4). Although these benign faults are no attack in the traditional sense (Charron-Bost and Schiper (2009)), they endanger the upper level software to stall (DE ANGELIS ET AL. (2018, p. 8); Greenspan (2015, p. 7)). As benign faults can barely be distinguished from e.g. 'unavailability on purpose', each implementation has to draw a line between the two cases.

3. Nothing at stake, long range and transparent forging

LIN AND LIAO (2017) describe two flaws of PoS, the *Nothing at stake* attack as well as the *long range* attack. Both attacks are applied from inside the network. The *Nothing at stake* attack is incentivized by cryptocurrency tokens from proposing a block even if there is no data to be written (LIN AND LIAO 2017, p. 301-302). PoT does not suffer from this as block

generation does not offer any cryptocurrency token rewards.

The *long range attack* derives from the *Longest Chain Rule* and an attacker who once had a high amount of computing power being able to rewrite the chain later on (LIN AND LIAO 2017, p. 302-303). PoT is resilient against this attack as well, as the data written by a node can only reflect the own turn's transactions. As long as the integrity of other nodes is not broken (e.g. published private keys), there is no way for an attacker to augment their moves. It has to be mentioned that PoT does not rely on a *Longest Chain Rule*, but rather on a *Most Progressive Chain Rule*, which is an adaption of the *Longest Chain Rule*.

Last, some CMs such as PoS allow to propose the next LN. If this is possible, a subgroup of nodes is able to pass the LN within their peers, which is called *transparent forging* (LIN AND LIAO 2017, p. 302). Due to the Round Robin procedure of PoT, *transparent forging* is not applicable.

4. Integrity of nodes

As details of encryption are seen out of scope, security of symmetric and asymmetric encryption are supposed. Still, the loss of the private encryption key(s) is critical to the integrity of any node. Once an attacker obtains access to the private key, the nodes identity can be augmented and used in hostile manner. Except the loss of the private encryption key(s), PoT seems secure on outside attacks regarding integrity. Likewise, Douceur (2002, p. 251) describes the 'Sybil Attack' wherein large-scale "[...] peer-to-peer systems face security threats from faulty or hostile remote computing elements". Here, if "[...] a single faulty entity can present multiple identities, it undermining this can control a substantial fraction of the redundancy." (Douceur 2002, p. 251) Hence, the integrity has to be kept throughout. Additionally, Douceur (2002)'s claim reflects the risk of BFT being captured by an entity, which represents the majority in a vote through multiple controlled network nodes.

5. Timing

Revisiting the PoA client *Aura* from section Consensus finality and Byzantine Fault Tolerance, malicious LNs could try to post their blocks as near to the end of their writing permission as possible to break the algorithm. Of course this is considered an inside attack with the implication to break the network. This flaw is prevented in PoT if *transition times* are enabled.

6. Silencing nodes

The prevention of service of any node of the network can be accomplish by any (larger) entity. The idea can be abstracted from a single attacking device accomplishing *Denial of Service*, therefore the term DDoS is used. DDoS can be both of internal and external nature (see WU ET AL. (2010)).

JOHNSON ET AL. (2014, p. 72) take a look at Bitcoin (PoW) mining pools and see both potential and incentive to use DDoS to gain competitive advantage over other mining pools. PoW does not assume any node to provide a block, hence the attack only harms the single node without hurting the remaining network. In PoT the targeted node has to be distinguished

between the LN and any other node.

Assuming that there is no data the network needs from the targeted node to continue, the DDoS attack is only harms the state of the single node (update prevention). This is applicable if the targeted is not the LN and does not have to deliver data (e.g. trigger/card draw/randomization).

The other way around, if the targeted node is the LN, the network is targeted entirely, as the progression stalls until the LN's turn is over. Still, information which has to be provided to the network by the LN, can not be served.

It can not be avoided, that the distributed and interconnected structure of BCT has to expose the needed network address information, which is needed to conduct the attack. The general solution would be to guarded nodes from each other by a central server, abstracting the network addresses, but this breaks the BCT's paradigm. Moreover, the DDoS could follow the round robin sequence and target one node after another, silencing the (whole) network. Nevertheless, if detected and once the attack is over, the round robin could be reset and continued from the last agreed state.

7. Double spending problem

One of the key problems PoW solved is the ability to spend assets/tokens in distributed systems twice (Nakamoto 2009, p. 1). PoT is able to guarantee this as well. If a node does not comly given SCs, the block/transaction is invalidated by vote and does not reach *effective* CF. Consequently, the published information which uses PoT has to deliver transparent data with every transaction. The double spending problem arises when hidden transactions become part of the upper level software. In this concern, DIB ET AL. (2018, p. 56) state that "[...] it is always possible to add encrypted data using the recipients public key, but then the validators cannot verify the semantics of the transaction (e.g., double spending) [...]". Therefore, specific (upper level) implementation has to solve related issues.

8. Effective network split

This refers to a node, sending different data to different parts of the network, resulting in a forced fork. This could either happen, **first**, due to lost identity or team gaming with consequent simultaneous gaming or **second** hostile emitted blocks by a single node. Now "[...] some of the nodes of the blockchain recognize a node as the next block writer, while other nodes recognize another node as the next block writer" (YUEN ET AL. 2019, p. 22-23). All three scenarios can be detected by the network as long as the network rechecks the last (transition) block of each LN to deliver equal fingerprints (Hashes) on each receiving node. Possible resolutions are described in section Forks & transition blocks. Hence, consistency can be maintained by preventing network partition.

9. Data flooding

Assuming PoT to be a lightweight algorithm for mobile games (e.g. on smartphones) and offering the possibility to write bloat blocks, enables to flood the network until some nodes have to drop out of the game due to allocated storage constraints. The mechanisms like pruning bloat blocks or meta-state blocks, wiping the history, are countermeasures but can

not be seen as all-embracing solution. Depending on the final use case, a maximum write permission may be needed as well. It has to be mentioned that a limit (rule) may be broken if the network decides it in a vote (changes the rule). Nevertheless, single nodes may clean for themselves to keep the 'important' parts of the BC. Thresholds for both the limit and the vote remain implementation dependent.

Further, more technical, attacks on BCT can be found in MIN AND CAI (2019). As MIN AND CAI (2019)'s attacks can not focus on PoT yet, they are out of scope.

4.10 Limitations

First it has to be mentioned that this document is (only) based on theoretical background. Many assumptions had to be made, as seen in table 4.2. Some of the assumptions, as a reminder, are given in the following:

- 1. Many BCT CMs, due to their extrinsic incentives of cryptocurrency tokens, prevent offline peers. A software publisher has to decide whether intrinsic incentives are satisfactory to provide a network with PoT. Taking the targeted platform with their consumers into consideration may lead to a decision.
 - Here, strong tiers alike a company's servers, personal computers and gaming consoles tend to be switched off after a gaming session and might not stay online over a longer period of time. Consequently extrinsic incentives are needed.
 - On the contrary mobile phones, likewise to WhatsApp usage, can be supposed stay online most of the time. Hence they do not need to be switched on on purpose. Nevertheless, offline nodes, stemming from flight mode during night times might stay a problem for PoT.
- Especially if choosing mobile clients, disk size limit has to be evaluated critically. Although new devices may offer excessive resources, if the targeted consumers are throughout the whole range of devices, average free resources may drop dramatically.
- 3. Although PoT is considered a lightweight solution among other CMs, it is still no solution for 'fast paced' upper level software. Transition times, needed to provide throughout consistency may be too long for immediate response. This is especially true for large groups of nodes aiming for simultaneous interaction. PoT is primarily designed for (long term) asynchronous data transfer.
- 4. PoT is not supposed and not able to reduce publishers fear of cloned replicas from open code due to SCs or unencrypted shared data for re-engineering. The incentive to save maintenance costs on servers have to suffice to cover re-engineering risks. Additionally, not updated clients, assuming different parameters, may lead to incompatibilities. Changing rules (SCs or parameters) after a PoT BC has been initialized is not recommended. Concluding: Once a software using PoT is released, the advantage from PoT are only the reduced network/server costs.

5. Less about PoT and more about games on BCT, distributed systems without fallback option are critical for long term usage. This can be obtained both from precise timestamps without helper nodes and shared decks of card. For the latter, private draws without helping nodes are already cumbersome. Hidden draws are not even possible without external help.

4.11 Interim Summary - Proof-of-Turn

The implications of PoT are promising. Chosen the right application's use case, PoT offers an intelligible framework for fast lightweight and fair distributed gaming experience. It outperforms PoW in terms of speed, whilst still managing to grant equal rights among the peers in comparison to PoS and PoA. There is no need for special Hardware as required for PoET and no semi-fair additional measure for 'effort' has to be determined as in PoP. Compared to the approaches *Multichain* and *Aura*, freeze-states can be prevented due to granted time frames for turns and the transition time slot between two turns.

Still, PoT meets a high BFT, adjustable to each use case needs, whilst offering fast CF reducing forks and computation in vain. Last, PoT can be seen as open for interoperability as it can be used as a sub-chain as well as it can use other chains below itself, likewise to Kraft (2016)'s *Game channels*. Summing up, PoT delivers many desirable features for its intended niche.

Therefore, after researching and evaluating different CMs, the second leading question ..

"What is the best fitting BCT CM to cover an asynchronous game play scenario?"

.. may be answered with "PoT" - depending on the application's use case. Additionally, by shifting the computation requirement from the publisher's central server to the peers nodes, the research hypothesis ..

"The PoT CM leads to a reduction of server running costs for game publishers."

.. appears to be admissible.

Finally, after **first** analyzing recent BCT CMs, **second** researching games, which use BCs in their ecosystem and **third** presenting a novel CM, PoT, to complement the existing solutions, a conclusion and outlook is given.

5

Chapter

Conclusion & Outlook

To answer the primary research question ..

"Can BCT be used to reduce publisher's server costs whilst providing (mobile) players a suitable gaming experience?"

.. the two leading questions have been discussed and answered within chapter Blockchain Technology and chapter Proof-of-Turn. Nevertheless, the primary research question cannot be answered within a straight boolean value spectrum.

From a purely algorithmic perspective, there is a solution to most challenges (e.g. Card draw). But that there is a viable algorithm is only part of the answer. This thesis brings to mind that every additional required algorithm is a trade-off: What is gained in pure feasibility on one side, is payed in additional development effort, higher complexity (and lower maintainability), lower system performance (and higher transaction costs). The risk of faulty technicality decreases, but other risks increase in return. In the case of games, dropping out players due to limited chances to win would be only one of the problematic cases. Especially the higher complexity should be evaluated throughout, before the need of cumbersome fixes arise.

Therefore, whilst CMs may function unexpectedly good in a theoretical document, the final 'design of upper level (gaming) software', 'additional programming effort' as well as 'cooperation and coordination of gamers' (e.g. players do only regularly quit game sessions) are major factors which have to be considered to answer the research question throughout.

This additional research would offer further insights to adjust and verify first the given working assumptions and second the plots containing (only) sample data. Hence, sample games have to be invented, implemented and analyzed throughout. Unluckily, the concomitant workload would have exceeded the scope of this document by far.

Nevertheless, the tight budgets in the gaming industry call for mechanisms to improve their games backend and PoT offers a solution for slow paced games promising low (networking) costs. Thus, finding sample applications conducting PoT is assumed not to be too far fetched, as described in the Outlook.

5 Conclusion & Outlook 87

5.1 Outlook

The outlook focuses on the PoT CM, but abstracts from the use case of games. Hence, research in other possible application's use cases with PoT and 'Adaption in the Wild' is aimed for:

1. A truly distributed chat

Given an initial interconnection, every pair of devices could establish a BC for their chat protocol. To the writer's knowledge, every recent messenger relies on central server technology. Still the most messengers lack funding. If users had an especially high need for privacy wherein only their devices had encryption keys, they could use a messenger build upon BCT using a lightweight CM as PoT. Interconnection could still suffer speed as the other node peer might be offline, but speed up turns, down to ~ten seconds seem possible. During inactivity, dynamic turn time may be used as well. Here, if turn time was set to long (sleeping chat), a direct message to the LN could trigger its turn's end and decrease transaction delivery times. Last, it would be reasonable to sacrifice instantaneous delivery for an increased level of security, as non existing central servers can not loose their integrity.

2. Cryptocurrencies and parliamentary elections

The ability to rewrite the chain, grounded on any network vote, makes PoT futile both for cryptocurrencies and parliamentary elections. Still, e.g. the possibility of flooding attacks and the setting in *permissioned networks* limits the practicality in this regard. Therefore, it is discouraged to use PoT in such critical infrastructures in general.

3. The citizens trust

Awarding of public administration contracts is mostly conducted using a public administration's central server. In this scenario, each company's offering is seen mutually exclusive to the others and all together create a race condition. Moreover, knowing the bids of other companies creates unfair advantage to the informed party - especially if the informed party still has to claim its price tag. Not only does the *offset revelation* mechanism (independent from BCT) promise a fair and transparent match making in the market. PoT offers here a lightweight (computation) and balanced (writing permission) distributed CM. Hence, PoT may help to lighten opaque administrative structures and raise the stakeholders' trust.

4. Responsible disclosure

Just found vulnerabilities in software systems, formally known as zero-day vulnerabilities as well as startling information gained from whistle-blowing activity endanger not only the targeted institution(s), but also the investigative journalist(s) and security researcher(s). The publishing party (person) is not dependent on any server to stay online or has to fear that the central server is taken down from any higher authority. Still *offset revelation* can be used to inform affected parties first, formally known as 'responsible disclosure'. If implemented accordingly, the publishing party may stay anonymous and gains additionally the ability to publish the encrypted data from a different node than the affiliated key(s). Here, if nodes stay offline the PoT algorithm may be adjusted to choose the next node according to

5 Conclusion & Outlook 88

online activity. Inactive nodes are - after a network vote - skipped. Hence, the publishing party is enabled to stay offline, until the right time to publish has come. This scenario, again, shows the adaptability of PoT. Last, as PoT assumes that nodes write their date themselves and nodes do not compete for tokens gained from mining, the possibility is high for the publishing party to release the data without intermediaries. Additionally, as some critical acts are sometimes discovered, not whilst conducting, but during the phaseout, transferring cryptocurrency tokens into traditional currencies was a risk for the publishing party. Therefore, the absence of cryptocurrency tokens in PoT protects publishing parties by design.

Next to PoT, this thesis gives some hints on future research fields like 'finding an optimal child-chain size' for different use cases.

All these and other scenarios call for further research and 'Adaption in the Wild'.



Appendix Appendix

The appendix offers data used throughout the document in further detail.

A.1 Data regarding gaming devices

First, the data relevant for used devices from (LIMELIGHT NETWORKS 2020, p. 7) is shown in relative measures (Figure A.1).

Second, a chart from 'Counterpoint Research' by WANG (2021) is given (Figure A.2), which supports the claim that "Average Smartphone NAND Flash Capacity Crossed 100GB in 2020".

Third, the data relevant for game characteristics from (LIMELIGHT NETWORKS 2020, p. 18) is shown in relative measures (Figure A.3).

Country	Computer	Tablet	Gaming console	Mobile phone
France	26%	17%	24%	32%
Germany	29%	16%	24%	31%
India	26%	17%	20%	37%
Italy	29%	15%	24%	32%
Japan	19%	16%	17%	48%
South Korea	29%	16%	15%	40%
Singapore	26%	18%	17%	40%
United Kingdom	25%	19%	27%	30%
United States	25%	18%	24%	33%
Global	26%	17%	21%	36%

Figure A.1: Share of devices by gaming time (LIMELIGHT NETWORKS 2020, p. 7)

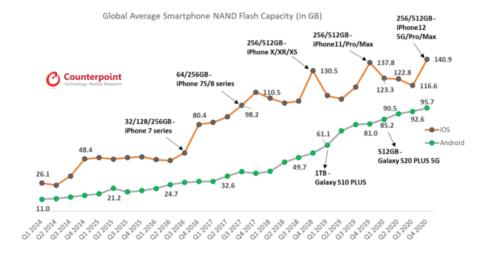


Figure A.2: Average smartphone storage space (From Wang (2021))

	Interesting storyline	Offline Activity	Fast performance	Multiplayer	Simple gameplay
France	21%	20%	21%	17%	21%
Germany	21%	20%	21%	17%	21%
India	20%	19%	21%	19%	21%
Italy	20%	21%	21%	18%	20%
Japan	22%	21%	22%	13%	23%
SouthKorea	20%	19%	22%	18%	21%
Singapore	19%	21%	22%	16%	21%
U.K.	20%	21%	22%	14%	23%
U.S.	20%	21%	22%	15%	22%
Global	20%	20%	22%	16%	21%

Figure A.3: Game characteristics data (LIMELIGHT NETWORKS 2020, p. 18)

A.2 Scripts for graph's plots

A.2.1 Plot: BFT shuffling cards

Python code for figure 3.12, 'BFT shuffling cards':

```
# Imports
import matplotlib
from matplotlib import pyplot as plt
from matplotlib.ticker import FuncFormatter
import numpy as np
# Output dimensions
plt.figure(figsize=(21*0.36,9*0.36))
# Case 1: n nodes and groups of 1 .. Y = 1
def f1(n):
    return n/n
# Case 2: n nodes and Groups of two nodes
def f2(n):
    return (n/2)/n
# Case 3: n nodes and Groups of three nodes
def f3(n):
    return (n/3)/n
# Case 5: n nodes and three groups
def f5(n):
    return 3/n
# Case 6: n nodes and four groups
def f6(n):
    return 4/n
# Case 7: n nodes and four groups
def f7(n):
    return 5/n
# Plot Ranges
rnge1 = np.arange(2.9, 54.01, 0.1)
rnge3 = np.arange(3, 55, 1)
rnge4 = np.arange(4, 55, 1)
rnge5 = np.arange(5, 55, 1)
rnge6 = np.arange(6, 55, 1)
rnge9 = np.arange(9, 55, 1)
# Plots
plt.plot(rnge3, f5(rnge3), '--', linewidth=2, label="\u03B1
```

```
: Three groups")
plt.plot(rnge4, f6(rnge4), '--', linewidth=2, label="\u03B2
  : Four groups")
plt.plot(rnge5, f7(rnge5), '--', linewidth=2, label="\u03B3
   : Five groups")
plt.plot(rnge1, f1(rnge1), 'tab:red', linewidth=2, label="\
  u03B4: Full mistrust")
plt.plot(rnge6, f2(rnge6), 'tab:brown', linewidth=2, label
  ="\u03B5: Groups of (~)two nodes")
plt.plot(rnge9, f3(rnge9), 'tab:olive', linewidth=2, label=
  "\u03B8: Groups of (~)three nodes")
# Axis & Labels
plt.xlabel('Network nodes')
plt.ylabel('Hostile nodes')
def to_percent(y, position):
    s = str(round(100*y,1))
    return s + '%'
formatter = FuncFormatter(to_percent)
plt.gca().yaxis.set_major_formatter(formatter)
# Finalize
plt.grid(b=True, which='major', color='#aaaaaa',
        linestyle='-')
plt.minorticks_on()
plt.grid(b=True, which='minor', color='#aaaaaa',
        linestyle='-.', alpha=0.2)
plt.legend(bbox_to_anchor=(0.575, 0.9), loc='upper left')
plt.tight_layout()
# Output
plt.show()
```

A.2.2 Plot: General storage allocation

Python code for figure 3.17, 'General BT storage allocation chart':

```
# Imports
from matplotlib import pyplot as plt
# Dimensions
plt.figure(figsize=(21*0.36,9*0.36))
# Data for 'Linear'
LINx = [0, 1300]
LINy = [0,1300]
# Data for 'Relevant Blocks'
RBx = [50,650,1300]
RBy = [0,300,625]
# Plotting
plt.plot(LINx, LINy, '--k')
plt.plot(RBx, RBy, '-.r')
# Nameing
plt.xlabel('Transactions (K)')
plt.ylabel('Megabyte')
plt.legend(["BC without optimization",
        "Historic relevant transactions"])
# Finalize
plt.grid(b=True, which='major', color='#dddddd',
        linestyle='-.', linewidth=0.4)
plt.tight_layout()
# Output
plt.show()
```

A.2.3 Plot: Storage allocation Prune procedure/Child-chain

Python code for figure 3.19, 'Prune procedure/Child-chain chart':

```
# Imports
from matplotlib import pyplot as plt
# Dimensions
plt.figure(figsize=(21*0.36,9*0.36))
# Data for 'Linear'
LINx = [0, 1300]
LINy = [0,1300]
# Data for 'Relevant Blocks'
RBx = [50,650,1300]
RBy = [0,300,625]
# Data for 'Prune Procedure/Sidechains'
SCx = [4, 150, 150, 300, 300, 450, 450, 600, 600, 750, 750,
        900,900,1050,1050,1200,1200,1300]
SCy = [0, 146, 75, 225, 150, 300, 225, 375, 300, 450, 375,
        525,450,600,525,675,600,700]
# Plotting
plt.plot(LINx, LINy, '-.k')
plt.plot(RBx, RBy, '--r')
plt.plot(SCx, SCy, 'C0')
# Nameing
plt.xlabel('Transactions (K)')
plt.ylabel('Megabyte')
plt.legend(["BC without optimization",
        "Historic relevant transactions",
        "Prune Procedure/Child-chains", ])
# Finalize
plt.grid(b=True, which='major', color='#dddddd',
        linestyle='-.', linewidth=0.4)
plt.tight_layout()
# Output
plt.show()
```

A.2.4 Plot: Storage allocation Meta State Block

Python code for figure 3.22, 'Meta-State Block chart':

```
# Imports
from matplotlib import pyplot as plt
# Dimensions
plt.figure(figsize=(21*0.36,9*0.36))
# Data for 'Linear'
LINx = [0, 1300]
LINy = [0,1300]
# Data for 'Relevant Blocks'
RBx = [50,650,1300]
RBy = [0,300,625]
# Data for 'Meta-State Block'
MSx = [4,200,200,390,390,580,580,770,770,
        960,960,1150,1150,1300]
MSy = [0,198, 75,265,150,340,183,373,205.5,
        395.5,220.3,410.3,230.2,380.2]
# Plotting
plt.plot(LINx, LINy, '-.k')
plt.plot(RBx, RBy, '--r')
plt.plot(MSx, MSy, 'C2')
# Nameing
plt.xlabel('Transactions (K)')
plt.ylabel('Megabyte')
plt.legend(["BC without optimization",
        "Historic relevant transactions",
        "Meta-State Block"])
# Finalize
plt.grid(b=True, which='major', color='#dddddd',
        linestyle='-.', linewidth=0.4)
plt.tight_layout()
# Output
plt.show()
```

A.2.5 Plot: Storage allocation comparison

Python code for figure 3.23, 'Storage allocation comparison':

```
# Imports
from matplotlib import pyplot as plt
# Dimensions
plt.figure(figsize=(21*0.36,9*0.36))
# Data for 'Linear'
LINx = [0,2050]
LINy = [0, 2050]
# Data for 'Relevant Blocks'
RBx = [50,650,1300,2050]
RBy = [0,300,625,1000]
# Data for 'Prune Procedure/Sidechains'
SCx = [4,150,150,300,300,450,450,600,600,750,750,
        900,900,1050,1050,1200,1200,1350,1350,1500,1500,
        1650, 1650, 1800, 1800, 1950, 1950, 2050]
SCy = [0, 146, 75, 225, 150, 300, 225, 375, 300, 450, 375,
        525,450,600,525,675,600,750,675,825,750,900,
        825,975,900,1050,975,1075]
# Data for 'Meta-State Block'
MSx = [4,200,200,390,390,580,580,770,770,960,960,1150,
1150, 1340, 1340, 1530, 1530, 1720, 1720, 1910, 1910, 2000, 2050]
MSy = [0, 198, 75, 265, 150, 340, 183.3, 373, 205, 395, 220, 410,
230,420.2,236,426.8,241.2,431,244.1,434,246.1,336,386.1]
# Plotting
plt.plot(LINx, LINy, '-.k')
plt.plot(RBx, RBy, '--r')
                   'C0')
plt.plot(SCx, SCy,
plt.plot(MSx, MSy, 'C2')
# Nameing
plt.xlabel('Transactions (K)')
plt.ylabel('Megabyte')
plt.legend(["BC without optimization",
        "Historic relevant transactions",
        "Prune Procedure/Child-chains",
        "Meta-State Block"])
# Finalize
plt.grid(b=True, which='major', color='#dddddd',
        linestyle='-.', linewidth=0.4)
plt.tight_layout()
# Output
```

plt.show()

- **Agosta JM, Crosby S (2003)** *Network integrity by inference in distributed systems*, 2003. https://www.researchgate.net/publication/228786554 (Accessed 11th June 2021)
- Ateniese G, Magri B, Venturi D, Andrade E (2017) Redactable Blockchain: or Rewriting History in Bitcoin and Friends. In: 2017 IEEE European Symposium on Security and Privacy pp.111–126. https://doi.org/10.1109/EuroSP.2017.37
- Back A, Corallo M, Dashir L, Friedenbach M, Maxwell G, Miller A, Poelstra A, Timon J, Wuille P (2014) Enabling Blockchain Innovations with Pegged Sidechains, 2014. https://blockstream.com/sidechains.pdf (Accessed 11th June 2021)
- Bergsma M, Spronck P (2008) Adaptive Spatial Reasoning for Turn-based Strategy Games. In: Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference 2008:161–166. https://www.aaai.org/Papers/AIIDE/2008/AIIDE08-027.pdf (Accessed 6th June 2021)
- **Besancon L, Da Silva CF, Ghodous P (2019)** *Towards Blockchain Interoperability: Improving Video Games Data Exchange.* In: 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). IEEE, pp.81–85. doi:10.1109/BLOC.2019.8751347
- **Bethke E (2003)** *Game development and production* (Wordware game developer's library.). Wordware Pub, Plano Tex. ISBN:1556229518
- Bodkhe U, Tanwar S, Parekh K, Khanpara P, Tyagi S, Kumar N, Alazab M (2020) Blockchain for Industry 4.0: A Comprehensive Review. In: IEEE Access 8:79764–79800. doi:10.1109/ACCESS.2020.2988579
- **Brewer E (2012)** *CAP Twelve Years Later: How the "Rules" Have Changed.* In: IEEE Computer Society 45(2):23–29. doi:10.1109/MC.2012.37
- **Buffa ES (1976)** *Towards a Typology of Production and Operations Management Systems.* In: Wiley, NewYork 8(3). doi:10.5465/amr.1983.4284369
- **Butijn BJ, Tamburri DA, van den Heuvel WJ (2020)** *Blockchains: A Systematic Multivocal Literature Review.* In: ACM Computing Surveys 53(3):1–37. doi:10.1145/3369052
- **Bux W (1989)** *Token-Ring Local-Area Networks and Their Performance.* In: Proceedings of the IEEE 1989(77):238–256. doi:10.1109/5.18625
- **By T (2011)** *Formalizing Game-play.* In: Simulation & Gaming 2011:1–31. doi:10.1177/1046878110388239

Castro M, Liskov B (2002) *Practical Byzantine Fault Tolerance and Proactive Recovery.* In: ACM Transactions on Computer Systems 20(4):398–461. doi:10.1145/571637.571640

- Catalini C, Gans JS (2016) Some Simple Economics of the Blockchain. In: NATIONAL BUREAU OF ECONOMIC RESEARCH 2016:1–28. doi:10.3386/w22952
- **Charron-Bost B, Schiper A (2009)** *The Heard-Of model: computing in distributed systems with benign faults.* In: Distributed Computing 22(1):49–71. doi:10.1007/s00446-009-0084-6
- Chatterjee K, Goharshady AK, Pourdamghani A (2019) Probabilistic Smart Contracts: Secure Randomness on the Blockchain. In: 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). IEEE, pp.403–412. doi:10.1109/BLOC.2019.8751326
- Chaudhry N, Yousaf MM (2018) Consensus Algorithms in Blockchain: Comparative Analysis, Challenges and Opportunities. In: 2018 12th International Conference on Open Source Systems and Technologies (ICOSST). IEEE, pp.54–63. doi:10.1109/ICOSST.2018.8632190
- **Courtois NT (2014)** On The Longest Chain Rule and Programmed Self-Destruction of Crypto Currencies, 2014. http://arxiv.org/pdf/1405.0534v11 (Accessed 11th June 2021)
- **CryptoKitties.co (2021)** *Collectibles game store: On Sale*, 2021. https://www.cryptokitties.co/search?include=sale (Accessed 11th June 2021)
- **Daniel F, Guida L (2019)** *A Service-Oriented Perspective on Blockchain Smart Contracts.* In: IEEE Internet Computing 23(1):46–53. doi:10.1109/MIC.2018.2890624
- Danzi P, Kalor AE, Stefanovic C, Popovski P (52018) Analysis of the Communication Traffic for Blockchain Synchronization of IoT Devices. In: 2018 IEEE International Conference on Communications (ICC). IEEE, pp.1–7. doi:10.1109/ICC.2018.8422485
- **Davis R, Lang B (2012)** *Modeling game usage, purchase behavior and ease of use.* In: Entertainment Computing 3(2):27–36. doi:10.1016/j.entcom.2011.11.001
- de Angelis S, Aniello L, Baldoni R, Lombardi F, Margheri A., Sassone V (2018) PBFT vs Proof-of-Authority: Applying the CAP Theorem to Permissioned Blockchain, 2018. https://eprints.soton.ac.uk/415083/2/itasec18_main.pdf (Accessed 11th June 2021)
- **decentraland.org (2021)** *Decentraland*, 2021. https://decentraland.org/ (Accessed 11th June 2021)
- **Demi S, Colomo-Palacios R, Sánchez-Gordón M (2021)** *Software Engineering Applications Enabled by Blockchain Technology: A Systematic Mapping Study.* In: Applied Sciences 11(7): 2960. doi:10.3390/app11072960
- **Deswarte Y, Quisquater JJ, Saïdane A (2004)** Remote Integrity Checking. In: Jajodia S, Strous L (Eds.) Integrity and Internal Control in Information Systems VI (IFIP International

Federation for Information Processing 140), pp.1–11. Kluwer Academic Publishers, Boston. doi:10.1007/1-4020-7901-X 1

- **Dib O, Brousmiche KL, Durand A, Thea E, Hamida EB (2018)** *Consortium Blockchains: Overview, Applications and Challenges.* In: International Journal on Advances in Telecommunications 2018:51–64. https://www.researchgate.net/publication/328887130 (Accessed 11th June 2021)
- **Douceur JR (2002)** *The Sybil Attack*. In: International workshop on peer-to-peer systems 2002: 251–260. doi:10.1007/3-540-45748-8 24
- **Ergen M, Lee D, Sengupta R, Varaiya P (2004)** *WTRP—Wireless Token Ring Protocol.* In: IEEE Transactions on Vehicular Technology 53(6):1863–1881. doi:10.1109/TVT.2004.836928
- **Ewerhart C (2020)** *Finite blockchain games.* In: Economics Letters 197:109614. doi:10.1016/j.econlet.2020.109614
- Finlow-Bates K. (2017) Adding Trust to CAP: Blockchain as a Strong Eventual Consistency Recovery Strategy, 2017. https://www.chainfrog.com/wp-content/uploads/2017/09/CAP-paper.pdf (Accessed 11th June 2021)
- **Forouzan BA (2003)** *TCP/IP Protocol Suite.* McGraw Hill. https://dl.acm.org/doi/abs/10.5555/572565
- **Gainsbury SM, Blaszczynski A (2017)** HOW BLOCKCHAIN AND CRYPTOCURRENCY TECHNOLOGY COULD REVOLUTIONIZE ONLINE GAMBLING. In: Gaming Law Review 21(7): 482–492. doi:10.1089/glr2.2017.2174
- **Gamesetwatch.com (2014)** *Gamasutra Salary Survey 2014*, 2014. http://www.gamesetwatch.com/2014/09/05/GAMA14_ACG_SalarySurvey_F.pdf (Accessed 11th June 2021)
- **Goebel J, Krzesinski AE (2017)** *Increased block size and Bitcoin blockchain dynamics.* IEEE, Piscataway. doi:10.1109/ATNAC.2017.8215367
- **Gramoli V (2017)** *From blockchain consensus back to Byzantine consensus.* In: Future Generation Computer Systems 107:760–769. doi:10.1016/j.future.2017.09.023
- Greenspan G (2015) MultiChain Private Blockchain: White Paper, 2015. https://www.multichain.com/download/MultiChain-White-Paper.pdf (Accessed 11th June 2021)
- Haeberlen A, Aditya P, Rodrigues R, Druschel P (2010) Accountable Virtual Machines. In: In Proceedings of the 9th Symposium on Operating Systems Design and Implementation 2010. https://www.usenix.org/legacy/event/osdi10/tech/full_papers/Haeberlen.pdf

Hagelbäck J, Johansson SJ (2008) Dealing with Fog of War in a Real Time Strategy Game Environment. In: 2008 IEEE Symposium on Computational Intelligence and Games (CIG'08) pp.55–62. doi:10.1109/CIG.2008.5035621

- Harkanson R, Chiu C, Kim Y, Jo JY (2020) A Framework for Decentralized Private Random State Generation and Maintenance for Multiplayer Gaming Over Blockchain. In: 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC). IEEE, pp.483–488. doi:10.1109/COMPSAC48688.2020.0-205
- **Hasbro (2021)** *Risiko*, 2021. https://products.hasbro.com/de-de/product/risk-game: 2C7C6F52-5056-9047-F5DD-EB8AC273BA4C (Accessed 11th June 2021)
- **Hu YC, Jakobsson M, Perrig A (2005)** *Efficient Constructions for One-Way Hash Chains.* In: Ioannidis J, Keromytis A, Yung M (Eds.) Applied cryptography and network security, pp.423–441. Springer, Berlin. ISBN:3540262237. https://link.springer.com/chapter/10.1007/11496137 29
- **itch.io (2021)** *Top games for Android with server-based network multiplayer*, 2021. https://itch.io/games/multiplayer-server/platform-android (Accessed 11th June 2021)
- Johnson B, Laszka A, Grossklags J, Vasek M, Moore T (2014) *Game-Theoretic Analysis of DDoS Attacks Against Bitcoin Mining Pools*. In: Böhme R, Brenner M, Moore T, Smith M (Eds.) Financial Cryptography and Data Security, pp.72–85. Springer Berlin Heidelberg, Berlin. ISBN:978-3-662-44773-4. https://link.springer.com/chapter/10.1007/978-3-662-44774-1_6
- Karafiloski E, Mishev A (2017) Blockchain Solutions for Big Data Challenges. In: Karadzinov L (Ed.) IEEE EUROCON 2017, pp.763–768. IEEE, Piscataway. doi:10.1109/EUROCON.2017.8011213
- Kejiao L, Hui L, Hanxu H, Kedan L, Yongle C (2017) Proof of Vote: A High-Performance Consensus Protocol Based on Vote Mechanism & Consortium Blockchain. In: 2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems 2017:466–473. doi:10.1109/HPCC-SmartCity-DSS.2017.61
- Khan D, Jung LT, Hashmani MA, Waqas A (2020) A Critical Review of Blockchain Consensus Model. In: 2020 3rd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET). pp.1–6. doi:10.1109/iCoMET48670.2020.9074107
- Khan N (2006) A Distributed Server Architecture for Massively Multiplayer Online Games, 2006. https://central.bac-lac.gc.ca/.item?id=TC-QMM-101150&op=pdf&app=Library&oclc_number=892079085 (Accessed 11th June 2021)
- Kim S, Kwon Y, Cho S (2018) A Survey of Scalability Solutions on Blockchain. In: 2018 International Conference on Information and Communication Technology Convergence (ICTC) pp.1204–1207. doi:10.1109/ICTC.2018.8539529

King S, Nadal S (2012) PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake, 2012. https://www.chainwhy.com/upload/default/20180619/126a057fef926dc286accb372da46955.pdf (Accessed 11th June 2021)

- Komiya K, Nakajima T (2019) Increasing Motivation for Playing Blockchain Games Using Proof-of-Achievement Algorithm. In: Fang X. (Ed.) Increasing Motivation for Playing Blockchain Games Using Proof-of-Achievement Algorithm, pp.125–140. Springer International Publishing, Cham. ISBN:978-3-030-22601-5. https://link.springer.com/content/pdf/10.1007%2F978-3-030-22602-2 11.pdf
- **Kooistra J. (2018)** *The State of the Battle Royale Genre*, 2018. https://resources.newzoo.com/hubfs/Reports/Newzoo_The_Rise_of_the_Battle_Royale_Genre.pdf (Accessed 11th June 2021)
- **Koster R (2018)** *The cost of games*, 2018. https://www.raphkoster.com/2018/01/17/the-cost-of-games/ (Accessed 11th June 2021)
- **Kraft D (2016)** *Game Channels for Trustless Off-Chain Interactions in Decentralized Virtual Worlds.* In: Ledger 1:84–98. doi:10.5195/ledger.2016.15
- **Kwon J (2014)** *Tendermint: Consensus without Mining*, 2014. https://www.weusecoins.com/assets/pdf/library/Tendermint%20Consensus%20without%20Mining.pdf (Accessed 11th June 2021)
- Laneve C, Ershov I (2019) Blockchain gaming: An analysis of the use of blockchain technology in the video gaming industry, 2019. https://amslaurea.unibo.it/20533/1/ershov_igor_tesi.pdf (Accessed 11th June 2021)
- **Lee D, Lin D, Bezemer CP, Hassan AE (2020)** Building the perfect game an empirical study of game modifications. In: Empirical Software Engineering 25(4):2485–2518. doi:10.1007/s10664-019-09783-w
- Lee Y, Agarwal S, Butcher C, Padhye J (2008) Measurement and Estimation of Network QoS Among Peer Xbox 360 Game Players. In: Claypool M, Uhlig S (Eds.) Passive and Active Network Measurement (Lecture Notes in Computer Science 4979), pp.41–50. Springer Berlin Heidelberg, Berlin. doi:10.1007/978-3-540-79232-1_5
- **Li M, Goh KY, Cavusoglu H, Jindal N (2014)** WHAT IS IT IN A NAME? AN EMPIRICAL EXAMINATION OF BRAND IMITATION IN MOBILE APP MARKET. In: Pacific Asia Journal of the Association for Information Systems 2014. http://www.pacis-net.org/file/2014/2121.pdf
- **Limelight Networks (2020)** *The State of Online Gaming 2020: Market Research*, 2020. https://www.limelight.com/resources/white-paper/state-of-online-gaming-2020 (Accessed 11th June 2021)
- **Lin IC, Liao TC (2017)** *A Survey of Blockchain Security Issues and Challenges.* In: International Journal of Network Security, 2017(5):653–659. doi:10.6633/IJNS.201709.19(5).01

Liu M, Yu FR, Teng Y, Leung VCM, Song M (2018) Computation Offloading and Content Caching in Wireless Blockchain Networks With Mobile Edge Computing. In: IEEE Transactions on Vehicular Technology 67(11):11008–11021. doi:10.1109/TVT.2018.2866365

- lostrelics.io (2021) Lost Relics, 2021. https://lostrelics.io/ (Accessed 11th June 2021)
- **Lowell DE, Saito Y, Samberg EJ (2004)** *Devirtualizable Virtual Machines: Enabling General, Single-Node, Online Maintenance.* In: ASPLOS XI, pp.211–223. Association for Computing Machinery, New York N.Y. doi:10.1145/1037947.1024419
- Ma G, Ge C, Zhou L (2020) Achieving reliable timestamp in the bitcoin platform. In: Peer-to-Peer Networking and Applications 13(6):2251–2259. doi:10.1007/s12083-020-00905-6
- magic.wizards.com (2021) Magic: The Gathering Arena, 2021. https://magic.wizards.com/en/mtgarena (Accessed 11th June 2021)
- Maister DH (1984) The Psychology of Waiting Lines, 1984. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.487.3938&rep=rep1&type=pdf (Accessed 11th June 2021)
- Mann CC (2000) The End of Moore's Law?, 2000. https://www.technologyreview.com/2000/05/01/236362/the-end-of-moores-law/ (Accessed 11th June 2021)
- McGraw R, Imsand E, Chinni MJ (Eds.) (2010) Proceedings of the 2010 Spring Simulation Multiconference on SpringSim '10. ACM Press, New York. doi:10.1145/1878537
- Miller A, Bentov I, Kumaresan R, McCorry P (2017) Sprites: Payment Channels that Go Faster than Lightning, 2017. http://arxiv.org/pdf/1702.05812v2 (Accessed 11th June 2021)
- Min T, Wang H, Guo Y, Cai W (2019) *Blockchain Games: A Survey.* In: 2019 IEEE Conference on Games (CoG). IEEE, pp.1–8. doi:10.1109/CIG.2019.8848111
- Min T, Cai W (2019) A Security Case Study for Blockchain Games. In: 2019 IEEE Games, Entertainment, Media Conference (GEM). IEEE, pp.1–8. doi:10.1109/GEM.2019.8811555
- Morris R, Thompson K (1979) Password Security: A Case History. In: Communications of the ACM 22(11):594–597. doi:10.1145/359168.359172
- Morris S (2003) WADs, Bots and Mods: Multiplayer FPS Games as Co-creative Media, 2003. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.190.3545&rep=rep1&type=pdf (Accessed 11th June 2021)
- Nagygyörgy K, Urbán R, Farkas J, Griffiths MD, Zilahy D, Kökönyei G, Mervó B, Reindl A, Ágoston C, Kertész A, Harmath E, Oláh A, Demetrovics Z (2013) Typology and Sociodemographic Characteristics of Massively Multiplayer Online Game Players.

In: International Journal of Human-Computer Interaction 29(3):192–200. doi:10.1080/10447318.2012.702636

- Nakamoto S (2009) Bitcoin_A Peer-to-Peer Electronic Cash System, 2009. https://www.bitcoin.org/bitcoin.pdf (Accessed 11th June 2021)
- Narayanan A (2018) Written Testimony of Arvind Narayanan, 2018. https://www.energy.senate.gov/services/files/8A1CECD1-157C-45D4-A1AB-B894E913737D (Accessed 11th June 2021)
- Netzer R. H. B., Miller B. P. (1992) What are race conditions? Some Issues and Formalizations. In: ACM Letters on Programming Languages and Systems 1(1):74–88. doi:10.1145/130616.130623
- **Nguyen G.-T.**, **Kim K (2018)** *A Survey about Consensus Algorithms Used in Blockchain.* In: Journal of Information Processing Systems 2018(14):101–128. doi:10.3745/JIPS.01.0024
- Oliveira MT, Carrara GR, Fernandes NC, Albuquerque CVN, Carrano RC, Medeiros DSV, Mattos DMF (2019) *Towards a Performance Evaluation of Private Blockchain Frameworks using a Realistic Workload.* In: 22nd Conference on Innovation in Clouds, Internet and Networks (ICIN 2019) 2019:180–187. doi:10.1109/ICIN.2019.8685888
- **Palmer JW (2002)** *Web Site Usability, Design, and Performance Metrics.* In: Information Systems Research 13(2):151–167. doi:10.1287/isre.13.2.151.88
- Pfeiffer A, Kriglstein S, Wernbacher T (2020) Blockchain Technologies and Games: A Proper Match? In: Yannakakis GN, Liapis A, Kyburz P, Volz V, Khosmood F, Lopes P (Eds.) International Conference on the Foundations of Digital Games. ACM, New York, pp.1–4. doi:10.1145/3402942.3402996
- **playhearthstone.com (2021)** *Hearthstone*, 2021. https://playhearthstone.com/ (Accessed 11th June 2021)
- **Poon J, Dryja T (2016)** *The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments*, 2016. https://lightning.network/lightning-network-paper.pdf (Accessed 11th June 2021)
- **Prugsamatz S, Lowe B, Alpert F (2010)** *Modelling consumer entertainment software choice: An exploratory examination of key attributes, and differences by gamer segment.* In: Journal of Consumer Behaviour 9(5):381–392. doi:10.1002/cb.325
- Quizduell (2021) Quizduell, 2021. http://www.quizduell-game.de/ (Accessed 11th June 2021)
- **Rachmawati D, Tarigan JT, Ginting ABC (2017)** *A comparative study of Message Digest 5(MD5) and SHA256 algorithm.* In: Journal of Physics: Conference Series 978:1–6. doi:10.1088/1742-6596/978/1/012116

Ranchal-Pedrosa A, Gramoli V (2020) Blockchain Is Dead, Long Live Blockchain! Accountable State Machine Replication for Longlasting Blockchain, 2020. http://arxiv.org/pdf/2007.10541v2 (Accessed 11th June 2021)

- **Red Blob Games (2020)** *2d Visibility*, 2020. https://www.redblobgames.com/articles/visibility/ (Accessed 11th June 2021)
- **rockpapershotgun.com (2021)** *Server Shutdowns*, 2021. https://www.rockpapershotgun.com/topics/server-shutdowns (Accessed 11th June 2021)
- Rong K, Ren Q, Shi X (2018) The determinants of network effects: Evidence from online games business ecosystems. In: Technological Forecasting and Social Change 134:45–60. doi:10.1016/j.techfore.2018.05.007
- **Saleh F (2020)** *Blockchain Without Waste: Proof-of-Stake.* In: Review of Financial Studies, Forthcoming doi:10.2139/ssrn.3183935
- Serada A, Sihvonen T, Harviainen JT (2020) CryptoKitties and the New Ludic Economy: How Blockchain Introduces Value, Ownership, and Scarcity in Digital Gaming. In: Games and Culture p. 155541201989830. doi:10.1177/1555412019898305
- **Setear JK (1989)** *Simulating the Fog of War: Research report.* In: RAND Corporation https://apps.dtic.mil/sti/pdfs/ADA228112.pdf
- **Sharma P, Jindal R, Borah MD (2020)** *Blockchain Technology for Cloud Storage.* In: ACM Computing Surveys 53(4):89–121. doi:10.1145/3403954
- **Smythe C (1999)** *Iso 8802/5 token ring local-area networks.* In: ELECTRONICS & COMMUNICATION ENGINEERING JOURNAL 1999:195–207. doi:10.1049/ecej:19990406
- Socios.com (2021) Socios, 2021. https://www.socios.com/ (Accessed 11th June 2021)
- **Sotamaa O, Svelch J (2021)** *Game Production Studies.* Amsterdam University Press, NL Amsterdam. doi:10.5117/9789463725439
- **statista.com (2021a)** *Number of available apps in the Apple App Store from 2008 to 2020*, 2021a. https:
 - //www.statista.com/statistics/268251/number-of-apps-in-the-itunes-app-store-since-2008/ (Accessed 11th June 2021)
- statista.com (2021b) Tablet operating systems market share in the United States from 2016 to 2021, 2021b. https://www.statista.com/statistics/271293/market-share-held-by-tablet-os-us/(Accessed 11th June 2021)
- **STIGLITZ JE (1989)** *Principal and Agent.* In: Allocation, Information and Markets pp.241–253. https://doi.org/10.1007/978-1-349-20215-7_25

Sultan K., Ruhi U, Lakhani R (2018) Conceptualizing Blockchains: Characteristics & Applications, 2018. https://arxiv.org/abs/1806.03693 (Accessed 11th June 2021)

- **teamfortress.com (2021)** *Windows dedicated server: Setup*, 2021. https://wiki.teamfortress.com/wiki/Windows dedicated server (Accessed 11th June 2021)
- VALVe (2021) Dedicated Servers List: A list of dedicated servers available on Steam/SteamCMD., 2021. https://developer.valvesoftware.com/wiki/Dedicated_Servers_List (Accessed 11th June 2021)
- vice.com (2021) Hackers Steal Wealth of Data from Game Giant EA, 2021. https: //www.vice.com/en/article/wx5xpx/hackers-steal-data-electronic-arts-ea-fifa-source-code (Accessed 11th June 2021)
- **Vukolić M (2016)** The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication. In: Camenisch J, Kesdoğan D (Eds.) Open Problems in Network Security, pp.112–125,. Springer International Publishing, Cham. doi:10.1007/978-3-319-39028-4
- Wang B (2021) Average Smartphone NAND Flash Capacity Crossed 100GB in 2020, 2021. https://www.counterpointresearch.com/average-smartphone-nand-flash-capacity-crossed-100gb-2020/ (Accessed 11th June 2021)
- Wang S, Dinh TTA, Lin Q, Xie Z, Zhang M, Cai Q, Chen G, Fu W, Ooi BC, Ruan P (2018) ForkBase: An Efficient Storage Engine for Blockchain and Forkable Applications, 2018. http://arxiv.org/pdf/1802.04949v1 (Accessed 11th June 2021)
- Wang X, Kwon T, Choi Y, Chen M, Zhang Y (2012) Characterizing the Gaming Traffic of World of Warcraft: From Game Scenarios to Network Access Technologies. In: IEEE Network 2012. doi:10.1109/MNET.2012.6135853
- Weber I, Gramoli V, Ponomarev A, Staples M, Holz R, Tran AB, Rimba P (2017) *On Availability for Blockchain-Based Systems*. In: 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS). IEEE, pp.64–73. doi:10.1109/SRDS.2017.15
- Weilbacher M (2012) Dedicated Servers in Gears of War 3: Scaling to Millions of Players, 2012. https://www.gdcvault.com/play/1015337/Dedicated-Servers-In-Gears-of (Accessed 11th June 2021)
- Wikipedia (2021a) Battlefield (video game series) Wikipedia, 2021a. https://en.wikipedia.org/wiki/Battlefield_(video_game_series) (Accessed 6th June 2021)
- **Wikipedia (2021b)** *Category:Play-by-email video games*, 2021b. https://en.wikipedia.org/wiki/Category:Play-by-email_video_games (Accessed 9th June 2021)
- Wikipedia (2021c) Game server, 2021c. https://en.wikipedia.org/wiki/Game server

Wikipedia (2021d) M.U.L.E., 2021d. https://en.wikipedia.org/wiki/M.U.L.E. (Accessed 9th June 2021)

- Wikipedia (2021e) Star Wars: Battlefront, 2021e. https://en.wikipedia.org/wiki/Star Wars: Battlefront (Accessed 6th June 2021)
- Williams D, Yee N, Caplan SE (2008) Who plays, how much, and why? Debunking the stereotypical gamer profile. In: Journal of Computer-Mediated Communication 13(4):993–1018. doi:10.1111/j.1083-6101.2008.00428.x
- Wu F, Yuen HY, Chan HCB, Leung VCM, Cai W (2020) Infinity Battle: A Glance at How Blockchain Techniques Serve in a Serverless Gaming System. In: Wen Chen C, Cucchiara R, Hua XS, Qi GJ, Ricci E, Zhang Z, Zimmermann R (Eds.) Proceedings of the 28th ACM International Conference on Multimedia. ACM, New York, pp.4559–4561. doi:10.1145/3394171.3414458
- Wu Q, Shiva S, Roy S, Ellis C, Datla, Vivek, D. (2010) On modeling and simulation of game theory-based defense mechanisms against DoS and DDoS attacks. In: McGraw R, Imsand E, Chinni MJ (Eds.) Proceedings of the 2010 Spring Simulation Multiconference on SpringSim '10. ACM Press, New York, p. 1. doi:10.1145/1878537.1878703
- xaya.io (2021) Huntercoin, 2021. https://xaya.io/huntercoin-legacy/ (Accessed 9th June 2021)
- Xu J, Wang S, Bhargava BK, Yang F (2019) A Blockchain-Enabled Trustless Crowd-Intelligence Ecosystem on Mobile Edge Computing. In: IEEE Transactions on Industrial Informatics 15(6): 3538–3547. doi:10.1109/TII.2019.2896965
- **Yan J, Randell B (2009)** *An Investigation of Cheating in Online Games.* In: IEEE Security & Privacy Magazine 7(3):37–44. doi:10.1109/MSP.2009.60
- Yuen HY, Wu F, Cai W, Chan HC, Yan Q, Leung VC (2019) Proof-of-Play: A Novel Consensus Model for Blockchain-based Peer-to-Peer Gaming System. In: Gai K, Raymond Choo KK, He D, Wen S (Eds.) Proceedings of the 2019 ACM International Symposium on Blockchain and Secure Critical Infrastructure BSCI '19. ACM Press, New York, pp.19–28. doi:10.1145/3327960.3332386
- **Zargar ST, Joshi J, Tipper D (2013)** A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. In: IEEE Communications Surveys & Tutorials 15(4): 2046–2069. doi:10.1109/SURV.2013.031413.00127
- Zhang X, Qin R, Yuan Y, Wang FY (2018) An Analysis of Blockchain-based Bitcoin Mining Difficulty: Techniques and Principles. In: 2018 Chinese Automation Congress (CAC). IEEE, pp.1184–1189. doi:10.1109/CAC.2018.8623140

Statutory Declaration

The statutory declaration needs to be in German, so the following excerpt is in German:

Eidesstattliche Versicherung

Ich versichere an Eides statt durch meine Unterschrift, dass ich die vorstehende Arbeit selbständig und ohne fremde Hilfe angefertigt und alle Stellen, die ich wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen habe, als solche kenntlich gemacht habe, mich auch keiner anderen als der angegebenen Literatur oder sonstiger Hilfsmittel bedient habe. Die Arbeit hat in dieser oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Essen, 18. April 2023	
Ort, Datum	Unterschrift