Cost Sensitive Moving Target Consensus

Sisi Duan
Oak Ridge National Laboratory
Email: duans@ornl.gov

Yun Li
University of California, Davis
Email: yunli@ucdavis.edu

Karl Levitt
University of California, Davis
Email: levitt@cs.ucdavis.edu

***Abstract*—Consensus is a fundamental approach to implementing fault-tolerant services through replication. It is well known that there exists a tradeoff between the cost and the resilience. For instance, Crash Fault Tolerant (CFT) protocols have a low cost but can only handle crash failures while Byzantine Fault Tolerant (BFT) protocols handle arbitrary failures but have a higher cost. Hybrid protocols enjoy the benefits of both high performance without failures and high resiliency under failures by switching among different subprotocols. However, it is challenging to determine which subprotocols should be used. We propose a moving target approach to switch among protocols according to the existing system and network vulnerability. At the core of our approach is a formalized cost model that evaluates the vulnerability and performance of consensus protocols based on real-time Intrusion Detection System (IDS) signals. Based on the evaluation results, we demonstrate that a safe, cheap, and unpredictable protocol is always used and a high IDS error rate can be tolerated.**

***Index Terms*—Consensus, state machine replication, crash fault tolerance, Byzantine fault tolerance, moving target defense.**

I. Introduction

Consensus is a generic technique that implements fault-tolerant services through replication. It is critical to achieve both reliability and availability in various online services and cloud computing applications, including Google's Chubby [6], Amazon Web Services [1], and VMware's vSphere [2, 3]. Depending on the types of failures we aim to tolerate, various protocols are designed with different security guarantees and performance characteristics. It is in general known that there exists a tradeoff between the resiliency and the cost of the consensus protocols. Namely, a low cost protocol with low redundancy and high performance can only handle limited types of failures. For instance, Crash Fault Tolerant (CFT) protocols are less redundant but can only tolerate crash failures. In comparison, Byzantine Fault Tolerant (BFT) protocols handle arbitrary failures but are very expensive, which may be an overkill to use most of the time.

Several approaches have been proposed to switch among different protocols depending on an estimation of failures in the system [16, 20, 23, 37], which enjoy the benefits of both high performance in the normal cases and high resiliency under failures. However, there still exist some issues when handling a wide range of failures. First, most approaches employ two subprotocols. Therefore, there is usually an obvious performance degradation even when it is not necessary to use a high cost protocol. Second, if we employ more than two subprotocols to not suffer from large performance degradation, it is hard to determine which one should be used according to existing system and network vulnerability. Third, the switching of protocols is usually deterministic and predictable, which makes the system more vulnerable since the attackers have enough time to collect protocol information, prepare, and complete an attack.

978-1-5090-3216-7/16/31.00

In this paper, we propose a cost sensitive approach motivated by Moving Target Defense (MTD) to build a resilient consensus model that handles various types of failures according to system and network vulnerability. At the core of our idea is a formalized cost model that evaluates the vulnerability and performance of leader-based state machine replicated fault-tolerant consensus protocols. We use *damage cost* to represent the vulnerability of the protocols and *operational cost* to evaluate the running cost. The input of the cost model is a set of probabilities from IDS, each of which represents the certainty of a replica being normal, crash, or Byzantine. We view the IDS as an oracle that provides the probabilities periodically. The underlying idea is that if the IDS can provide a rough reference about which replica(s) might be faulty without actively participating the protocols, we will be able to use a protocol that causes the lowest damage to the system while achieving high performance.

Due to the use of our approach, a protocol that is safe, cheap, and unpredictable is always used. According to the cost model, we select a cluster of protocols with low cost according to existing system vulnerability. By switching protocols randomly among the cluster, an unpredictable protocol is always used. In addition, the formalized cost model can also be viewed as a theoretical model to analyze the consensus protocols. We illustrate our cost model with 8 consensus protocols and our evaluation results show that the selection of the protocols naturally follows the properties of the protocols. In addition, we handle crashing IDS through the use of a fail-safe protocol and we also show in the evaluation that our approach tolerates a high error rate for the IDS.

The contributions of the paper are summarized as follows. (1) We propose a formalized cost model to evaluate the vulnerability and performance of consensus protocols based on real-time IDS signals, which can also be viewed as a theoretical analysis model (§IV); (2) We illustrate our cost model using 8 different consensus protocols (§V); (3) Based on the cost model, we present a moving target consensus approach to select a cluster of safe, cheap, and unpredictable protocols (§VI); (4) Our evaluation results show that a cluster of both safe and fast protocols is always selected. In addition,

we tolerate a high IDS error rate and also handle the case where the IDS crashes (§VII).

II. Related Work

Most consensus protocols proceed in rounds [3, 7, 9, 13, 16, 23, 24, 35]. A number of approaches rely on (small) trusted components to prevent equivocation and handle Byzantine failures using fewer than $3f + 1$ replicas [8, 12, 20]. For instance, ByzID [12] relies on specification-based IDS [22] to passively monitor the consistency of the messages. We also rely on a trusted IDS that may fail by crashing. In contrast, we employ the IDS to evaluate the vulnerability of the replicas from the network and view it as an oracle.

A number of previous efforts have been made to evaluate distributed algorithms [21, 34]. We use similar measures to evaluate operational cost. In addition, we also include damage cost to evaluate the vulnerability of the system.

The cost factors and cost model have previously been proposed and their definitions are usually subjective to the specific problems [29]. Lee et al. [25] discuss cost factors related to intrusion detection: damage cost, response cost, and operational cost. They assign values empirically to cost factors based on the IDS results and improve the model by reducing some of the cost factors. We use similar terms but with different definitions for consensus protocols.

MTD has been applied to various of areas such as cyber security [18] and mobile wireless networks [15, 33] using techniques such as randomization [19]. In order to take advantage of cost-sensitive model as well as randomization, we do not always choose the minimum-cost solutions [25, 32] but take into concerns of both vulnerability and running cost. Turtle consensus [28] uses a MTD approach that switches between CFT protocol in each round to handle DoS failures but the switching of protocols is deterministic. In comparison, our paper considers a wide range of failures.

III. Preliminaries

In this section, we introduce the system model, our IDS model, and the background of consensus protocols.

A. System Model

We consider a distributed system with n replicas $\mathcal{P} = \{p_0, p_1, \cdots p_{n-1}\}$. Each replica can be viewed as a state machine following some protocol, where a protocol specifies the communication between replicas. We distinguish the types $\mathcal{T}$ of correctness for each replica: correct, crash, or Byzantine. A correct node faithfully follows the corresponding protocol. Faulty replicas may fail by crashing (stop executing the protocol) or be Byzantine (behave arbitrarily). The Byzantine failures we aim to handle are mainly caused by adversary attacks from the network. We assume fair-loss links, where if a message is sent infinitely often by a correct replica, a correct receiver will receive the message infinitely often. Liveness is ensured under partial synchrony [14]: synchrony holds after some unknown global stabilization time.

B. IDS Model for Moving Target Defense

An Intrusion Detection System (IDS) monitors the correctness of each replica p_i (near) real-time, which can fail by crashing. It monitors the host and network devices and detects events that could indicate an ongoing attack [11, 22, 27]. We view the IDS as an oracle that outputs a set of signals (N, C, A) with three probabilities $P_{i,N}$, $P_{i,C}$, and $P_{i,A}$. The three signals represent replica p_i being Normal (correct), Crashed, or under Attack (Byzantine). The three probabilities represent the confidence of each signal, which are refreshed periodically. The value of $P_{i,A}$ is set to 0 initially or after recovery. The IDS also evaluates the cost using our cost model and notifies the replicas the protocol to run. There are several ways to deploy the IDS. For example, we can deploy a network-based IDS over a LAN with a passive architecture. In a passive architecture, it monitors a copy of actual network traffic while no traffic passes through the sensor [30].

C. Consensus Protocols

Consensus protocols tolerate a certain number of failures through the use of redundant replicas. Correctness includes *safety* and *liveness*. Safety guarantees that all the correct replicas decide on the same value and liveness guarantees that all the correct replicas eventually decide. In order to handle f failures, different protocols may vary significantly regarding the minimum number of replicas. We specify three types of protocols: $\boldsymbol{C}$**-1** represents CFT protocols that require at least $f + 1$ replicas where safety may not be guaranteed, $\boldsymbol{C}$**-2** denotes CFT protocols that require at least $2f + 1$ replicas, and $\boldsymbol{B}$**-1** represents BFT protocols that require at least $3f + 1$ replicas. Without loss of generality, for each protocol that requires at least n replicas, we assume there are n replicas. When each protocol is run, safety of the protocol is guaranteed only when the number of corresponding $\mathcal{T}$ failures does not exceed f. Although replicas may be temporarily inconsistent, they can be consistent after switching to another protocol in our approach.

We consider leader-based consensus protocols that operate in *rounds*, where in each normal round a client request is received by replicas, assigned with a sequence number by the leader, agreed by the replicas, executed, and eventually the result is received by the client. The leader is also called the *primary*, which initializes each round. We assume by default the primary is p_0. Other nodes are called *backups* or *backup nodes*. Each protocol consists of several *phases*, where in each phase each replica receives and authenticates certain number of messages and sends messages to some replica(s).

We consider two classes of protocols: *broadcast-based* and *chain-based* [13, 35]. Among broadcast-based protocols, we further distinguish *primary-backup* approaches [5, 10, 17], where primary can communicate with backups but backups do not communicate with each other. In comparison, in regular broadcast-based approaches [7, 20, 23, 24], replicas are fully connected. In contrast, chain-based protocols organize replicas into a logical chain and the head is considered the primary.

Most protocols have *view change* scheme where a backup node takes over when existing primary is faulty. Some protocols have *reconfiguration* scheme to replace some faulty replicas. Protocols may use MACs or digital signatures for authentication. Unless otherwise mentioned, we assume MACs are used. Every protocol has a *checkpoint* scheme, where a replica periodically generates a snapshot of its state, signs a checkpoint message, and sends to other replicas. After the checkpoint becomes stable, previous messages are discarded.

IV. Cost Model

In this section, we formalize the measurement of the costs based on notations in TABLE I. Cost evaluation plays a very important role in our approach to select protocols. Therefore, to precisely evaluate cost factors and select appropriate protocols, we aim at cost metrics that follow these principles: 1) Cost metrics should be measured consistently [26]. 2) Source data should be cheap to gather in terms of time or money. Based on our formalized cost model, the cost can be easily calculated according to the IDS signals. 3) Cost metrics should be evaluated to a value with an associated unit of measures that characterize the value. We define *damage cost* as the number of lost or delayed requests, and *operational cost* as the time of running a normal round of the protocols. The cost factors considered here are standard quantities that all consensus protocols can adopt to perform the cost analysis.

Notation	Meaning
$P_{i,N}$	The probability replica p_i is Normal/correct.
$P_{i,C}$	The probability replica p_i has Crashed.
$P_{i,A}$	The probability replica p_i is being Attacked/Byzantine.
t_0	Transmission time between two replicas.
t_c	Transmission time between client and a replica.
t_1	The time it takes for an IDS to report a crash or an attack.
T_m, T_d	The time it takes to verify or generate a MAC/digital signature.
T_V	The time it takes for replicas to move to a new view.
T_R	The time it takes for recovery/reconfiguration.
λ	The number of requests in each checkpoint.
δ	Number of incoming requests per second.
Δ	Number of incoming requests since last checkpoint.

TABLE I
Notations.

Damage Cost (C_D). The damage cost evaluates the vulnerability of the protocol. We measure it as the sum of expected number of requests that will be lost or delayed due to the faulty replicas, as shown in Equ. (1). The input is a set of probabilities $P_{i,N}, P_{i,C}$, and $P_{i,A}$ for $i = 0, \cdots, n-1$. The $R_{i,c}$ and $R_{i,A}$ are fixed values that denote the number of requests that may be lost or delayed due to the failure of the replica p_i being $\mathcal{T}$ type where $\mathcal{T} = C$ or A. When we consider the cost for each individual replica p_i, we assume p_i is faulty and the number of $\mathcal{T}$ faulty replicas in $\mathcal{P}$ does not exceed f.

$$C_D = \sum_{i=0}^{n-1} (R_{i,C} P_{i,C} + R_{i,A} P_{i,A}) \tag{1}$$

The values for $R_{i,C}$ and $R_{i,A}$ depend on the identity of replica p_i and the protocol, as summarized below.

The primary crashes. For protocols that have view changes and $T_V < t_1$, the incoming requests during T_V time are lost since nodes cannot process any other requests, i.e., $R_{0,C} = T_V \delta$. Otherwise, all the requests since the failure of the primary will be lost, i.e., $R_{0,C} = t_1 \delta$. For simplicity, we only include the case where $T_V < t_1$.

The primary is Byzantine. We assume that the last checkpoint is stable for all the replicas. For CFT protocols with arbitrary failures, all the requests since the last checkpoint are considered lost since there is no guarantee of the safety of the protocol. Therefore, $R_{0,A} = \Delta$. In comparison, the BFT approaches handle this case through view changes. Therefore, the cost is the same with the previous case, i.e., $R_{0,A} = T_V \delta$.

A backup node crashes. All the broadcast-based protocols naturally handle the crash of backup nodes, i.e., there is no cost for this case. However, chain-based protocols suffer from backup failures. For instance, Chain [35] reconfigures faulty replicas. Therefore, requests during reconfiguration are delayed, i.e., $R_{i,C} = T_R \delta$.

A backup node is Byzantine. The correctness of the protocols is closely related to the primary. In C-1 protocols, the requests since last checkpoint will be lost, i.e., $R_{i,A} = \Delta$. This is because the case is indistinguishable from the case where the primary is Byzantine. In contrast, in a broadcast-based C-2 protocol, correct replicas are still consistent if the primary is correct. This is because the primary always sends consistent messages to the replicas. On the other hand, most broadcast-based BFT protocols handle backup failures. An exception happens to protocols like Zyzzyva [23] since it employs two subprotocols. Also, similar to the previous case, chain-based protocols also suffer from backup failures.

Operational Cost (C_O). The operational cost evaluates the cost to run the protocols. We present two types, latency ($C_{O,L}$) and throughput ($C_{O,T}$), and both are evaluated in terms of time according to similar metrics of previous work [34]. The smaller the latency $C_{O,L}$, the smaller the throughput $C_{O,T}$, and the larger the actual throughput will be. In addition, if the checkpoint is not required frequently, the cost of the checkpoint is not included in the operational cost.

Latency $C_{O,L}$. The latency cost is measured as the time of a consensus round starting from the leader receiving the request to the end of the round of agreement. It includes the transmission time and the time for authentication. $C_{O,L}$ is computed according to the following equation.

$$C_{O,L} = \sum_{j=1}^{\#phases} (n_{j,c} T_m + t_0 + n_{j,1} T_m + n_{j,2} T_m) + \varepsilon \tag{2}$$

In the above equation, $n_{j,c}$ is the number of MACs the replica p_j needs to verify and generate for the client, $n_{j,1}$ is the minimum number of MACs p_j needs to authenticate in each phase, and $n_{j,2}$ is the number of MACs each replica needs to generate in each phase. The cost is measured for the normal case when there is no message congestion. Notice that, although in broadcast-based protocols, all the replicas need to run the same steps, they run concurrently. Therefore, we measure the cost for each phase as the cost of a single

replica if all the replicas execute the same step. The total cost will be the sum of cost for each phase. We also include a variable ε, which includes the cost caused by switching to the new protocol, e.g. physical cost of starting a new replica. For simplicity, we do not include it in the examples in §V.

Throughput $C_{O,T}$. $C_{O,T}$ evaluates the time for authenticating and generating MACs or digital signatures of the bottleneck node. This is due to the fact that the bottleneck replica can continue processing new messages before a consensus round completes. Therefore, the transmission time is not included.

$$C_{O,T} = \sum_{j=1}^{\#phases} (n_{j,c}T_m + n_{j,1}T_m + n_{j,2}T_m) \tag{3}$$

V. Consensus Cost

We introduce the costs of 8 protocols to illustrate our cost model, with both CFT protocols and BFT protocols. Specifically, we survey the cost of two C-1 protocols, Remus and Semi-Active, which can guarantee safety only when all the replicas are correct. We also include two C-2 protocols, Paxos and Chain, which are the state-of-the-art CFT protocols. Finally, we present four BFT protocols, Aliph-Chain, BChain, PBFT, and Zyzzyva, which have different performance characteristics and are perfect for case study. In this section, we first briefly introduce the protocols and then show their cost using the notations in TABLE I.

Remus [9]: A primary-backup C-1 approach where the backups periodically obtain checkpoints from the primary to maintain the latest state. It can also be viewed as a semi-passive approach [10]. In the presence of failures, the requests since last one or two checkpoints will be lost depending on the time IDS detects the failure. Namely, if $t_1\delta$ is greater than λ, a node has already been faulty the last stable checkpoint. Therefore, two checkpoints will be lost. Otherwise, only one checkpoint will be lost.

$$C_D = \begin{cases} 2\lambda P_{0,C} + \Delta P_{0,A} + \sum_{i=1}^{f} \Delta P_{i,A} & \text{if } t_1\delta > \lambda \\ \lambda P_{0,C} + \Delta P_{0,A} + \sum_{i=1}^{f} \Delta P_{i,A} & \text{otherwise} \end{cases} \tag{4}$$

$$C_{O,L} = C_{O,T} = \frac{2\lambda T_m + T_d}{\lambda} \tag{5}$$

Semi-Active [3]: A primary-backup C-1 approach where the primary notifies the backups each incoming request so that all the replicas execute them directly. Since backup nodes receive all the requests from the primary, only those requests during view changes will be lost, i.e., $T_V\delta$.

$$C_D = T_V\delta P_{0,C} + \Delta P_{0,A} + \sum_{i=1}^{f} \Delta P_{i,A} \tag{6}$$

$$C_{O,L} = C_{O,T} = (2+f)T_m \tag{7}$$

Paxos [24]: A broadcast-based C-2 approach with two phases: the leader first notifies the backups of the incoming request; each replica sends a message to all other replicas. If a replica collects at least $f+1$ matching messages, it executes the request and sends a reply to the client.

$$C_D = T_V\delta P_{0,C} + \Delta P_{0,A} \tag{8}$$

$$C_{O,L} = (5f+3)T_m + 2t_0 \tag{9}$$

$$C_{O,T} = (3f+2)T_m \tag{10}$$

Chain [35]: A chain-based CFT approach. The head receives a request from a client and then sends along the chain towards the tail and the tail replies to the client. When a replica crashes, a non-faulty master node reconfigures the chain. When a node fails, all the request during reconfiguration will be lost.

$$C_D = T_R\delta P_{0,C} + \Delta P_{0,A} + \sum_{i=1}^{2f} T_R\delta(P_{i,C} + P_{i,A}) \tag{11}$$

$$C_{O,L} = 2(2f+1)T_m + 2ft_0 \tag{12}$$

$$C_{O,T} = 2T_m \tag{13}$$

Aliph-Chain [16]: A chain-based BFT approach. Each replica needs to verify MACs from at most $f+1$ previous replicas and also append MACs for up to $f+1$ subsequent replicas or the clients. The client accepts the reply message when it receives a message from the tail with $f+1$ valid MACs. In the equations, function $F(i)$ represents the latency up to replica p_i.

$$C_D = t_1\delta(P_{0,C} + P_{0,A}) + \sum_{i=1}^{3f}(t_1 + F(i))\delta(P_{i,C} + P_{i,A}) \tag{14}$$

$$C_{O,L} = \sum_{i=0}^{f-1}(f+i+2)T_m + \sum_{i=f}^{2f-1}(2f+2)T_m + \sum_{i=2f}^{3f}(4f-i+2)T_m + 3ft_0 \tag{15}$$

$$C_{O,T} = (2f+2)T_m \tag{16}$$

BChain [13]: A chain-based BFT approach. Being different from Chain and Aliph-Chain, only the first $2f+1$ replicas form a chain and the last f replicas serve as backups which are reconfigured periodically and the message is sent from the head to the $(2f+1)^{th}$ replica. It uses similar authentication scheme with Aliph-Chain. All the first $2f+1$ replicas notify the rest f replicas the execution order so that they are also up-to-date. When failures occur, the chain is reordered by the head with at most f rounds of reconfigurations.

$$C_D = T_V\delta(P_{0,C} + P_{0,A}) + \sum_{i=1}^{2f} T_R\delta P_{i,C} + 3fT_R\delta \sum_{i=1}^{2f} P_{i,A} \tag{17}$$

$$C_{O,L} = \sum_{i=0}^{f-1}(2f+2i+3)T_m + \sum_{i=f}^{2f}(6f-2i+3)T_m + 4ft_0 \tag{18}$$

$$C_{O,T} = (4f+3)T_m \tag{19}$$

PBFT [7]: A leader-based BFT approach with three phases: in the first phase the leader notifies the replicas of the incoming request; replicas exchange their messages until each correct replica collects at least $2f+1$ matching messages in the second and third phase. Replicas then reply to the clients.

$$C_D = T_V\delta(P_{0,C} + P_{0,A}) \tag{20}$$

$$C_{O,L} = (13f+3)T_m + 3t_0 \tag{21}$$

$$C_{O,T} = (10f+2)T_m \tag{22}$$

Zyzzyva [23]: A leader-based BFT approach where clients participate. The leader first notifies the replicas and the replicas directly send a reply to the client. If the client receives matching replies from all the replicas, it accepts the message. If it receives at least $2f+1$ matching messages, it sends them to all the replicas. The replicas then commit the request.

$$C_D = T_V\delta(P_{0,C} + P_{0,A}) + \sum_{i=1}^{3f}\left(\frac{t_1}{(25)(a)} - \frac{t_1}{(25)(b)}\right)(P_{i,C} + P_{i,A}) \tag{23}$$

$$C_{O,L} = \begin{cases} (6f+3)T_m + t_0 & \text{if } 3f+1 \text{ matching} \\ (10f+5)T_m + t_0 + 2t_c & \text{if } 2f+1 \text{ matching} \end{cases} \tag{24}$$

$$C_{O,T} = \begin{cases} (6f+2)T_m & \text{if } 3f+1 \text{ matching} \quad (a) \\ (8f+3)T_m & \text{if } 2f+1 \text{ matching} \quad (b) \end{cases} \tag{25}$$

VI. A Moving Target Consensus Approach

In this section, we first briefly overview our approach and introduce the procedures for switching protocols. Then we show our moving target algorithm for selecting protocols in details. Finally, we show the lower bound for the IDS values and discuss the case when IDS crashes.

Overview of the Protocol. We illustrate our approach in Fig. 1. It contains three components: the IDS, a set of available consensus protocols, and a set of replicas. The IDS monitors the correctness of the replicas and periodically evaluates the costs of the protocols based on the *Moving Target Algorithm*. It selects a protocol and sends configuration messages to the replicas. Namely, by default a cluster of protocols is selected and a random one is used periodically on a set of replicas $\mathcal{P}$. If the damage cost of existing protocol is higher than a threshold, the IDS selects a new cluster of protocols. The IDS can also select a set of new replicas $\mathcal{P}'$ according to the correctness of the replicas. After receiving the configuration message from the IDS, the replicas switch to the new protocol following the procedures in *Moving Target Consensus*.

The underlying idea is that given the IDS indication of failures of some replicas, running the same protocol may cause a large number of lost or delayed requests. If the damage is higher than expected, we should select a set of protocols that causes lower damage while still achieving good performance. The cost model provides the flexibility of selecting the right protocols according to both network and system vulnerability and user requirements.

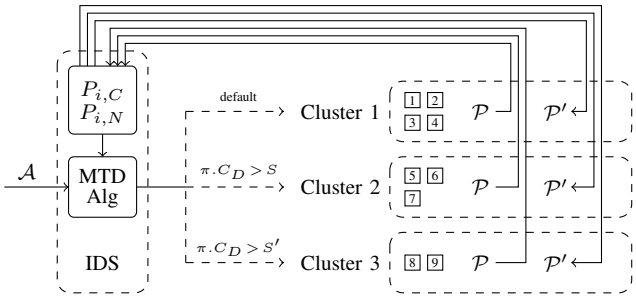


Fig. 1. Moving Target Consensus

Moving Target Consensus. We assume existing protocol π is run on a set of replicas $\mathcal{P}$. Replicas periodically generate checkpoints and authenticate them using digital signatures. In addition to the regular checkpoint steps of the protocols, if π is a C-1 protocol, we let the replicas also forward their checkpoint messages to the IDS.

The IDS updates the costs for all the protocols and sends a configuration message to the replicas periodically. A configuration message includes a protocol π', a set of replicas $\mathcal{P}'$, and the id of a default primary. A protocol π' is randomly selected 1) periodically in the same cluster of π, or 2) when the damage cost of π is higher than the threshold S. In the latter case, a new cluster is selected and a random one is used, as we will discuss in Algorithm 1. All the replicas in both $\mathcal{P}$ and $\mathcal{P}'$ also forward the configuration message to each other to guarantee that the configuration is learned.

If $\mathcal{P}'$ is a subset of $\mathcal{P}$, the leader in $\mathcal{P}'$ *initializes protocol* π' (more details later) and replicas start executing the protocol. Otherwise, replicas need to first *obtain the last stable checkpoint* and then *the new primary initializes* π'.

There are two cases for replicas to obtain the last stable checkpoint. If π is a C-1 protocol, the replicas in $\mathcal{P}'$ obtain a stable checkpoint directly from the IDS. In all other cases, replicas in $\mathcal{P}'$ need to obtain checkpoints from $\mathcal{P}$. Namely, after receiving a configuration message from the IDS, replicas in both $\mathcal{P}'$ and $\mathcal{P}$ send checkpoints to each other. In a checkpoint, with a sequence number greater than the last stable checkpoint, a replica includes all the committed requests in $\mathcal{O}$ and all the accepted but uncommitted requests in $\mathcal{U}$. For C-1 protocols and Aliph-Chain, all the requests are included in $\mathcal{U}$. For C-2 protocols, if a replica receives matching message from $2f+1$ replicas during protocol π, the request is included in $\mathcal{O}$ and other requests are included in $\mathcal{U}$. For B-1 protocols besides Aliph-Chain, each replica includes the committed requests according to the protocols, e.g., a valid *ack* in BChain, etc. If the new primary receives matching checkpoints from at least $f+1$ replicas, it starts π'.

In order to initialize π', the primary selects the last stable checkpoint and uses the state and sequence number l. The primary then determines L where $l+L$ is the largest sequence number found in $\mathcal{O}$ and $\mathcal{U}$. For each sequence number, the new primary selects a request M if at least $f+1$ replicas include M in $\mathcal{O}$ or at least $2f+1$ replicas include M in $\mathcal{U}$ (or $f+1$ for C-1 protocols). It then sends a message to all the replicas in $\mathcal{P}'$. The message includes the last stable checkpoint and a set of selected requests. After receiving the message, replicas process the requests according to π'.

We show in Theorem 1 the switching of protocols is both safe and live. We include the proofs for all the theorems in the Appendix.

Theorem 1. *Let protocol π on a set of replicas $\mathcal{P}$ be switched to protocol π' on a set of replicas $\mathcal{P}'$. If π' tolerates failures with type $\mathcal{T}$ and there are fewer than f $\mathcal{T}$ failures in both π and π', the switching of protocols is both safe and live.*

The Moving Target Algorithm. The underlying idea of out algorithm for selecting protocols is that based on the IDS signals, we evaluate the cost of the existing protocol. If the existing protocol is considered vulnerable regarding a threshold S, we select a new cluster of protocols that is safe and cheap. As shown in Algorithm 1, $\mathcal{A}$ represents all the available protocols we can use, which initially includes all the protocols. The function $top(x, \mathcal{B}.y)$ selects the x^{th} largest value according to the y value in set $\mathcal{B}$. We set up the threshold S to be the $\sigma|\mathcal{A}|^{th}$ largest of the damage cost for protocols in $\mathcal{A}$ where $\sigma \in (0,1)$. When the damage cost of existing protocol is higher than S, indicating that existing protocol may cause larger damage than expected, we start selecting a new cluster. We first filter all the protocols with higher damage cost from $\mathcal{A}$. Then we select protocols with operational cost smaller than the $\theta|\mathcal{A}|^{th}$ protocol according to the operational cost, where $\theta \in (0,1)$. Finally, we do an optional step among protocols in $\mathcal{R}$ to further filter protocols with outstanding

damage cost. Namely, we set up another threshold Λ for damage cost and filter the protocols with damage cost higher than $h + \Lambda$ where h represents the lowest damage cost for protocols in $\mathcal{R}$.

Algorithm 1 The Moving Target Algorithm

```
S ← top(σ|A|, A.C_D)
if π.C_D > S then                {Damage cost of existing protocol is high}
    A ← A.C_D < S                {Remove protocols with high damage cost}
    O ← top(θ|A|, A.C_{O,L})
    R ← A.C_{O,L} < O            {Select protocols with low operational cost}
    h ← top(|R|, R.C_D)
    C ← R.C_D < h + Λ            {Remove protocols with outstanding cost}
```

Notice that we use three parameters: σ, θ, and Λ. σ is used for threshold S in order to determine whether the damage cost of existing protocol is higher than a portion of protocols in $\mathcal{A}$. Similarly, θ is a threshold that is used to select a set of protocols in $\mathcal{A}$ with the lowest operational cost. It is important to select protocols with the similar performance given the damage cost is lower than S. Lastly, Λ is an optional threshold that is used to further choose protocols with low damage cost based on the previous selection. The values of σ and θ can be set up according to the requirement. However, the value of Λ is important to guarantee that we select the right protocols. As shown in Theorem 2, it is also related to the P values from IDS.

Theorem 2. *Let Ω be the damage cost caused by backups for BFT protocols and Λ be the threshold for selecting a cluster. In order for the approach to be safe, the following requirement for IDS holds, where $min(\Omega)$ represents the BFT protocol with minimum damage cost caused by all the backups.*

$$P_{0,A} > \frac{\Lambda + min(\Omega)}{\Delta + T_V \delta} \tag{26}$$

Dealing with Crashing IDS. The IDS generates configuration messages periodically. In order to handle the case where IDS crashes, each replica starts a timer after receiving a configuration message and waits for the next configuration message. If the replica does not receive any configuration message before its timer expires, it sends a $[cids]$ message to other replicas. If a replica receives more than $f + 1$ $[cids]$ messages, it also sends a $[cids]$ messages to other replicas. All the replicas then learn that the IDS has crashed. Then replicas switch to a default fail-safe protocol, in our case PBFT. This is due to the fact that PBFT, in general, has the lowest damage cost among all the protocols we use. This guarantees that the protocol is still safe when IDS crashes. Notice that C-1 protocols require only $f + 1$ replicas. In this case, the failure of IDS can only be detected if all the replicas are correct.

MTD Entropy. Based on Shannon's information entropy [31], MTD entropy is formalized to evaluate the randomness and effectiveness of the MTD model [36]. Specifically, the greater the entropy of the configuration of an MTD system, the more effective the approach is to prevent future attacks. We show the entropy of our approach in Theorem 3.

Theorem 3. *Let $\mathcal{A} = \{\pi_1, \pi_2, \cdots, \pi_m\}$ represent the m protocols we can use. $H(\mathcal{A})$ represents the MTD entropy, which can be denoted as:*

$$H(\mathcal{A}) = H(\pi_1, \pi_2, \cdots, \pi_m) = \sum_{i=1}^{m} p(\pi_i) log(p(\pi_i)) \tag{27}$$

$p(\pi_i)$ represents the possibility π_i is selected:

$$p(\pi_i) = \frac{1}{\sigma|A|} \tag{28}$$

Given that Λ is large enough, after selecting a cluster, the probability of each protocol being used is:

$$p(\pi_i) = \frac{1}{(1-\theta)(1-\sigma)|A|} \tag{29}$$

Based on this theorem, if the switching of protocols is deterministic, the entropy is 0 since the probability of each protocol is 1. In comparison, in our example, $|A| = 8$ and let $\sigma = \Lambda = 0.2$, the entropy for our approach is 8.68 initially and 96.51 after switching. If we simply switch among all the 8 protocols, the entropy is 192.00. We conclude that due to the unpredictability of our approach, we can also prevent further attacks using the randomization method. If there are more protocols in the same cluster, the effectiveness can be further increased.

VII. Evaluation

In this section, we show the evaluation of the effectiveness of our cost model in selecting protocols and the IDS error rate our approach can handle.

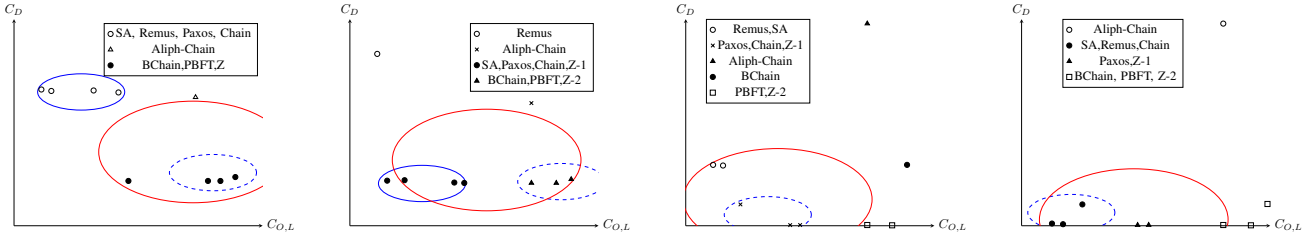
Setting	λ	T_m	T_d	T_R/T_V	$t_1/t_0/t_c$	δ	Δ	P
1	10	0.5	1.0	0.5	1.0	10	15	0.001
2	10	0.5	1.0	0.5	5.0	10	60	0.01
3	10	0.5	1.0	0.1	1.0	10	20	0.01
4	10	0.5	1.0	0.1	10.0	10	105	0.1

TABLE II
Experiment settings. T_m, T_d, T_R, T_V, t_1,t_0, and t_c are measured in ms. P is the default value of the replicas unless specified.

Implementation and Settings. The implementation of the protocols is based on Castro et al.'s implementation of PBFT. We evaluate throughput under failures based on our cost model using $0/0$ benchmark, where the clients issue 0kB request and receive 0kB replies. We test the throughput to demonstrate the effectiveness of our cost model. Experiments are carried out on DeterLab [4], utilizing a cluster of up to 20 identical machines connected through a 100 Mbps switched LAN. Each machine is equipped with a 3 GHz Xeon processor and 2 GB of RAM.

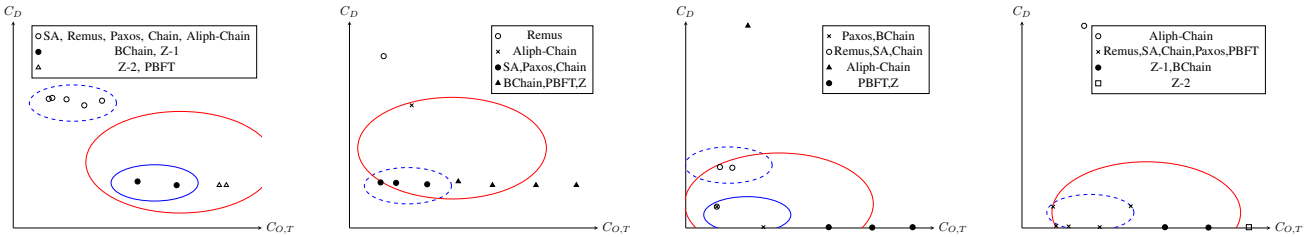
We use several parameters in our cost model. Among them, the P values are the output of the IDS. The values of δ, λ, and Δ are all fixed or preset. In comparison, the values of t_0, t_1, T_m, T_V, and T_R can be obtained through testing. Although the values can be different for different protocols or even for different rounds of each protocol, we can still test them and use average values to measure the cost. For instance, we measure t_0 as the half of the average round trip transmission time between any two correct nodes.

Additionally, we use IDS as an oracle in our cost model. We assign different values to assess our cost model. We evaluate our cost model using 4 settings, as shown in TABLE II, where P represents the default values unless specified.



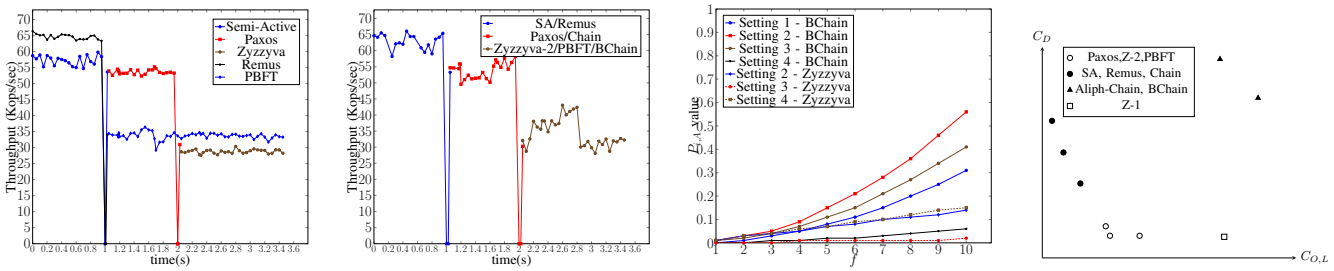
(a) $P_{0,A} = 0.9$. (b) $P_{0,C} = 0.9$. (c) $P_{1,A} = 0.9$. (d) $P_{1,C} = 0.9$.

Fig. 2. Damage cost (C_D) vs. Operational Cost - Latency ($C_{O,L}$) under setting 1 and $f = 1$. SA, Z, Z-1, Z-2 represent Semi-Active, Zyzzyva, Zyzzyva with normal run, and Zyzzyva when at least one backup node fails, respectively.



(a) $P_{0,A} = 0.9$. (b) $P_{0,C} = 0.9$. (c) $P_{1,A} = 0.9$. (d) $P_{1,C} = 0.9$.

Fig. 3. Damage cost (C_D) vs. Operational Cost -Throughput ($C_{O,T}$) under setting 1 and $f = 1$.



(a) Throughput under setting 1 with $f = 2$ and 10 clients. Protocols are not switched periodically in the same cluster.

(b) Throughput under setting 1 with $f = 2$ and 10 clients. Protocols are switched in the same cluster every 0.2 ms.

(c) Minimum requirement for $P_{0,A}$ between Paxos and BChain or Zyzzyva under different settings.

(d) C_D vs. $C_{O,L}$. Setting 1 with $f = 2$ and $P_{0,A} = P_{1,A} = P_{2,A} = P_{0,C} = P_{1,C} = P_{2,C} = 0..9$.

Fig. 4. Evaluation of the cost model and the protocol.

Selection of Protocols. We first evaluate the effectiveness of our cost model in selecting the "right" cluster of protocols, as shown in Fig. 2 and Fig. 3 under setting 1 and $f = 1$. In each experiment, the IDS reports a relatively high P value for one replica, either crash or Byzantine. Based on the figures, we notice that the protocols naturally fall into clusters. For the case where the primary is Byzantine, protocols fall into two clusters. As observed from Fig. 2(a) and Fig. 3(a), the damage cost is high for the CFT protocols and much lower for the BFT protocols. This observation correctly reflects the nature of the protocols. On the other hand, in the case where the primary crashes, the damage cost remains low since most protocols have view changes, as shown in Fig. 2(b) and Fig. 3(b). We observe similar results from the cases where a backup is faulty. Being different from the case where the primary is Byzantine, only Remus and Semi-Active (SA) have very high damage cost compared to other protocols. This can be explained by the fact that Remus and SA require only $f + 1$ replicas and the protocols are no longer safe. For other CFT protocols like Paxos, since the primary is correct, correct replicas are still consistent.

Parameter θ. We evaluate the protocols to determine an empirical value for θ, which is used to select protocols with low operational cost. We use ellipses to show the selection of the protocols in the figure. In practice, the threshold values represent ranges of cost values. As observed in Fig. 2 and Fig. 3, the threshold value can largely impact the selection of protocols. For instance, if we use a tight value, as illustrated in the small ellipses, protocols in general fall into the same category (either CFT or BFT). The only exception we notice is the case where a backup node fails. In this case, CFT protocols and BFT protocols fall into the same cluster, as shown in Fig. 3(b) and Fig. 3(d). However, the protocols are still safe since the primary is correct. The downside is the possibility that very few number of protocols are selected and the selection of protocol becomes predictable. In comparison, more protocols will be selected if we use a larger threshold. However, it is possible that "wrong" protocols will be included. In most cases for the protocols we illustrate, 2 or 3 is an appropriate number, which indicates that $\theta = [0.25, 0.375]$.

Throughput. We assess the throughput under failures. We use 10 concurrent clients and let $f = 2$ and $\sigma = \theta = 0.375$ based on setting 1. We inject a crash failure at time $t = 1s$ and a "Byzantine" failure at $t = 2s$ where the IDS reports a high probability within 0.2 ms. As illustrated in Fig. 4(a), we first do not include periodic switching among protocols and show two typical cases. In the first case, SA is run in the beginning, Paxos is used after a crash failure is injected, and Zyzzyva is selected after Byzantine failures. In the second case, Remus is first run and PBFT is used after failures, where the performance degrades suddenly. We then show in Fig. 4(b) with the same setting but protocols in the same cluster are switched every 0.2 ms. It can be observed that if protocols are switched with a tight bound on operational cost, the performance is in general consistent, where the switching of protocols generates some overhead.

Threshold Λ and IDS Error Rate. The value Λ is used to select the protocol with certain damage cost in $\mathcal{C}$ and we have shown the theoretical bound. In order to determine an appropriate Λ value, we show the minimum requirement for $P_{0,A}$ so that the damage cost of any CFT protocol is lower than the highest of the BFT protocols. This is considered the worst case where the CFT protocol might fall in the same cluster with BFT protocols. We evaluate the costs for all the settings by changing the $P_{i,A}$ values. In each setting, we compare the damage cost of Paxos with that of BChain and Zyzzyva. This is because, in general, Paxos has the lowest damage cost among CFT protocols while BChain and Zyzzyva have the highest damage cost among BFT protocols. Notice that Aliph-Chain itself may have high damage cost, but our approach filters the protocols with outstanding cost. As shown in Fig. 4(c), there exist some settings where the IDS must report a high $P_{0,A}$ value, especially when f is large. In most cases, the IDS does not need to report a P with value higher than 0.5. Based on our observation, we can handle a high IDS error rate so as for the approach to be safe.

Limitations. Our cost model has several limitations. First, it cannot be used to evaluate the case where the number of faulty nodes exceeds f. As shown in Fig. 4(d), we use setting 1 and $f = 2$. $P_{i,C}$ and $P_{i,A}$ for replica 0, 1, and 2 are 0.9. It can be observed that the damage costs for the protocols such as Paxos and PBFT are still low. This is because the cost is measured by assuming that fewer than f faulty replicas are present. Second, as we have shown previously, a high IDS error rate can be tolerated. However, performance can be degraded due to inaccurate IDS results, i.e., when IDS reports an attack while the replicas do not fail. Third, the protocols with high damage cost are removed when the IDS reports more failures. However, we do not provide a scheme to add protocols into the set. This problem can be resolved by periodically recovering the replicas and adding protocols to $\mathcal{A}$.

VIII. Conclusion

In this paper, we present a moving target consensus approach. At the core of our approach is a cost model that can be used to evaluate the damage cost and the operational cost for leader-based consensus protocols that operate in rounds. Based on real-time Intrusion Detection System signals about each replica being correct, crash, or Byzantine, the damage cost evaluates the vulnerability of the protocols while the operational cost evaluates the performance of the protocols. Our approach enables the use of a safe, fast, and unpredictable protocol according to existing system vulnerability. In addition, the cost model can also be viewed as a theoretical model to analyze the characteristics of the consensus protocols.

IX. Acknowledgments

Sisi Duan was sponsored in part by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the Department of Energy. Yun Li and Karl Levitt were sponsored in part by the Army Research Laboratory under Cooperative Agreement Number W911NF-13-2-0045(ARL Cyber Security CRA).

References

[1] *Amazon Web Services (AWS)*. https://aws.amazon.com,.
[2] White paper: VMware high availability concepts, implementation, and best practices. Technical report, VMware, 2007.
[3] White paper: Protecting mission-critical workloads with VMware fault tolerance. Technical report, VMware, 2009.
[4] T. Benzel. The science of cyber security experimentation: the deter project. In *ACSAC*, pages 137–148, 2011.
[5] N. Budhiraja, K. Marzullo, F. B. Schneider, and S. Toueg. *Distributed Systems*. ACM Press/Addison-Wesley, 1993.
[6] M. Burrows. The chubby lock service for loosely-coupled distributed systems. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 335–350. USENIX Association, 2006.
[7] M. Castro and B. Liskov. Practical byzantine fault tolerance. In *OSDI*, pages 173–186, 1999.
[8] B.-G. Chun, P. Maniatis, S. Shenker, and J. Kubiatowicz. Attested append-only memory: making adversaries stick to their word. In *SOSP*, pages 189–204, 2007.
[9] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson, and A. Warfield. Remus: High availability via asynchronous virtual machine replication. In *NSDI*, pages 161–174, 2008.
[10] X. Défago and S. André. Semi-passive replication and lazy consensus. *Parallel and Distributed Computing*, 64:1380–1398, 2004.
[11] D. E. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, SE-13(2):222–232, 1987.
[12] S. Duan, K. N. Levitt, H. Meling, S. Peisert, and H. Zhang. ByzID: Byzantine fault tolerance from intrusion detection. In *SRDS*, pages 253–264, 2014.
[13] S. Duan, H. Meling, S. Peisert, and H. Zhang. BChain: Byzantine replication with high throughput and embedded reconfiguration. In *OPODIS*, pages 91–106, 2014.
[14] C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *Journal of ACM*, 32(2):288–323, 1988.
[15] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 41–47. ACM, 2002.
[16] R. Guerraoui, N. Knežević, V. Quéma, and M. Vukolić. The next 700 bft protocols. *ACM Transactions on Computer Systems*, 32(4):12:1–12:45, 2015.
[17] R. Guerraoui and A. Schiper. Software-based replication for fault tolerance. *Journal of Computer*, 30(4):68–74, 1997.
[18] W. House. Trustworthy cyberspace: Strategic plan for the federal cyber security research and development program. *Report of the National Science and Technology Council, Executive Office of the President*, 2011.
[19] S. Jajodia, A. K. Ghosh, V. Subrahmanian, V. Swarup, C. Wang, and X. S. Wang. Moving target defense ii. *Application of game Theory and Adversarial Modeling. Series: Advances in Information Security*, 100:203, 2013.
[20] R. Kapitza, J. Behl, C. Cachine, T. Distler, S. Kuhnle, S. V. Mohammadi, W. Schröder-Preikschat, and K. Stengel. CheapBFT: Resource-efficient Byzantine fault tolerance. In *EuroSys*, pages 295–308, 2012.

[21] I. Keidar. Challenges in evaluating distributed algorithms. In *Future directions in distributed computing*, pages 40–44. Springer, 2003.
[22] C. Ko, M. Ruschitzka, and K. N. Levitt. Execution monitoring of security-critical programs in distributed systems: a specification-based approach. In *Security and Privacy*, pages 175–187, 1997.
[23] R. Kolta, L. Alvisi, M. Dahlin, A. Clement, and E. Wong. Zyzzyva: speculative byzantine fault tolerance. *ACM Transactions on Computer Systems*, 27(4):7:1–7:39, 2009.
[24] L. Lamport. The part-time parliament. *ACM Transactions on Computer Systems*, 16(2):133–169, 1998.
[25] W. Lee, W. Fan, M. Miller, S. J. Stolfo, and E. Zadok. Toward cost-sensitive modeling for intrusion detection and response. *Journal of Computer Security*, 10(1, 2):5–22, 2002.
[26] E. B. Lennon, M. Swanson, J. Sabato, J. Hash, and L. Graffo. It security metrics. *ITL Bulletin, National Institute of Standards and Technology*, 2003.
[27] T. F. Lunt and R. Jagannathan. A prototype real-time intrusion-detection expert system. In *Security and Privacy*, pages 59–66, 1988.
[28] S. Nikolaou and R. van Renesse. Turtle consensus: Moving target defense for consensus. In *Middleware*, pages 185–196, 2015.
[29] N. Poolsappasit, R. Dewri, and I. Ray. Dynamic security risk management using bayesian attack graphs. *Dependable and Secure Computing, IEEE Transactions on*, 9(1):61–74, 2012.
[30] K. Scarfone and P. Mell. Guide to intrusion detection and prevention systems (idps). *NIST special publication*, 800(2007):94, 2007.
[31] C. E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.
[32] N. Stakhanova, S. Basu, and J. Wong. A cost-sensitive model for preemptive intrusion response systems. In *AINA*, volume 7, pages 428–435, 2007.
[33] P. Traynor, H. Choi, G. Cao, S. Zhu, and T. La Porta. Establishing pair-wise keys in heterogeneous sensor networks. In *INFOCOM*, 2006.
[34] P. Urbán, X. Défago, and A. Schiper. Contention-aware metrics for distributed algorithms: Comparison of atomic broadcast algorithms. In *Computer Communications and Networks, 2000. Proceedings. Ninth International Conference on*, pages 582–589. IEEE, 2000.
[35] R. van Renesse and F. B. Schneider. Chain replication for supporting high throughput and availability. In *OSDI*, pages 91–104, 2004.
[36] R. Zhang, S. A. DeLoach, and X. Ou. Towards a theory of moving target defense. In *MTD*, pages 31–40, 2014.
[37] P. Zieliński. Low-latency atomic broadcast in the presence of contention. In *DISC*, pages 505–519, 2006.

APPENDIX

A. Proof of Theorem 1

Proof. We first prove the following lemmas and then show the correctness of the theorem.

Lemma 4. *For all the BFT protocols, if a correct replica includes a request M in $\mathcal{O}$ with the same sequence number N, at least $2f+1$ replicas accept M with N.*

Proof of Lemma 4: The lemma simply follows the correctness of the protocols and we ignore the details here. ■

Lemma 5. *If π' is run on the same set or a subset of replicas of π, i.e., $\mathcal{P}' \subseteq \mathcal{P}$, the switching to protocol π' is both safe and live.*

Proof of Lemma 5: If π' is run on the same set or subset of replicas $\mathcal{P}$, all the replicas maintain the same state from π. The lemma can then be proved by showing that 1) The new primary can select a set of requests based on the execution history for the switching to be live, and 2) If a correct replica has committed a request, the request will be selected by the new primary for the switching to be safe. Notice that if the primary is not correct, view changes will occur until a correct primary is selected. The correctness of view change is proved according to protocol π' and we only consider the case where a correct primary ensures the safety during the switching of protocols.

If both π and π' are CFT protocols, all the committed and uncommitted requests must be consistent since there are only crash failures. Therefore, the primary can order the requests based on the execution history and the switching of protocols is both safe and live.

Otherwise, if π is a BFT protocol, the primary is able to proceed since if fewer than $2f+1$ replicas have committed a request, it selects null. It is also not possible where two sets of $f+1$ replicas both include M in the $\mathcal{O}$. This can be proved by contradiction where in each set there is at least one correct replica, e.g., p_1 committed M and p_2 committed M'. In both cases, according to Lemma 4, at least $2f+1$ replicas that have already accepted M or M'. Therefore, at least one correct replica has accepted both M and M', a contradiction. Therefore, the switching of protocols is live.

We then prove that if a correct replica has committed a request then the primary will select it. This can also be proved by contradiction assuming a correct replica p_i has committed a request M with sequence number N but the primary selects M'. If the primary assigns M' with N, there are at least $f+1$ replicas that include M' in the $\mathcal{O}$ set, among which there is at least one correct replica. Based on Lemma 4, it indicates that in π at least $2f+1$ replicas has accepted M'. However, since p_i has committed the request M, at least $2f+1$ replicas has accepted M. Therefore, there must be at least one correct replica that has accepted both M and M', a contradiction. Therefore, the switching of protocol is safe and the lemma follows. ■

Lemma 6. *A correct latest checkpoint can be collected based on the replicas in $\mathcal{P}$ running π.*

Proof of Lemma 6: If we use new replica(s) for protocol π', each new replica obtains checkpoints from $\mathcal{P}$ and the new primary selects one with at least $f+1$ signatures and orders the requests with a sequence number greater than the latest stable checkpoint. If π is a C-1 protocol, the replicas must send their signed checkpoint by the primary to the IDS so that IDS can transfer the checkpoint to the new replicas. Since we assume IDS is benign and can only fail by crashing, the new replicas will receive matching checkpoints and the correctness follows. Otherwise, if π is a B-1 protocol, the correctness simply follows the checkpoint scheme for BFT protocols and we ignore the details here. Lastly, if π is a C-2 protocol and there are Byzantine failures, it is possible that correct replicas have inconsistent states and checkpoints. However, the new primary is able to find a stable checkpoint if there exists a checkpoint with at least $f+1$ signatures and it is not possible that there exist two inconsistent checkpoints since there are $2f+1$ replicas. Since checkpoints from replicas can be verified by any replicas due to the use of digital signatures, all the new replicas will accept the checkpoint by the new primary. ■

Lemma 7. *If π' is run on a larger number of replicas than π, i.e., $|\mathcal{P}| < |\mathcal{P}'|$, the switching to protocol π' is both safe and live.*

Proof of Lemma 7: As we show in Lemma 6, if we use new replicas, the primary is able to select a stable checkpoint. Therefore, during the switching of protocols there are three cases for ordering requests with sequence number greater than last stable checkpoint: 1) π is a C-1 protocol and π' is a C-2 protocol, 2) π is a C-1 protocol and π' is a B-1 protocol, and 3) π is a C-2 protocol and π' is a B-1 protocol. The first case is trivial due to the fact that all the replicas are benign and the uncommitted requests are consistent. Therefore, the primary will be able to select a request for each sequence number. We then show the correctness the other two cases.

In the second case, if π is a C-1 protocol like Remus, the new primary selects null request for all the sequence numbers from l to $l+L$ since backups keep their states consistent from the checkpoints. Otherwise, replicas will include their executed requests in $\mathcal{U}$ instead of $\mathcal{O}$. The new primary only selects requests for each sequence number if uncommitted requests are matching for all the replicas. In this case, at least one correct replica has accepted the request. Therefore, the primary can select requests easily and the requests must be accepted by correct replicas. The correctness therefore follows.

In the third case, $\mathcal{O}$ includes requests where the replica collects $2f+1$ matching messages in π for C-2 protocols according to our consensus model. We first show safety that any committed requests by a correct replica will be included by the new primary. We prove by contradiction by assuming a correct replica p_i commits a request M with sequence number N and the new primary includes M' during the switching of protocols. According to our approach, if p_i includes M in $\mathcal{O}$, p_i receives matching messages for M with N from all the $2f+1$ replicas, among which at least $f+1$ of them are correct. Similarly, if the new primary in π' selects M' for N, it finds that at least $f+1$ replicas include M' for N in $\mathcal{O}$ or at least $2f+1$ replicas include M' in $\mathcal{U}$. If at least $f+1$ replicas include M' in $\mathcal{O}$, at least one correct replica includes M' for N and the correct replica receives $2f+1$ matching messages with M', among which at least $f+1$ replicas are correct. If at least $2f+1$ replicas include M' in $\mathcal{U}$, it is straightforward that at least $f+1$ correct replicas accept M'. Since there are only $2f+1$ replicas in π, there exists at least one correct replica that accepts both M and M' for N, a contradiction. Therefore, the protocol is safe.

We only need to prove liveness for the third case where the primary will be able to select a request for each sequence number. We show that it is not possible where there exists M and M' with the same sequence number, at least $f+1$ replicas include in $\mathcal{O}$ or $2f+1$ replicas include in $\mathcal{U}$. It is trivial that if $2f+1$ replicas include a request in $\mathcal{U}$, all the replicas accept the request. If at least $f+1$ replicas include a request in $\mathcal{O}$, at least one of them is correct. The correct replica must have received matching messages from $2f+1$ replicas in protocol π. Therefore, it is not possible that there exists M and M' with the same sequence number. The switching to protocol π' is live and the correctness of the lemma follows. ■

We now show the correctness of the theorem. During the switching of protocols, since we use $f+1$ replicas for C-1 protocols, $2f+1$ for C-2 protocols, or $3f+1$ for B-1 replicas, there are in total three cases: 1) the new protocol runs on the same number of replicas, 2) the new protocol runs on more replicas, and 3) the new protocol runs on fewer replicas. We have already show in Lemma 5 the first two cases are safe and live if $\mathcal{P}' \subseteq \mathcal{P}$. We also include the case where if new replicas are used in π', all the replicas will be able to use a consistent checkpoint and state in Lemma 6. Notice that if there are new replicas in $\mathcal{P}'$, replicas must be able to obtain consistent checkpoint from $\mathcal{P}$ since if π' runs on the same number or smaller number of replicas, the type of failures π' tolerates is weaker than or the same with π. Therefore, all the new replicas in $\mathcal{P}'$ can obtain consistent state. Finally, we also show in Lemma 7 the last case is safe and live. The correctness of the theorem then follows. □

B. Proof of Theorem 2

Proof. In order for our approach to be safe, we always need to guarantee that in the worst case when the primary is Byzantine, the damage cost of any CFT protocol is high enough so that it does not fall into the same cluster with other BFT protocol. Therefore, when the CFT protocol that has the lowest damage cost is greater than any BFT protocol, no CFT protocols will fall into the same cluster with BFT protocols. This requires a value of $P_{0,A}$ that is high enough regarding the threshold for selecting a cluster. We notice that Paxos has the lowest damage cost among the CFT protocols we illustrate, which has damage cost as shown in Equ. (8) and all the BFT protocols with view changes have the same pattern as follows.

$$C_D = T_V \delta(P_{0,C} + P_{0,A}) + \Omega \tag{30}$$

Ω represents the expected damage cost from the failures of backup nodes. Therefore, the following equation follows according to the selection of $\mathcal{C}$ in Algorithm 1.

$$T_V \delta P_{0,C} + \Delta P_{0,A} > T_V \delta(P_{0,C} + P_{0,A}) + min(\Omega) + \Lambda \tag{31}$$

In Equ. (31), $min(\Omega)$ represents the minimum damage cost by backups among all the BFT protocols. We then have the following:

$$(\Delta - T_V \delta) P_{0,A} > min(\Omega) + \Lambda \tag{32}$$

Therefore, Theorem 2 follows. □

C. Proof of Theorem 3

Proof. Since the threshold S is set to $\sigma|\mathcal{A}|$, in the beginning there are $\sigma|\mathcal{A}|$ replicas and the probability follows.

According to Algorithm 1, we first filter the protocols with damage cost greater than $\sigma|\mathcal{A}|$ protocols and there are $|\mathcal{A}'| = (1-\sigma)|\mathcal{A}|$ protocols. Next, we filter protocols with operational cost greater than $\theta|\mathcal{A}'|$ protocols. Therefore, there are $(1-\theta)(1-\sigma)|\mathcal{A}|$ protocols in $\mathcal{R}$. If we assume a large enough Λ value, set $\mathcal{C}$ has $(1-\theta)(1-\sigma)|\mathcal{A}|$ protocols and we switch among them, the theorem then follows. □