Threat Adaptive Byzantine Fault Tolerant State-Machine Replication

Simoes Silva, Douglas; Graczyk, Rafal; Decouchant, J.E.A.P.; Völp, Marcus; Esteves-Veríssimo, Paulo

DOI
10.1109/SRDS53918.2021.00017

Publication date
2021

Document Version
Final published version

Published in
The 40th International Symposium on Reliable Distributed Systems

2021 40th International Symposium on Reliable Distributed Systems (SRDS) | 978-1-6654-3819-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/SRDS53918.2021.00017

Threat Adaptive Byzantine Fault Tolerant State-Machine Replication

Douglas Simões Silva*, Rafal Graczyk*, Jérémie Decouchant†, Marcus Völp*, and Paulo Esteves-Verissimo‡
*SnT - University of Luxembourg, †Delft University of Technology, ‡RC3 - KAUST
*[douglas.simoes-silva, rafal.graczyk, marcus.voelp]@uni.lu, †j.decouchant@tudelft.nl, ‡paulo.verissimo@kaust.edu.sa

***Abstract*—Critical infrastructures have to withstand advanced and persistent threats, which can be addressed using Byzantine fault tolerant state-machine replication (BFT-SMR). In practice, unattended cyberdefense systems rely on threat level detectors that synchronously inform them of changing threat levels. However, to have a BFT-SMR protocol operate unattended, the state-of-the-art is still to configure them to withstand the highest possible number of faulty replicas f they might encounter, which limits their performance, or to make the strong assumption that a trusted external reconfiguration service is available, which introduces a single point of failure. In this work, we present *ThreatAdaptive* the first BFT-SMR protocol that is automatically strengthened or optimized by its replicas in reaction to threat level changes. We first determine under which conditions replicas can safely reconfigure a BFT-SMR system, i.e., adapt the number of replicas n and the fault threshold f, so as to outpace an adversary. Since replicas typically communicate with each other using an asynchronous network they cannot rely on consensus to decide how the system should be reconfigured. *ThreatAdaptive* avoids this pitfall by proactively preparing the reconfiguration that may be triggered by an increasing threat when it optimizes its performance. Our evaluation shows that *ThreatAdaptive* can meet the latency and throughput of BFT baselines configured statically for a particular level of threat, and adapt 30% faster than previous methods, which make stronger assumptions to provide safety.**

I. Introduction

Cyber infrastructures, which are used in domains such as finance, public administration (e-government), social networks, or e-health, as well as cyber-physical systems (CPS), such as the power grid or autonomous vehicles, increasingly face cyber attacks of various threat levels [1]. Some of these attacks might successfully compromise a subset of the machines used in an infrastructure, which imposes periodical verification of a system's integrity and protection measures. However, the nature of cyber-physical systems, and the increasing complexity of both cyber-physical and cyber-only systems prevent manual attack surveillance and mitigation [2]. Instead, systems have to operate through times of ongoing and possibly persistent incidents autonomously and unattended.

Fluctuations in the threat a system faces naturally arise from variations of environmental effects (e.g., radiation levels vary while planes taxi on ground and during flight [3]), or when it comes to attacks, from the number and skill of adversarial actors having put their attention to a system and from the sophistication of the tools they use. In practice, cyber systems rely on threat detectors [4] that indicate the level of threat they are facing and that allows them to automatically adapt their performance and resilience.

Byzantine fault tolerant state machine replication (BFT-SMR) [5], combined with rejuvenation [6] and diversification methods [7], [8] is a method that can be used to replicate servers and tolerate powerful adversaries. BFT-SMR protocols are typically configured with $n \geq 3f+1$ replicas to tolerate the largest number f of faulty replicas [5] it might encounter, and require $2k$ additional replicas (i.e., $n \geq 3f+2k+1$) if up to k replicas are simultaneously rejuvenated every T_R seconds [6]. However, the latency and throughput of BFT-SMR systems deteriorate when f increases. Our work is the first to allow BFT-SMR protocols to safely adapt to evolving threats by leveraging threat detectors.

Traditional protocols [5] define a value for f once and for all at deployment time, and therefore have a lower performance than less resilient systems. Adaptive protocols [9] reclaim some performance, but they also maintain f constant. To some extent, dual mode and abortable protocols [10]–[13] can optimize their performance by executing a protocol switch at runtime, but they also keep the fault threshold f constant. On the other hand, group membership protocols adjust the system and quorum sizes using consensus [14]–[16]. However, group membership protocols cannot guarantee that the system will enter a sufficiently resilient configuration before it gets compromised, in particular when network synchrony is lost. Finally, relying on an external reconfiguration service introduces a single point of failure.

Fortunately, as we show in this paper, it is possible to circumvent the limitations of these approaches by leveraging threat detectors, which provide a lightweight trusted functionality. Threat detectors issue warnings well in advance of imminent increases of adversarial strength [17]–[20]. However, threat detectors also report about threat level decreases, when there is room for optimization.

We describe how to adapt to fluctuating adversarial threats. We start by carefully analyzing the timing properties a threat detector should have to allow the system to react in time to adversarial strength increases. Building on those insights we describe a reconfiguration protocol, ThreatAdaptive, which allows the replicas of a system to: (i) use consensus to agree on and switch to a less resilient, but better performing

[1] This work is partially funded by FNR through Pearl grant IISD and the Core project ThreatAdapt C18/IS/1269492.

2575-8462/21/$31.00 ©2021 IEEE
DOI 10.1109/SRDS53918.2021.00017

configuration that is still resilient enough when possible; and (ii) return to a configuration that is strong enough, which has been agreed upon during the optimization phase, when the threat detector informs them about an imminent increase of adversarial strength. ThreatAdaptive allows a BFT-SMR protocol to save the resources it does not require to ensure safety at a given moment in time while increasing its efficiency by operating with fewer active replicas whenever possible.

Overall, this work makes the following contributions: (i) We establish when and how it is safely possible to reconfigure BFT protocols when the adversarial strength evolves over time. (ii) We present ThreatAdaptive, a BFT reconfiguration protocol that allows replicas to optimize its use of system resources and increase its resilience threshold. with respect to the perceived adversarial strength. (iii) We implement our Threat-Adaptive system design and evaluate its performance.

This paper is organized as follows. §II surveys the related work. §III presents our system model and objectives. §IV discusses threat detectors and presents the required conditions for the safe reconfiguration of a BFT-SMR protocol that relies on rejuvenation. §V describes our threat adaptive reconfiguration protocol for BFT-SMR protocols. §VI presents our performance evaluation. §VII concludes this paper.

II. Related work

Dynamic performance optimization of BFT-SMR. AWARE [9] periodically optimizes its performance by having replicas evaluate the network latencies, and subsequently distribute among themselves voting weights and the leader role. AWARE maintains constant the resilience parameters, namely the number of replicas n, the fault threshold f, while we aim at modifying these parameters depending on perceived threats (along with the rejuvenation period T_R).

Speculative agreement. Several works strive for protocol intrinsic (i.e., situation independent) optimization by relaxing the state machine semantics to accept transactions optimistically [21], [22] or to execute requests without causality properties [23]. We believe that our threat adaptative mechanisms could also be used in combination with these systems.

Adaptive BFT-SMR protocols. CheapBFT [10] and ReBFT [11] operate through error-free phases using only $n{-}f$ replicas (e.g., $2f{+}1$ for ReBFT), and do not adjust the fault threshold f. They switch to n replicas to mask faults. Abortable protocols [12], [13] switch between several algorithms to improve performance. The maximum fault threshold is also maintained constant in those protocols. Reconfiguration services in crash or Byzantine settings [24], [25] are also closely related to our work. These services rely on an external and centralized reconfiguration system, while we rely on a simpler and smaller threat detector. Kuznetsov and Tonkikh recently described a framework to reconfigure BFT systems [26]. This interesting work did not consider the threat adaptivity problem that we tackle in this work.

Group membership protocols. Group membership protocols, closely integrated with replication management protocols [27], can adjust the composition of the replica group, and hence n and f. However, replicas have to reach consensus to perform these adjustments or must do so one replica at a time [15], [16] with payload consensus interleaved between subsequent additions. When confronted with an increasing adversarial strength, these protocols cannot ensure that the system's fault threshold will be reconfigured early enough to be safe. In such a scenario, our mechanisms allow a system to automatically transition to more resilient configurations.

Architectural hybridization. Architectural hybridization [28] has been introduced to provide synchrony guarantees in an otherwise partially synchronous system. Veríssimo et al. [29] show how to introduce such a notion, either locally or in a distributed manner, with synchronously connected trusted-trustworthy components, called *wormholes*. To adapt to an increasing threat level, one could assume the existence of a wormhole that would reconfigure the system and inform replicas using synchronous communications. We assume a smaller and simpler threat detector whose role is to inform replicas synchronously of an increasing threat level.

III. System Model and Problem Statement

A. System Model

We assume a replicated service for which replicas coordinate agreement using a payload BFT-SMR protocol. We use PBFT [5] for illustration purposes, but our dynamic membership protocol is generic and can be applied to other BFT-SMR protocols, or even to reliable broadcast protocols [30]. Replicas are interconnected by a partially synchronous network. We assume the availability of strong cryptographic primitives and abstract from the steps necessary to establish trust in replicas and their key material, initially and after they have been rejuvenated.

The configuration of a BFT-SMR system is captured by a tuple $(N_i, f_i, q_i, k_i, T_{R,i})$, where N_i is the set of active replicas that execute the payload protocol, f_i is the fault threshold, q_i is the quorum size, and, to support rejuvenation, k_i and $T_{R,i}$ respectively denote the number of replicas that can be rejuvenated simultaneously and the time a proactive rejuvenation of k_i replicas takes. After a duration of $\left\lceil \frac{n_i}{k_i} \right\rceil T_{R,i}$, the system has proactively rejuvenated all n_i replicas once, and at most k_i simultaneously. In general, the values of k_i and $T_{R,i}$ can be freely chosen as long as one takes into account that replicas cannot be rejuvenated arbitrarily quickly.

For simplicity, we focus here exclusively on homogeneous payload protocols (i.e., there are no trusted components that replicas can use for the payload protocol, but they rely on a threat detector for reconfigurations), where $q_i{\leq}n_i{-}f_i$ and $2q_i{-}n_i{\geq}f_i{+}1$ must hold for safe and live configurations, which implies that $n_i{\geq}3f_i{+}2k_i{+}1$ and $q_i{\geq}2f_i{+}k_i{+}1$. We assume an initial set of replicas N_{max} and define a *World Configuration* $C_{max} = (N_{max}, f_{max}, q_{max}, k_{max}, T_{R,max})$ as the initial system configuration. We require the world configuration to be safe, live and capable of masking faults and rejuvenating replicas as this will be the configuration the system returns to in the most severe circumstances. During times when the system experiences synchrony, group membership protocols can be used to adjust this world configuration. Finally, we assume that the system is equipped with a threat detector (TD), which we discuss in §IV.

B. Threat Adaptive Reconfigurations

We consider an adversary whose strength evolves over time. We consider two ways in which this evolution can happen: i) the adversary can corrupt less or more replicas overall (the adversarial fault threshold f_{adv} evolves); and ii) the adversary requires less or more time to corrupt f_{adv} replicas. We capture those two aspects in the notion of adversarial strength, which is a function $T_A(t, f)$ of time t and fault threshold f. We do not instantiate the strength function for our analysis (cf. §IV).

The system can be configured to run the payload protocol either more efficiently or with more resilience. A replica r transitions through configurations and into a passive but responsive operation mode [10], [11], e.g., a deep sleep mode with wake-on-LAN, if it is not involved in the current configuration (i.e., if $r \notin N_i$). We assume that passive replicas are correct upon wake up and substantiate this assumption by frequently rejuvenating passive replicas. Such a rejuvenation is only limited by the rejuvenation time T_R and not by k_i. Passive replicas can be rejuvenated simultaneously.

Classically, system reconfigurations are decided by a system administrator. We opt for an automatic and internal reconfiguration orchestrated by the replicas themselves. Assuming an external reconfiguration service only relocates the problem we tackle, since this service has to be safe at all times. In order not to introduce a single point of failure, this service would also need to be replicated and either configured to the maximum fault threshold or have the capability to reconfigure itself.

Our goal is to define a resilient system able to automatically react to evolving threats safely and rapidly enough to outpace the adversary in its attempt to compromise the system. Replicas reconfigure the system to counter imminent threats, or to optimize its performance when it is safely possible, based on a threat detector's signal. We call *reactions* and *optimizations* the adaptations in consequence of an increasing or decreasing adversarial strength, respectively.

IV. Threat Detectors and Requirements for Adapting to Changing Adversarial Strength

This section first discusses threat detectors. We then present our threat model, which encapsulates the notion of a changing adversary, and state more precisely the threat detector requirements (i.e., how well in advance it should warn of a treat change) using the proactive recovery threat model.

A. Threat Detectors

Threat detectors are fundamentally different from intrusion detectors. Whereas the latter has to identify whether parts of the system have been compromised, threat detectors merely have to assess the risk of severe faults happening.

Threat levels have been defined by Singer as a product of the estimated capabilities of malicious actors and their intent [31]. Defense against cybersecurity threats requires identifying such actors, their points of entry, attack vectors and known vulnerabilities of the system to protect. The perception of adaptive threats requires continuous monitoring of changes of malicious actor capabilities (i.e., whether new, exploitable vulnerabilities have been exposed or old ones patched), as well as changes of a malicious actor's intent (system in focus of enemy military or intelligence actors, higher/lower black market financial incentives for breaching the system).

The answer to this threat perception challenge lies in implementing Cyber Threat Intelligence, which allows for continuous and responsive cybersecurity information collection, dissemination and processing, and, as a result, enables educated decisions on how to prepare the system to face (perceived) threats [19]. Operational frameworks (STIX) and standards of information exchange (TAXII) have been designed to automatically evaluate threat levels [32]. Open threat feeds [4], [33] are already available and the feasibility of such systems is confirmed by their existence in notable institutions like the Bank of England [20].

The threat detector (TD) generates and delivers indications of changes in the perceived adversarial strength T_A to replicas. TD endpoints at each replica are connected through a synchronous network, that is separated from the regular partially synchronous network that replicas use for the payload protocol. This separated network is a wormhole [29].

B. Adversarial strength

We now determine how much in advance replicas should be informed of an increasing adversarial strength so that they can reconfigure the system to maintain it safe. Sousa et al. [6] assume that during any time interval of duration T_A the adversary can compromise at most f replicas. A system with n replicas is then said to be *exhaustion safe* if all faulty replicas are repaired faster than T_A. In the absence of perfect failure detectors, this can be achieved by rejuvenating all n replicas proactively faster than T_A. For example, if k replicas can be rejuvenated simultaneously well within T_R, the system is exhaustion safe if and only if $\left\lceil \frac{N}{k} \right\rceil \; T_R \leq T_A$. In this scheme, the number of replicas should be at least $n \geq 3f + 2k + 1$ to ensure safe and live quorums.

We extend Sousa et al.'s model and their exhaustion safety notion by characterizing the combined adversarial strength T_A as a function of time t. However, while their adversary model faces systems with a constant fault threshold f, we strive for systems that adapt f in response to changing adversarial strength. Therefore, to understand how strong an adversary of strength $T_A(t)$ is against different configurations of the systems, $T_A(t)$ must itself be a function mapping each configuration's fault threshold f to the length of the time interval during which no more than f replicas can be compromised. That is, $T_A(t, f)$ resembles Sousa et al.'s time interval for an adversary with strength $T_A(t)$ at time t when it faces a system that is capable of tolerating up to f simultaneous faults.

Definition 1 (Adversarial strength). *Let $T_A : \mathbb{R} \times \mathbb{N} \to \mathbb{R}$ be a function that maps every point in time $t \in \mathbb{R}$ and*

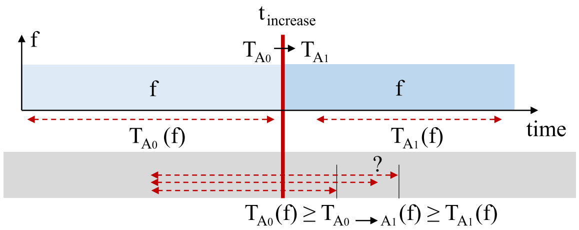
$t_{increase}$
f
$T_{A_0} \rightarrow T_{A_1}$
f
f
time
T_{A_0} (f)
T_{A_1} (f)
?
$T_{A_0}(f) \geq T_{A_0 \rightarrow A_1}(f) \geq T_{A_1}(f)$

Fig. 1: Time to compromise before, during and after the increase of adversarial strength at $t_{increase}$. During red-dashed intervals, the adversary cannot compromise more than f replicas. The length of intervals including $t_{increase}$ is not known precisely and hence requires careful consideration.

every $f \in \mathbb{N}$ to a duration $T_A(t, f)$ such that at time t the adversary cannot corrupt more than f replicas before a duration $T_A(t, f)$ has elapsed. We shall assume in this work that T_A remains constant for extended periods of time and that the duration of such a period $P_i = [t^i, t^{i+1})$ is larger than $T_A(t, f)$ for all involved fault thresholds f (e.g., when the system transitions from f_1 to f_2). We call $T_A(t, f)$ the strength of the adversary *during this period against a system with fault threshold f.*

In the following, for simplicity, we shall write $T_{A_i}(f)$ instead of $T_A(t, f)$ where $t \in P_i$ and $T_{A_{i+1}}(f)$ for $T_A(t, f)$ while $t \in P_{i+1}$, to denote these noticeable changes in the adversarial strength during subsequent periods of time P_i, P_{i+1}. We say that the adversary becomes *stronger* relative to a given f if it evolves from being characterized by T_{A_i} and is now characterized by $T_{A_{i+1}}$ where $T_{A_{i+1}}(f) < T_{A_i}(f)$. We define similarly a *weaker* adversary. We require T_{A_i} to be monotonic (i.e., $T_{A_i}(f_0) \leq T_{A_i}(f_1)$ for all $f_0 \leq f_1$), but make no further assumptions on T_{A_i} (such as linearity).

We define the time an adversary takes to corrupt f replicas:

Definition 2 (Time-to-compromise intervals)**.** *A time-to-compromise interval $[t_l, t_r)$ is any interval, between t_l and t_r such that the adversary cannot corrupt more than f replicas. If T_{A_i} remains constant between t_l and t_r it implies that $t_r - t_l \leq T_{A_i}(f)$.*

The duration of time-to-compromise intervals that include a change in the adversary's strength requires careful consideration. We denote by $T_{A_i \rightarrow A_{i+1}}(f)$ the adversarial strength over such an interval and require only that $T_{A_i}(f) \geq T_{A_i \rightarrow A_{i+1}}(f) \geq T_{A_{i+1}}(f)$. Fig. 1 illustrates this point.

In the following, we discuss how a system can react to an increasing adversarial strength, namely: i) by increasing the rejuvenation rate; or ii) by activating additional replicas. For each situation, we describe how the system should be reconfigured, and obtain a temporal bound before which the reconfiguration should be effective. We finally combine both results to identify all possible reconfigurations of the system.

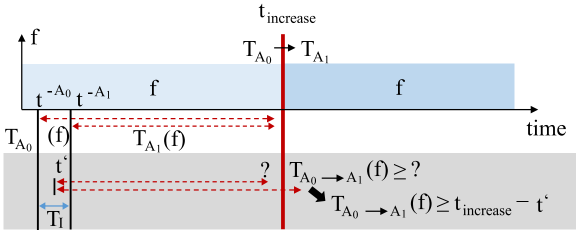
$t_{increase}$
f
$T_{A_0} \rightarrow T_{A_1}$
t^{-A_0} t^{-A_1}
f
f
time
T_{A_0}
(f)
T_{A_1}(f)
t'
?
$T_{A_0 \rightarrow A_1}(f) \geq ?$
$T_{A_0 \rightarrow A_1}(f) \geq t_{increase} - t'$
T_I

Fig. 2: Increase of adversarial strength.

C. Accelerating rejuvenation

Let us assume an adversary that becomes stronger at time $t_{increase}$, characterized through the functions T_{A_0} and T_{A_1}. In particular, we have $T_{A_0}(f) \geq T_{A_1}(f)$. We first study how and when the rejuvenation parameters can evolve so that no more than f replicas are simultaneously faulty. This situation is illustrated in Figure 2.

Exhaustion safe systems allow up to f faults to happen in any window of size $T_{A_0}(f)$ that starts before $t^{-A_0} = t_{increase} - T_{A_0}(f)$, and in any window of size $T_{A_1}(f)$ that starts after $t_{increase}$, provided the system is as well exhaustion safe with the adjusted rejuvenation rate and the strong adversary. We now investigate what happens over the time intervals where the adversary becomes stronger, i.e., those that contain $t_{increase}$. Those intervals start at a time t such that $t^{-A_0} < t < t_{increase}$, and have a duration comprised between $T_{A_1}(f)$ and $T_{A_0}(f)$. We can therefore over-approximate the adversary's strength and state our first theorem:

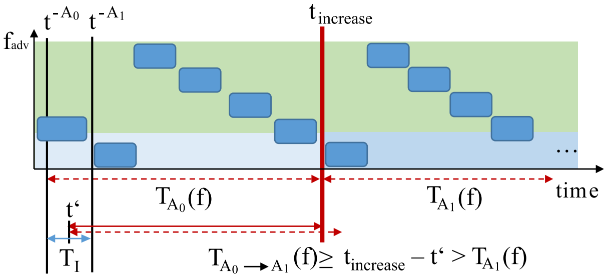

Fig. 3: More frequent rejuvenation to counter a stronger adversary. A blue rectangle shows the rejuvenation of a replica.

Theorem 1 (Reacting by accelerating rejuvenation)**.** *From an exhaustion safe configuration (N, f, q, k_0, T_{R_0}), if the adversary's strength evolves from T_{A_0} to T_{A_1} at time $t_{increase}$, the system remains safe if it is reconfigured to an exhaustion safe configuration (N, f, q, k_1, T_{R_1}), where $\left\lceil \frac{|N|}{k_1} \right\rceil T_{R_1} \leq T_{A_1}(f)$ before $t_{increase} - T_{A_1}(f)$.*

Proof. We know that before $t_{increase}$, the additional adversary strength has no effect. Therefore, there exists an interval included in $T_I = [t^{-A_0}, t^{-A_1} = t_{increase} - T_{A_1}(f)]$ such that the time-to-compromise windows that start in this interval

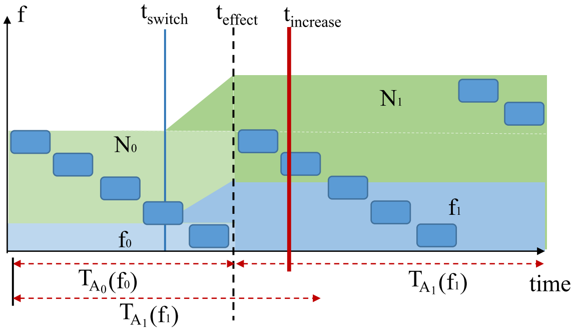

Fig. 4: Adding replicas to counter a stronger adversary.

(e.g., at $t' \in [t^{-A_0}, t^{-A_1}]$) cannot compromise the system in less than $t_{increase} - t'$ because this window is entirely included in the interval where the adversary is still weak.

The existence of this interval allows us to derive the point in time when the system has to adjust its proactive rejuvenation rate to counter the increase of adversarial strength. Having over-approximated the adversary's strength with $T_{A_1}(f)$ for any window that starts after t^{-A_1}, any attack launched during any of these windows is countered by rejuvenating all replicas in N faster than $T_{A_1}(f)$ (i.e., after adjustment, $\left\lceil \frac{|N|}{k_1} \right\rceil T_{R_1} \leq T_{A_1}(f)$ holds, which guarantees exhaustion safety for all sliding windows after t^{-A_1}).

The system does not have to switch to this rate before t^{-A_1} because all time-to-compromise windows that start in the interval T_I contain interval $[t^{-A_1}, t_{increase}]$ during which all n replicas are rejuvenated. The time-to-compromise windows that start earlier than t^{-A_0} find all replicas repaired before $t_{increase}$ since we assume the weak system to be exhaustion safe while the adversary is weak, and their possible overlap with interval $[t^{-A_1}, t_{increase}]$ only speeds up rejuvenation. Figure 3 illustrates this point. It is important to note that despite the change of rejuvenation parameters of the configuration (from k_0 to k_1 and from T_{R_0} to T_{R_1}), the order in which replicas are rejuvenated must be preserved. □

D. Adding replicas

Replicas cannot be rejuvenated arbitrarily fast. For example, Garcia et al. found that rejuvenating a replica running Ubuntu 16.04 requires 40s [8]. Therefore, it might happen, that a system is already rejuvenating replicas as fast as possible. In this case, it will no longer be possible to react only by accelerating the rejuvenation rate. Fortunately, the system can also react to an increasing adversarial strength by increasing its fault threshold, i.e., by adding replicas. Figure 4 illustrates this strategy and Theorem 2 captures its effectiveness.

Theorem 2 (Reacting by adding replicas)**.** *From an exhaustion safe configuration $C_0 = (N_0, f_0, q_0, k, T_R)$, if the adversarial strength evolves from T_{A_0} to T_{A_1} at time $t_{increase}$, the system remains exhaustion safe if it is reconfigured before $t_{increase}$ to a configuration $C_1 = (N_1, f_1, q_1, k, T_R)$, such that C_1 is exhaustion safe relative to the stronger adversary (i.e., $\left\lceil \frac{|N_1|}{k} \right\rceil T_R$ holds), provided that $T_{A_0}(f_0) \leq T_{A_1}(f_1)$, and replicas continue to be rejuvenated in the same order.*

Proof. The system starts with $n_0 = |N_0|$ replicas and fault threshold f_0, and is reconfigured to involve $n_1 = |N_1|$ replicas with fault threshold f_1. Let t_{switch} be the point in time where the system starts reconfiguring itself, and t_{effect} the time when the new configuration is operational and capable of tolerating f_1 faults. During a time-to-compromise window that starts before $t_{effect} - T_{A_0}(f_0)$ the replicas in N_0 are rejuvenated by the rejuvenation scheme of the weak configuration. Similarly, time-to-compromise windows that start after t_{effect} are exhaustion safe by our assumption that the strong configuration fulfills this property.

We now consider the windows that overlap with t_{effect}. Let us assume that f_1 was chosen so that $f_0 < f_1$ and $T_{A_0}(f_0) \leq T_{A_1}(f_1)$, and that the rejuvenation order of replicas is maintained through the reconfiguration. In this case, any window that starts before t_{effect} and goes beyond $t_{increase}$ does not lead to the compromise of more than f_1 replicas.

The windows that start in $[t_{effect} - T_{A_0}(f_0), t_{effect}]$ overlap with the new configuration. However, their length is larger than $T_{A_1}(f_1)$, because the adversary's strength over them is sometimes lower than T_{A_1}. However, since replicas that get activated at t_{effect} are correct, maintaining the same rejuvenation order ensures that the replicas that were active in the previous configuration, and which might be faulty, are rejuvenated first. In fact one could regard such a correct instantiation as a free simultaneous and instantaneous rejuvenation at t_{effect}. As a consequence, no more than f_1 replicas get compromised within any window of size $T_{A_1}(f_1)$ that starts after $t_{effect} - T_A(f_0)$, since all n_1 replicas in N_1 are rejuvenated in such a window. □

E. Accelerating rejuvenation and adding replicas

We have seen that the system can adjust the rejuvenation rate to $T_{A_1}(f_1)$ before time $t_{increase} - T_{A_1}(f_1)$ so that the $|N_0|$ replicas are rejuvenated over all time-to-compromise windows located before or containing $t_{increase}$ (Theorem 1). We have also seen that replicas can be added so that they are active before $t_{increase}$, so that the $|N_1|$ replicas are rejuvenated over all time-to-compromise windows that start after $t_{increase}$ (Theorem 2). These results can be combined as follows.

Theorem 3 (Reacting by accelerating rejuvenation and adding replicas)**.** *From an exhaustion safe configuration $(N_0, f_0, q_0, k_0, T_{R_0})$, if the adversarial strength evolves from T_{A_0} to T_{A_1} at time $t_{increase}$, the system remains safe if it is reconfigured to an exhaustion safe configuration $(N_1, f_1, q_1, k_1, T_{R_1})$, where*

1) *f_1 is such that $f_0 \leq f_1$ and $T_{A_0}(f_0) \leq T_{A_1}(f_1)$*
2) *the rejuvenation frequency is changed before $t_{increase} - T_{A_1}(f_1)$ so that $\left\lceil \frac{|N_1|}{k_1} \right\rceil T_{R_1} \leq T_{A_1}(f_1)$*
3) *the passive replicas become active before $t_{increase}$*
4) *replicas that appear in both configurations are rejuvenated in the same order after t_{effect}*

Proof. by Theorem 1 and 2. □

F. Consistent and Synchronized Information of Replicas

In addition to advance notice, consistency of the information replicas receive and possible synchronization requirements are further concerns to be considered in the threat detector network. Clearly, if all replicas receive the same information about imminent increases/decreases of adversarial strength at the same time, no confusion can arise when some lagging replicas are still reacting while others already optimize their configuration. However, such a time synchronized response introduces significant complexity and overhead in the threat detector wormhole.

In Section V-D, we therefore show how our protocol avoids this complexity, by first returning to the strongest required configuration since the replica last reacted to increasing threats before allowing this replica to advance with optimizations since then. Threat detectors inform replicas about both the strongest required configuration and whether it is safe to enter the targeted configuration during optimization.

V. A Threat-Adaptive Reconfiguration Protocol

Assuming the availability of a threat detector that reports threat increases sufficiently in advance, as discussed in the previous section, we now present our reconfiguration protocol that replicas use to reconfigure the system.

A. Intuition

Receiving an adversarial strength decrease signal from the threat detector allows replicas to optimize performance and switch into a more efficient configuration that involves fewer active replicas and possibly an adjusted rejuvenation scheme. If threat detection gives replicas enough confidence that T_A remains constant for a period of time long enough to optimize, execute the payload in the less resilient but more performant configuration and return from there should threats increase, replicas reach consensus about a less resilient configuration and switch to it. Replicas involved in this configuration remain active and return to the payload protocol after adopting the rejuvenation scheme of this configuration. Replicas not involved in this configuration passivate themselves, but remain attentive to activation requests. In the mean time, they rejuvenate themselves without coordinating with other replicas to remain available (remember, in configuration C_i active replicas are limited to rejuvenate at most k_i replicas at a time; passive replicas are not constrained in this way).

After the threat-detector wormhole informs the replicas about an imminent increase of adversarial strength, correct replicas must react within a bounded amount of time (see Section IV-C), which, as we have seen, rules out using consensus. Key to circumventing this impossibility is the observation that the decision about how to react to a given threat can be already taken well before the threat manifests itself. In this paper, we will suggest preparing for the reaction to increasing adversarial strength well ahead when reaching consensus about how to optimize. Clearly, a first decision must be taken at this point in time, but it is as well possible to revisit this decision several times during the execution of the target configuration (e.g., to compensate for permanent failure in passive replicas).

For simplicity, in this paper we shall only discuss reaction by returning to the very same configurations from where the system came from. That is, if the system, starting in the world configuration C_{max} progressed through a sequence of configurations $C_1, C_2, \ldots C_i$, where C_i is the current configuration. It will react by returning along this chain (i.e., from C_i to C_{i-1} to $C_{i-2}, \ldots$) until it has reached a configuration that is capable of withstanding the currently reported adversarial strength. Since any such chain starts with C_{max}, which is by definition the most resilient configuration, it is always possible to find such a configuration, provided the system can withstand an adversary of this strength in the first place.

B. Reaching Consensus for Optimizations

The goal of our reconfiguration protocol is to safely transition the system from its current configuration C_s to a proposed new configuration C_t, which is resilient enough to withstand the current adversary of decreased strength. At the same time, the protocol prepares for a later increase of adversarial strength by creating the possibility to return to C_s and to previous configurations in the chain without having to reach consensus. In general, the replica set N_t of configuration C_t does not need to be a subset of the current configuration's replica set N_s. In particular, quorums formed in N_t do not have to be safe quorums in N_s and vice versa.

Our reconfiguration protocol has the following properties.

P.1 Replicas can prove to lagging replicas (and clients) that C_t is the next configuration.

P.2 If C_t becomes active, enough replicas in this configuration must know it so that they can report any progress made in C_t in case the system requires to return to C_s. Otherwise replicas in C_s or in a previous configuration would wait forever for the progress of C_t and the system would not be live.

P.3 If C_t becomes active, enough replicas in C_s need to know about this fact so that no quorum can be formed in C_s to agree on a second configuration C'_t. Otherwise, both C_t and C'_t could become active simultaneously and the system would not be safe (i.e., clients could receive inconsistent replies if both C_t and C'_t return to the payload protocol).

We provide P.1 thanks to configuration certificates $\mathbb{C}_{s \to t}$, which loosely resemble view-change certificates of PBFT. To enforce P.2 all $n_t = |N_t|$ replicas of the target configuration C_t need to participate in a configuration certificate such that at least $n_t - f_t$ correct replicas are able to report any progress made in C_t. The payload protocol liveness is guaranteed despite waiting for a response from all n_t replicas in C_t, as long as subsequent optimization attempts are interleaved with minimal progress in the payload protocol. Note that an optimization may never succeed if target configurations continue to include non-responsive replicas, but the above guarantees progress nonetheless. Property P.3 requires a quorum of replicas in the source configuration C_s to witness the activation of C_t. We shall return to this aspect in Section V-E.

```
1  reconfiguration ($v$,$seq$):
2    // leader $s_l \in N_s$
3    compute $C_t$
4    propose $\langle \mathit{PrePrepareCfg}, l, v, seq, C_s, C_t \rangle_{\sigma(s_l)}$
5    // backups $s_i \in N_s$
6    relay $\mathit{PrePrepareCfg}$ as $\langle \mathit{PrepareCfg}, i, v, seq, C_s, C_t \rangle_{\sigma(s_i)}$
7    // all replicas $s_i, s_l \in N_s$
8    wait for $q_s$ matching $\mathit{Pre/PrepareCfg}$ messages
9    if TD.is_valid($C_t$)
10     send $\langle \mathit{CommitCfg}, i, v, seq, \mathbb{C}_{s \to t} \rangle_{\sigma(s_i)}$
11         to all replicas in $N_s \cup N_t$
12   // new mode replicas $s_j \in N_t$
13   wait for $q_s$ matching $\mathit{CommitCfg}$ messages
14   if $\mathbb{C}_{s \to t}$ is valid
15     send $\langle \mathit{ConfirmCfg}, j, v, seq, \mathbb{C}_{s \to t} \rangle_{\sigma(s_j)}$
16         to all replicas in $N_s$
17   // witnesses $s_i \in N_s$
18   wait for $n_t$ matching $\mathit{ConfirmCfg}$ messages
19   send $\langle \mathit{AckCfg}, i, v, seq, \mathbb{C}_{s \to t} \rangle_{\sigma(s_i)}$ to all replicas in $N_t$
20   if $s_i \notin N_t$ wait passively for $C_t$ to return
21   // new mode replicas $s_j \in N_t$
22   wait for $q_s$ matching $\mathit{AckCfg}$ messages
23   resume payload in $C_t$ with view $v+1$
```

Fig. 5: Reconfiguration Protocol.

Fig. 5 shows the pseudocode of our optimization reconfiguration protocol. Let s_l be the current leader of the payload protocol (i.e., the x^{th} replica in the set N_s where $x = v \bmod |N_s|$). We assume that the leader s_l progressed in view v to request sequence number $seq - 1$ and proposes with seq the reconfiguration of the system as $\langle \mathit{PrePrepareCfg}, l, v, seq, C_s, C_t \rangle_{\sigma(s_l)}$ (Ln 4). Here $\langle \ldots \rangle_{\sigma(s_i)}$ denotes a message signed by replica s_i. C_s, C_t are the source and target configurations. Proposing this configuration in view v and with sequence number seq ensures that backups (i.e., non-leader replicas) will try to optimize the system at the same relative point in time. Following PBFT, backups relay the leader proposal as $\langle \mathit{PrepareCfg}, i, v, seq, C_s, C_t \rangle_{\sigma(s_i)}$ (Ln 6) before both leader and backups in C_s wait for q_s matching $\mathit{PrePrepageCfg}$ and $\mathit{PrepareCfg}$ messages from different replicas (Ln 8). Forming configuration change certificate $\mathbb{C}_{s \to t}$ out of these q_s messages, the replicas in C_s validate with their threat detector TD, whether C_t is a valid configuration given the current adversarial strength (Ln 9) and if so, they send this certificate in a *CommitCfg* message to both the replicas of the source configuration and all target configuration replicas (Ln 10), which wake up passive replicas in N_t.

Replicas of both configurations then proceed with a handshake to transition the system to C_t. Replicas of the target configuration C_t confirm the receipt of a valid configuration change certificate (Ln 14) and the replicas in the source configuration C_s acknowledge the receipt of such confirmations from all replicas in C_t (Ln 18-19). We call *witnesses* the source configuration replicas that have seen these confirmations. Whereas target configuration replicas resume execution of the payload protocol in view $v+1$ (Ln 23), replicas that are not in the target configuration become passive but they can still react to messages (Ln 20).

```
24 on TD strength increase or reconfig timeout:
25   // replica $s_j \in N_t$ in view $v$ at $seq$
26   stop payload protocol
27   create local history $lh_j$
28   send $\langle \mathit{History}, j, v, seq+1, lh_j \rangle_{\sigma(s_j)}$ to all replicas in $N_s$
29   stop processing view $v$ messages
30   // replica $s_i \in N_s$
31   wait for $q_t$ $\mathit{History}$ messages
32       from different replicas (=Fig.4 Line 22)
33   combine $lh_j$ into global history $gh$
34   if $C_s$ is strongest visited
35     apply $gh$
36     resume in $C_s$ with view $v+1$
37   else continue returning to the next config.
38        in the chain with $lh_i = gh$
```

Fig. 6: Protocol to react to increasing adversarial strength, by returning from C_t to the configuration C_s that activated C_t.

For better readability, we omit timeout handling in Fig. 5. All participating replicas (i.e., all $s_i {\in} N_s$ and after activation all $s_j {\in} N_t$) set a timer when engaging in the configuration change. Timeouts cause replicas to abort their attempt to change to configuration C_t, which typically leads replicas in C_s to retry the configuration change. However, there are two exceptions: (i) having sent *AckCfg* witnesses wait for replicas from C_t to return, even if their timeout fires (Ln 31). This ensures that progress in the target configuration C_t is not lost; and (ii) replicas in the target configuration C_t return if the timeout fires during the configuration change (see Figure 6 and Ln 24–29). They will stop doing so after having received q_s matching *AckCfg* messages, which marks a successful reconfiguration.

C. Reacting to Increasing Adversarial Strength

Reaction closely resembles the switch protocol of ReBFT [11] extended to traverse the chain of visited configurations to the more resilient one(see Section IV).

Once in the targeted safe configuration, replicas resume executing the payload protocol, possibly after catching up with the progress their peers made relative to the history, which has to be done using the synchronous wormhole. The construction of local histories lh_j and the application of the global history gh depend on the payload protocol (Ln 27, 35). For example, for PBFT, the latest checkpoint and the progress made since then have to be reported. Prepared messages (i.e., those receiving q_t matching *PrePrepare* or *Prepare* messages from replicas in C_t) are executed. Client requests not in this state are proposed again by the leader of the current configuration. The combined global history merely reports all received local histories and confirms with the signature of s_i the transition if C_s is not the strongest visited.

Notice that although the return may proceed through several configurations in the chain, only the final configuration will resume the payload protocol. Moreover, the replicas initiating this process will continue being rejuvenated in the pattern of their configuration while it is active. All other replicas and those that become passive will undergo frequent reconfigurations without first having to coordinate with others. Therefore, between the moment where the reconfiguration starts t_{switch} and the moment t_{effect} where it is effective even if multiple configurations are passed, Theorems 2 and 3 hold because none of these transitionally passed configurations become active in the sense of entering the payload protocol.

To prevent a client from ever accepting replies from a faulty quorum of replicas, in particular after a threat increase, replicas drop the private key they were using to interact with the client when they receive a threat increase notification. Replicas then generate a new set of keys to interact with the clients and broadcast the public keys to other replicas. To identify the currently active configuration, clients contact the world configuration and successively walks down the optimizations. Note that we could also rely on a forward-secure digital signature scheme [26] to replace the keys of replicas.

D. Lagging Replicas and Successive Reconfigurations

One important detail concerns passive or lagging replicas that have not been able to enter the current configuration before the system optimizes to decreasing threats. Without further precautions, faulty replicas could activate a configuration with the lagging replicas while agreeing on a different operation to activate with the replicas that aim to optimize the system. To avoid this issue, replicas do not engage in optimizations before they have returned to the stronger configuration that can withstand the increasing adversarial strength. (see Section IV-F). From this configuration, lagging replicas follow the configurations the system was reconfigured into until they reach the currently active configuration or become passive in the progress. Only in this active configuration will they resume participating in potential optimizations.

E. Witnesses

To conclude the discussion of our protocol, let us give further details on the witness role (Ln 17–20) and show how it ensures property P.3, i.e., safety and liveness of the system.

Liveness. Replicas in the target configuration C_t react to increasing adversarial strength once they receive q_s matching $\mathit{CommitCfg}$ messages (Ln 13). However, they only start processing the payload protocol after receiving the same number of q_s matching AckCfg messages (Ln 22). This step ensures that replicas from C_t will already return control to C_s, even if they are not yet ready to advance to entering the payload protocol. Conversely, witnesses in C_s will only acknowledge the configuration transition if they are sure that enough replicas in C_t will report the progress C_t might make (Ln 13ff). This is the case after all n_t replicas confirmed, because then $n_t - f_t \geq q_t$ correct replicas are guaranteed to communicate back their progress. In combination, this ensures liveness, even if the target configuration C_t is only partially activated.

Safety. We have to ensure that no two configurations ever execute the payload protocol (i.e., configurations may only be simultaneously active as part of the above transition protocols). Witnesses ensure this by refusing to participate in re-executing the reconfiguration protocol in Figure 5 (e.g., for agreeing on a different configuration in case the optimization failed) unless the activated target configuration returned. For a partially activated target configuration (i.e., one receiving only fewer than q_s acknowledgments from witnesses), this is the case after the reconfiguration timeout expires. But in this case, already n_t replicas of C_t confirmed, which guarantees that those witnesses that have acknowledged C_t receive a correct history (although with no progress made since the payload protocol did not restart). This refusal to participate in a re-election before C_t returns ensures safety.

Replay Attacks. Replay attacks would not lead to the activation of two configurations. Key to preventing such replay attacks is the fact that to obtain all messages required for the activation (i.e., q_s AckCfg messages), a quorum in C_t must confirm this configuration. From this moment onward, replicas no longer react to messages with the confirmed view v after replicas in C_t returned (see Line 32 in Figure 6). Moreover, correct witnesses do not produce AckCfg messages before they receive confirmation from C_t. Consequently, if the required messages are available, C_t can be activated only until the point in time when C_t returns, but such a return is necessary for the witnesses to resume in the protocols of Fig. 5 and 6 and hence to agree on a different configuration to activate.

VI. Performance Evaluation

Setup. We implemented our reconfiguration protocol, which we name Threat-Adaptive, on top of BFT-SMaRt [25]. For our experiments we use 4 Ubuntu 18.04+ desktops each equipped with an Intel i7 6th generation processor and 8 GB of memory. The desktops are interconnected with a Linksys WRT54G router. For each experiment, replicas are evenly distributed on the machines. We ran BFT-SMaRt's microbenchmarks with 160 clients that send 100 Bytes requests every $150ms$ to keep replicas continuously busy. Messages are delivered in batches of 1024 and experiments start with fresh views. Our goal is to demonstrate that despite the threat adaptation mechanisms we described, the performance of the payload protocol is maintained close to the respective native configurations.

Baselines. We compare the performance of Threat-Adaptive to the following five baselines.

1) *BFT-SMaRt-4* and *BFT-SMaRt-10* are unmodified BFT-SMaRT [25] deployments with n=4 (i.e., f=1) and n=10 (i.e., f=3) without rejuvenation (i.e., k=0). *BFT-SMaRt-6* and *BFT-SMaRt-12* are the equivalent configurations (i.e., f=1 or 3) with rejuvenation and k=1.
2) *StoppableBFT* stops the system execution before restarting it in a new configuration. This protocol emulates the unrealistic scenario where a sysadmin would immediately reconfigure the system before each threat increase.
3) *GroupMembership* is a consensus-based reconfiguration protocol, similar to Rampart et al. [27].
4) *ReBFT* [11] is the optimistic mode of BFT-SMaRt where the system runs with $n - f$ replicas (here 7), but would

Protocol	**Safety depends on**
BFT-SMaRt-4	`#faults` ≤ 1
BFT-SMaRt-10	`#faults` ≤ 3
Stoppable	sysadmin being faster than attackers to reconfigure (not guaranteed)
Group Membership	reaching consensus during attack (not guaranteed)
ReBFT	`#faults`$\leq f$ (mode if `#faults`$\neq$0)
Speculative	`#faults` $\leq f$ (gracious execution if `#faults`=0)
ThreatAdaptive (this work)	Safe if TD signal arrives early enough so that Thm. 3's conditions are verified.

TABLE I: Safety conditions of the protocols considered.

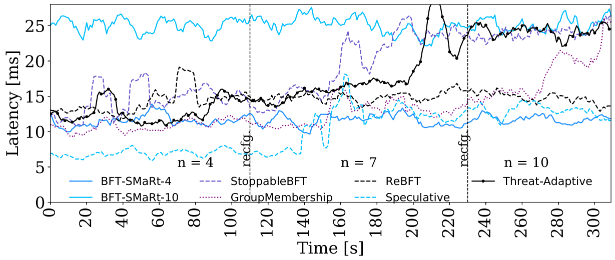

Fig. 7: Latency with increasing adversarial strength. σ=3 during transitions, and 2 otherwise (no rejuvenation).

need to return to a configuration with $n = 10$ replicas to handle faults.

5) *Speculative* implements an optimization presented where clients wait for $3f{+}1$ replies [22]. Replicas rollback and re-execute requests if these replies are not received within a bounded time.

Metrics. We measure the throughput and latency of the payload protocol, in particular when reconfigurations happen, and the delay that is required to react to increasing threat levels. We measure latency as the average time required to process all client requests received within a one second interval at the replica side. To improve the readibility of the graphs, we plot the average of 11 measurements per sliding-window and indicate the standard deviation σ in the figure captions.

A. Reacting to Increasing Threat Levels

We first consider the situation where the threat level increases. Fig. 7 illustrates ThreatAdaptive's latency when the threat level increases from f=1 to 2 at 110s, and from f=2 to 3 at 230s. ThreatAdaptive's latency evolves from the one of BFT-SMaRt-4 before 110s to the one of BFT-SMaRt-10 at 230s, while obtaining an intermediary performance between 110s and 230s, and remains stable despite the reconfigurations. Protocols with a lower latency are those that do not guarantee safety or that are executing without faulty replicas. To precise this point, we summarize in Table I the safety conditions for the baselines and ThreatAdaptive in that situation. The GroupMembership and the BFT-SMaRt-4 baselines may no longer be safe in the scenario we consider, and StoppableBFT would be safe only if the administrator reacts quickly enough. GroupMembership replaces replicas using consensus, which would always finish in time if consensus is provided as a functionality of a synchronous wormhole. ThreatAdaptive assumes a simpler synchronous wormhole whose task is to inform replicas sufficiently in advance of a threat level increase so that histories can be transmitted in time (through the synchronous wormhole) to the replicas involved in the next configuration. We now precise this condition.

System	Recfg. increasing f (n=3f+1)			
	1 $\rightarrow$ **2**	**2** $\rightarrow$ **3**	**3** $\rightarrow$ **4**	**4** $\rightarrow$ **5**
GroupMembership	3561	3656	3825	3948
ThreatAdaptive (inclusive)	2404	2445	2589	2781
ThreatAdaptive (overlapping)	2690	2856	3012	3379

TABLE II: Reaction time ($t_{effect}-t_{switch}$) in ms (no rejuvenation).

B. Outpacing Adversaries

Table II shows the time required for ThreatAdaptive to return to a safe configuration after the adversarial strength increases. We evaluate two extremes supported by our witness scheme: (i) *inclusive* returns from C_t to an inclusive configuration C_s, where $N_t \subseteq N_s$; and (ii) *overlapping* returns to a configuration that activates passive replicas that have never been active in previous configurations. The *inclusive* scheme outperforms GroupMembership by 30% since the reconfiguration decision is made in advance. As expected, replicas s_i that remain active in both configurations (i.e., $s_i \in N_t \cap N_s$) speed up the reconfiguration process. These results precise condition 3 of Theorem 3 to indicate the time required for replicas to transmit their histories and for passive replicas to become active. For example, the threat detector would need to inform replicas that the threat level evolves from f=1 to 2 only 3,561 ms in advance with the GroupMembership baseline, whereas with our approach (with the inclusive witness scheme) only 2,404 ms would be required.

C. Optimization Reconfiguration

We now consider the case where the threat level decreases from f=3 to 2 at 110s, and from f=2 to 1 at 230s. Fig. 8 shows that the throughput of ThreatAdaptive during two optimization reconfigurations is similar to the two BFT-SMaRt baselines (i.e., BFT-SMaRT-10 and BFT-SMaRT-4) when they are configured for a similar threat level. In addition, between 110s and 230s ThreatAdaptive is able to use 7 replicas, which provides a better latency and throughput than BFT-SMaRt-10 and is safe, contrary to BFT-SMaRt-4 in that interval.

For the same experiment, Fig. 9 illustrates the latency of the protocols. ThreatAdaptive comes close to the BFT-SMaRt baselines when the threat level allows it. In the second phase, we see slightly higher latencies than StoppableBFT,

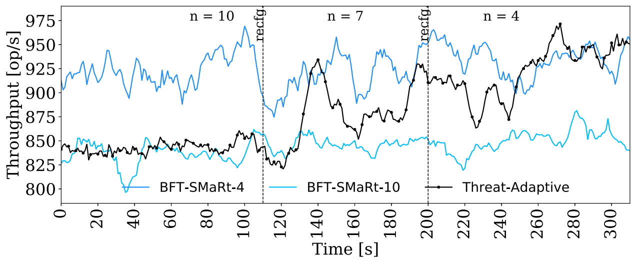

Fig. 8: Throughput with optimizations at 110s and 200s (no rejuvenation, σ=65 at 110s and 200s, and 34 otherwise).

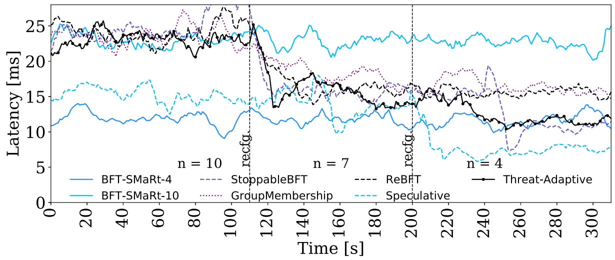

Fig. 9: Latency of ThreatAdaptive and the 5 baseline protocols (no rejuvenation, $\sigma \leq 3$).

this is because outliers present in StoppableBFT reconfigurations were averaged out (i.e., masked in the sliding window average). StoppableBFT's restart costs are much higher than other protocols because the payload protocol has to be stopped, reconfigured and relaunched, which is much slower than executing view changes.

VII. Conclusion

In this paper, we established the conditions that allow a BFT-SMR protocol, which potentially rejuvenates its replicas, to safely reconfigure itself to tolerate an increasingly powerful adversary based on a threat detector that communicates synchronously with replicas. We designed ThreatAdaptive, a novel BFT-SMR protocol that proactively agrees on more resilient fall-back reconfigurations before optimizing its configuration. Our results allow a protocol to dynamically and safely optimize its performance by reducing the number of replicas that actively participate in the protocol's execution. ThreatAdaptive is the first protocol that guarantees safe reconfigurations directly executed by the replicas assuming that a threat increase signal is received sufficiently in advance. Our experiments showed that our threat adaptive protocol achieves a throughput and latency comparable to the non-adaptive baselines in stable phases, and that reconfigurations are 30% faster than using previous methods, which make stronger assumptions to provide safety. Future work includes extending our methods to BFT protocols that use trusted components and to consider weaker threat detector models.

References

[1] P. W. Coopers, "The global state of information security® survey 2018," *Price Waterhouse Coopers*, 2014.

[2] Accenture", "State of Cybersecurity report 2020," Tech. Rep., 2020.

[3] C. J. B. L. Jones, "Space weather effects on aircraft operations," in *Effects of Space Weather on Technology Infrastructure*, I. A. Daglis, Ed. Dordrecht: Springer, 2005, pp. 215–234.

[4] "Proactive detection of network security incidents," ENISA, European Union Agency for Cybersecurity, Tech. Rep., Dec 2011.

[5] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *OSDI*, 1999.

[6] P. Sousa, N. F. Neves, and P. Veríssimo, "Proactive resilience through architectural hybridization," in *SAP*, 2006.

[7] S. Hosseinzadeh, S. Rauti, S. Laurn *et al.*, "Diversification and obfuscation techniques for software security: A systematic literature review," *Information and Software Technology*, vol. 104, pp. 72 – 93, 2018.

[8] M. Garcia, A. Bessani, and N. Neves, "Lazarus: Automatic management of diversity in bft systems," in *Middleware*, 2019.

[9] C. Berger, H. P. Reiser, J. Sousa *et al.*, "Resilient wide-area byzantine consensus using adaptive weighted replication," in *SRDS*, 2019.

[10] R. Kapitza, J. Behl, C. Cachin *et al.*, "Cheapbft: Resource-efficient byzantine fault tolerance," in *EuroSys*, 2012.

[11] T. Distler, C. Cachin, and R. Kapitza, "Resource-efficient byzantine fault tolerance," *IEEE Trans. on Comput.*, vol. 65, no. 9, pp. 2807–2819, 2015.

[12] P.-L. Aublin, R. Guerraoui, N. Knežević *et al.*, "The next 700 bft protocols," *ACM Trans. Comput. Syst.*, vol. 32, no. 4, 2015.

[13] J.-P. Bahsoun, R. Guerraoui, and A. Shoker, "Making BFT protocols really adaptive," in *PDPS*, 2015.

[14] T. D. Chandra, V. Hadzilacos, S. Toueg *et al.*, "On the impossibility of group membership," in *PODC*, 1996.

[15] M. K. Reiter, "A secure group membership protocol," *IEEE Transactions on Software Engineering*, vol. 22, no. 1, pp. 31–42, 1996.

[16] A. Das, I. Gupta, and A. Motivala, "Swim: Scalable weakly-consistent infection-style process group membership protocol," in *DSN*, 2002.

[17] R. Mitchell and I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, pp. 1–29, 2014.

[18] A. Dehghantanha, *Cyber Threat Intelligence*. Springer, 2018.

[19] S. Boeke, "National cyber crisis management: Different european approaches," *Governance*, vol. 31, no. 3, pp. 449–464, 2018.

[20] "Cbest intelligence-led testing: Understanding cyber threat intelligence operations," Bank of England, Tech. Rep., 2016.

[21] R. Pass and E. Shi, "Thunderella: Blockchains with optimistic instant confirmation," in *EuroCrypt*, 2018.

[22] R. Kotla, L. Alvisi, M. Dahlin *et al.*, "Zyzzyva: speculative byzantine fault tolerance," in *SOSP*, 2007.

[23] L. Luu, V. Narayanan, C. Zheng *et al.*, "A secure sharding protocol for open blockchains," in *CCS*, 2016.

[24] R. Rodrigues, B. Liskov, K. Chen *et al.*, "Automatic reconfiguration for large-scale reliable storage systems," *IEEE TDSC*, vol. 9, pp. 145–158, 2012.

[25] A. Bessani, J. Sousa, and E. Alchieri, "State machine replication for the masses with bft-smart," in *DSN*, 2014.

[26] P. Kuznetsov and A. Tonkikh, "Asynchronous reconfiguration with byzantine failures," in *DISC*, 2020.

[27] M. Reiter, "Distributing trust with the rampart toolkit," *Commun. ACM*, vol. 39, no. 4, pp. 71–74, 1996.

[28] P. Verssimo, A. Casimiro, and C. Fetzer, "The timely computing base: Timely actions in the presence of uncertain timeliness," in *DSN*, 2000.

[29] P. E. Veríssimo, "Travelling through wormholes: A new look at distributed systems models," *SIGACT News*, vol. 37, no. 1, pp. 66–81, 2006.

[30] R. Guerraoui, J. Komatovic, P. Kuznetsov *et al.*, "Dynamic byzantine reliable broadcast," in *OPODIS*, 2020.

[31] J. D. Singer, "Threat-perception and the armament-tension dilemma," *The Journal of Conflict RSesolution*, vol. 2, no. 1, pp. 90–105, 1958.

[32] S. Qamar, Z. Anwar, M. A. Rahman *et al.*, "Data-driven analytics for cyber-threat intelligence and information sharing," *Computers & Security*, vol. 67, pp. 35–58, 2017.

[33] B. Gourley, "Cyber threat intelligence feeds," Available at https://thecyberthreat.com/cyber-threat-intelligence-feeds/ (2021/04/07).