TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Vita Krainik 165637IVCM

DISTRIBUTED CONSENSUS PROBLEMS AND PROTOCOLS: A SYSTEMATIC LITERATURE REVIEW

Master's thesis

Supervisors:
Ahto Buldas
Dirk Draheim

Tallinn 2019

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Vita Krainik 165637IVCM

HAJUSLEPPE PROBLEEMID JA PROTOKOLLID: SÜSTEMAATILINE KIRJANDUSÜLEVAADE

Magistritöö

Juhendajad:
Ahto Buldas
Dirk Draheim

Tallinn 2019

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Vita Krainik

May 13, 2019

Abstract

Consensus problems are fundamental in distributed computing. Depending on a system, fault model or additional requirements, consensus problems can become harder, or even impossible to solve. They emerged in the late 1970s – early 1980s and have been studied in different shapes under various assumptions since. This thesis provides a Systematic literature review on consensus and its forms. The paper aims to collect various problems that emerged over time, and the solutions – original protocols and their improved versions. The review results in a survey on consensus and is focused primarily on problem definitions, rather than protocols. We present the reviewed problems in style composed of the motivation of the proposed consensus problem, its system and fault models, goals to achieve consensus and known solutions – consensus protocols. Such structure outlines the significant aspects of consensus and allows further comparison of consensus problems. This thesis provides a theoretical ground for researchers who wish to learn what are the different problems of consensus and what has been achieved in this area so far.

The thesis is in English and contains 55 pages of text, 6 chapters, 2 figures, 2 tables.

Contents

1 Introduction

This chapter is to introduce the master's thesis topic to the reader, describe the aim of the thesis, relevance and novelty of the research and provide the research questions. First, a background of the subject is given along with an informal introduction to consensus in distributed computing. Then, the research problem is presented, the motivation and reasoning why this research is important. Finally, the purpose of the thesis and thesis overview is specified. This chapter aims to provide clearly what this thesis is about, what is its contribution and novelty, why it is relevant and what exactly has been done in the research.

This thesis is a study on the formal definitions of consensus, problems and their solutions proposed during 1980 – 2019. The research method for collecting the papers on the topic is based on the systematic literature review methodology described by Barbara Kitchenham [1]. In her paper, Barbara Kitchenham provides guidelines for performing literature reviews on software engineering topics and defines the methodology as following: "A systematic literature review is a means of identifying, evaluating and interpreting all available research relevant to a particular research question, or topic area, or phenomenon of interest." [1]

In terms of the research scope, formal consensus problem proposals, original protocols, and improved solutions are the main focus during the review process. The general overviews of consensus problem and informal description are not relevant for this thesis since in this way the research questions cannot be answered. Further selection criteria one can find listed in the research methodology chapter.

1.1 Background

Consensus is a problem in distributed computing field, it derives straight from the design of a distributed system. Unlike centralized systems, in distributed environment data is maintained among processors, which have to synchronize and agree on the correct version of this data. Informally, agreeing on a common value between participants is considered a consensus (or agreement) problem.

Reaching consensus becomes less trivial if some of the processors are faulty. Faults in real-case scenario may include system errors, network issues or even malicious intent. Therefore, correct processes need to be able to achieve consensus even where there are faulty processes in the system.

Consensus has many faces. Besides the various environments of distributed systems, over time, technology changes, introducing mobile networks, faster internet, so consensus problems and algorithms need adjustment to the new environments, thus, generating new definitions. Therefore, the consensus in distributed computing is not one problem, it is a family of problems modeled under various assumptions.

Reaching consensus in distributed systems with failures has been an interesting subject to many researchers for the past 40 years. Consensus underlines the technology behind nowadays popular distributed applications for cryptocurrencies, smart contracts etc. This popularity motivates researchers to improve the technical properties of existing consensus protocols and design new algorithms. Therefore, a study about consensus problems and protocols is relevant to the academic community, as well as engineers.

1.2 Research problem

To our knowledge, a proper literature review on consensus definitions is missing. While searching for similar work, only brief surveys on consensus problem have appeared but no systematic literature review related to the topic. The existing surveys on consensus are rather old or focused mostly on protocols. At the same time searching papers by keywords "consensus (problem OR protocol)" gives us about half a million results. This amount of literature highlights the need in performing a systematic literature review on consensus problems and protocols – there are numerous studies in this area, which points on high interest, and there are attempts to write a basic consensus review in forms of short surveys. So the researchers' community would benefit from a comprehensive systematic literature review on consensus problems and protocols.

There is another argument to why this review is necessary. As distributed consensus developed, researchers discovered additional issues, for example, scalability, performance, other flaws in definitions and algorithms. The problems were either spotted by academics or derived directly from real-world applications' issues. These findings, however, might be missed in a pile of articles. Therefore, it would be valuable to see the progress that has been made on consensus theory over time in one place.

In this thesis, we use a systematic approach to find and evaluate the papers with the consensus problem proposals. We also search for the improvements in solutions of this problem and present our collection of problem-solution consensus groups.

1.3 Purpose of the research

This thesis aims to gain insight into this grey area and provide an up-to-date literature review of consensus definitions in distributed computing. We try to grasp a picture on existing consensus problems and link them with up-to-day solutions. By saying consensus definitions, we want to emphasize that the focus is made on consensus problem itself rather than on algorithms. We study motives behind a problem, the modeled environment, and the technical properties of the proposed protocols.

This paper is to encourage researchers and developers to learn, exploit and improve theoretical aspects of consensus, revisit definitions, problem statements and consensus protocols. Practitioners can get a better understanding of consensus and apply gained knowledge in their applications. Theoreticians can review and compare one consensus problem to another, learn about the achievements in the distributed consensus and identify the opportunities for further research.

The novelty of the thesis is in 1) the subject itself – consensus algorithms have been a trending topic for the past few years; 2) the methodology – systematic literature review has not been used (to our knowledge) for a study on consensus problems and protocols; 3) an up-to-date review on consensus – the rising popularity of consensus is the perfect time to revisit theoretical achievements in this area.

1.4 Research questions

The following research questions are to be answered:

1. What are the definitions of consensus?
2. What are the motives behind the definitions?
3. What are the additional assumptions that allow achieving the consensus?
4. What are the fault models of the given consensus problem?
5. What are the technical properties of the consensus protocols (e.g., space, communication, time complexity – memory, messages sent, computations)?

1.5 Thesis overview

Chapter 2 describes the chosen research methodology – Systematic literature review, the justification of the method and the detailed steps of the performed research. The followed procedure includes search keywords for paper selection, inclusion and exclusion criteria, research questions to be answered and the list of papers selected for the review.

Chapter 3 contains related work on consensus, which inspired to write this thesis. It is also mentioned there why this thesis is different from the existing papers and how this literature has formed a direction of the current research.

In Chapter 4 one can find the result of the aforementioned systematic literature review. The data extracted from the review is structured into a survey (or a study), which describes the establishment of consensus, reviews the definitions of honest and malicious parties and lists various consensus agreement problems proposed during 1980 – 2019.

Chapter 5 and 6 provide analysis and conclusions of the systematic literature review, the findings, and suggestions for future research.

2 Research Methodology

In this chapter the protocol for the systematic review is described step-by-step. As mentioned previously, for the research method, a systematic literature review is chosen, following the guidelines from Barbara Kitchenham's paper [1], where she thoroughly describes how to conduct systematic reviews for software engineering researches.

By definition, "A systematic literature review is a means of identifying, evaluating and interpreting all available research relevant to a particular research question, or topic area, or phenomenon of interest" [1]. This research method helps to develop a plan for finding and evaluating the papers on the current thesis's topic. A detailed description of the method and the research procedure justifies the paper selection for this study.

The current chapter is structured as follows. First the plan for performing the research is presented. Secondly, the actual execution of the protocol is described along with the outcomes of each step of the protocol. The final list of the papers is presented in the table 2.1, which are later used for composing a survey in Chapter 4. The analysis section describes how the final papers are classified and groupped around a consensus problem.

2.1 Introduction to the methodology

The general guidelines provided by Barbara Kitchenham [1] are adopted to the needs of this thesis and three main stages of the protocol are presented below:

1. Planning:
   (a) search for related work, define the need in performing a review;
   (b) define research questions;
   (c) select resources – tools, databases;
   (d) define search keywords for primary selection;
   (e) define inclusion and exclusion criteria for further filtering of papers.
2. Execution:
   (a) select primary papers based on search keywords;
   (b) perform secondary search by reading titles and abstracts;
   (c) select the relevant papers considering inclusion/exclusion criteria;

    (d) add papers from the references, if needed;
    (e) extract the data.
3. Analysis:
    (a) process the collected data;
    (b) answer research questions;
    (c) aggregate the results and present the outcome.

2.2 Planning

The need for a systematic literature review coincides with setting a problem statement and relevance of the thesis mentioned in the previous chapter. The main task here is to verify if there is similar work and what is missing in the existing literature. While performing the search, there was no systematic literature review found on consensus, only a few surveys, which we describe in the literature review chapter. Therefore, we assume that systematic literature review is needed and we can proceed further.

The materials for the thesis itself (e.g., articles to support the relevance of the topic, problem statement) are found through communication with supervisors, taking into account the feedback of the reviewer, and trial searches online via Google Scholar and PRIMO portal. However, the paper selection for a systematic literature review is more structured, and it follows the pre-defined protocol. The review is conducted online, with tools and e-resources available in TalTech library, mainly through PRIMO search portal, also TORU.ttu.ee VPN to access the resources. PRIMO allows searching through all the e-databases available in TalTech network but, of course, only some of them are relevant for this thesis meaning the papers are selected only from e-resources related to Information Technology. To reproduce the steps of the research, one can use any search engine that provides access to Lecture Notes in Computer Science, ACM Digital Library, SpringerLink, Taylor and Francis, Wiley Online Library, ScienceDirect, IEEE Xplore, and Safari databases.

Search keywords is one of the essential components of the systematic review. We defined the search query while preparing for the thesis, which included reading materials related to the consensus to understand the topic better. We discovered that various searches in PRIMO solely by keywords “consensus”, “distributed consensus”, “consensus problems”, “consensus protocols”, “byzantine agreement” result in thousands of papers physically impossible to process in a limited amount of time. Considering the time frames and realistic expectations of the number of papers one person can process in one semester, the expectation was that the search keywords would be specific enough to produce a result of up to 1000 papers as a primary selection. Therefore, it was decided to take advantage of PRIMO's feature to search by subject. Most of the more or less relevant papers on the consensus we found were in the Computer science category, also Distributed computing, Consensus problem and that was a solid ground to generate search keywords.

The term "consensus" was found to be also studied in Control theory and Pure Mathematics disciplines. Even though the concepts looked similar, in Distributed computing they approach the topic differently. It would be interesting to compare these concepts, but that would overextend the study. So the research scope was limited to Distributed computing field of study.

Thus, the final version of the search pattern is the following:
Any field contains "consensus"
AND
Subject contains "(Computer Science OR Distributed computing OR Distributed Systems) AND (Consensus OR Distributed OR Byzantine) NOT Control"
Apply filter: peer-reviewed Journals, English language, exclude online collections arXiv, PMC(PubMed Central).

To guarantee the quality of the research, we consider only peer-reviewed papers for this systematic review. Surprisingly, including a "peer-reviewed journals" filter cut the search results in half. Perhaps, failing to follow standard academic requirements by researchers in this field is another issue that needs addressing. Filter by the English language has removed less than 2% of the total results, so this is not crucial for this research.

We particularly exclude PubMed Central (PMC) archive as the collection is not relevant to computer science. Also, arXiv was assumed not to be a reliable source, so we filtered it out. However, even if there are some relevant papers published and peer-reviewed in arXiv, they will likely to appear in other more influential journals, so those can still end up in our search results.

After the primary papers are selected, the next step is to filter them by the pre-defined inclusion and exclusion criteria. Therefore, every paper is evaluated based on the following inclusion criteria:

1. The research is done in the distributed computing field.
2. A formal definition of the consensus problem is studied in the paper.
3. The paper contains the answers to the research questions of the thesis.

Also, If the paper matches the exclusion criteria below, it is removed:

1. Duplicate papers and content – if some papers appear multiple times, only one of

them ends up in the final selection.

2. Consensus used in another context than about distributed computing, for example, control theory consensus, the consensus in pure mathematics, medicine or economics related.
3. The paper has an informal description of the consensus problem, with no formalized consensus model, definitions, its assumptions, and technical properties.

The strategy also includes a backward reference search by searching for relevant literature through references. We assume that the references would be an excellent source to find the consensus definitions and their origins. Many papers can lead us to the original concepts proposed in 80s, which can show us where the consensus definitions come from. Therefore, we include the reference search to perform the study.

Finally, the research questions for this thesis are the following:

1. What are the definitions of consensus?
2. What are the motives behind the definitions?
3. What are the additional assumptions that allow achieving the consensus?
4. What are the fault models of the given consensus problem?
5. What are the technical properties of the consensus protocols (e.g., space, communication, time complexity – memory, messages sent, computations)?

2.3 Execution

This systematic literature review consists of several rounds. For the primary selection of papers, PRIMO search portal is used. Running the search string specified earlier resulted in 1073 papers, as of January 18th, 2019. For convenience, the secondary selection results were kept in BibDesk tool. During secondary selection, these 1073 papers were checked for relevance by reading the title and, if needed, the abstract. Also, the duplicated results were ignored. After the first round, 171 publications were filtered for further research. The second round resulted in 136 papers.

The next step included going through the remaining papers and selecting those that contain the answers to the research questions of this thesis, namely consensus definitions. The key interest is the papers that have formal definitions of consensus under different assumptions, first proposals of consensus problems and improvements on the topic. These 136 papers were reviewed more closely, by reading through the content to see if there is relevant data for extraction. Simultaneously, the references were checked, and some of those articles we added to the list. We expected that many papers would point to the original definitions

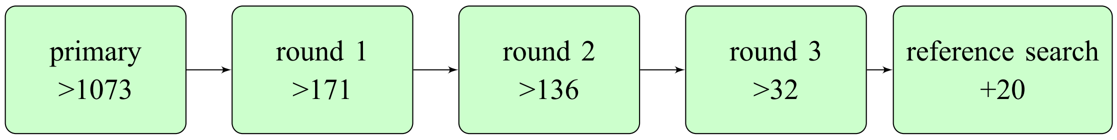

Figure 2.1: Summary of paper selection procedure

or the places where these definitions were first mentioned. Finally, there were 32 papers left, plus some added through citation search. So there were 52 papers selected in total for further research.

To summarize, the paper selection can be presented in a diagram 2.1. A full list of the papers, which ended up in final selections, are presented in a table 2.1.

Table 2.1: Summary of the review

No	Ref. no	Year	Authors	Title	Publisher	Ref. search
1	[2]	1980	Pease et al.	Reaching Agreement in the Presence of Faults	ACM	1
2	[3]	1982	Dolev et al.	Requirements for agreement in a distributed system	DDB	1
3	[4]	1982	Lamport et al.	The Byzantine Generals Problem	ACM	1
4	[5]	1983	Rabin	Randomized byzantine generals	IEEE	1
5	[6]	1985	Bracha & Toueg	Asynchronous Consensus and Broadcast Protocols	ACM	0
6	[7]	1985	Fischer et al.	Impossibility of Distributed Consensus with One Faulty Process	ACM	0
7	[8]	1986	Dolev et al.	Reaching approximate agreement in the presence of faults	ACM	1
8	[9]	1987	Loui et al.	Memory requirements for agreement among unreliable asynchronous processes	JAI press	1
9	[10]	1987	Patnaik & Balaji	Byzantine-resilient distributed computing systems	Springer	0
10	[11]	1988	Abrahamson	On achieving consensus using a shared memory	ACM	1
11	[12]	1988	Dwork et al.	Consensus in the presence of partial synchrony	ACM	1
12	[13]	1989	Berman & Garay	Asymptotically optimal distributed consensus: Extended abstract	Springer	1
13	[14]	1993	Berman & Garay	Randomized distributed agreement revisited	IEEE	1
14	[15]	1993	Barborak et al.	The consensus problem in fault-tolerant computing	ACM	0

Table 2.1: Summary of the review

No	Ref. no	Year	Authors	Title	Publisher	Ref. search
15	[16]	1994	Hadzilacos &Toueg	A Modular Approach to Fault-Tolerant Broadcasts and Related Problems	Cornell University	1
16	[17]	1994	Neiger, Gil	Distributed consensus revisited	Elsevier	1
17	[18]	1994	Attiya et al.	Bounds on the time to reach agreement in the presence of timing uncertainty	ACM	0
18	[19]	1995	Guerraoui, R.	Revisiting the Relationship between Non-Blocking Atomic Commitment and Consensus	Springer	1
19	[20]	1996	Chandra & Toueg	Unreliable failure detectors for reliable distributed systems	ACM	1
20	[21]	1999	Krings & Feyer	The Byzantine agreement problem: optimal early stopping	IEEE	1
21	[22]	1999	Gärtner	Fundamentals of fault-tolerant distributed computing in asynchronous environments	ACM	0
22	[23]	2002	Nesterenko & Arora	Dining philosophers that tolerate malicious crashes	IEEE	1
23	[24]	2002	Hsiao et al.	Reaching strong consensus in a general network	Inst Information Science	0
24	[25]	2002	Castro & Liskov	Practical byzantine fault tolerance and proactive recovery	ACM	0
25	[26]	2002	Attie	Wait-free Byzantine consensus	Elsevier	0
26	[27]	2003	Charron-Bost et al.	Comparing the Atomic Commitment and Consensus Problems	Springer	1
27	[28]	2003	Keidar & Rajsbaum	A simple proof of the uniform consensus synchronous lower bound	Elsevier	0
28	[29]	2004	Charron-Bost & Schiper	Uniform consensus is harder than consensus	Elsevier	1
29	[30]	2004	Défago & Schiper	Semi-passive replication and Lazy Consensus	Elsevier	0
30	[31]	2004	Izumi & Masuzawa	Synchronous condition-based consensus adapting to input-vector legality	Springer	0
31	[32]	2005	Lamport	Generalized Consensus and Paxos	Microsoft Research	1
32	[33]	2005	Wang et al.	A bivalency proof of the lower bound for uniform consensus	Elsevier	0

Table 2.1: Summary of the review

No	Ref. no	Year	Authors	Title	Publisher	Ref. search
33	[34]	2005	Izumi & Masuzawa	An improved algorithm for adaptive condition-based consensus	Springer	0
34	[35]	2006	Martin & Alvisi	Fast Byzantine Consensus	IEEE	0
35	[36]	2007	Anceaume et al.	Managed Agreement: Generalizing Two Fundamental Distributed Agreement Problems	Elsevier	0
36	[37]	2008	Mizrahi & Moses	Continuous consensus via common knowledge	Springer	1
37	[38]	2008	Okun & Barak	Efficient Algorithms for Anonymous Byzantine Agreement	Springer	0
38	[39]	2008	Correia et al.	On Byzantine generals with alternative plans	Elsevier	0
39	[40]	2009	Kotla et al.	Zyzzyva: Speculative Byzantine fault tolerance	ACM	0
40	[41]	2010	Mizrah & Moses	Continuous consensus with ambiguous failures	Elsevier	0
41	[42]	2011	Bouzid et al.	Anonymous Agreement: The Janus Algorithm	Springer	0
42	[43]	2012	Cheng & Tsai	Eventual strong consensus with fault detection in the presence of dual failure mode on processors under dynamic networks	Elsevier	0
43	[44]	2012	Ramesh & Kumar	An Optimal Novel Byzantine Agreement Protocol (ONBAP) for Heterogeneous Distributed Database Processing Systems	Elsevier	0
44	[45]	2012	Widder et al.	Consensus in the presence of mortal Byzantine faulty processes	Springer	0
45	[46]	2013	Delporte-Gallet et al.	Byzantine agreement with homonyms	Springer	0
46	[47]	2013	Veronese et al.	Efficient Byzantine Fault-Tolerance	IEEE	0
47	[48]	2015	Cheng & Tsai	A recursive Byzantine-resilient protocol	Elsevier	0
48	[49]	2015	Mostéfaoui et al.	Signature-Free Asynchronous Binary Byzantine Consensus with $t < \frac{n}{3}$, $O(n^2)$ Messages, and $O(1)$ Expected Time	ACM	0
49	[50]	2017	Gramoli	From blockchain consensus back to Byzantine consensus	Elsevier	0
50	[51]	2017	Pires et al.	Generalized Paxos Made Byzantine (and Less Complex)	Springer	0

Table 2.1: Summary of the review

No	Ref. no	Year	Authors	Title	Publisher	Ref. search
51	[52]	2018	Alchieri et al.	Knowledge Connectivity Requirements for Solving Byzantine Consensus with Unknown Participants	IEEE	0
52	[53]	2019	Liu et al.	Scalable byzantine consensus via hardware-assisted secret sharing	IEEE	0

The analyzed papers propose a consensus problem, a solution to it – a consensus protocol, or some improvements on either of those. The consensus problem components, which would answer the research questions of the current thesis are the following:

- system model – an environment for the protocol, e.g., synchronous/asynchronous;
- fault model;
- conditions – goals to achieve consensus, e.g. agreement, validity;
- a proposed solution – a consensus protocol, its technical properties;
- an improved solution if any.

For a survey, all found consensus problems are tried to put into the structure above. Such consistency would help to see the differences between those problems.

2.4 Analysis

The collected data from the papers we selected for our research can be put into classification by its content:

A: A consensus problem and a solution to this problem are proposed.
B: A consensus problem defined in another paper is revisited and studied from a different angle.
C: A new protocol is designed, which solves the consensus problem defined in another paper, and improves the complexity of the previous protocols.
D: A new protocol is designed, which solves the consensus problem defined in another paper, and offers a solution under different assumptions (e.g., in Byzantine fault model).
E: Other improvements on the topic on consensus problems and protocols – new upper/lower bounds, impossibility results and so on.

This classification helped us structure the main outcome of this systematic literature review – a survey on consensus problems and protocols. The data extracted during the review has been grouped around each consensus problem definition. On the example of strong consensus [17], we can observe how we aggregated the data. First, we have a paper with the original definition of a strong consensus of 1994 [17]. Secondly, another paper of 2002 ended up in our search results, which describes strong consensus in a general network [24]. Finally, we found a study of 2012, which proposes an early-stopping protocol for the strong consensus problem [43]. In the survey, these papers go together and provide a general review of the strong consensus. The rest of consensus problems are structured similarly.

The research questions concern each consensus problem definition separately meaning we look at the research questions and try to answer them every time we have a new consensus problem. The collected data is grouped into consensus problem-solution sections, and every section follows the same structure – a motivation behind the new problem, environment, conditions, and solutions. The outcome is a well-structured consensus study, which is presented further below in this thesis.

3 Related Work

This chapter provides a list of papers that are similar to this thesis topic in some way. Here one can find out how the topic, the research scope, the research problem have developed and why this research is different from the existing literature reviews. This review was the preliminary study to find the ideas for the thesis.

The majority of the papers below are surveys on consensus related topics. A short description of each paper is presented, its limitations and how this thesis contributes to improve the current state of things. The last two articles are to support the idea of the review, namely the motivation to revisit distributed consensus theory.

The consensus problem in unreliable distributed systems (a brief survey) by Michael. J. Fischer

The author in [54] provides a summary of the traditional consensus problem. The article has been published in 1983 and included most prominent studies on consensus at that time. In his paper, M.J. Fischer explains basic concepts of the consensus problem, requirements to solve it, upper/lower bounds and models of computation.

The survey can serve as a starting point for those who begin discovering consensus problems. It is short and easy to read; the author considers simplified models like systems with only $\{0, 1\}$-bit messages. This paper also discusses some essential concepts, which help understand the general picture.

On the downside, the article has many limitations since it was published more than thirty years ago. It covers only the work of the late 70s/early 80s and distributed systems have been improved significantly since then. The survey is also brief, as specified in the title, so it covers only basic ideas of the consensus problem and refers to other articles for detailed proofs. The outdated results lack applicability of the concepts and protocols learned from the paper. M.J. Fischer also points on that: “The abstract versions of agreement problems considered in this survey are not general enough to be directly applicable to many practical situations.”[54]. A more comprehensive study of the consensus including recent articles and modern applications would expand the topic and provide a fresh overview of the problem.

The consensus problem in fault-tolerant computing by Michael Barborak, Anton Dahbura, and Miroslaw Malek

This paper [15] is focused mostly on system diagnosis, faulty elements detection and provides a comparison of Byzantine agreement and system diagnosis approaches. It is thorough and detailed about fault models classification, which gives practical input to the developers of consensus protocols.

However, in the paper, the main focus is put on Byzantine agreement, not on the consensus problems in general like in [54]. Moreover, again, this work is of June 1993, which means nowadays there is much more content to include on the subject. Therefore, there are ways to expand the topic, for example, an up-to-date review with a focus on different consensus problems.

Consensus in asynchronous distributed systems: A concise guided tour

Rachid Guerraoui et al. covers one of the chapters on consensus problems in asynchronous environments [55]. The primary focus of the paper is the classical consensus problem in crash/recovery and crash/no recovery fault models. It was published in 2000, so the paper is neither new nor old. This paper is rather short and covers only a small part of the consensus problems. So the existence of this paper does not eliminate the need for another survey on consensus problems, which could be more extensive.

Consensus in synchronous systems: a concise guided tour

In 2002 [56] another survey was presented by Michel Raynal. Despite the title overlap with the previous paper, this study is different – it is twice shorter, 8 pages long in total, and it reviews mostly the consensus protocols, rather than formalized problems. The consensus problem did not seem to be the primary interest of the author, the definition takes only a few paragraphs in the survey. This review on consensus protocols lacks more information on consensus problems, and this is what we offer in our thesis.

Survey of consensus protocols on blockchain applications by Lakshmi Siva Sankar, Sindhu M., and M. Sethumadhavan

This survey [57] is 4 pages long, it introduces the reader to different concepts related to blockchain, where blockchain is defined as "a distributed, transparent, immutable ledger"[57], and compares how some mainstream blockchain applications work. However, this paper does not provide much formal terminology to a reader or answer any of the research questions set in the current thesis.

The main difference between this paper and our thesis is in the approach to the distributed consensus. This paper is one of the many studies with the focus shifted to the implementations of consensus algorithms, rather than theoretical research backed by peer-reviewed academic literature. Additionally, while the authors do not explain why they selected those particular projects, the papers for our systematic review are selected based on the inclusion/exclusion criteria, which maximizes the probability that the collected data reflects the general picture on the subject. Thus, the existence of this paper does not eliminate the value of the current research.

Bitcoins academic pedigree by Arvind Narayanan and Jeremy Clark

Narayanan and Clark [58] offer a direction for thinking and further research. On the example of the history of bitcoin's key components, they point on a gap between academics and practitioners while both could benefit from the knowledge of each other: "Both practitioners and academics would do well to revisit old ideas to glean insights for present systems"[58]. Indeed, instead of reinventing the wheel, it is useful to see if the existing theoretical base already has any ideas related to the problem.

From blockchain consensus back to Byzantine consensus by Vincent Gramoli

The author in [50] outlines the gap between consensus theoretical base and existing implementations of consensus protocols. This gap is particularly noticeable in blockchain protocols, where a significant number of white papers are not peer-reviewed, and the definitions are often skipped or vaguely defined: "While the source code of most blockchain protocols is publicly available, the theoretical ramifications of the blockchain abstraction are rather informal" [50]

The author points out on the need for having formal definitions for modern protocols and designing proper theoretical models before implementing those. He proposes a new consensus problem definition in context of blockchain and attempts to formalize the consensus models of some mainstream blockchain algorithms. The theoretical base for these algorithms is far behind, at the same time when proper definitions would allow to avoid potential problems caused by design errors: "Very little work has however been devoted to explore its theoretical ramifications. As a result, existing proposals are sometimes misunderstood and it is often unclear whether the problems arising during their executions are due to implementation bugs or more fundamental design issues." [50]

The problems highlighted by Vincent Gramoli support the ideas mentioned in this thesis. Although it is hard to force engineers to start using proper definitions and formalized models for their algorithms, let us revisit the achievements on consensus theory and provide an easier access to the existing consensus problems.

4 Consensus Study

This chapter presents the result of the research, and it is also the main contribution of the thesis. The data, collected during the systematic literature review, has been extracted, analyzed and put into a survey. First, we introduce the consensus, how it emerged and what are the most prominent achievements of the 1980s on this topic. Secondly, we talk about fault models and types of participants in a protocol. Finally, a collection of consensus problems is presented, along with their solutions.

The review of consensus problems resulted in the majority results to have a message-passing model, where processes communicate via exchanging messages. However, we also found a few problems in a shared-memory model, where processes communicate not directly but through a shared-memory [11] (first studies on consensus using shared memory appear in [9, 11]).

4.1 Introduction

What is a distributed consensus? In distributed systems, processors communicate with each other to keep their data in sync. The fundamental problem is to agree, even when some of the processors are not honest. The more participants in the system, the more likely it is that some of them behave arbitrarily. This basic problem generates many other consensus problems, modeled in various environments.

So what differentiates one consensus problem from another? The model consists of several components, one of them is the value processes need to agree on. The simplest form of consensus is binary consensus, where the processes decide on a value from $\{0, 1\}$ range. In another case, they try to achieve consensus on a set of values, which is called multi-valued consensus. Another extension of consensus, where processes aim to agree to a set of values, is called k-set agreement [59]. However, the k-set agreement is outside of the scope of the current thesis. These are the basic consensus types which differentiate the values to agree on.

Another different category is whether processes communicate synchronously or asynchronously. Here we have to mention another significant paper in the world of distributed systems – the proof of the impossibility of consensus in asynchronous settings by M.J. Fischer [7]. According to Google Scholars, this article is cited by nearly 5000 academic papers. The results in this paper are prominent because they show that no protocol can achieve consensus in a completely asynchronous environment in the presence of even a single fault; therefore, there must be further conditions and additional assumptions about the environment to reach agreement.

To cope with the impossibility result above, one of the options introduced is partial synchrony [12]. The authors of the concept model a system, where processes are synchronous and their communication is partially synchronous (communication bound Δ is either unknown or holds eventually [12]). Furthermore, as a response to Fischer's work, Gabriel Bracha and Sam Toueg described a model where the protocol is assumed to terminate with probability 1 [6]. They also showed in this paper that the majority ($\frac{n+1}{2}$) of correct processes is the lower bound to reach agreement in a fail-stop model, and for the malicious model it is $\frac{2n+1}{3}$ [6].

4.2 Participants

In consensus protocols, good and bad processes may participate. When talking about good processes, there are not many definitions of a well-behaved process, but generally speaking it is considered that "a good process is a process that behaves as expected" [55], or essentially fault-free. The meaning of a faulty (or bad, malicious, corrupted) process varies from article to article, for example, Felix Gärtner [22] gives a formal definition to a fault as "action on the possibly extended state of a process" [22]. Consensus problems particularly depend on how bad processes are assumed to behave, that is why we want to mention what failure models are out there.

When setting a consensus problem, the assumptions about the behavior of a faulty process affects further restrictions and the complexity of the consensus algorithm. There are several failure classifications, Michael Barborak et al. [15] presented a detailed taxonomy of fault types in 1993. These types are fail-stop, crash, omission, timing, incorrect computation, authenticated Byzantine and Byzantine [15] (see the summary on [15] in the table 4.1).

From the Barborak's classification [15], a Byzantine fault looks like it implies other kinds of faults, e.g., crash, however, in different papers, authors put their own meaning into this concept, often Byzantine means controlled by the adversary but not corrupted in a "natural" manner. It makes sense to distinguish arbitrary faults from others as crash, omission

Table 4.1: Fault-model types

fault	main characteristics
fail-stop	terminates, other processes are aware of termination
crash	terminates due to corruption, other processes do not know from a failed process about its termination
omission	process fails partially(e.g., sending or receiving message fails)
timing(or performance)	process fails to perform a task at a specified time
incorrect computation	process miscalculates the output
authenticated Byzantine (arbitrary, malicious)	controlled by the adversary, can behave in an arbitrary manner
Byzantine	any kind of a fault

faults, which can be recognized by fault-detection mechanisms. In such cases, protocol bounds can be significantly improved as the fault model turns to be more specific and realistic. This division to malicious/nonmalicious fault can also allow modeling single and dual failure modes, where either one type of fault can occur or both [43].

The participants in consensus protocol typically consist of n – the total amount of processes, t – the tolerated amount of faulty processes, f – the actual amount of faults during protocol execution. The f allows to take into account that in practice the number of failures may differ from what is assumed or allowed, and this assumption can help with developing a more optimized protocol.

4.3 Consensus Problems and Protocols

4.3.1 Traditional Consensus Variations

Consensus problems form a whole class of problems. Let us talk about the original consensus problem definition, as well as some of the problems very similar to consensus. These problems are atomic commit and atomic broadcast.

Chandra et al. [20] have formalized the consensus problem, by defining two properties $propose(v)$ and $decide(v)$, which are executed by processes $p_i, i \in \overline{0, n}$ [20]. Here v is some value $v_i, i \in \overline{0, n}$ proposed or decided by a prosess p_i.

Based on these primitives, Chandra et al. specify uniform consensus by following the conditions:

"*Termination.* Every correct process eventually decides some value.
Uniform integrity. Every process decides at most once.
Agreement. No two correct processes decide differently.
Uniform validity. If a process decides v, then v was proposed by some process." [20]

This paper [20] also introduces an unreliable failure detector abstraction – a component in a system able to detect crash failures with some probability. Some of the consensus problems listed further specifically assume the presence of one or multiple failure detectors, or similar abstractions.

Atomic commit is the problem of agreeing on committing/aborting a transaction. An extended version of the atomic commit is non-blocking atomic commit (NB-AC). Atomic commit and consensus have been compared by Rachid Guerraoui in 1995 [19] and by Bernadette Charron-Bost in 2003 [27]. NB-AC and consensus have a similar goal – to make a decision despite failures. In [19], Rachid Guerraoui shows that in an asynchronous system with failure detectors as described in [20], NB-AC cannot be solved as opposed to consensus. In the same paper, on the other hand, Rachid Guerraoui defines a weak form of atomic commit (NB-WAC) that can be reduced to uniform consensus in this system model. The conditions to satisfy NB-AC and NB-WAC are then following [19]:

"*Uniform Agreement.* No two participants AC-decide different outcomes.
Uniform Validity. If a participant AC-decides commit, then all participants have voted yes.
Termination. Every correct participant eventually AC-decides.
NonTriviality. If all participants vote yes, and there is no failure, then every correct participant eventually AC-decides commit.
(Weak form) NonTriviality. If all participants vote yes, and no participant is ever suspected, then every correct participant eventually AC-decides commit." [19]

More updates on the atomic commit problem are collected in the paper of Bernadette Charron-Bost [27]. The results of the work show that non-blocking atomic commit and consensus are very similar in synchronous environments in terms of complexity but again, in asynchronous systems NB-AC is non-solvable.

Atomic broadcast is another problem that is proven to be redundant to consensus and vice versa in asynchronous systems with crash failures [20]. The formal definition of atomic broadcast is characterized by the properties:

"Validity. If a correct process R-broadcasts a message m, then it eventually R-delivers m.
Agreement. If a correct process R-delivers a message m, then all correct processes eventually R-deliver m.
Uniform integrity. For any message m, every process R-delivers m at most once, and only

if m was previously R-broadcast by sender(m)." [20], this definition Chandra et al. cite from [16].
"Total order. If two correct processes p and q deliver two messages m and m‘, then p delivers m before m‘ if and only if q delivers m before m'" [20]
Validity, agreement and uniform integrity define reliable broadcast, adding total order condition transforms it to atomic broadcast [20].

4.3.2 Byzantine Generals/Byzantine Agreement

The Byzantine agreement is a huge topic, and researchers put a significant amount of effort into developing Byzantine consensus protocols, which are getting better, faster, more efficient, optimal, scalable over time. Let us review the original definition and then list some of its variations and researches made on the topic.

The original Byzantine agreement definition comes from Pease, Shostak and Lamport's paper "The byzantine generals problem" of 1982 [4]. They describe a problem of agreeing on a common battle plan (reaching consensus) among generals (processors) in the presence of traitors (failures). A few years earlier, Pease, Shostak and Lamport introduce the interactive consistency [2] which is, in fact, the same Byzantine agreement problem; even though the authors did not use the terms "byzantine fault" or consensus, they implied that bad processes might behave arbitrarily or "may lie", and processors need to reach agreement on a value.

System environment: Processes (Generals) need to agree on a value (a battle plan) no matter how malicious processes (Byzantine generals) try to conflict the parties. Pease, Shostak and Lamport [4] describe a synchronous model, where processes exchange messages in rounds.

Goals: The problem relies on the interactive consistency properties:

"IC1. All loyal lieutenants obey the same order.
IC2. If the commanding general is loyal, then every loyal lieutenant obeys the order he sends." [4]

Proposed solution/protocol: The proposed algorithms are not efficient in terms of time and message complexities. They require $(n-1)(n-2)...(n-m-1)$ message exchanges to achieve agreement.

Fault model: Pease, Shostak, and Lamport introduce the concept of a Byzantine fault, which can behave in an arbitrary manner. This fault type is the most generic in distributed systems. The authors of the concept show that the solution to the Byzantine generals problem requires more than $\frac{2}{3}$ of the processes to be correct if messages are not signed. They also show that with signed messages the agreement can be achieved with any number of faulty processes.

Consensus problems and Byzantine generals: Patnaik and Balaji [10] in 1987 reviewed distributed agreement problems and highlighted three main classes of agreement problems – consensus problem, interactive consistency, and Byzantine generals. They mention the classical definitions of consensus, namely two conditions must be satisfied for the protocol to achieve consensus [10]:

"*Agreement.* All non-faulty processes agree on a common value.
Validity. If all non-faulty processes choose the same initial value, then all non-faulty processes agree on this value." [10]

The conditions to solve generals problem are very similar to the consensus problem, allowing us to focus on both of the terms, and defined as following [10]:

"*Agreement.* All non-faulty processes agree on a common value.
Validity. If the general does not fail, then all nonfaulty processes agree on x." [10]

One solution can solve both problems as we can convert one to another: "Given a protocol for the 'consensus problem', the Byzantine generals problem may be solved by having each process choose the value broadcast by the general as its initial value. On the other hand, given a protocol for the Byzantine generals problem, the consensus problem may be solved allowing each process to execute a copy of the 'Byzantine agreement protocol'." [10]

Updates on the protocols: Since in this paper the focus is on problem definition and there are tons of Byzantine agreement protocols, we mention only some of them, which ended up in our search results and which seem prominent in this area.

Axel W. Krings and Thomas Feyer have designed an early-stopping algorithm in non-authenticated Byzantine model [21]. The protocol is a modification of the original Byzantine generals problem solution [4]. This protocol takes into account that there might be less than the tolerated amount of faults $f \leq t$. The proposed algorithm terminates at maximum min $\{f + 2, t + 1\}$ rounds [3], where f is the number of actual faults, and t is the number of tolerated faults.

When talking about anonymous systems, there is an interesting study of the year 2013 on Byzantine agreement with homonyms [46]. The target of the research is the identifiers of the processes, where processes with distinct and same (homonyms) identifiers can be present in a system. The authors of the article claim that for synchronous Byzantine agreement there has to be at least $3t + 1$ identifiers, and for a partially synchronous case it is more than $\frac{n+3t}{2}$. This numbers can be improved to $t+1$ if there is a restriction that a faulty process cannot send more than one message to the same process in one round [46].

Optimal novel Byzantine agreement protocol (ONBAP) [44] is proposed in 2012 by Dharavath Ramesha and Chiranjeev Kumar. They claim to improve the results of the protocols proposed in the 80-s in the articles of Pease, Wang, Lamport, and Fischer and develop a Byzantine consensus protocol with reduced time complexity. ONBAP solves traditional Byzantine agreement in three rounds of message exchanges and with quadratic message complexity $(O(n^2))$. Here we can see that the technical properties of this protocol are better than in the original BA papers.

A recursive Byzantine-resilient protocol (RBR) [48] improves the classical Byzantine consensus protocols in terms of time and space complexity. The novel protocol can achieve consensus with time complexity $O(\lg n)$ and space complexity $O(nk^{\lg n})$. The protocol can tolerate less than $\frac{n-1}{3}$ Byzantine faults, specifically $2^h(\frac{\frac{n}{4^h}-1}{3}+1)-1$ with $h = \frac{\lg n-2}{2}$. The authors claim that in practice the amount of faulty Byzantine processors is much less than the estimated amount in the original papers, and they improve the complexity by lowering this number.

Practical Byzantine Fault Tolerant (PBFT) consensus protocol [25] is an algorithm widely used in practice, for example, its implementation is a foundation of Hyperledger Fabric [60] at IBM. PBFT uses state machine replication mechanism, which is "a general method for implementing a fault-tolerant service by replicating servers and coordinating client interactions with server replicas" [61], and tolerates up to $\frac{n-1}{3}$ of Byzantine faults [25]. A couple more protocols have been proposed, that are claimed to be more efficient than PBFT. On the contrast of PBFT that require $3t + 1$ replicas, algorithms MinBFT and MinZyzzyva require only $2t + 1$ and have better throughput and latency [47].

To address the issue of scalability of Byzantine consensus protocols, a FastBFT was introduced in 2019 [53]. To improve performance, FastBFT uses message aggregation to achieve linear message complexity $O(n)$, and follows optimistic paradigm with $f + 1$ nodes expected to participate in the protocol actively. The creators of the protocol claim that it can process 100000 transactions per second "assuming 1 MB blocks and 250 byte transaction records" [53], and it is claimed to be 6 times faster than Zyzzyva [40] – a speculative BFT protocol that also targets to achieve better performance than previous protocols.

Solida [62] protocol is designed using Byzantine consensus with added reconfiguration. Solida implements PBFT algorithm with a reconfiguration step to satisfy permissionless settings. The protocol is designed in a synchronous environment, and PBFT provides responsiveness and therefore improves the protocol performance. Malicious party here is assumed to be less than $\frac{1}{3}$ of all participants.

To cover some of the Byzantine consensus protocols in the asynchronous message-passing system, let us mention a signature-free binary consensus protocol [49], that tolerates up to $\frac{n-1}{3}$ Byzantine failures with $O(n^2)$ message exchanges per round and $O(1)$ bit complexity. The protocol foundation is a double synchronized binary value broadcast abstraction [49], and it is actually a randomized consensus algorithm (the details on the randomization are in other sections).

4.3.3 Byzantine Generals with Alternative Plans

In 2008 another variation of Byzantine generals problem was proposed by Miguel Correia, Alysson Neves Bessani and Paulo Veríssimo called "Byzantine generals with alternative plans" [39]. It is essentially a multi-valued Byzantine consensus problem.

Motivation: This problem has a practical use for scenarios where Byzantine consensus problem is needed.

System environment: BGAP has asynchronous message-passing model, where processes are required to satisfy $n \geq 3t + 1$. An oracle is present to avoid the impossibility of consensus in asynchronous systems.

Goals: BGAP is described by the following conditions:

"*Validity 1.* If there is a value v such that for any correct process p_i , vG_i , then any correct process that decides, decides a value v' such that $v' \in G_j$ for a correct process p_j.
Validity 2. No correct process p_i decides a value v if there is a correct process p_j with $v \in B_j$.
Agreement. No two correct processes decide differently.
Termination. Every correct process eventually decides." [39]

It is worth to mention that in this problem variation processes may have a set of good values $G_i = \{v_{i1}, ..., v_{ik_i}\}$ and a set of bad values $B_i = \{v'_{i1}, ..., v'_{il_i}\}$ [39].

Fault model: Byzantine generals with alternative plans use the same Byzantine fault model as all Byzantine consensus problems.

Proposed solution/protocol: Authors have proposed several solutions to the problem. The most difficult case, when there are no additional restrictions on good and bad value sets, is claimed to be unsolvable [39]. Another non-trivial variation holds the following assumption: $\forall i, j : (correct(i) \land correct(j)) \Rightarrow (Bi = Bj)$. The proposed algorithm has the same time complexity as Byzantine binary agreement, and the message complexity is binary consensus complexity multiplied by V – the size of possible decisions.

4.3.4 Anonymous Byzantine Agreement

Motivation: In 2008 Michael Okun and Amnon Barak proposed an anonymous Byzantine consensus [38] and explain their motivation as follows: "The predominant motivation of this study is to find the minimal conditions which still allow to reach BA" [38]. Additionally, the authors point out on the enhanced privacy due to the anonymity of the processors.

System environment: The binary consensus in the synchronous system is considered in the paper. All processes are anonymous, meaning the receiver of the messages does not know who the sender is.

Goals: The goal of the problem is to achieve Byzantine eventual agreement:

"*Termination.* Every correct processor eventually decides.
Agreement. All the correct processors decide on the same value from V.
Validity. If the input to all the correct processors is $v \in V$, then v is the only possible output value." [38]

Fault model: The problem is set in Byzantine environment, which makes sense as this consensus problem is an extension of the original Byzantine consensus.

Proposed solution/protocol: The authors [38] offer their solution to anonymous Byzantine agreement and provide the following technical properties of the algorithms. The proposed algorithm tolerates $t < \frac{n}{3}$ Byzantine faults in anonymous network and time complexity is $\frac{3(n-t)t}{n-2t} + 4$ rounds. Regarding communication complexity, the paper shows that it is $O(n^2t)$ messages and $O(n^2t \cdot \log n)$ message bits. In the same work [38], there is also an early-stopping version of the protocol where the maximum amount of rounds is $\min\{\frac{3(n-t)t}{n-2t} + 4, \frac{3(n-f)f}{n-t-f} + 3f + 9\}$ and communication complexity is $O(n^2f)$ messages and $O(n^2f \cdot \log n)$ message bits.

4.3.5 Mortal Byzantine Consensus

Motivation: Byzantine fault model is known to be too pessimistic while the crash model does not reflect many scenarios in practice [45]. Therefore, a consensus problem in the mortal Byzantine failure model has been introduced, as a compromise to the fault types mentioned above, by Josef Widder et al. in 2012 [45]. This fault type is taken from earlier research by Nesterenko and Arora [23] where they name it "malicious crashes".

System environment: The described problem is a binary consensus in the message-passing model. The system is synchronous and round-based.

Goals: To solve Byzantine consensus with mortal Byzantine failures, a protocol must meet the following requirements:

"*Agreement.* No two correct processes decide differently.
Validity. If some correct process decides v, then v is the initial value of some correct process.
Decision. Every correct process eventually decides.
Halting. Every correct process eventually halts." [45]

Fault model: Fault model of this consensus pattern is what differentiates it from other consensus definitions. By mortal Byzantine fault the authors [45] mean a fault that eventually crashes, but before that can behave arbitrarily. Such behavior reflects some of the real faulty behaviors in practice.

Proposed solution/protocol: A proposed protocol solves consensus with up to $t < \frac{n}{2}$ failures, it is claimed to be an optimal number of faulty processes in this model [45].

Improvements: This paper also shows that in partially synchronous systems the lower bound on the number of faults for mortal Byzantine consensus protocols is $t < \frac{n}{3}$ [45].

4.3.6 Byzantine Consensus with Unknown Participants (BFT-CUP)

Motivation: Alchieri et al. in 2018 study consensus in a system with the unknown number of participants [52], reasoning that while static networks are studied the most in distributed systems, the technology dictates different – wireless mobile ad-hoc, P2P networks might not be aware of the exact number of participants.

System environment: Processes communicate in a network in the message-passing model via authenticated reliable channels. A process knows about a subset of participants. The authors of the paper [52] state that the system model does not require any synchronization besides what is necessary to implement Byzantine consensus black box.

Goals: The definition of consensus is used from [20]; it is a traditional definition with propose and decide functions. A safe Byzantine failure pattern assumption is added in [52].

Fault model: Fault model used is Byzantine – a process that does not follow the protocol is considered faulty, otherwise – correct. Even though the total amount of participants is unknown, it is expected that at most f processes are faulty and at least $2f + 1$ processes in a sink are correct.

Proposed solution/protocol: BFT-CUP protocol reaches consensus in an asynchronous system with $2f + 1$ correct processes in a sink and must satisfy a safe Byzantine failure pattern [52].

4.3.7 Generalized Consensus and Paxos Protocols

Motivation: The very first generalized paxos was proposed by Lamport in 2005 [32]. Lamport explains that the weak consensus (or generalized consensus) can be achieved at lower costs and proposes protocol named Paxos that solves it. But we can get into more details of it in its simplified version by Miguel Pires et al. [51] first published in 2017, which also extends generalized paxos to Byzantine model. Interestingly, generalized paxos has weakened traditional consensus problem definition to achieve better communication complexity [51].

System environment: The environment is asynchronous and processes communicate via exchanging messages over reliable channels. A process may play learner, proposer or acceptor roles.

Goals: The following requirements [51] are defined for the generalized consensus problem:

"*Nontriviality.* $learned_l$ can only contain proposed commands.
Stability. If $learned_l = v$ then, at all later times, v is a eq-prefix of $learned_l$, for any l and v.
Consistency. At any time and for any two correct learners l_i and l_j, $learned_{l_i}$ and $learned_{l_j}$ can subsequently be extended to equivalent sequences.
Liveness. For any proposal s and correct learner l, eventually $learned_l$ contains s."[51]

Fault model: This problem is set in authenticated Byzantine model meaning that processes may follow the protocol, crash or behave maliciously.

Proposed solution/protocol: The proposed Byzantine fault tolerant generalized paxos protocol [51] solves agreement with acceptor processes $\geq 3f + 1$ and quorums of $N - f$ processes. By definition, "quorums are any set of $N{-}f$ processes" [51] where N is the total size of the system and f is the tolerated number of faults.

Improvements: Paxos is a family of protocols, let us mention some of its developments below. In 2006 a Fast Byzantine (FaB) consensus protocol was proposed [35] (>200 citations), where the authors improve Paxos to reach agreement in two rounds for the common case in Byzantine fault model. The motives behind FaB protocol is to improve performance for the common case in Byzantine environment. The system has synchronous settings, so-called common case. By common case the authors mean a scenario that will most likely happen in the majority of executions, more specifically "1) there is a unique correct leader, 2) all correct acceptors agree on its identity, and 3) the system is in a period of synchrony." [35]. FaB runs in unreliable network with unreliable channels and "authenticated asynchronous fair links" [35], the protocol tolerates f tolerated faulty processes, out of $5f + 1$ total. This solution is a two-step protocol, which terminates in 2 steps in common case.

Also, Paxos consensus algorithm has been implemented at Google in their fault-tolerant system called Chubby [63]. Chandra, Griesemer and Redstone in this paper point out on the complexity of the implementation theoretical algorithms into the real-world systems.

4.3.8 Consensus with Timing Uncertainty

Goals: Another consensus problem definition extends the classical round-based model to a more realistic case with "timing uncertainty" [18]:

"*Agreement.* No two different processes decide on different values;
Validity. If some process decides on v, then an event input(i, v) occurs in α;
Termination and Time Bound. Every process either has a failure event or makes a decision by time start$(\alpha) + B($delay$(\alpha))$." [18], where v is the decision value, B is the time during which the protocol terminates, α – a timed execution prefix.

Motivation: The motivation behind this definition is related to the common case when participants do not start or receive input at the same time. The authors claim that the "initial synchronization is not very realistic in a distributed network" [18], hence the input events are introduced to the definitions.

System environment: The protocols run in message-passing settings, where communication is based on rounds. The system is neither completely synchronous nor asynchronous. The authors propose a compromise claiming that both those settings are "too extreme" [18].

Fault model: This consensus problem is described in a fail-stop model, and these failures can be detected by introducing a failure detector [18].

Solution: The particular focus in this paper is put on the time complexity of the algorithms which solve this agreement problem. While for classic round-based model the upper and lower bounds to reach an agreement are $f+1$ synchronous round, the authors aim to study how introducing timing uncertainty C affects the time complexity of the protocols. The results have shown that the lower bound for the worst case scenario in the timing-based model is $(f-1)d+Cd$ and the upper bound $2fd+Cd$, where d is message delivery time, and C is timing uncertainty [18].

4.3.9 Uniform Consensus

Goals: Uniform consensus is a more strict version of traditional consensus. Uniform consensus keeps the termination and validity of the consensus problem and adds a uniform agreement requirement: "every two processes (correct or faulty) that decide, decide on the same output" [28].

System environment: Uniform consensus described in [33] and [28] runs in a synchronous round-based environment.

Fault model: Idit Keidar, Sergio Rajsbaum [28] and Xianbing Wang et al. [33] proved that reaching uniform consensus in synchronous crash-prone system requires at least $f+2$ rounds (lower bound), if $0 \leq f \leq t-2$, here f – number of failures during protocol execution, t – maximum tolerated amount of failures. On the other hand, uniform consensus does not have solutions in Byzantine fault model [29]. Thus, most of the protocols that aim to solve uniform consensus would likely assume crash failures only.

Solutions: Let us name some of the consensus protocols that solve uniform consensus problem. [31] describes an algorithm that runs in a synchronous message-passing system with reliable channels, where processes may crash, and correct processes remain a majority. It is a condition-based protocol and it is solved "(1) in 1 round if $f=0$ and $l(I) \geq t$ holds, (2) within 2 rounds if $l(I) \geq f$ holds, and (3) within min $\{f+2-l(I), t+1\}$ rounds otherwise" [31]. Here $l(I)$ is a difficulty of input vectors, time complexity of the protocol is adapting to (depending on) its value. Authors claim this protocol to be the fastest from existing ones (as for 2004 when the paper was published). A year later the time complexity of the same protocol has been improved for $f = t$ and $l > f$ by one round [34].

4.3.10 Randomized Consensus

Motivation: Randomization is one of the solutions to avoid the impossibility consensus in an asynchronous system shown by Fischer et al. [7]. In 1983 a randomized Byzantine generals problem was considered [5], and the definition of the problem in the current thesis is based on this paper by Michael O. Rabin. This consensus variation was proposed as a solution to the original Byzantine generals problem [4].

System environment: The message-passing model is used for randomized consensus; the system is asynchronous but can be as well modified to synchronous one [5]. The current description is applied to the asynchronous model, if not stated otherwise. Randomization is assumed to be present in a system – the lottery model is described in the paper. Participants in consensus use authentication.

Goals: The requirements to reach randomized Byzantine consensus are the same as in the original Byzantine agreement problem and are described by the following definitions:

Agreement. Correct processes reach agreement if $message(i) = message(j)$ for all correct processes G_i and G_j [5].
Termination and Validity. "All the proper processes reach an agreement. If all proper G_i have the same initial message $M_i = M$ then the proper processes agree on M as the value of the message." [5]

Here $G_i, 1 \leq i \leq n$ are processes, and a proper message means the correct behavior of the process.

Fault model: The authors of the randomized consensus consider a Byzantine agreement model. Therefore faulty processes may behave arbitrarily.

Proposed solution/protocol: The original protocol proposed is BAP – Byzantine agreement protocol [5], is claimed to reach agreement in four rounds and requires cnt message exchanges in total, where n is the number of processes, t – faulty processes, c – a small constant. Also, according to the paper, synchronous case requires $n > 4t$ processes.

Improvements on the topic, more protocols: The randomized consensus has certainly been studied further, let us name one of the examples. One protocol was proposed by Berman and Garay in 1993 [14], it reaches agreement in asynchronous system with $n > 5t$

participants and $\theta(\log n)$ message complexity.

4.3.11 Approximate Consensus

An approximate consensus is also a variation of Byzantine generals problem. The definition is discussed in the context of [8] that was published in 1986.

Motivation: The motivation behind the approximate consensus is to reach agreement on approximate values within some margin, rather than the exact value. This consensus problem can be used, for example, for clock synchronization [8]. Comparing to Byzantine generals, an approximate algorithm may terminate in less than $t + 1$ rounds, furthermore, approximation allows achieving consensus in some cases, where exact consensus is not possible [8].

System environment: Dolev et al. [8] considers both synchronous and asynchronous model. Processes communicate through reliable channels by exchanging messages with each other.

Goals: Approximate consensus problem is solved when a protocol satisfies the conditions:

"*Agreement.* All nonfaulty processes eventually halt with output values that are within ϵ of each other.
Validity. The value output by each nonfaulty process must be in the range of values of the nonfaulty processes." [8]

Fault model: Dolev et al. [8] describes approximate consensus in the Byzantine fault model. The total number of processes must be $n > 5t$ in asynchronous case. For synchronous system it is $n > 3t$, these are the lower bounds.

Proposed solution/protocol: A proposed synchronous consensus algorithm successfully reaches an agreement with $n > 3t$ (optimal solution) [8]. An asynchronous protocol, defined in the same paper, terminates in $\log_c(\frac{\delta(V)}{\epsilon})$ rounds with $n > 5t$, plus initialization round, where V is multiset of values from initialization round, $\delta(V)$ is the diameter $\delta(V) = \max(V) - \min(V)$ and c is a factor $c = \frac{n-3t-1}{2t} + 1$ [8].

4.3.12 Strong Consensus

Motives behind the definition: The definition of strong consensus originates from Gil Neiger's article published in 1994 [17]. Strong consensus adds value in case of multi-valued consensus when input values are not only 0 or 1, but there are more options. This definition is more reasonable than the traditional version and the author claim that classical consensus validity condition fits fine to binary consensus while for multi-valued consensus strong validity is more appropriate.

System environment: Strong consensus problem is set in synchronous message passing model in a fully connected network with a known number of participants.

Goals: While keeping the original Termination and Agreement conditions to reach an agreement as it is, this problem modifies Validity into Strong Validity:

"*Strong Validity.* The output value of each correct processor must be the input value of some correct processor" [17].

Fault model: Correct process behaves according to protocol – send messages and updates its state following the specs. A faulty process is defined as the one that is not correct, and Byzantine faulty behavior is assumed. Gil Neiger's strong consensus requires $n > \max\{mt, 3t\}$ of total processes, where m is the number of values in multi-valued agreement, t – the number of faulty processes.

Proposed solutions: The adjusted version of exponential-time algorithm to solve Byzantine generals problem [4] and phase-king algorithm [13] solve strong consensus if $n > \max\{mt, 3t\}$ and $n > \max\{2mt, 4t\}$ respectively.

Further improvements: Hsiao et al. in 2002 [24] have added up a study on strong consensus in general network. By general network, authors mean that the network may not be fully connected as assumed in the original problem statement. Also, the failure model in this paper assumes the possibility of both processes and link failures. The motivation behind this paper is to model a more realistic and practical environment. Therefore, the updated fault model on link and process failures looks like following [24] :

Process failures – P_a arbitrary, P_s symmetric, P_m manifest;
Link failures – L_a arbitrary, L_m manifest.

Hsiao et al. as a solution to strong consensus problem in general network proposed generalized protocol with minimum $t + 1$ message exchange rounds for correct processes and $(t + 1)cn^2$ messages, and maximum number of faulty components.

There are a couple more protocols that solve strong consensus, but it is worth to mention the most recent to our knowledge article on SC. In 2012 eventual strong consensus has been introduced [43] with a focus on dynamic networks. Here time and space complexity have been improved for protocols, that satisfy strong consensus, with adding an early stopping rule.

The dynamic network reflects the development of network technologies over the past 25 years when the strong consensus was initially proposed. Chien-Fu Cheng and Kuo-Tang Tsai explicitly point on the outdated environmental assumptions regarding system settings of the previous studies on strong consensus: “Static networks built upon coaxial cables, twisted pair cables, and optical fibers have been gradually replaced by dynamic networks.” [43]

For eventual strong consensus the definitions of faults are kept the same as original – correct process must behave correctly, otherwise it is faulty. The authors classify faults into Byzantine and dormant faults: crash, omission or invariant and consider dual failure mode [43]. Regarding the technical properties of the new protocol, the researchers in [43] claim that $n > \max\{mf_m + f_d + f_a, 3f_m + f_d + f_a\}$ – constraints on total number of participants, where f_a – away processor, f_d – dormant processor, f_m – malicious processor, m – size of the domain range of initial values. The number or message exchange rounds required is $\min\{f_m + 2, t^{\#} + 1\}$, where $t^{\#} = \frac{(n-1-f_d-f_a)}{\max\{m,3\}} + 1$. This protocol is well-performing, the complexity is reduced comparing to the original paper.

From analyzing strong consensus problem, we can conclude that there is a need to keep up the consensus problem and protocols with the speed of technological advancement and update their environment assumptions and fault models, which would satisfy the realistic conditions.

4.3.13 Wait-free Consensus

This section is on a wait-free Byzantine consensus problem described in 2002 by Paul Attie [26]. This problem is one of the few problems in the shared-memory model studied in this thesis.

System model: Wait-free consensus model is different from all the previous ones in terms of process communication – instead of passing messages to each other, they perform operations on shared data objects [26].

Fault-model: The primary motivation of the author is to combine Byzantine consensus in asynchronous shared-memory settings. Naturally, the problem adopts Byzantine fault model, and the behavior of the fault may be malicious.

Conditions: To solve wait-free Byzantine consensus problem in the proposed model, a protocol must satisfy the further conditions, besides traditional Agreement and Validity:

"*Wait-free termination.* If process P_i is nonfaulty in an infinite fair execution α, then P_i decides exactly once in α.
Uniform-initial-state. Every combination of the initial local state for each process and initial local state for each shared object is a possible initial global state." [26]

A weak form of wait-free consensus contains weak validity requirement:

"*Weak validity.* If there are no faulty processes and all processes start with the same initial value $val \in V$, then val is the only possible decision value." [26]

Bounds, Impossibility results: There is no protocol in the paper but there are some impossibility results presented, for instance, only a weak form of wait-free consensus can be achieved and only if shared objects cannot be reset to the initial local state [26].

4.3.14 Lazy Consensus

Motivation: Lazy consensus problem was discussed in 2004 by Xavier Défago and André Schiper [30]. The motivation behind this problem mentioned in the paper is to allow processes to compute their initial values when needed. From the paper we can guess that lazy consensus works efficiently for semi-passive replication algorithm defined in the same paper [30], where the cost of input value computation is high, and lazy consensus allows to perform it when requested.

Environment: The problem is described in the asynchronous message-passing model. The impossibility result can be avoided with failure detectors [30].

Fault model: Xavier Défago and André Schiper define lazy consensus in crash/no recovery fault model, while a correct process is specified as the one that does not crash.

Conditions: Lazy consensus is achieved when uniform consensus is satisfied along with one of the invented laziness properties:

"*Termination.* Every correct process eventually decides some value.
Uniform integrity. Every process decides at most once.
Agreement. No two correct processes decide differently.
Uniform validity. If a process decides v, then v was proposed by some process.
Proposition integrity. Every process proposes a value at most once."[30]

The authors provide definitions for weak, quazi-strong and strong lazy consensus, therefore generate three new consensus problems.

"*Weak laziness.* If two processes p and q propose a value, then at least one of p and q is suspected by some process in the set of processes P_S.
Quasi-strong laziness. If two processes p and q propose a value, then p and q are not both correct.
Strong laziness. If a process p proposes a value, then no process q proposes a value before p has crashed unless q has crashed before p proposes a value." [30]

Protocol: The proposed solution is a protocol adapted from [20]. As in the original protocol, lazy consensus protocol requires the majority of the processes to be correct. Overall, all other assumptions remain the same.

4.3.15 Managed Agreement

A managed agreement was presented by Emmanuelle Anceaume et al. [36] in 2007. It is a concept for generalizing non-blocking atomic commit and consensus problem. The managed agreement is a family of problems, where consensus and NBAC are the two special cases of it.

Motivation: The motivation behind this new definition is not very clear to us, but it is probably about generalizing two significant problems in distributed computing.

System environment: The problem is set in the asynchronous system, message-passing model, where processes communicate via reliable channels. A protocol is equipped with one or multiple failure detectors.

Goals: To achieve managed agreement the following conditions must be satisfied:

"*(Uniform) Agreement.* No two processes decide differently.
Termination. Every correct process eventually decides on some value.
Managed-Obligation If the decision value is $M(Default)$, then either one of the aristocrats proposes $Default$ or crashes.
Managed-Justification If the decision value v is different from $M(Default)$, then v corresponds to a proposed value and all aristocrats propose a non default value." [36]

The interesting elements of the problem is a default value and a subset of participants called aristocrats. Managed agreement equals to NBAC when all nodes are aristocrats [36]. So, in case of NBAC, if any aristocrat proposes default("abort") or crashes, the decided value is default("abort"). The decision to commit is made when no aristocrat has crashed, and none of them proposed a default value. Regarding the traditional consensus problem, it is a case when there are no aristocrats in the system [36].

Fault model: Lazy consensus described in [36] uses crash model, where a process may crash permanently and $f < n$ processes may be faulty. A correct process follows the specs and does not crash.

Proposed solution/protocol: The authors introduce their own failure detector along with the schema of a generic managed agreement protocol, that would use the aforementioned failure detector and solve the managed agreement problem.

4.3.16 Anonymous Consensus

Motivation: In 2011 the anonymous consensus problem was studied by François Bonnet and Michel Raynal [64]. Anonymity often implies user's privacy, and researching this area is a reasonable motive.

System environment: The problem in [64] describes an asynchronous message-passing system. Processes are anonymous and communicate via reliable channels. The failure detector is also present in a system.

Goals: This consensus model is using the uniform agreement goals.

Fault model: The authors of the article claim to be the first ones who studied asynchronous anonymous consensus with crash failures. Processes may crash but not recover during the run, where $0 \leq t \leq n-1$ (at least one process must be correct) [64].

Proposed solution/protocol: The protocol in the paper terminates in $2t+1$ asynchronous rounds, an early-deciding version reaches consensus in min $\{2f+2, 2t+1\}$ rounds.

4.3.17 Anonymous Consensus in a Shared-Memory System

Motivation: In 2011 a Janus algorithm was presented, which solves the consensus in a shared-memory model with anonymous processors [42]. The particular interest of the study was to investigate the write step complexity of the specified consensus [42].

System environment: This consensus problem has an asynchronous shared-memory model with anonymous processes.

Goals: The authors mention $propose$ and $decide$ functions, which seems like a definition from Chandra et al. [20]. The following goals must be achieved to reach consensus: "*(Agreement)* two processes cannot decide different values; *(Validity)* if a process decides some value v, then v was proposed before; and *(Termination)* every correct process eventually decides." [42]

Fault model: It is assumed that processes can crash, but the system is equipped with a failure detector.

Proposed solution/protocol: The authors propose an anonymous consensus algorithm called Janus, which solves the previously defined problem in $O(\sqrt{n})$ individual writes [42].

4.3.18 Continuous Consensus

Motivation: Continuous consensus problem was introduced in 2008 by Tal Mizrahi and Yoram Moses [37]. They describe a problem of maintaining consistency of the information among nodes in distributed systems prone to failures. This problem is exciting to us, as it is not the typical problem of agreeing on a value. If searching for the roots of the problem in the literature, Tal Mizrahi and Yoram Moses explain: "Continuous consensus is a strict generalization of simultaneous agreement."[37]

System environment: The protocol to solve continuous consensus must run in a synchronous network, where processes communicate by sending messages.

Goals: Formally continuous consensus problem is presented as follows. Each process i maintains a copy of events E. A protocol to solve continuous consensus must satisfy the requirements:

"*Accuracy.* All events in $M_i[k]$ occurred in the run.
Consistency. $M_i[k] = M_j[k]$ at all times k.
Completeness. If an event $e \in E$ is known to process j at any point, then $e \in M_i[k]$ must hold at some time k." [37]

Here k is time step, $M_i[k]$ is a core list of events, that have to maintained by each correct process continuously.

Fault model: The paper describes continuous consensus in crash and omission fault models. Later in 2010 continuous consensus was studied by the same authors in a system with ambiguous failures, namely general omissions and authenticated Byzantine [41].

Proposed solution/protocol: The ConCon protocol is an optimal protocol that solves continuous consensus in crash and omission failure models [37].

Improvements on the topic, more protocols: In 2010 a few more protocols were considered by Tal Mizrahi and Yoram Moses [41]. The authors present authenticated continuous consensus protocol, which solves CC for $n > 3t$ processes. Another protocol proposed in the same paper called AccD is an early-stopping protocol and solves CC for $n > t$. These algorithms are claimed to be valid for both general omissions and authenticated Byzantine models.

4.3.19 Blockchain Consensus

Motivation: One of the latest consensus problems is blockchain consensus [50]. Vincent Gramoli in his paper points out that the main reasons of defining a separate definition for blockchain related protocols is that it is put more in the context of blockchain applications and it is claimed to improve the model in terms of scalability, on the contrast of using the classical Byzantine consensus. The blockchain consensus is specifically designed for the blockchain environment and, therefore, has more relaxed conditions than the Byzantine consensus.

Goals: The blockchain consensus also requires agreement, termination and validity conditions but the definitions are adapted to use the language of blockchain applications: "*Agreement:* no two correct processes decide different blocks; *Termination:* all correct processes eventually decide a block; *Validity:* a decided block is valid, it satisfies the predefined predicate valid." [50]

Solution: Vincent Gramoli does not offer his own solution to blockchain consensus in this paper. However, he mentions that one protocol already solves blockchain consensus, namely Democratic BFT [65].

5 Analysis

In the previous chapter, a systematic survey of consensus problems and protocols have been presented. Here the aim is to show our findings in a diagram and demonstrate how these problems are interconnected.

This survey is a collection of consensus problems for the past 40 years. To see the progress on consensus problems in each decade, a timeline of consensus problems is presented below, which shows the earliest (to our knowledge) study on this particular problem:

- 1980 – 1989: interactive consistency, Byzantine generals, uniform consensus, randomized Byzantine consensus, approximate consensus;
- 1990 – 1999: consensus with timing uncertainty, consensus by Chandra et al., strong consensus;
- 2000 – 2009: wait-free Byzantine consensus, lazy consensus, generalized consensus, managed agreement, Byzantine generals with alternative plans, anonymous Byzantine agreement, continuous consensus;
- 2010 – 2019: anonymous consensus, anonymous consensus in a shared-memory system, mortal Byzantine consensus, blockchain consensus, Byzantine consensus with unknown participants.

A consensus problem has many components. A formalized consensus problem must specify how the system looks like, how processes can behave and so on. However, one consensus problem is mainly distinguished from another by its conditions, which are required to reach an agreement. Termination, validity, agreement conditions and their extensions – these are the goals that define a consensus problem, and this is the main interest in this chapter.

Thus, a goal-based map of consensus definitions is presented (see fig. 5.1). This research has collected 17 problems in a message-passing system and 2 other consensus problems in a shared-memory model, 19 problems in total. This map contains all consensus problems in a message-passing model found during the systematic literature review. For a comprehensive taxonomy on consensus in a shared-memory model, more data than collected is needed, that is why these problems have not been included in the map, though they are still present in the survey. The rectangles on the map demonstrate the goals introduced by the problems and oval shapes are the consensus problems themselves. The arrows show the

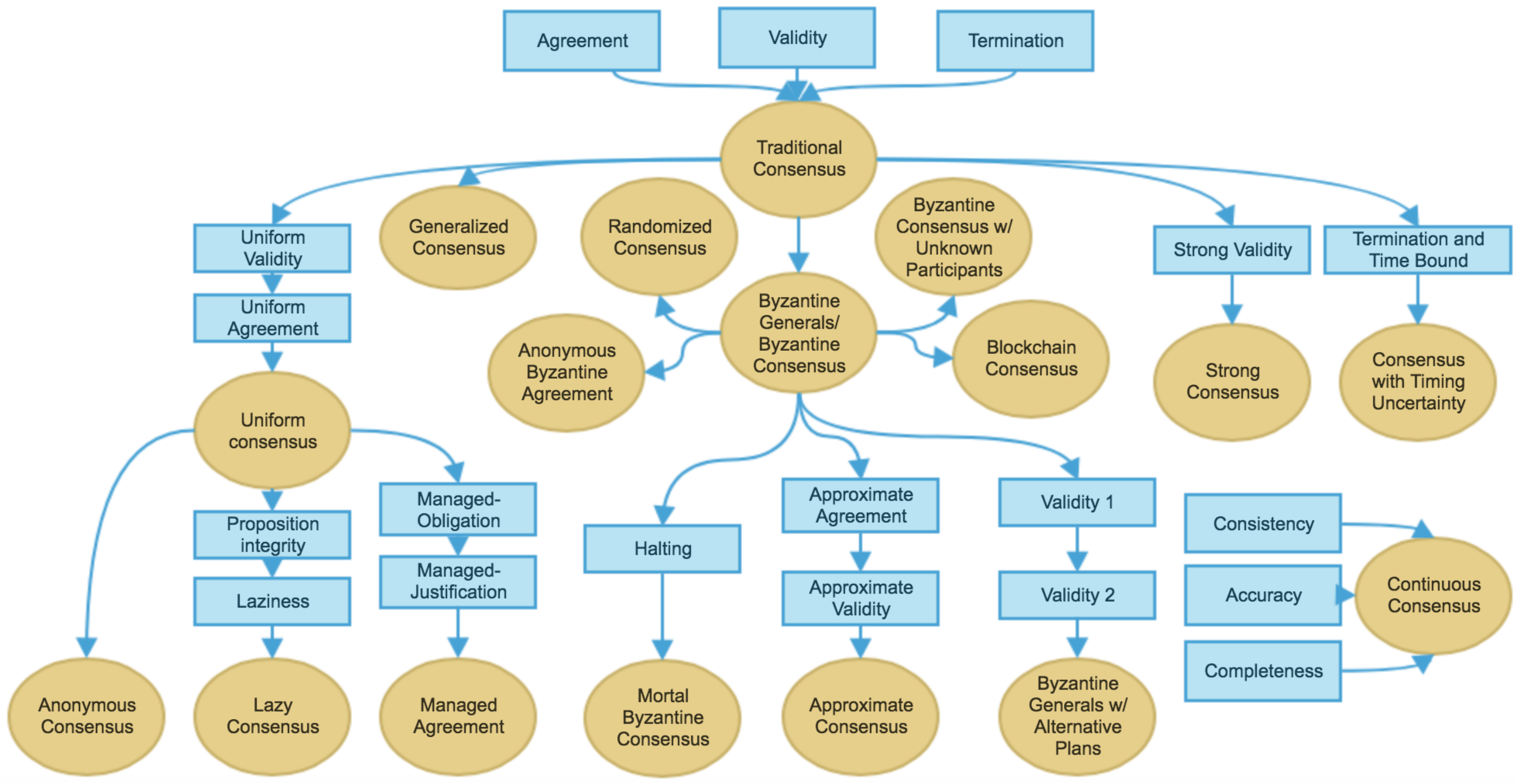

Figure 5.1: A goal-based map of consensus problems in message-passing model

links between problems and point to the additional goals defined by the child problems.

When aggregating the data, the attention is put on the links between the problems. Some of the papers explicitly state that the proposed problem extends the already existing consensus definition (e.g., it was noticed that many problems were variations of the Byzantine generals problem). On the other hand, it is possible to guess from the goals, that the new consensus problem solves, for example, the uniform consensus as well, so it is considered it as its extension.

From the diagram, one can see that a high amount of consensus problems extends the Byzantine consensus. The anonymous Byzantine agreement, randomized consensus, Byzantine consensus with unknown participants have the same goals, while mortal Byzantine consensus adds halting condition, approximate consensus redefines agreement and validity to reach consensus on the approximate value, and Byzantine generals with alternative plans plot their battle based on the newly introduced validity 1 and validity 2 requirements. The continuous consensus is defined by consistency, accuracy and completeness properties, which is different from other problems studied in this thesis, therefore the goal-links between continuous consensus and any other one were not established. When the problem uses whether a uniform agreement or uniform validity conditions, it is put to the uniform consensus branch. The generalized consensus is considered a weakened form of the traditional consensus but in the reviewed papers it was not clear what exact goals are specified, therefore on the map, it is only mentioned where the roots come from.

To summarize, this map is an additional contribution to this thesis, besides the survey. The visual representation shows how the consensus has evolved and what are the different goals proposed by the authors of the consensus problems. It reveals the connections between these problems, which might not be obvious from solely reviewing the literature.

6 Conclusion

Researchers strive to define a valid consensus problem, which would be based on realistic assumptions and reflect the modern system environment. Engineers are interested in having a protocol, which is secure, scalable and well-performed. With the amount of research done for the past 40 years, it makes sense to revisit consensus problems and protocols, see what has been achieved so far and make updates reflecting the technological development. The current thesis presents such a study.

This systematic literature review is research on distributed consensus, which brings the various consensus problems and protocols in one place. In total 52 papers on consensus are analized, they were published during 1980 – 2019 and 19 different consensus problems found, each of them serves its purpose. Additionally, a goal-based map of the consensus problems in message-passing model is presented.

For future research, more consensus problems in a shared-memory model can be reviewed. Another suggestion is to evaluate the models of the existing consensus problems. One could check if the system model reflects the nowadays network architecture, how measurable are the assumptions in the consensus problems, if it is realistic, for example, to assume $\frac{2n}{3}$ or $\frac{4n}{5}$ correct processors. Answering these questions could significantly improve the current state of distributed consensus. It could also bring returns in forms of practical protocols, that would serve its purpose not only on paper but also in real applications.

References

[1] B. Kitchenham, “Procedures for performing systematic reviews,” *Keele, UK, Keele Univ.*, vol. 33, Aug. 2004.

[2] M. Pease, R. Shostak, and L. Lamport, “Reaching agreement in the presence of faults,” *J. ACM*, vol. 27, no. 2, pp. 228–234, Apr. 1980. [Online]. Available: http://doi.acm.org/10.1145/322186.322188

[3] D. Dolev, H. R. Strong *et al.*, “Requirements for agreement in a distributed system,” in *DDB*, 1982, pp. 115–129.

[4] L. Lamport, R. Shostak, and M. Pease, “The byzantine generals problem,” *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982. [Online]. Available: http://doi.acm.org/10.1145/357172.357176

[5] M. O. Rabin, “Randomized byzantine generals,” in *Proceedings of the 24th Annual Symposium on Foundations of Computer Science*, ser. SFCS ’83. Washington, DC, USA: IEEE Computer Society, Nov 1983, pp. 403–409. [Online]. Available: https://doi.org/10.1109/SFCS.1983.48

[6] G. Bracha and S. Toueg, “Asynchronous consensus and broadcast protocols,” *Journal of the ACM (JACM)*, vol. 32, no. 4, pp. 824–840, Oct. 1985. [Online]. Available: http://doi.acm.org/10.1145/4221.214134

[7] M. J. Fischer, N. A. Lynch, and M. S. Paterson, “Impossibility of distributed consensus with one faulty process,” *Journal of the ACM (JACM)*, vol. 32, no. 2, pp. 374–382, Apr. 1985. [Online]. Available: http://doi.acm.org/10.1145/3149.214121

[8] D. Dolev, N. Lynch, S. Pinter, E. Stark, and W. Weihl, “Reaching approximate agreement in the presence of faults,” *Journal of the ACM (JACM)*, vol. 33, no. 3, pp. 499–516, May 1986. [Online]. Available: http://doi.acm.org/10.1145/5925.5931

[9] M. C. Loui and H. H. Abu-Amara, “Memory requirements for agreement among unreliable asynchronous processes,” in *Advances in Computing Research*, F. P. Preparata, Ed. JAI press, 1987, vol. 4, pp. 163–183.

[10] L. Patnaik and S. Balaji, “Byzantine-resilient distributed computing systems,” *Sadhana*, vol. 11, no. 1, pp. 81–91, Oct. 1987. [Online]. Available: https://doi.org/10.1007/BF02811312

[11] K. Abrahamson, "On achieving consensus using a shared memory," in *Proceedings of the Annual ACM Symposium on Principles of Distributed Computing*, ser. PODC '88. New York, NY, USA: ACM, 1988, pp. 291–302. [Online]. Available: http://doi.acm.org/10.1145/62546.62594

[12] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony," *Journal of the ACM (JACM)*, vol. 35, no. 2, pp. 288–323, Apr. 1988. [Online]. Available: http://doi.acm.org/10.1145/42282.42283

[13] P. Berman and J. A. Garay, "Asymptotically optimal distributed consensus," in *Automata, Languages and Programming*, G. Ausiello, M. Dezani-Ciancaglini, and S. R. Della Rocca, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1989, pp. 80–94.

[14] P. Berman and J. A. Garay, "Randomized distributed agreement revisited," in *FTCS-23 The Twenty-Third International Symposium on Fault-Tolerant Computing*. IEEE, June 1993, pp. 412–419. [Online]. Available: https://doi.org/10.1109/FTCS.1993.627344

[15] M. Barborak, A. Dahbura, and M. Malek, "The consensus problem in fault-tolerant computing," *ACM Comput. Surv.*, vol. 25, no. 2, pp. 171–220, Jun. 1993. [Online]. Available: http://doi.acm.org/10.1145/152610.152612

[16] V. Hadzilacos and S. Toueg, "A modular approach to fault-tolerant broadcasts and related problems," Ithaca, NY, USA, Tech. Rep., 1994.

[17] G. Neiger, "Distributed consensus revisited," *Information Processing Letters*, vol. 49, no. 4, pp. 195–201, Feb. 1994. [Online]. Available: http://dx.doi.org/10.1016/0020-0190(94)90011-6

[18] H. Attiya, C. Dwork, N. Lynch, and L. Stockmeyer, "Bounds on the time to reach agreement in the presence of timing uncertainty," *Journal of the ACM (JACM)*, vol. 41, no. 1, pp. 122–152, Jan. 1994. [Online]. Available: http://doi.acm.org/10.1145/174644.174649

[19] R. Guerraoui, "Revisiting the relationship between non-blocking atomic commitment and consensus," in *Distributed Algorithms*, J.-M. Hélary and M. Raynal, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 87–100. [Online]. Available: https://doi.org/10.1007/BFb0022140

[20] T. D. Chandra and S. Toueg, "Unreliable failure detectors for reliable distributed systems," *Journal Of The Acm*, vol. 43, no. 2, pp. 225–267, Mar. 1996. [Online]. Available: http://doi.acm.org/10.1145/226643.226647

[21] A. W. Krings and T. Feyer, "The byzantine agreement problem: optimal early stopping," in *Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences. 1999. HICSS-32. Abstracts and CD-ROM of Full Papers*, vol. 8, Jan 1999. [Online]. Available: https://doi.org/10.1109/HICSS.1999.772983

[22] F. C. Gärtner, “Fundamentals of fault-tolerant distributed computing in asynchronous environments,” *ACM Comput. Surv.*, vol. 31, no. 1, pp. 1–26, Mar. 1999. [Online]. Available: http://doi.acm.org/10.1145/311531.311532

[23] M. Nesterenko and A. Arora, “Dining philosophers that tolerate malicious crashes,” in *Proceedings 22nd International Conference on Distributed Computing Systems*. IEEE, July 2002, pp. 191–198. [Online]. Available: https://doi.org/10.1109/ICDCS.2002.1022256

[24] H. Hsiao, Y. Chin, and W. Yang, “Reaching strong consensus in a general network,” *Journal Of Information Science And Engineering*, vol. 18, no. 4, pp. 601–625, 2002.

[25] M. Castro and B. Liskov, “Practical byzantine fault tolerance and proactive recovery,” *Acm Transactions On Computer Systems*, vol. 20, no. 4, pp. 398–461, Nov. 2002. [Online]. Available: http://doi.acm.org/10.1145/571637.571640

[26] P. Attie, “Wait-free byzantine consensus,” *Information Processing Letters*, vol. 83, no. 4, pp. 221–227, 2002. [Online]. Available: https://doi.org/10.1016/S0020-0190(01)00334-9

[27] B. Charron-Bost, “Comparing the atomic commitment and consensus problems,” in *Future Directions in Distributed Computing: Research and Position Papers*, A. Schiper, A. A. Shvartsman, H. Weatherspoon, and B. Y. Zhao, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 29–34. [Online]. Available: https://doi.org/10.1007/3-540-37795-6_6

[28] I. Keidar and S. Rajsbaum, “A simple proof of the uniform consensus synchronous lower bound,” *Information Processing Letters*, vol. 85, no. 1, pp. 47–52, 2003. [Online]. Available: https://doi.org/10.1016/S0020-0190(02)00333-2

[29] B. Charron-Bost and A. Schiper, “Uniform consensus is harder than consensus,” *Journal of Algorithms*, vol. 51, no. 1, pp. 15–37, Apr. 2004. [Online]. Available: https://doi.org/10.1016/j.jalgor.2003.11.001

[30] X. Défago and A. Schiper, “Semi-passive replication and lazy consensus,” *J. Parallel Distrib. Comput.*, vol. 64, no. 12, pp. 1380–1398, Dec. 2004. [Online]. Available: http://dx.doi.org/10.1016/j.jpdc.2004.08.006

[31] T. Izumi and T. Masuzawa, “Synchronous condition-based consensus adapting to input-vector legality,” in *Distributed Computing*, R. Guerraoui, Ed., vol. 3274. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 16–29. [Online]. Available: https://doi.org/10.1007/978-3-540-30186-8_2

[32] L. Lamport, “Generalized consensus and paxos,” Tech. Rep., Mar. 2005. [Online]. Available: https://www.microsoft.com/en-us/research/publication/generalized-consensus-and-paxos/

[33] X. Wang, Y. M. Teo, and J. Cao, “A bivalency proof of the lower bound for uniform consensus,” *Information Processing Letters*, vol. 96, no. 5, pp. 167–174, 2005. [Online]. Available: https://doi.org/10.1016/j.ipl.2005.08.002

[34] T. Izumi and T. Masuzawa, “An improved algorithm for adaptive condition-based consensus,” in *Structural Information And Communication Complexity. Lecture Notes in Computer Science*, A. Pelc and M. Raynal, Eds., vol. 3499. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 170–184. [Online]. Available: https://doi.org/10.1007/11429647_15

[35] J.-P. Martin and L. Alvisi, “Fast byzantine consensus,” *IEEE Transactions on Dependable and Secure Computing*, vol. 3, no. 3, pp. 202–215, 2006. [Online]. Available: https://doi.org/10.1109/TDSC.2006.35

[36] E. Anceaume, R. Friedman, and M. Gradinariu, “Managed agreement: Generalizing two fundamental distributed agreement problems,” *Information Processing Letters*, vol. 101, no. 5, pp. 190—198, 2007. [Online]. Available: https://doi.org/10.1016/j.ipl.2006.10.001

[37] T. Mizrahi and Y. Moses, “Continuous consensus via common knowledge,” *Distributed Computing*, vol. 20, no. 5, pp. 305–321, Feb. 2008. [Online]. Available: https://doi.org/10.1007/s00446-007-0049-6

[38] M. Okun and A. Barak, “Efficient algorithms for anonymous byzantine agreement,” *Theory of Computing Systems*, vol. 42, no. 2, pp. 222–238, 2008. [Online]. Available: https://doi.org/10.1007/s00224-007-9006-9

[39] M. Correia, A. N. Bessani, and P. Veríssimo, “On byzantine generals with alternative plans,” *Journal of Parallel and Distributed Computing*, vol. 68, no. 9, pp. 1291–1296, 2008. [Online]. Available: https://doi.org/10.1016/j.jpdc.2008.04.008

[40] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, “Zyzzyva: Speculative byzantine fault tolerance,” *ACM Transactions on Computer Systems (TOCS)*, vol. 27, no. 4, pp. 1–39, 2009. [Online]. Available: http://doi.acm.org/10.1145/1658357.1658358

[41] T. Mizrahi and Y. Moses, “Continuous consensus with ambiguous failures,” *Theoretical Computer Science*, vol. 411, no. 34, pp. 3031–3041, 2010. [Online]. Available: https://doi.org/10.1016/j.tcs.2010.04.025

[42] Z. Bouzid, P. Sutra, and C. Travers, “Anonymous agreement: The janus algorithm,” in *OPODIS’11 - 15th International Conference On Principles Of Distributed Systems*, ser. Lecture Notes in Computer Science, vol. 7109. Springer, 2011, pp. 175–190. [Online]. Available: https://doi.org/10.1007/978-3-642-25873-2_13

[43] C.-F. Cheng and K.-T. Tsai, “Eventual strong consensus with fault detection in the presence of dual failure mode on processors under dynamic networks,” *Journal of Network and Computer Applications*, vol. 35, no. 4, pp. 1260–1276, 2012. [Online]. Available: https://doi.org/10.1016/j.jnca.2012.01.011

[44] D. Ramesh and C. Kumar, “An optimal novel byzantine agreement protocol (onbap) for heterogeneous distributed database processing systems,” *Procedia Technology*, vol. 6, pp. 57–66, 2012. [Online]. Available: https://doi.org/10.1016/j.protcy.2012.10.008

[45] J. Widder, M. Biely, G. Gridling, B. Weiss, and J. Blanquart, “Consensus in the presence of mortal byzantine faulty processes,” *Distributed Computing*, vol. 24, no. 6, pp. 299–321, 2012. [Online]. Available: http://doi.org/10.1007/s00446-011-0147-3

[46] C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, A. Kermarrec, E. Ruppert, and H. Tran-The, “Byzantine agreement with homonyms,” *Distributed Computing*, vol. 26, no. 5-6, pp. 321–340, 2013. [Online]. Available: https://doi.org/10.1007/s00446-013-0190-3

[47] G. Veronese, M. Correia, A. Bessani, L. Lung, and P. Verissimo, “Efficient byzantine fault-tolerance,” *IEEE Transactions On Computers*, vol. 62, no. 1, pp. 16–30, 2013. [Online]. Available: https://doi.org/10.1109/TC.2011.221

[48] C.-F. Cheng and K.-T. Tsai, “A recursive byzantine-resilient protocol,” *Journal of Network and Computer Applications*, vol. 48, pp. 87–98, 2015. [Online]. Available: https://doi.org/10.1016/j.jnca.2014.10.010

[49] A. Mostéfaoui, H. Moumen, and M. Raynal, “Signature-free asynchronous binary byzantine consensus with t < n/3, o(n2) messages, and o(1) expected time,” *J. ACM*, vol. 62, no. 4, pp. 31:1–31:21, Sep. 2015. [Online]. Available: http://doi.acm.org/10.1145/2785953

[50] V. Gramoli, “From blockchain consensus back to byzantine consensus,” *Future Generation Computer Systems*, 2017. [Online]. Available: https://doi.org/10.1016/j.future.2017.09.023

[51] M. Pires, S. Ravi, and R. Rodrigues, “Generalized paxos made byzantine (and less complex),” in *Stabilization, Safety, and Security of Distributed Systems*, P. Spirakis and P. Tsigas, Eds. Cham: Springer International Publishing, 2017, pp. 203–218. [Online]. Available: https://doi.org/10.1007/978-3-319-69084-1_14

[52] E. A. P. Alchieri, A. Bessani, F. Greve, and J. D. S. Fraga, “Knowledge connectivity requirements for solving byzantine consensus with unknown participants,” *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 2, pp. 246–259, Mar. 2018. [Online]. Available: https://doi.org/10.1109/TDSC.2016.2548460

[53] J. Liu, W. Li, G. Karame, and N. Asokan, “Scalable byzantine consensus via hardware-assisted secret sharing,” *IEEE Transactions on Computers*, vol. 68, no. 1, pp. 139–151, Jan. 2019. [Online]. Available: https://doi.org/10.1109/TC.2018.2860009

[54] M. J. Fischer, “The consensus problem in unreliable distributed systems (a brief survey),” in *Foundations of Computation Theory*, M. Karpinski, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1983, pp. 127–140. [Online]. Available: https://doi.org/10.1007/3-540-12689-9_99

[55] R. Guerraoui, M. Hurfin, A. Mostefaoui, R. Oliveira, M. Raynal, and A. Schiper, “Consensus in asynchronous distributed systems: A concise guided tour,” *Advances In Distributed Systems. Lecture Notes in Computer Science*, vol. 1752, pp. 33–47, 2000. [Online]. Available: https://doi.org/10.1007/3-540-46475-1_2

[56] M. Raynal, “Consensus in synchronous systems: a concise guided tour,” in *2002 Pacific Rim International Symposium on Dependable Computing, 2002. Proceedings.*, Dec 2002, pp. 221–228. [Online]. Available: https://doi.org/10.1109/PRDC.2002.1185641

[57] L. S. Sankar, M. Sindhu, and M. Sethumadhavan, “Survey of consensus protocols on blockchain applications,” in *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Jan 2017, pp. 1–5. [Online]. Available: https://doi.org/10.1109/ICACCS.2017.8014672

[58] A. Narayanan and J. Clark, “Bitcoin’s academic pedigree,” *Commun. ACM*, vol. 60, no. 12, pp. 36–45, Nov. 2017. [Online]. Available: http://doi.acm.org/10.1145/3132259

[59] S. Chaudhuri, “Agreement is harder than consensus: Set consensus problems in totally asynchronous systems,” in *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, ser. PODC ’90. New York, NY, USA: ACM, 1990, pp. 311–324. [Online]. Available: http://doi.acm.org/10.1145/93385.93431

[60] IBM, “Ibm blockchain,” [Accessed Mar. 16, 2019]. [Online]. Available: https://www.ibm.com/blockchain

[61] F. B. Schneider, “Implementing fault-tolerant services using the state machine approach: A tutorial,” *ACM Comput. Surv.*, vol. 22, no. 4, pp. 299–319, Dec. 1990. [Online]. Available: http://doi.acm.org/10.1145/98163.98167

[62] I. Abraham, D. Malkhi, K. Nayak, L. Ren, and A. Spiegelman, “Solida: A Blockchain Protocol Based on Reconfigurable Byzantine Consensus,” in *21st International Conference on Principles of Distributed Systems (OPODIS 2017)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), J. Aspnes, A. Bessani, P. Felber, and J. Leitão, Eds., vol. 95. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, pp. 25:1–25:19. [Online]. Available: http://dx.doi.org/10.4230/LIPIcs.OPODIS.2017.25

[63] T. D. Chandra, R. Griesemer, and J. Redstone, “Paxos made live: An engineering perspective,” in *Proceedings of the Twenty-sixth Annual ACM Symposium on Principles of Distributed Computing*, ser. PODC ’07. New York, NY, USA: ACM, 2007, pp. 398–407. [Online]. Available: http://doi.acm.org/10.1145/1281100.1281103

[64] F. Bonnet and M. Raynal, “The price of anonymity: Optimal consensus despite asynchrony, crash, and anonymity,” *ACM Trans. Auton. Adapt. Syst.*, vol. 6, no. 4, pp. 23:1–23:28, Oct. 2011. [Online]. Available: http://doi.acm.org/10.1145/2019591.2019592

[65] T. Crain, V. Gramoli, M. Larrea, and M. Raynal, “(leader/randomization/signature)-free byzantine consensus for consortium blockchains,” *CoRR*, vol. abs/1702.03068, 2017.

List of Figures

List of Tables