Proceedings of the 16th Conference on Computer Science and Intelligence Systems pp. 291–300
DOI: 10.15439/2021F38
ISSN 2300-5963 ACSIS, Vol. 25

# Bernoulli Meets PBFT: Modeling BFT Protocols in the Presence of Dynamic Failures

Martin Nischwitz, Marko Esche
Physikalisch-Technische Bundesanstalt
Berlin, Germany
Email: {martin.nischwitz, marko.esche}@ptb.de

Florian Tschorsch
Department for Distributed Security Infrastructures
Technische Universität Berlin
Berlin, Germany
Email: florian.tschorsch@tu-berlin.de

***Abstract*—The publication of the pivotal state machine replication protocol PBFT laid the foundation for a body of BFT protocols. We introduce a probabilistic model for evaluating BFT protocols in the presence of dynamic link and crash failures. The model is derived from the communication pattern, facilitating an adaptation to other protocols. The state of replicas is captured and used to derive the success probability of the protocol execution. To this end, we examine the influence of link and crash failure rates as well as the number of replicas. A comparison in protocol behavior of PBFT, Zyzzyva and SBFT is performed.**

## I. Introduction

THE rapidly increasing connectivity of devices, as for example envisioned by the Internet of things, entices the development of large-scale, globally distributed systems. The European Metrology Cloud project [1], which aims to coordinate the digital transformation of legal metrology, is a prime example. The scale and complexity of such systems leads to a higher risk of failure and/or malicious behavior. The demand for trust and reliability, however, remains unchanged.

A technique to offer higher fault tolerance and availability for distributed systems is state machine replication (SMR). It requires processes to find agreement on the order of state transitions and, thus, consensus on the system state. One of the most prominent Byzantine fault tolerant (BFT) protocols is Practical Byzantine Fault Tolerance (PBFT) [2]. Many modern systems, including the recent surge of blockchain applications [3], utilize PBFT, or a variation of it, as their core consensus algorithm, e.g., BFT-SMaRt [4], Tendermint [5], RBFT [6], CheapBFT [7], and Hyperledger Fabric v0.6 [8].

While many advanced BFT protocols exist, the impact of dynamic failures in general and unreliable links in particular is often ignored. Many protocols, however, require that messages arrive within a defined timespan, i.e., there is a bound on the message delay. If that bound is not met, the performance of the protocols might deteriorate. To the best of our knowledge, there is unfortunately no technique to assess the impact of unreliable links on the performance of BFT protocols without requiring a comprehensive implementation. Instead, benchmarks on either real systems or simulations are deployed.

In this paper, we fill the gap and present a probabilistic modeling approach for BFT protocols to measure the impact of dynamic link and crash failures on their performance. The model is derived from the communication pattern and therefore transferable to many BFT protocols. It predicts the system state assuming the so-called dynamic link failure model [9], that is, unreliable communication links with message losses and high delays. More specifically, we assume a constant failure probability for all links and processes, model state transitions as Bernoulli trials, and express the resulting system state as probability density functions. Thus, our model can provide feedback already during the design and development phase of BFT protocols as well as support to parameterize timeouts.

Our model validation confirms that the model accurately predicts the probability for successful protocol executions of PBFT as well as BFT-SMaRt [4]. Moreover, we employ our model to Zyzzyva [10], and SBFT [11] to showcase its applicability to other BFT protocols. In our evaluation, we analyze the mentioned protocols, most notably PBFT, with respect to the impact of various failures and protocol stability. Accordingly, the paper's contributions can be summarized as follows:

- We develop a probabilistic model for PBFT to quantify the performance impact in the presence of dynamic link and crash failures. Since the model is based on communication patterns, it is implementation independent.
- We generalize our modeling approach and show that it can be applied to other BFT protocols.
- We validate the approach in a study by comparing it to a simulation of PBFT and BFT-SMaRt and apply it to Zyzzyva and SBFT.
- We identify critical values for dynamic link failure and crash failure rates at which the previously mentioned protocols become unstable.

The remainder of the paper is organized as follows. In Section II, we discuss related work with a focus on BFT modeling and failures. In Section III, we define the system model. Next, we describe the detailed derivation of our modeling approach for PBFT in Section IV, and present a simulation-based model validation in Section V. In Section VI, we use our model to reveal structural differences between PBFT, BFT-SMaRt, Zyzzyva, and SBFT, before we conclude the paper in Section VII.

## II. Related Work

### A. Preliminaries

The main properties of BFT protocols are described by the notions of *safety* and *liveness* [2]. Safety indicates that the protocol satisfies serializability, i.e., it behaves like a centralized system. Liveness, on the other hand, indicates that the system will eventually respond to requests. In order to tolerate $f$ faulty processes, at least $3f + 1$ processes are necessary [12]. Aside from process-related failures, the network, i.e., the communication between processes, also impacts the performance of BFT protocols and is often overlooked.

The network can be described as either synchronous or asynchronous. To bridge the gap between completely synchronous/asynchronous systems, the term *partially synchronous* was introduced [13]. A partially synchronous system may start in an asynchronous state but will, after some unspecified time, eventually return to a synchronous state. This captures temporary link failures, for example. A different perspective on a partially synchronous system is to assume a network with fixed upper bounds on message delays and processing times, where both are unknown a priori. The partially synchronous system model is utilized by many BFT protocols, e.g., PBFT, to circumvent the FLP impossibility [14] and guarantee liveness during the synchronous states of the system, without requiring it at all times. Deterministic BFT protocols guarantee safety, even in the asynchronous state, but require synchronous periods to guarantee liveness.

To detect Byzantine behavior, most BFT protocols utilize timeouts (and signatures). If the happy path of a protocol fails to make progress, a sub-protocol, e.g., a view change protocol, is triggered to recover [15]. To optimize performance, the timeout values should depend on the bounded message delay in the synchronous periods of the network, which plays an important role for deployments [16]. In addition to message delay characteristics, some networks, e.g., wireless networks, might be susceptible to link failures, leading to message omissions or corruptions. These failures are formally captured by the so-called *dynamic link failure model* [9], where the authors prove that consensus is impossible in a synchronous system with an unbounded number of transmission failures. Schmid et al. [17], [18] introduced a hybrid failure model to capture process and communication failures and derived bounds on the number of failures for synchronous networks.

### B. BFT Models

Other modeling techniques to analyze the performance of fault tolerant systems have previously been proposed in the literature. The framework HyPerf [19] combines model checking and simulation techniques to explore the possible paths in BFT protocols. While model checking usually proves correctness, their framework uses simulations to explore the possible paths in the model checker and evaluate the performance of the protocol. The model is validated against an implementation of PBFT to predict latencies and throughput.

A method to model PBFT with Stochastic Reward Nets (SRNs) was proposed in [20]. The authors deployed Hyperledger Fabric v0.6, which implements PBFT, and evaluate the mean time to consensus against the number of nodes in the system.

Singh et al. [21] provided the simulation framework BFTSim to evaluate BFT protocols. It builds upon the high-level declarative language P2 to implement three different BFT protocols [2], [10], [22] as well as ns-2, to explore various network conditions.

While the previously listed works offer the possibility to evaluate BFT protocols, they all require *comprehensive implementations* of the respective protocol. The model presented in this paper, however, is derived from the *communication pattern*. Moreover, no simulations or measurements are required to employ our model; all system states can be evaluated with closed-form expressions at low computational cost. Finally, the main focus of our work is to present a model for fault tolerant protocols that captures the impact of unreliable communication and varying message delays, which the other models only considered as a minor aspect.

### C. Link and Crash Failures

Fathollahnejad et al. [23] examined the impact of link failures on their leader election algorithm in a traffic control system to predict the probability for disagreement, based on Bernoulli trials. As in our paper, the number of received messages of an all-to-all broadcast is modeled with Bernoulli trials. Their protocol, however, does not require the consecutive collection of quorums which is implemented in most fault tolerant (FT) protocols and thus the main focus of the model presented in this paper.

Xu et al. presented RATCHETA [24], a consensus protocol which was designed for embedded devices in a wireless network that might be prone to dynamic link failures. They included an evaluation with artificially induced packet losses, measuring the number of failing consensus instances. RATCHETA requires a trusted subsystem that prevents a process from casting differing votes during the same consensus instance, eliminating the possibility of equivocations. It therefore yields a $2f + 1$ resilience, allowing $f$ Byzantine failures.

In addition, there is a body of literature that covers the theoretical limits of failures of consensus protocols [18], [25], [26], which are based on a hybrid failure model [17] and therefore also capture link failures. Existing models, however, rarely consider the actual impact of unreliable network conditions, such as dynamic crash and link failures, on the protocol algorithm. Since all BFT protocols have a built-in protocol to recover from crashed processes, e.g., view changes, their impact on the performance is tied to the frequency of the recovery algorithm execution.

## III. System Model

### A. Process Model

The distributed system consists of a fixed number of $n$ processes (we use the term process, node, and replica interchangeably). Typically, no more than $f$ processes are allowed to be subject to Byzantine faults and $n \geq 3f + 1$ replicas are required to guarantee safety [12].

To tolerate Byzantine (or crash) failures in an asynchronous setting, distributed systems rely on timeouts in combination with message thresholds to make progress. Consequently, it is common to describe the protocols in phases, i.e., system states in which each process, e.g., awaits the reception of a certain amount of messages or, alternatively, a timeout. Our model is time-free in that all events are mapped to the respective phases of the protocol.

In order to capture diverse failure cases, e.g., congestion due to high traffic load, we introduce the term *dynamic crash failures*, along the lines of the *dynamic link failure* model by Santoro et al. [9]. That is, processes can become unavailable in each phase. It is assumed that every crashed process will recover almost immediately, upholding its pre-crash state, and may thus be available in the next phase of the protocol. Since most BFT protocols (including PBFT) are based on consecutive phases, a crashed replica will remain inactive until the protocol-specific recovery algorithm, e.g., view-change protocol for PBFT, has recovered crashed replicas. In this paper, dynamic crash failures are assumed to be independent and identically distributed (i.i.d.) random variables for all processes during each phase.

Since our model is derived from the communication pattern of the protocol, special roles such as the primary in PBFT which follow a different communication pattern, are incorporated into the model.

### B. Network Model

We assume that each network node has a peer-to-peer connection to all other nodes. The network model in this paper allows for (i) messages to be delayed indefinitely, i.e., past the configured timeout parameter of PBFT, and (ii) message omissions as well as corruptions, as they may appear in e.g. wireless networks. The former case acknowledges BFT protocols that rely on synchronous periods to guarantee liveness and are based on timeouts to detect process and/or link failures. PBFT, for example, makes use of timeouts to detect if progress is being made and as a consequence to initiate the view-change protocol. Messages that arrive after a configured timeout can therefore be considered as message omissions. The same applies to invalid or corrupted messages. The resulting failure model can be described with the *dynamic link failure model* [9]. While in practice many BFT protocols rely on the network layer to guarantee reliable communication, e.g., TCP, they should implement means to handle lost messages due to crashed or malicious processes. We therefore assume unidirectional links, which implies unreliable communication.

If assumptions made regarding the bound of message delays fail, i.e., the timeouts are not configured appropriately, the protocol can be considered to operate in an asynchronous network with unbounded message delays. This does not apply if an attacker is considered to have control over the scheduling of messages, as this could easily lead to stopping a BFT protocol altogether [27]. As with process failures, the link failures are assumed to be i.i.d. for all links.

## IV. Modeling PBFT

The model presented in this section offers means to evaluate PBFT in the presence of dynamic link failures and crash failures. For the sake of clarity, we provide an overview of our modeling approach and introduce our notation first. Next, we unroll our model for various failure types step-by-step starting with dynamic crash failures, before we incorporate dynamic link failures.

### A. Overview

In PBFT, the happy path consists of five phases of message exchanges, as depicted in Figure 1. The first and last phase consist of transmissions from and to the client. In the first phase, the leader of the current view will collect and serialize client requests. This is followed by a phase in which the primary will disseminate the requests to all other replicas in so-called `pre-prepare` messages. If a replica receives and accepts a `pre-prepare` message, it stores that message and enters the third phase, broadcasting and collecting a quorum, i.e., at least $2f + 1$, of `prepare` messages that match the stored `pre-prepare` message. The fourth phase mirrors the third phase, except with `commit` messages. If a quorum of valid `commit` messages is collected, the node will commit (and execute) the state transition. In the fifth and last phase of the protocol, replicas reply to the client, confirming that the client's request was executed from the replicated system.

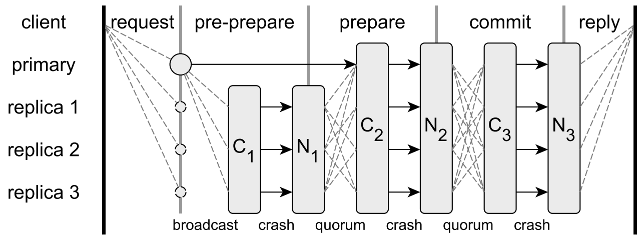


Fig. 1: Modeled view on PBFT's happy path communication pattern. Each phase is modeled via alternating predictions for crash ($C_i$) and link failures ($N_i$).

Omitting client interactions, the first and last phase can be disregarded and PBFT's happy path can be reduced to three phases. These phases can be summarized as a broadcast phase and two quorum collection phases. In the following, we assume that the primary is ready to initiate the consensus algorithm. Consequently, only the communication between the replicas is captured in our model.

In each phase of the protocol, the communication between and the availability of replicas is modeled as a combination of Bernoulli trials. More specifically, we model link and crash failures in alternating rounds for each of PBFT's phases, as depicted at the bottom of Figure 1. We use random variables $N_i$ and $C_i$ to express the success probabilities for the respective failure type in phase $i$.

In a first step, only faulty nodes are modeled as crash failures in a series of interdependent Bernoulli trials, i.e., $N_1 \rightarrow N_2 \rightarrow N_3$. In a second step, we extend the model by incorporating link failures. The communication is modeled along the lines of the three transmission phases $C_1$, $C_2$, and $C_3$. Combined with the node failures, our model yields an interleaving series of dependent system states, i.e., $C_1 \rightarrow N_1 \rightarrow C_2 \rightarrow N_2 \rightarrow C_3 \rightarrow N_3$.

In summary, the system state of all replicas at each protocol phase is captured by a series of probability density functions (PDFs), each constituting the calculation of the following. Please note, that each PDF allows for precise prediction of the protocol behavior and can be transformed into more common performance metrics, e.g., latency, with statistics or other models that predict the duration of individual phases.

## B. Notation

In Table I, we summarize relevant probabilities, events, and random variables, which are used in our model. For ease of comprehension, the link and crash failure distributions are now reduced to single probabilities, i.e., $p_l$ and $p_c$, respectively. The assumption to have identical link failure probabilities for all links is not an uncommon practice in this field of research [21], [23], [24]. The system state of the protocol is modeled by calculating PDFs that describe each replica's state. To this end, the random variables and events listed in Table I are indexed according to PBFT's phases. In particular, $C_1$, $C_2$, and $C_3$ represent the number of replicas that received a `pre-prepare` message, received a quorum of `prepare` messages and received a quorum of `commit` messages, respectively. Additionally, the number of active replicas after each phase is described with $N_1$, $N_2$, and $N_3$. Due to the nature of the PBFT algorithm, the distributions are dependent on each other, i.e., a replica that crashed or failed to collect the required messages will not be able to complete the happy path.

A key building block of our model are Bernoulli trials. Therefore, we use the notation $B(n,p,k) = \binom{n}{k} \cdot p^k \cdot (1-p)^{n-k}$ to express the probability to get exactly $k$ successes in a Bernoulli experiment with $n$ trials and a success probability of $p$. Furthermore, we define $B(n,p,[k,l]) = \sum_{i=k}^{l} B(n,p,i)$ as the sum over all Bernoulli trials with at least $k$ and up to $l$ successes. Finally, the notations $P_X(x) = P(X = x)$ and $P_{X|Y}(x|y) = P(X = x|Y = y)$ are used to abbreviate (conditional) probabilities.

## C. Modeling crash failures

We start by modeling one of the most prominent failures, crash failures. In particular, a crash failure implies no participation of the crashed process in the phase in which the crash occurred. Hence, the replica will neither receive nor send any messages. In the following, the three random variables $N_1, N_2$, and $N_3$ are derived for a crash failure probability $p_c$, assuming reliable communication links.

The happy path of PBFT is initiated with the primary broadcasting a `pre-prepare` message to all other replicas.

TABLE I: Model notation for PBFT.

| Symbol | Description | Range |
|---|---|---|
| | PROBABILITIES | |
| $p_c$ | Probability for a crash failure. | $[0,1]$ |
| $p_l$ | Probability for a link failure. | $[0,1]$ |
| | EVENTS | |
| $C_p$ | Indicates a successful reception of a quorum of `prepare` messages at the current primary. | |
| | RANDOM VARIABLES | |
| $C_1$ | The number of replicas that have received a `pre-prepare` message. | $[0,n-1]$ |
| $N_1$ | The number of replicas, excluding the primary, that did not crash in the `pre-prepare` phase. | $[0,n-1]$ |
| $C_{2,n}$ | The number of replicas, excluding the primary, that received a `pre-prepare` message as well as collected a quorum of `prepare` messages. | $[0,n-1]$ |
| $C_2$ | The number of replicas that received a `pre-prepare` message as well as collected a quorum of `prepare` messages. | $[0,n]$ |
| $N_2$ | The number of replicas that did not crash in the `prepare` phase. | $[0,n]$ |
| $C_3$ | The number of replicas that have received a `pre-prepare` message as well as collected a quorum of both, `prepare` and `commit` messages. | $[0,n]$ |
| $N_3$ | The number of replicas that have did not crash in the `commit` phase and successfully executed the algorithm. | $[0,n]$ |

For the sake of simplicity, it is assumed the primary cannot crash during the `pre-prepare` phase. Since the probability for a crash is uniform across all nodes, the PDF of the still active replicas $N_1$ is given by $P_{N_1}(n_1) = B(n, 1-p_c, n_1)$. The distribution of $N_1$ describes the number of replicas that will now broadcast `prepare` messages to all other replicas in the second phase of the protocol.

Following this procedure, the distribution of active nodes in further phases is calculated conditioned on the previous phase, meaning

$$P_{N_i}(n_i) = \sum_{n_{i-1}} B(n_{i-1}, 1-p_c, n_i) \cdot P_{N_{i-1}}(n_{i-1}). \qquad (1)$$

for $i = 2, 3$. Adding up the values of $P(N_3 \geq 2f+1)$ allows to predict the success probability for the happy path of PBFT. As replicas cannot skip a phase in PBFT, a crashed replica will not recover during the happy path rendering dynamic crashes similar to permanent ones.

## D. Modeling crash and link failures

We now extend our model by introducing link failures, i.e., the links are no longer considered reliable and are subject to a link failure probability $p_l$. The three random variables $C_1, C_2$, and $C_3$ are introduced to model the behavior of the protocol during the three communication phases. Due to the special behavior of the primary in the second phase, $C_2$ is divided into the event $C_p$ and random variable $C_{2,n}$ to capture the communication of the primary and other replicas, respectively. Assuming the dynamic link failure model, all links are subject to the same failure probability $p_l$ and can be, as with crash failures before, described with Bernoulli trials. In the following, we therefore start alternating between $C_i$ and $N_i$ (cf. Figure 1) to model the success of the message delivery and node availability, respectively.

*Calculating $C_1$:* The primary broadcasts a `pre-prepare` message to all other replicas. Since the success probability for each message transmission is equal to $1-p_l$ and independent of other transmissions, the number of successful transmissions can be calculated with a Bernoulli trial. The PDF of $C_1$ is given by $P_{C_1}(c_1) = B(n-1, 1-p_l, c_1)$ and describes the number of replicas that have received a `pre-preapre` message from the primary.

*Calculating $N_1$:* Based on the distribution of $C_1$, some replicas might crash in this phase, leading to

$$\begin{aligned} P_{N_1}(n_1) &= \sum_{c_1=0}^{n-1} P_{N_1|C_1}(n_1 \,|\, c_1) \cdot P_{C_1}(c_1) \\ &= \sum_{c_1=0}^{n-1} B(c_1, 1-p_c, n_1) \cdot P_{C_1}(c_1). \end{aligned} \tag{2}$$

*Calculating $C_2$:* The communication in the second phase is composed of the following: (i) whether the primary can collect $2f$ `prepare` messages (i.e., event $C_p$) (ii) the number of non-primary replicas that collect at least $2f+1$ `prepare` messages (i.e., $C_{2,n}$).

The primary can only collect at least $2f$ `prepare` messages if at least $2f$ active replicas have received the previous `pre-prepare` message, i.e., $N_1 \geq 2f$. In this case, at least $2f$ transmissions of `prepare` messages of the $N_1$ replicas have to successfully reach the primary. This can be expressed as the sum over all favorable Bernoulli trials, i.e., all trials with at least $2f$ successes out of $N_1$. The conditional probability $P(C_p \,|\, N_1 = n_1)$ for the primary to collect the `prepare` message is given by

$$P(C_p \,|\, N_1 = n_1) = \begin{cases} 0, & n_1 < 2f \\ B(n_1, 1-p_l, [2f, n]) & \text{otherwise.} \end{cases} \tag{3}$$

For a non-primary node, i.e., a replica, to advance to $C_2$, two requirements need to be met: (i) the replica has received a respective `pre-prepare` message, and (ii) the replica has collected a quorum of matching `prepare` messages. For a quorum, only $2f-1$ `prepare` messages are required, since a replica's own `prepare` message and the primary's `pre-prepare` message count towards the $2f+1$ required messages. The previous requirements translate to

1) there cannot be more replicas that receive $2f-1$ `prepare` messages than replicas that have previously received a `pre-prepare` message, i.e., $C_{2,n} \leq N_1$, and
2) a replica can only receive $2f-1$ `prepare` messages if at least $2f$ replicas, including itself, have received a `pre-prepare`, i.e., $N_1 \geq 2f$.

The calculation of $C_{2,n}$ can thus be divided into the following cases, assuming that $n_1$ replicas have received a `pre-prepare` message. First, for $n_1 < 2f$, no replica will be able to gather the required quorum of `prepare` messages, thus, the probability for $c_{2,n} = 0$ is always one. Second, if $c_{2,n} > n_1$, the probability has to be zero. Finally, for all other cases, of the $n_1$ replicas that broadcast `prepare` messages, excluding the primary, the probability for $c_{2,n}$ replicas to receive $2f-1$ of those messages can be modeled as another Bernoulli trial. The probability of success in that Bernoulli trial is identical to a replica receiving at least $2f-1$ messages of the $n_1 - 1$ possible. Thus, the conditional PDF of $C_{2,n}$ for $n_1 \geq 2f$ and $c_{2,n} \leq n_1$ is given by

$$P_{C_{2,n}|N_1}(c_{2,n} \,|\, n_1) = B(n_1, p_2(n_1), c_{2,n}) \tag{4}$$

with $p_2(n_1)$ being the probability that a replica will receive at least $2f-1$ `prepare` messages, given that $n_1$ replicas, including the replica itself, are broadcasting that message, which implies they have received the `pre-prepare` message as well. This can be calculated with another Bernoulli trial to get $2f-1$ receptions from $n_1 - 1$ messages of the other replicas: $p_2(n_1) = B(n_1 - 1, 1-p_l, [2f-1, n])$.

Combining (3) and (4) yields the conditional PDF of $C_2$. The calculation is split into multiple cases as follows

$$P_{C_2|N_1}(c_2|n_1) = \begin{cases} P_{C_{2,n}|N_1}(0 \,|\, n_1) \cdot P(\overline{C_p} \,|\, N_1 = n_1), & c_{2,n} = 0 \\ P_{C_{2,n}|N_1}(n-1 \,|\, n_1) \cdot P(C_p \,|\, N_1 = n_1), & c_{2,n} = n \\ P_{C_{2,n}|N_1}(c_{2,n} \,|\, n_1) \cdot P(\overline{C_p} \,|\, N_1 = n_1) & \\ + P_{C_{2,n}|N_1}(c_{2,n} - 1 \,|\, n_1) \cdot P(C_p \,|\, N_1 = n_1), & c_{2,n} \leq n_1 + 1 \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

The final PDF of $C_2$ is given by applying the law of total probability to (5), which yields $P_{C_2}(c_2) = \sum_{n_1=0}^{n-1} P_{C_2|N_1}(c_2 \,|\, n_1) \cdot P_{N_1}(n_1)$.

*Calculating $N_2$:* As with $N_1$ and (2), the distribution of replicas that are still active, based on $C_2$, is $P_{N_2}(n_2) = \sum_{c_2=0}^{n} B(c_2, 1-p_c, n_2) \cdot P_{C_2}(c_2)$.

*Calculating $C_3$:* Now, let us turn to the states $C_3$ and $N_3$. In the third phase, the primary behaves in the same way as every other replica, simplifying many calculations regarding the communication as we do not need to mind so many exceptions. As with $C_2$, there are two requirements necessary for a replica to reach $C_3$: (i) the replica must be in state $C_2$ and (ii) it must have received at least $2f$ `commit` messages, not counting its own. Thus, we can conclude that

1) there cannot be more replicas that have received $2f$ `commit` messages than replicas that have reached $C_2$, i.e., $C_3 \leq C_2$, and
2) a replica can only receive $2f$ `commit` messages if at least $2f+1$ replicas, including itself, have reached state $C_2$, i.e., $C_2 > 2f$.

Deriving $C_3$ is similar to $C_2$. The conditional probability of $C_3$ for $c_2 > 2f$ and $c_3 \leq c_2$ is accordingly

$$P_{C_3|C_2}(c_3 \,|\, c_2) = B(c_2, p_3(c_2), c_3) \tag{6}$$

where $p_3(c_2)$ is the probability that a replica will receive at least $2f$ `commit` messages if $c_2$ replicas, including itself, are broadcasting that message, i.e.,

$$p_3(c_2) = B(c_2 - 1, 1 - p_l, [2f, n]). \tag{7}$$

Applying the law of total probability yields $P_{C_3}(c_3) = \sum_{c_2=0}^{n} P_{C_3|C_2}(c_3 \,|\, c_2) \cdot P_{C_2}(c_2)$.

*Calculating $N_3$:* Finally, $P_{N_3}(n_3) = \sum_{c_3=0}^{n} B(c_3, 1 - p_c, n_3) \cdot P_{C_3}(c_3)$, which denotes the PDF of all active nodes after the last phase.

If more than $2f$ replicas have completed the last phase, i.e., $P(N_3 > 2f)$, the happy path of PBFT was successful. For the system to provide liveness in regards to the current request, only $f + 1$ replicas are sufficient.

### E. Generalization

For the sake of simplicity, we so far assumed constant failure probabilities for links and processes, i.e., $p_l$ and $p_c$. We also assumed in our calculations, that those probabilities be constant for each phase of PBFT. This is not a requirement and could be expanded to reflect more sophisticated failure models that include time-based correlations as long as they remain i.i.d. for each phase.

Since the model is derived solely from communication patterns, it can be adapted to other fault tolerant protocols. This is facilitated by the modular design of the model, i.e., the expression of communication phases, e.g., broadcast, quorum, as PDFs which can be combined to describe the overall system state. To demonstrate the adaptability, we show the application of the model to BFT-SMaRt, Zyzzyva and SBFT. The detailed adaptations are available in the extended pre-print of this paper [28], where we showcase how the model can be applied to a variety of communication patterns, including client interaction and the possibility to branch into a fast or slow path.

We deliberately chose to highlight the model derivation in this section, leaving the formal definition of the modular components for future work.

## V. Model Validation

To verify the correctness of the model, a discrete-event simulator was written in Rust. The simulation is publicly available on Github[1]. In a first instance, the simulator implements the happy path of PBFT for single requests without batching. The dynamic link failure model is realized by discarding each message reception event with a configurable probability $p_l$. In addition, each node will miss all messages belonging to a certain communication phase with probability $p_c$, simulating a crash failure. By doing this, we can compare the simulated state with the predictions of our model. The simulation can easily scale to larger numbers of nodes (above 100) since only the state transitions in the happy path are of interest and no actual SMR is implemented, i.e., requests are not executed.

[1]https://github.com/mani2416/bft_simulation

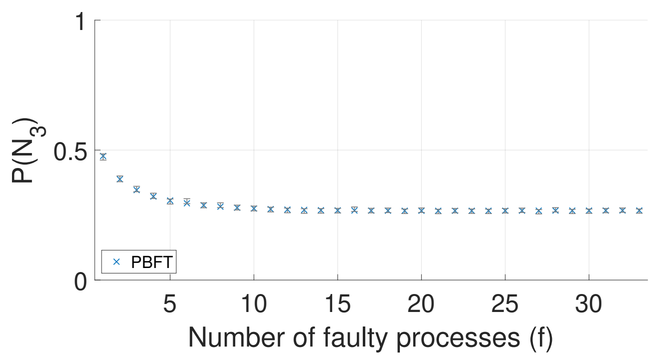
In order to validate our model with an independent source, we also deployed the Java-based public BFT SMR library BFT-SMaRt [4] as a reference implementation. It implements a consensus protocol that bears a high resemblance to PBFT: it utilizes epochs, an equivalent to the views in PBFT, and operates in three phases with respective message types [29]. For simplicity, we stick to PBFT's terminology, when discussing BFT-SMaRt While mostly similar, the communication pattern of BFT-SMaRt differs from PBFT in two details, which required minor model adaptations. Firstly, nodes do not count the primary's `pre-prepare` message as a `prepare` message for the second phase. Secondly, nodes are allowed to skip the second phase if a quorum of other nodes were able to complete that phase. We describe the model adaptations in the extended pre-print of this paper [28].

In order to apply dynamic link and crash failures to BFT-SMaRt, artificial message omission probabilities were implemented into the library. Accordingly, all messages are dropped with the probability $p_l$ and each node discards all messages of a whole phase with probability $p_c$. The changes necessary to implement the aforementioned failures affected the class that is responsible for handling incoming messages only and consisted of less than 50 lines of additional code. The library was executed on a single computer and up to 10 replicas and one client were instantiated to execute the requests.

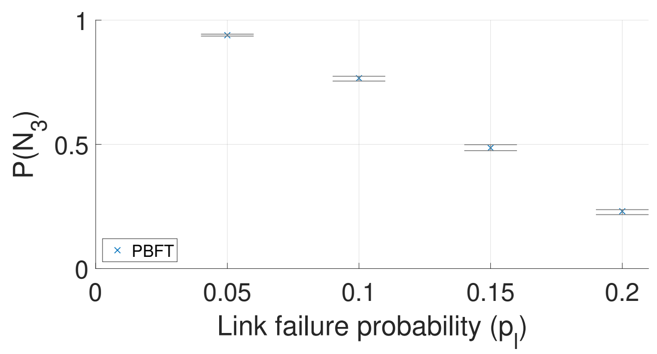
Increasing the number of processes while keeping the maximum number of faulty processes constant leads to an increased robustness of both protocols against link and crash failures, because more messages are available to build a quorum while the required quorum size remains equal. We therefore evaluate both protocols for the most interesting scenario $n = 3f + 1$, i.e., the minimum number of processes required to tolerate $f$ faulty processes.

To validate the model for a larger parameter space, we evaluated PBFT for different numbers of processes, link failure rates, and crash failure rates. Figures 2a to 2c show the probability of a single (representative) process to successfully reach phase $N_3$ for different $n$, $p_l$ and $p_c$, respectively. The simulation results for 5,000 protocol executions are plotted with $99\%$ confidence intervals and model predictions are depicted as crosses. Increasing the number of processes in the network can have, depending on the number of processes and failure rates, either a stabilizing or destabilizing effect on the performance. A more detailed analysis of this behavior is given in Section VI-C. Increasing the failure rate of either links or processes causes a constant decrease for $P(N_3)$.

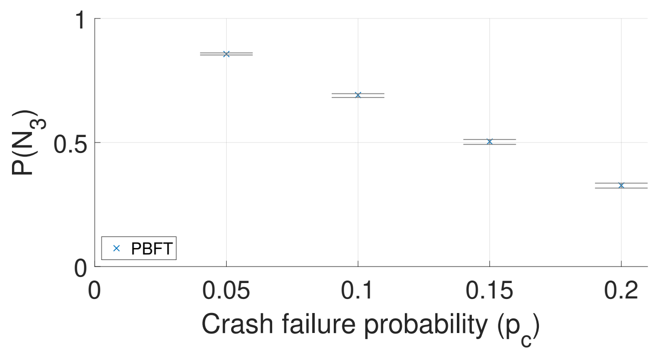
The comparison between model predictions and experimental results for BFT-SMaRt are shown in Figures 2d to 2f. Since BFT-SMaRt implements actual SMR and the execution was unstable due to the previously mentioned halts during the view-change protocol, we evaluated 1,000 protocol executions for each parameter combination only (without batching). Figure 2d depicts the measured and by the model predicted PDF of $P(N_3)$. The impact of link and crash failures on BFT-SMaRt is similar to PBFT. The small deviations visible between Figures 2b and 2c and Figures 2e and 2f stem from the



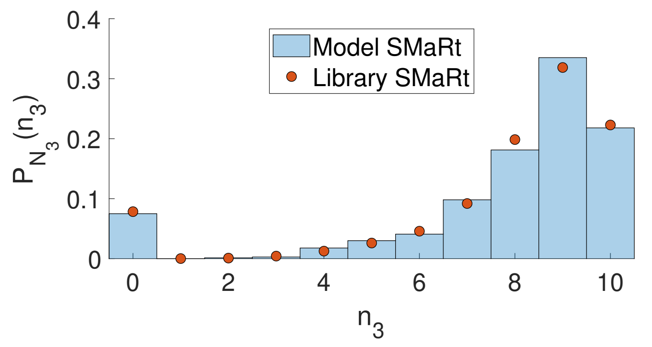
(a) $p_l = p_c = 0.1$



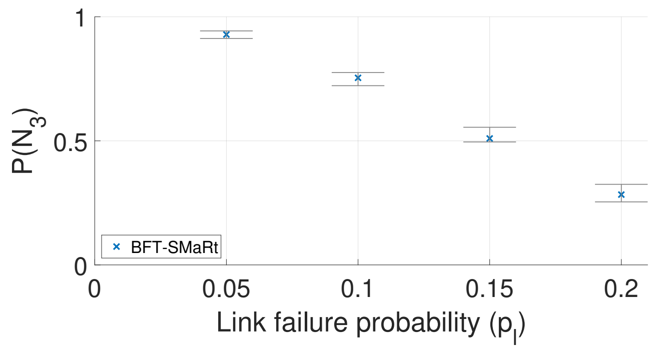
(b) $n = 10$, $p_c = 0$



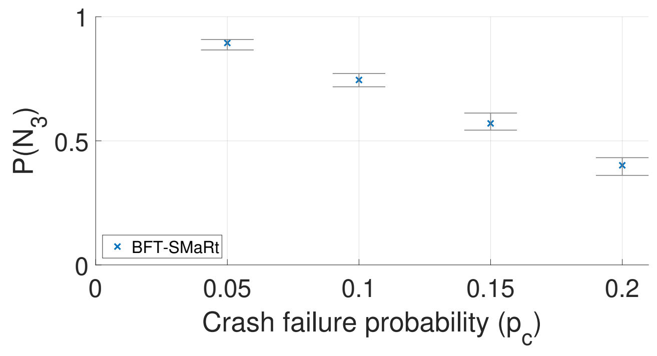
(c) $n = 10$, $p_l = 0$



(d) $f = 3, p_l = p_c = 0.05$



(e) $n = 10$, $p_c = 0$



(f) $n = 10$, $p_l = 0$

Fig. 2: Model validation results for PBFT (a-c) and BFT-SMaRt (d-f).

algorithmic differences described above. The overall results confirm that our model predictions for PBFT and BFT-SMaRt align accurately with the simulations and experimental results.

## VI. Evaluation

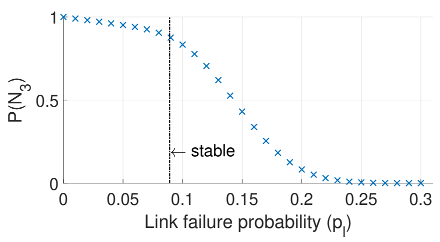
### A. Protocol stability

The quorum collection phase in fault tolerant protocols ,i.e., for $2f + 1$ processes to collect $2f$ out of $3f$ possible messages (not counting its own), is inherently resilient against link failures. A node cannot collect a quorum if at least $f + 1$ out of its $3f$ incoming links are failing. Consequently, even in the worst case, at least $f + 1$ nodes with at least $f + 1$ link failures, i.e., $(f + 1)^2$ overall link failures, are necessary for the quorum collection phase to potentially fail.

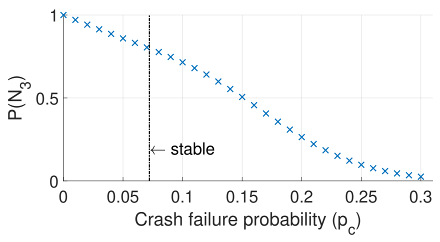
Given our model assumptions, we can calculate the theoretical failure rate necessary for a quorum phase in PBFT to fail. Since the number of processes that partake in each quorum phase is dependent on previous phases, the boundary for each phase is calculated as

$$\frac{((f + 1) - (n - E[N_{i-1}]))^2}{E[N_{i-1}](E[N_{i-1}] - 1)} \tag{8}$$

with $E[N_{i-1}]$ being the expected number of nodes that are still currently active. Depicted in Figure 3 are the predicted probabilities for $P(N_3)$ for increasing link failures (Figure 3a) and crash failures (Figure 3b). The line labeled "stable" marks



(a) $P(N_3)$ with $p_c = 0$.



(b) $P(N_3)$ with $p_l = 0$.

Fig. 3: $P(N_3)$ for PBFT with $n = 25$.

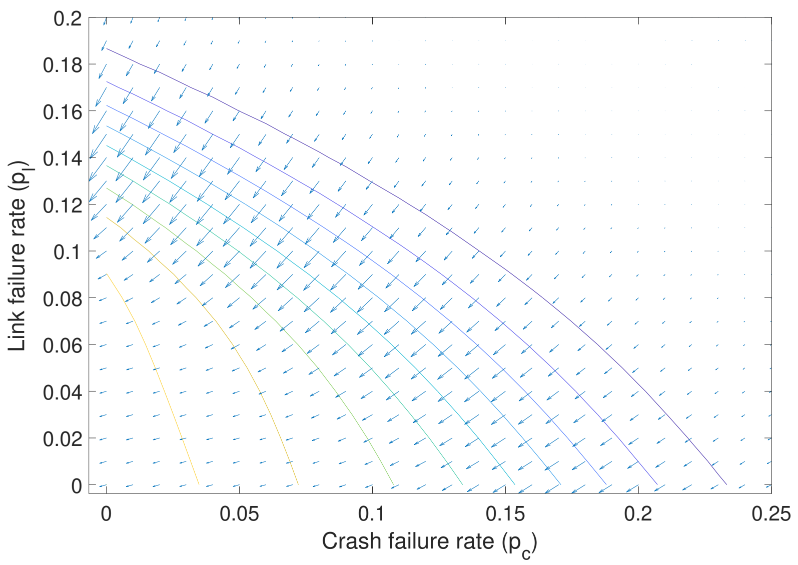


Fig. 4: Contour plot of $P(N_3)$ as predicted by our model for PBFT with $n = 40$; the gradient shows a vector field over $p_c$ and $p_l$.

the boundary given by (8). The linear decrease to the left of the boundary in Figure 3a originates from the previous phases of the protocol. Since the first phase implements a one-to-all broadcast, the failing nodes will increase linearly with the failure rates. The same effect is even more pronounced for increasing crash failures in Figure 3b, albeit with an even steeper linear phase. Because processes cannot recover within the happy path of PBFT, each successive phase with crash failures will decrease the number of available nodes for further phases, leading to the steeper decline before the boundary.

The evaluation methodology and the respective results can be used to parameterize the protocol to ensure that the protocol execution remains stable even for a given failure rate. Since most BFT protocols treat delayed messages as link failures, the model can, e.g., be utilized to fine-tune timeouts. That is, for a given delay distribution, a timeout parameter can be translated to a failure rate. A small timeout leads accordingly to a higher failure rate, but at the same time is able to quickly detect (genuinely) lost messages and make progress. For instance, let us assume that the message delay on all links can be described with a normal distribution of mean $\mu = 100\,\mathrm{ms}$ and standard deviation $\sigma = 10\,\mathrm{ms}$. Further, we assume the result of (8) to be $0.1$ for some arbitrary protocol. The timeout that keeps the protocol in the stable region is derived by finding an upper bound, where the integrated PDF of the delays is equal to $1 - b_{\text{stable}} = 0.9$. In our example, the timeout should be $> 87.19\,\mathrm{ms}$. To conclude, our model allows to evaluate various failure scenarios and adjust parameters accordingly.

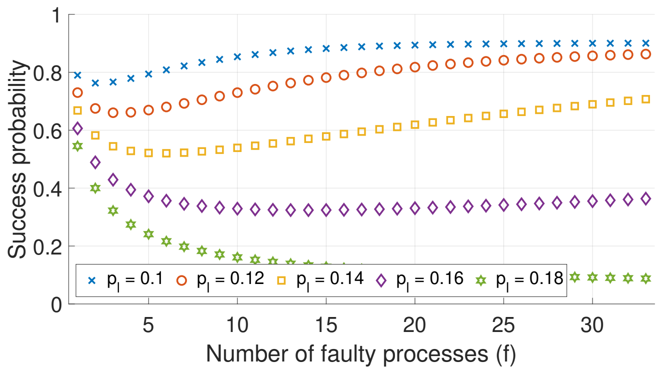
## *B. Impact of number of processes, link and crash failures*

To better demonstrate the predictive capabilities of the model, a contour plot of $P(N_3)$ for PBFT is provided in Figure 4, for varying link and crash failure rates. Additionally, the gradient is displayed, derived from the operating points for different $p_l$ and $p_c$, as they are predicted by the model. The orientation of the arrows indicates the impact of variations in either failure rate on $P(N_3)$. The more pronounced the horizontal component of a vector, the more dominant is the impact of crash failures on $P(N_3)$ and the same applies to the vertical component and link failures. The contour plot allows to quickly discern the impact of either failure rate on the protocol. Figure 4 shows that for low link failure rates, changes in the crash failure rate will dominate the success probability of the protocol, while for very low rates of crash failures and a moderate number of link failures ($p_l > 0.1$), the link failure rate dominates. Figure 4 also validates the observations made in Figure 3, i.e., the crash failure rates dominates the linear decline before the stable lines, while the link failures gain in impact for higher failure rates.

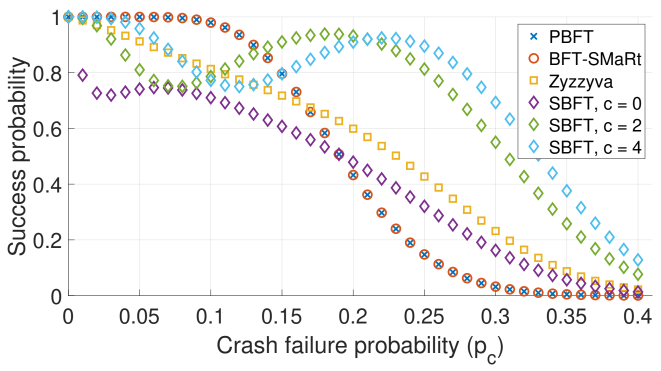
Although it is well known that most BFT protocols do not scale well with the number of processes due to the quadratic message complexity, it generally offers means to increase stability in the presence of dynamic link failures. In the extended version of the paper [28], we show that the probability to collect a quorum for $n \to \infty$ converges to 0 or 1, depending on $p_l$ and the quorum size. As a consequence, the number of nodes can increase the success probability of the quorum collection phase for failure rates below a certain threshold.

## *C. Comparison: PBFT, BFT-SMaRt, Zyzzyva, and SBFT*

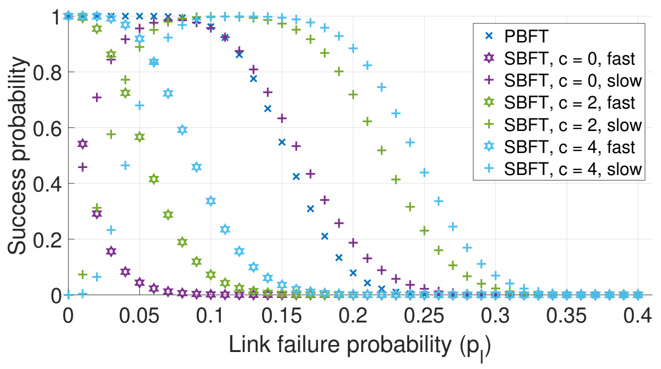
To showcase the adaptability of our model, we applied it to Zyzzyva [10] and SBFT [11]. An exemplary comparison of all protocols for different crash failure rates is given in Figure 5b. Depicted are the overall success probabilities, i.e., for Zyzzyva and SBFT the combination of fast and slow



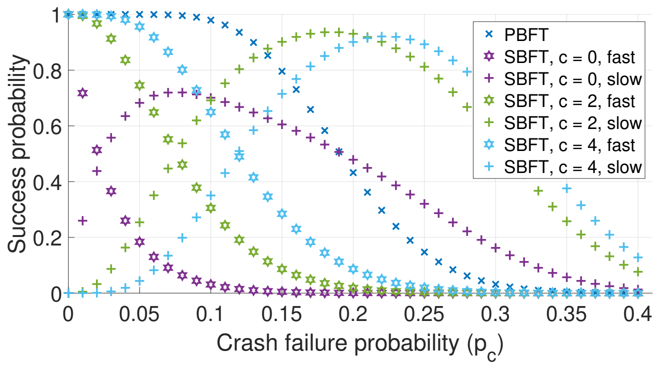
(a) $p_c = 0$.



(b) $f = 10, p_l = 0$.



(c) $f = 10, p_c = 0$.



(d) $f = 10, p_l = 0$.

Fig. 5: (a) happy path success probabilities of PBFT, BFT-SMaRt, Zyzzyva and SBFT dependent on crash failure rates, (c) and (d): detailed analysis of SBFT.

paths. To better demonstrate the capabilities of the model, the individual success probabilities for the fast and slow path of SBFT for different link and crash failure rates are plotted in Figures 5c and 5d. Since SBFT allows for optional, additional replicas, denoted as $c$, the model allows to quickly assess the protocol behavior for different failure rates and configurations of $c$. The plots show that SBFT outperforms PBFT for increasing numbers of $c$ and higher failure rates, while PBFT is more stable if SBFT transitions from the fast path to the slow path.

## VII. Conclusion

The probabilistic predictions of the presented model were validated with implementations of PBFT and BFT-SMaRt for various numbers of processes and dynamic link and crash failure rates. It was demonstrated with BFT-SMaRt, Zyzzyva and SBFT, that the model can be adapted with little effort to other communication patterns of BFT protocols. The model gives a prediction of the distribution of process states during execution, allowing prediction of protocol behavior (e.g. how many view changes will occur) and therefore performance evaluation. Additionally, if the message delay statistics are known, the model can be deployed to tune the timeouts for BFT protocols, since most protocols cannot differentiate between a delayed or an omitted message, making them indifferent in their impact on the algorithm. The model allows to assess the impact of crash and link failures for various operating points of a protocol to identify key boundaries regarding protocol stability.

As was demonstrated with BFT-SMaRt, Zyzzyva and SBFT, the model can be applied to different BFT protocols by modifying the respective equations for the distributions or adding further random variables should the protocol consist of more phases (as is the case with SBFT). Further adaptations are facilitated by the fact that a body of BFT protocols are derived from the core structure of PBFT and consist of interdependent phases.

In further work, we are planning to apply the model to more BFT protocols and evaluate their performance regarding dynamic failures. Furthermore we are exploring possibilities to extend the model to predict more sophisticated key performance indicators, such as throughput and latency. Lastly, we will consider adaptations to our model in order to account for correlated link failures, e.g., as was proposed with a model by Nguyen [30].

## References

[1] F. Thiel, M. Esche, F. Grasso Toro, A. Oppermann, J. Wetzlich, and D. Peters, "The European Metrology Cloud," in *Proceedings of the 18th International Congress of Metrology*, Jan. 2017. doi: 10.1051/metrology/201709001

[2] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance and Proactive Recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, Nov. 2002. doi: 10.1145/571637.571640

[3] P. Sazonova, "The general universal model of blockchain technology based on an analysis of some implementations," in *Communication Papers of the 2020 Federated Conference on Computer Science and Information Systems*. PTI, Sep. 2020. doi: 10.15439/2020f190. [Online]. Available: https://doi.org/10.15439/2020f190

[4] A. Bessani, J. Sousa, and E. E. P. Alchieri, "State Machine Replication for the Masses with BFT-SMART," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, Jun. 2014. doi: 10.1109/DSN.2014.43. ISSN 1530-0889 pp. 355–362.

[5] E. Buchman, J. Kwon, and Z. Milosevic, "The latest gossip on BFT consensus," *CoRR*, vol. abs/1807.04938, 2018.

[6] P. Aublin, S. B. Mokhtar, and V. Quéma, "RBFT: Redundant Byzantine Fault Tolerance," in *2013 IEEE 33rd International Conference on Distributed Computing Systems*, Jul. 2013. doi: 10.1109/ICDCS.2013.53. ISSN 1063-6927 pp. 297–306.

[7] R. Kapitza, J. Behl, C. Cachin, T. Distler, S. Kuhnle, S. V. Mohammadi, W. Schröder-Preikschat, and K. Stengel, "CheapBFT: Resource-efficient Byzantine Fault Tolerance," in *Proceedings of the 7th ACM European Conference on Computer Systems*, ser. EuroSys '12. New York, NY, USA: ACM, 2012. doi: 10.1145/2168836.2168866. ISBN 978-1-4503-1223-3 pp. 295–308.

[8] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick, "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, ser. EuroSys '18. New York, NY, USA: ACM, 2018. doi: 10.1145/3190508.3190538. ISBN 978-1-4503-5584-1 pp. 30:1–30:15.

[9] N. Santoro and P. Widmayer, "Time is not a healer," in *STACS 89*, B. Monien and R. Cori, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1989. doi: 10.1007/BFb0028994. ISBN 978-3-540-46098-5 pp. 304–313.

[10] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, "Zyzzyva: Speculative Byzantine Fault Tolerance," *SIGOPS Oper. Syst. Rev.*, vol. 41, no. 6, pp. 45–58, Oct. 2007. doi: 10.1145/1323293.1294267

[11] G. Golan Gueta, I. Abraham, S. Grossman, D. Malkhi, B. Pinkas, M. Reiter, D. Seredinschi, O. Tamir, and A. Tomescu, "Sbft: A scalable and decentralized trust infrastructure," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2019. doi: 10.1109/DSN.2019.00063 pp. 568–580.

[12] G. Bracha and S. Toueg, "Asynchronous Consensus and Broadcast Protocols," *J. ACM*, vol. 32, no. 4, pp. 824–840, Oct. 1985. doi: 10.1145/4221.214134

[13] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the Presence of Partial Synchrony," *J. ACM*, vol. 35, no. 2, pp. 288–323, Apr. 1988. doi: 10.1145/42282.42283

[14] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of Distributed Consensus with One Faulty Process," *J. ACM*, vol. 32, no. 2, pp. 374–382, Apr. 1985. doi: 10.1145/3149.214121

[15] B. Liskov, "From Viewstamped Replication to Byzantine Fault Tolerance," in *Replication: Theory and Practice*, ser. Lecture Notes in Computer Science, no. 5959, 2010.

[16] A. S. de Sá, A. E. Silva Freitas, and R. J. de Araújo Macêdo, "Adaptive Request Batching for Byzantine Replication," *SIGOPS Oper. Syst. Rev.*, vol. 47, no. 1, pp. 35–42, Jan. 2013. doi: 10.1145/2433140.2433149

[17] U. Schmid, "How to model link failures: a perception-based fault model," in *2001 International Conference on Dependable Systems and Networks*, Jul. 2001. doi: 10.1109/DSN.2001.941391 pp. 57–66.

[18] U. Schmid, B. Weiss, and I. Keidar, "Impossibility Results and Lower Bounds for Consensus under Link Failures," *SIAM Journal on Computing*, vol. 38, no. 5, pp. 1912–1951, 2009. doi: 10.1137/S009753970443999X

[19] R. Halalai, T. A. Henzinger, and V. Singh, "Quantitative Evaluation of BFT Protocols," in *2011 Eighth International Conference on Quantitative Evaluation of SysTems*, Sep. 2011. doi: 10.1109/QEST.2011.40 pp. 255–264.

[20] H. Sukhwani, J. M. Martínez, X. Chang, K. S. Trivedi, and A. Rindos, "Performance Modeling of PBFT Consensus Process for Permissioned Blockchain Network (Hyperledger Fabric)," in *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, Sep. 2017. doi: 10.1109/SRDS.2017.36 pp. 253–255.

[21] A. Singh, T. Das, P. Maniatis, P. Druschel, and T. Roscoe, "BFT Protocols Under Fire," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, ser. NSDI'08. Berkeley, CA, USA: USENIX Association, 2008. ISBN 111-999-5555-22-1 pp. 189–204.

[22] M. Abd-El-Malek, G. R. Ganger, G. R. Goodson, M. K. Reiter, and J. J. Wylie, "Fault-scalable Byzantine Fault-tolerant Services," *SIGOPS Oper. Syst. Rev.*, vol. 39, no. 5, pp. 59–74, Oct. 2005. doi: 10.1145/1095809.1095817

[23] N. Fathollahnejad, E. Villani, R. Pathan, R. Barbosa, and J. Karlsson, "On reliability analysis of leader election protocols for virtual traffic lights," in *2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W)*, Jun. 2013. doi: 10.1109/DSNW.2013.6615529. ISSN 2325-6664 pp. 1–12.

[24] W. Xu and R. Kapitza, "RATCHETA: Memory-Bounded Hybrid Byzantine Consensus for Cooperative Embedded Systems," in *2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)*, Oct. 2018. doi: 10.1109/SRDS.2018.00021. ISSN 2575-8462 pp. 103–112.

[25] U. Schmid, B. Weiss, and J. Rushby, "Formally verified Byzantine agreement in presence of link faults," in *Proceedings 22nd International Conference on Distributed Computing Systems*, Jul. 2002. doi: 10.1109/ICDCS.2002.1022311. ISSN 1063-6927 pp. 608–616.

[26] M. Biely, U. Schmid, and B. Weiss, "Synchronous consensus under hybrid process and link failures," *Theoretical Computer Science*, vol. 412, no. 40, pp. 5602–5630, 2011. doi: 10.1016/j.tcs.2010.09.032 Stabilization, Safety and Security.

[27] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The Honey Badger of BFT Protocols," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: ACM, 2016. doi: 10.1145/2976749.2978399. ISBN 978-1-4503-4139-4 pp. 31–42.

[28] M. Nischwitz, M. Esche, and F. Tschorsch, "Bernoulli meets pbft: Modeling bft protocols in the presence of dynamic failures," 2020.

[29] C. Cachin, "Yet another visit to Paxos," Apr. 2011.

[30] H. H. Nguyen, K. Palani, and D. M. Nicol, "Extensions of Network Reliability Analysis," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Jun. 2019. doi: 10.1109/DSN.2019.00023 pp. 88–99.