

---

# Attack and Defense Strategies for Intrusion Detection in Autonomous Distributed IoT Systems

Hamid Al-Hamadi<sup>1</sup>, Ing-Ray Chen<sup>2</sup>, Ding-Chau Wang<sup>3</sup>, and Meshal Almashan<sup>4</sup>

<sup>1</sup>Department of Computer Science, Kuwait University, Safat 13060, Kuwait

<sup>2</sup>Department of Computer Science, Virginia Tech, 7054 Haycock Road, Falls Church, Virginia 22043, USA

<sup>3</sup>Department of Information Management, Southern Taiwan University of Science and Technology, Tainan 71005, Taiwan

<sup>4</sup>Graduate School of Engineering, the University of Tokyo, Bunkyo-ku, Tokyo 113-0033, Japan

Corresponding author: Hamid Al-Hamadi (e-mail: hamid@cs.ku.edu.kw).

This work was supported and funded by Kuwait University Research Grant #RQ02/18. This work is also supported by the U.S. AFOSR under grant number FA2386-17-1-4076.

**ABSTRACT** In this paper, we develop a methodology to capture and analyze the interplay of attack-defense strategies for intrusion detection in an autonomous distributed Internet of Things (IoT) system. In our formulation, every node must participate in lightweight intrusion detection of a neighbor target node. Consequently, every good node would play a set of defense strategies to faithfully defend the system while every bad node would play a set of attack strategies for achieving their own goals. We develop an analytical model based on Stochastic Petri Net (SPN) modeling techniques. Our methodology allows the optimal defense strategies to be played by good nodes to maximize the system lifetime when given a set of parameter values characterizing the distributed IoT system operational environment. We conduct a detailed performance evaluation based on an experiment dataset deriving from a reference autonomous distributed IoT system comprising 128 sensor-carrying mobile nodes and show how IDS defense mechanisms can counter malicious attack mechanisms under the ADIoTS system while considering multiple failure conditions.

**INDEX TERMS** Intrusion Detection, Internet of Things, mission-oriented IoT systems, Stochastic Petri Net, attack/defense behavior models.

## I. INTRODUCTION

Security of Internet of Things (IoT) is of paramount importance given its widespread adoption. This is especially critical for security-sensitive IoT systems tasked with disaster recovery, evacuation, and military operations. In this paper, we develop a methodology to capture and analyze the interplay of intrusion detection attack-defense strategies in an Autonomous Distributed Internet of Things System (ADIoTS). An instance of ADIoTS is a mission-oriented IoT system populated with autonomous, smart IoT devices including smart sensors, actuators, and control nodes, for executing a specific mission. Possible application scenarios may involve a team of Unmanned Aerial Vehicles (UAVs), soldiers, automobiles, or robots monitoring and patrolling a combat area, and relaying critical information to the base for combat advantages. Such IoT devices (called nodes in this paper for short) can be compromised via capture attacks (through physical or cyber space) and turned into insiders performing various malicious attacks with the objective to fail the mission. Thus, an Intrusion Detection System (IDS) is called for to detect and remove inside attackers in the

ADIoTS to ensure successful mission execution. Given the high threat of attacker strategies on the system, defense strategies must be put in place to counter such threats.

We design the ADIoTS such that all nodes in the ADIoTS are expected to perform not only tasks assigned to them but also IDS duties to defend the system. Malicious nodes, however, can choose from a set of attack strategies with the objective to retain malicious nodes (thus causing false negatives) and evict good nodes (thus causing false positives) with the ultimate goal to fail the mission. Good nodes, on the other hand, can choose from a set of defense strategies to prolong the system lifetime. The attack/defense behaviors manifest into the false negative probability (i.e., missing a malicious node as a good node) and false positive probability (i.e., misidentifying a good node as a malicious node) which together affect the system lifetime. Here an attacker refers to an inside attacker and a defender refers to a good node.

Our approach is based on distributed voting-based detection. We utilize SPNs as a mathematical tool to model smart attack and defense behaviors of nodes in a mission-oriented ADIoTS operating under our collusion-aware

voting-based IDS scheme, with the objective to analyze and identify the optimal parameter settings of our collusion-aware voting-based IDS design that can optimize the system lifetime. More specifically, we develop node SPN models each keeping track of the status of one node in an ADIoTS as events happen in the system and identify optimal defense strength parameters (in terms of the detection interval length and the number of voters) in response to attacker characteristics and the sensed environment to maximize the system lifetime. While the importance of designing effective IDS strategies for detecting malicious nodes is well recognized, the literature is thin in modeling the interplay of attack/defense strategies and their effects on system reliability. Our work follows model-based evaluation. The novelty lies in setting up IDS duties that every node must participate in, thus forcing attack/defense interplay to go in a direction toward the designer’s desirable outcome, i.e., prolonging the system lifetime.

Our work has the following unique contributions:

- 1) We develop a methodology to capture and analyze the interplay of attack-defense strategies while attackers/defenders execute their required IDS functions in the form of voting-based intrusion detection in an ADIoTS.
- 2) We develop an analytical model based on Stochastic Petri Net (SPN) modeling techniques [1] to describe the dynamics of IDS attack/defense strategies and examine their effect on system lifetime.
- 3) We develop a novel iterative computational procedure with computational complexity of  $O(n)$  where  $n$  is the number of nodes in an ADIoTS to make it computationally feasible to analyze a large ADIoTS.
- 4) We provide a detailed security evaluation in Section VI showing how defense strategies in our collusion-aware voting-based IDS design can effectively counter malicious attack strategies in an ADIoTS while considering multiple failure conditions.
- 5) We compare our proposed collusion-aware voting-based IDS (CAVBIDS) scheme with baseline IDS schemes, and show how it outperforms these systems under smart and collusion-based attacks.

Table I below lists acronyms and abbreviations used in the paper. Table II lists the parameters used by the analytical model. The rest of the paper is organized as follows:

Section II surveys related work. Section III discusses the system model including intrusion detection attack-defense strategies. Section IV develops an analytical model and an iterative computational procedure for quantifying the effect of attack/defense strategies on system lifetime. Section V discusses how optimal defense strategies are applied to maximize system lifetime. Section VI conducts security evaluation. Finally, Section VII summarizes the paper and outlines future work.

TABLE I  
ACRONYMS AND ABBREVIATIONS

Acronym	Meaning
---------	---------

CAVBIDS	Collusion-Aware Voting-Based IDS
IoT	Internet of things
IDS	Intrusion detection system
ADIoTS	Autonomous distributed internet of things system
SPN	Stochastic petri net
MTTF	Mean time to failure
TEU	Task execution unit

TABLE II  
NOTATIONS

Symbol	Meaning
$P_a$	Random attack probability
$m$	Number of voters per IDS voting
$T_{IDS}$	IDS interval
$P_{alc}$	Random attack probability for low capability node
$P_{ahc}$	Random attack probability for high capability node
$P_i^g(t)$	Probability of node $i$ being good at time $t$
$P_i^b(t)$	Probability of node $i$ being bad at time $t$
$P_i^e(t)$	Probability of node $i$ being evicted at time $t$
$P_{li}^l(t)$	Probability that node $i$ is located in area $l$ at time $t$
$P_{fn}^{IDS}$	System IDS false negative probability
$P_{fp}^{IDS}$	System IDS false positive probability
$H_{pfn}$	Host IDS false negative probability
$H_{pfp}$	Host IDS false positive probability
$P_e$	Percentage of energy spent per $T_{IDS}$
TEU	Task execution unit
$RT$	Residence time in current location
$LOC$	Location subnet of node SPN model
$UCN$	Undetected Compromised Node subnet of node SPN model
$DCN$	Detected Compromised Node subnet of node SPN node
$TASKS$	Tasks subnet of node SPN model indicating tasks completed

## II. RELATED WORK

Many studies have focused on providing intrusion detection capability to protect vulnerable IoT-based networks from malicious attacks. Benkhelifa et al. [2] discuss advancements in intrusion detection practices in IoT. They discuss how preventive security measures for IoT systems are difficult to implement and stress the importance of a second line of defense such as using an IDS. They further suggest that due to the distributed nature of IoT, a distributed or hierarchical IDS is most suitable. We adopt the same approach in this paper. In [3], a protocol for centralized and active malicious node detection is proposed. Malicious nodes are detected by identifying changes by a genetic algorithm-based data gathering scheme through a matrix comparison method

---

executed by a mobile sink with unlimited energy capability. They consider the energy exhaustion of IoT nodes and lifetime duration. Furthermore, they focus on collecting digital forensic evidence to be used against malicious nodes. Their protocol however only focuses on detecting energy exhaustion and fake information reporting attacks and does not consider more sophisticated attacks against IoT deployments. In [4], the authors propose and analyze an intrusion detection architecture for resource constrained IoT networks. They use Raspberry Pi as a commodity single board computer with the snort open-source Intrusion Detection System and evaluate the resulting performance of the actual deployment. They prove that their proposed architecture can effectively run the IDS on Raspberry Pi in a distributed IoT system with limited resources and explain how the Raspberry Pi hardware mitigated problems with deploying snort on wireless mesh networks. Similarly, [5] proposes an intrusion detection model based on machine learning where feature selection is linked to specific types of expected attacks on the IoT system. They also run their IDS on a Raspberry Pi system claiming their lightweight protocol does not sacrifice on detection performance. In [6], a host-based intrusion detection and mitigation framework for smart devices deployed in home environments is proposed. Their framework detects intruders and blocks them from reaching their target. The system is designed with the flexibility to dynamically include patterns of known attacks, thus specifying related features to employ with customized machine learning techniques. It provides a comprehensive architecture for intrusion detection in smart homes, with a clear implementation following the OpenFlow protocol [7]. They, however, take a centralized approach to intrusion detection as they rely on a centralized intrusion detection manager, and their work concentrates on protecting limited devices in a home IoT setting. You et al. [8] present a monitoring solution to identify misbehaving embedded IoT devices. Their solution is based on a lightweight behavior rule specification system. In their study they have concluded that the misbehavior detection techniques that are based on rule specifications outperform the contemporary ones that are anomaly-based, for a UAV cyber physical system. Compared to the above cited work, our work does not focus on host-level (i.e., one-to-one) detection methods for identifying host-level IDS security measures such as the host-level false positive probability and the host-level false negative probability. Rather, our work focuses on a system-level detection method in the form of majority voting by requesting each node that is assigned to monitor a target node to report its host-level intrusion detection results obtained through host-level detection methods. The system-level voting outcome subsequently determines if the target node is behaving or not. Furthermore, we develop a methodology to capture and analyze the interplay of attack-defense strategies while attackers/defenders execute their required IDS functions during majority voting.

Machine learning techniques have been used in IDS design [9-12]. In [9] the authors propose a misuse intrusion detection framework for a wireless local area network based on majority voting that differentiates between attacker and legitimate node patterns by examining mac-layer frames. Their system uses several machine learning techniques where the best performing classifiers are chosen to get strong generalization. Then, majority voting is performed to get better accuracy. In [10], the authors develop a novel machine learning based IDS for IoT. The authors stress the importance of securing IoT devices which are considered as the weakest link and vulnerable to a variety of attacks. Their proposed protocol aims to detect malicious IoT nodes by first learning the behavior of the IoT based network, then applying a rule-based approach configured by the administrator. Their work however is only focused on detecting network scanning and simple forms of denial of service attacks. Similarly, the authors in [11] use a machine learning IDS approach by collecting data through dedicated sniffers, followed by stages of generating correctly classified instances, an iterative linear regression stage, and then finally a detection threshold is being applied to separate normal from malicious nodes. Their work does not consider lifetime and only considers blackhole and flooding attacks without considering colluding and smart attackers. In [12], a smart approach for intrusion detection and prevention system in mobile ad hoc networks is proposed. Their proposed system relies on machine learning approaches methods where intrusion detection depends on the four entities of packet analyzer, preprocessing unit, feature extraction unit and classification unit. They perform security analysis by considering false positive rate and detection rate. These works cited above [9-12] all utilize machine learning techniques for providing host-level intrusion detection. Our work is different in that we aim to provide system-level intrusion detection in the form of IDS voting by which each voting node selected (which can be good or bad) reports its host-level intrusion detection outcome as input. We aim to obtain the best system-level intrusion detection outcome by analyzing the interplay of attack-defense strategies employed by IoT devices (good or bad) and identifying the best defense strategies to be employed in an ADIoTS.

In [13], the authors discuss the limitations in the state-of-the-art counter-measures against the security threats in health IoT. They identify internal attacks, where cryptographic keying material can be compromised, as serious threats to the IoT system especially due to the difficulty in detecting such compromised nodes. The authors identify key aspects to counter security threats in health IoT including using a distributed system instead of centralized, considering energy efficiency, and countering colluding nodes orchestrating attacks that are difficult to be detected by traditional methods. In our work, we also consider these aspects in our model and identify optimal intrusion detection parameters to maximize the IoT system lifetime under colluding attackers. In [14], a distributed anomaly detection system for IoT is

proposed. The authors discuss how internal attackers can establish themselves as legitimate nodes within the network, and how an internal anomaly detection system is required to prevent such internal attackers. They devise a control message scheme integrated with a routing protocol for low-power and lossy networks, to report the anomaly to an edge-router node. The system has a configurable profile in which detection and grading parameters are defined. They further specify the system architecture operating at the network and link layers. While their work exhibits distributed attributes and considers inside attackers, the final decision regarding the anomaly in their protocol is done at the edge router, thus creating the possibility of single point of failure vulnerabilities. Unlike [14] our work does not have a single point of failure. Rather, the system-level intrusion detection method designed in the form of IDS voting is totally distributed and resilient to failures or compromises of one third of IoT nodes in an ADIoTS.

Another line of related work for IoT systems in the literature focuses on lifetime maximization in the presence of inside attackers. The authors in [15] propose and analyze a technique that is based on a behavior-rule specification for medical devices intrusion detection. The medical devices studied here are those embedded in Medical Cyber Physical Systems (MCPSs) where the patient's safety is of the highest priority. The authors propose a methodology of transforming behavior rules into a state machine enabling the identification of monitored devices that are changing their behavior from the behavior specifications. The analysis of system lifetime is studied after defining system failure conditions. In [16], the authors propose an adaptive network defense management system for defending against smart attacks and selective capture that disrupt the basic functionality of data delivery in a wireless sensor network. The authors develop an analysis methodology for identifying the best settings of the defense protocol by which the lifetime of the sensor network against smart attacks and selective captures is maximized. Also, a simulation study has been conducted to validate this methodology. The protocol settings include the redundancy level for multi-path routing, the radio adjustment, the intrusion invocation interval and the number of voters. In [17], the authors present an intrusion detection and prevention mechanism suitable for low-power IoT deployments, where an intelligent security architecture is implementing using random neural networks by learning the normal behavior of the system and then embedding the base station with the trained random neural networks. Their method additionally relies on creating tags linked to memory accesses in order to detect out of bounds memory accesses indicating malicious activity. Their work follows anomaly-based mechanisms for intrusion detection in IoT where computation is offloaded to a base station. In [18], the authors design and evaluate a trust-based distributed intrusion detection mechanism for IoT where nodes use trust-based mechanisms to manage neighbor reputation scores. A border router or cluster head is used to calculate trust values based on collected direct trust and reputation trust values of

neighbors. They claim the proposed mechanism is suited to small IoT devices and can be applied to the health domain. Their work however only considers attacks of selective forwarding and sink hole on RPL (Routing Protocol for Low-Power and Lossy Networks) performed by inside attackers, and does not consider smart attacks, collusion of nodes, or finding optimal IDS parameters to maximize system lifetime. Compared to the works cited above [15-18], our work also adopts model-based analysis, i.e., we develop an analytical model based on Stochastic Petri Nets (SPNs) [19] to analyze the interplay of attack-defense strategies employed by voting members during IDS voting and identifying the best defense strategies to be employed to maximize the IoT system lifetime, given system failure conditions as input. Unlike [15-18], we do not consider the use of a centralized entity for conducting intrusion detection. Specifically, the proposed system-level intrusion detection is conducted in the form of IDS voting which is totally distributed. The only requirement is that each node when acting as a host IDS uses lightweight host-level detection techniques in order to cast a vote regarding the behavior of a target node in its vicinity.

An SPN model is an analytical model allowing a system's behaviors and states to be clearly defined since it will be transformed (by SPNP [1]) into a Markov model, thus providing a state-based strong modeling foundation. Unlike simulation which is laborious and lacking accuracy unless it is run sufficiently long to satisfy statistical significance, SPN models facilitate answering why if types of design questions accurately and rapidly because it merely involves computation. We use Stochastic Petri Nets (SPNs) [19] to model the interplay of attack-defense strategies employed by voting members during IDS voting with the goal to identify the best defense strategies to be employed to maximize the IoT system lifetime. In the literature, SPN models have also been used by many scholars [20-29] to analyze reliability and performance characteristics of cyber physical systems. In particular, the authors in [25] propose an analytical model based on SPNs for cyber-physical systems to capture the dynamics between defense and adversary behavior. In their work, they consider different types of failures that can occur in cyber-physical systems such as pervasion failure, exfiltration failure, and attrition failure. They further illustrate the parameterization process by using a modernized electrical grid and discuss the optimal design conditions by which the mean time to failure of the modernized electrical grid is maximized. In [26] Mitchell and Chen study the reliability of a cyber-physical system utilizing an IDS. A probability model that is based on SPNs is developed to analyze the CPS's behavior with the existence of malicious nodes over a wide range of attackers' behaviors. In addition, an Intrusion Detection and Response System (IDRS) is also presented in this study. In [27], the authors propose a SPN-based approach for modeling and analysis of disaster recovery solutions for IoT infrastructures where mission-critical IoT systems need real-time decision-making. A disaster recovery cloud is used in the case of a disaster

TABLE III  
SUPPORTING FEATURES OF THE PROPOSED SYSTEM COMPARED TO THE AVAILABLE LITERATURE

Previous Work	Intrusion Detection	IoT-Based	Tackles Inside Attackers and Collusion	Considers Lifetime and Energy Consumption	Distributed	SPN Model-Based Approach
[5]	✓	✓	-	Partial	-	-
[10],[11]	✓	✓	-	-	-	-
[6],[8],[12],[17]	✓	✓	Partial	-	-	-
[15]	✓	✓	Partial	Partial	-	-
[4]	✓	✓	-	✓	✓	-
[13]	✓	✓	✓	Partial	✓	-
[14],[18]	✓	✓	Partial	-	✓	-
[25]	✓	-	Partial	✓	-	✓
[16]	✓	-	✓	✓	✓	-
[26]	✓	-	✓	✓	✓	✓
[27],[28]	✓	✓	-	✓	-	✓
[29]	✓	-	-	Partial	-	✓
Proposed system	✓	✓	✓	✓	✓	✓

affecting the primary site’s components including medical servers and physician and emergency workstations. IoT network components are incorporated into the model to represent the connectivity between IoT medical devices and the primary site. Using the Petri net-based modeling approach, the system availability, cost, and recovery time of the disaster are analyzed. The authors show the applicability of their work through modeling a real-world healthcare IoT system. In [28], the authors propose a SPN model of a defensive maneuver cyber platform utilizing moving target defense and deceptive defense tactics. Their aim is to utilize defense cyber maneuver techniques to provide survivability and operational continuity. In their model, each system node can be in one of the operational, idle, or deceptive states as a defense response based on the attack, such that all states satisfy the specified constraints ensuring the stability of the maneuver system. By changing node states, and the rates in which nodes transition between states, a more defendable platform can be provided. Finally, they show trade-offs between security and operations. In [29], a SPN model is used to model and analyze threats in smart factories. They consider various threat scenarios including attacks and errors the cause availability risks, and their impact on the components of information networks in the area of connected production environments.

Our work is different from the above cited works [20-29] in four aspects. First, to the best of our knowledge we are the first to explore SPNs for modeling a mission-oriented ADIoTS, whereas the above cited works [20-29] utilize SPN techniques mainly for reliability and performance analysis of wireless networks or cyber physical systems. In our work, we adopt SPN modeling techniques to specifically model and analyze the security property of a mission-oriented ADIoTS

operating under our proposed collusion-aware voting-based IDS (CAVBIDS) scheme. Second, we develop an SPN model that allows us to analyze defense strategies to counteract attack strategies performed by inside attackers that would cause system (i.e., mission) failures. Our SPN model considers smart attacks such that attackers may attack strategically in order to maximize system failure probability. Our SPN model also considers colluding malicious nodes with the ability to work together to evade intrusion detection and evict good nodes. Third, our SPN model considers both energy and lifetime of the system as well as the interplay of attack/defense strategies for finding optimal IDS parameters to adaptively counter these smart attacks such that the system lifetime is maximized while achieving specified mission objectives. Finally, we demonstrate how CAVBIDS can apply the optimal IDS parameters identified from our SPN model to dynamically defend against smart collusion attacks.

Table III above summarizes our proposed system compared to the available surveyed literature.

### III. ADIoTS SYSTEM MODEL

We consider an ADIoTS comprising mission-oriented members or IoT devices (i.e. nodes) where all nodes in the ADIoTS are expected to perform mission tasks and IDS duties in order to achieve mission goals. In this work we do not differentiate between a member and an IoT device but instead treat them synonymously. IDS activities are performed by the ADIoTS where each node is responsible for participating in intrusion detection periodically at every intrusion detection interval (namely  $T_{IDS}$ ). IoT devices can be compromised via capture attacks (through physical or cyber space) and turned into insiders performing various malicious attacks. An insider is a legitimate member of the mission group and thus has



access to the group security key for group communication, and it will remain undetected unless it exhibits malicious behavior and is detected by the IDS. An inside attacker can perform packet-dropping attacks by randomly dropping packets passing through it, integrity attacks by packet modification, and denial of service attacks by overwhelming the network/server with service requests. In this paper, we propose a voting-based distributed IDS design to detect and remove inside attackers in the ADIoTS to ensure successful mission execution.

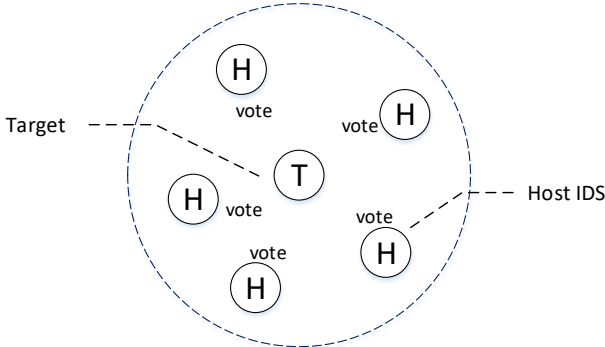


FIGURE 1. Distributed voting-based Intrusion Detection.

At the host-level, a node that is assigned to monitor a neighbor target node will use its host IDS capability based on lightweight anomaly detection mechanisms to judge if the neighbor target node is behaving or misbehaving (see Fig. 1). At the system level, we assume that the mission commander in a well-protected area will send a mobile sink node (e.g., a drone) at every intrusion detection interval (namely  $T_{IDS}$ ) to collect votes from IoT nodes who have been assigned to monitor a target node and then based on the voting outcome make a decision about whether the target node is behaving or misbehaving. Note that the mobile sink node sent by the mission commander is not a single point of failure because its only function is to collect votes from participating nodes that perform host-level intrusion detection on a target node. Should the mobile sink node fail to return votes to the mission commander, the mission commander can send another one immediately. When asked to express its opinion about whether a target node in the neighborhood is behaving, a node must vote “yes” (meaning behaving) or “no” (meaning misbehaving) toward the target node. A malicious node can perform “ballot-stuffing” attacks by voting “yes” toward another malicious node to keep the malicious target node in the system. A malicious node can also perform advanced collusion attacks, including “ballot-stuffing” attacks by voting “yes” toward another malicious node to keep the malicious target node in the system, and “bad-mouthing” attacks by voting “no” toward a good node to evict the good target node from the system. When the majority of votes is “no” the target node is evicted. For the case in which a malicious node is voted “yes” by a majority, the system results in a false negative. For the case in which a good node is voted “no” by a majority, the system results

in a false positive. Malicious nodes would apply the “best” attack strategies with the goal of shortening the system lifetime. Good nodes (i.e., defenders) on the other hand would select the “best” defense strategies to prolong the system lifetime. The attack/defense behavior therefore is set up within the context of IDS voting whose effectiveness is measured by the false negative probability and false positive probability which together affect the system lifetime. We note that a good node’s host IDS is not perfect, so it may miss detecting a bad node. That is, a good node can miss detecting a bad node with a host-level false negative probability  $H_{pfn}$  and it can misidentify a good node as a bad node with a host-level false positive probability  $H_{pfp}$ . Such values are frequently small (e.g., less than 5%) and are assumed to be known before each node is released to operation by software engineering testing.

Furthermore, the mission’s success is dependent on the nodes collectively completing the required mission tasks. We consider that the ADIoTS nodes are heterogeneous with respect to memory and processing capability. While a low capability node may complete a small number of tasks, high capability nodes may have the capability to complete a larger number of tasks within the same time interval. We consider that nodes with similar capability will execute the same number of tasks within the same time duration. We map the amount of task work to a unit of task completion denoted by task execution unit (TEU), to effectively measure the mission group’s tasks completion, as opposed to counting the number of tasks. Fig. 2 depicts the intrusion detection and mission task execution of the ADIoTS. Each IoT device is responsible for completing TEUs based on its capability. IoT devices communicate with each other based on IoT machine-to-machine (M2M) wireless communication protocols such as MQTT [30] and LWM2M [31] without the need to connect to the broader Internet.

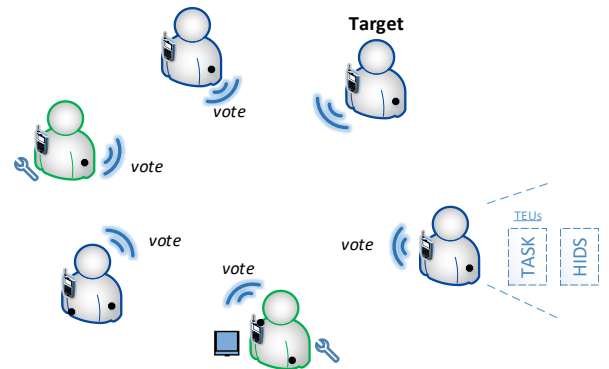


FIGURE 2. ADIoTS nodes perform distributed voting-based intrusion detection and execute mission tasks. ADIoTS nodes are heterogeneous with low capability (blue) and high capability (green). Each node is equipped with modules for executing intrusion detection and executing tasks (i.e., TEUs).

#### A. SYSTEM FAILURE TYPES

We consider the following system failure types:

- Byzantine failure [32]: A Byzantine failure occurs if one third or more IoT devices in the ADIoTS have been

compromised as there is no way to reach a consensus for decision making.

- **Attrition failure:** An attrition failure occurs if the ADIoTS does not have enough IoT devices left to carry out its mission.
- **Resource depletion failure:** A resource depletion failure occurs if the energy of IoT devices is too depleted to be able to accomplish the mission.
- **Application failure:** If the number of tasks completed does not meet the minimum threshold required to meet the mission’s objectives. Such objectives could include gathering location-based measurements, performing certain calculations while deployed, or performing physical tasks (actuating) while deployed. These tasks can only be performed by nodes that have not been evicted. The number of tasks (or equivalently, completion percentage) is application dependent, where critical and security sensitive applications may have a higher threshold (i.e. lower tolerance) for task completion.

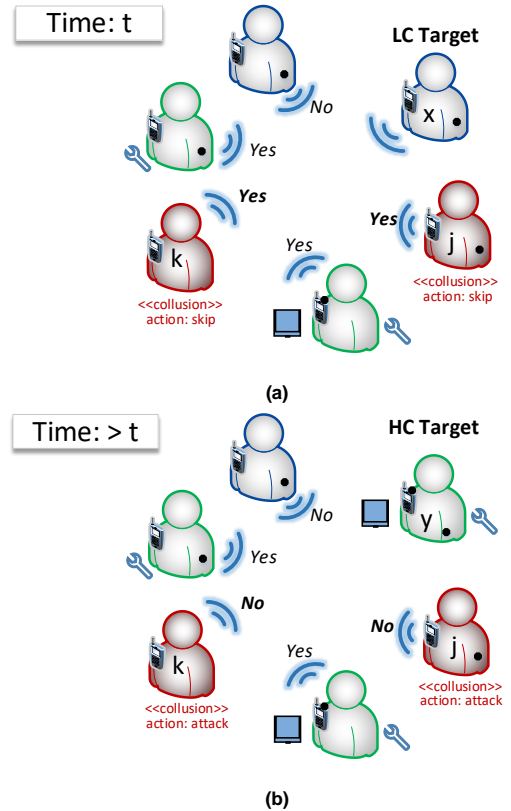
### B. ATTACK STRATEGIES

Attack strategies used by a malicious node during IDS majority voting include:

- **Persistent:** A malicious node attacks recklessly. When serving as a voter during IDS majority voting, it will always vote “no” to evict a good node (to cause a false positive), and “yes” to retain a bad node (to cause a false negative).
- **Random:** The attack behavior is the same as a persistent attacker except that a malicious node only attacks randomly with probability  $p_a$  (0 to 1) to avoid detection.
- **Opportunistic:** The attack behavior is the same as a persistent attacker except that a malicious node only attacks opportunistically. That is, when serving as a voter, a malicious node will vote to evict a good node, or to retain a bad node, only if there is a majority of bad nodes among  $m$  nodes being selected to perform majority voting.
- **Selective:** The attacker uses the strategies of Random and Opportunistic attacks, but selectively performs actions based on the target node under IDS evaluation. When there is a majority of bad voters, a bad voter will vote against a good target confidently (without concern of being detected by the IDS). When the bad voters are less than a majority, they perform voting attacks in a random fashion, where they vote against lower capability IoT nodes with a lower probability than higher capability nodes. Thus attack with  $P_a$  (where  $P_a = P_{alc} + P_{ahc}$ ) if bad nodes are less than a majority, where the malicious voter attacks low capability and high capability with  $P_{alc}$  and  $P_{ahc}$  respectively, where  $P_{alc} \ll P_{ahc}$ . The strategy of malicious voters here is to prioritize evicting high capability IoT nodes over lower capability ones while keeping IDS suspicion low. Thus, malicious nodes may vote for a good low capability target, not in the hopes of the target remaining in the system, but in order to maintain a random attack behavior that evades the IDS, thus maximizing the probability of the malicious voter remaining in the system. The malicious voter does this

with the aim of remaining in the system for subsequent IDS rounds to vote against a high capability target and maximize the damage inflicted on the system. It also does this since low nodes are of a lower benefit to the system and are a lower threat to malicious nodes (as they typically have a higher host-level false positive probability and a higher false negative probability) in comparison to high capability nodes.

Fig. 3 (a) illustrates a scenario of how malicious nodes have decided to disregard an opportunity to attack, at time  $t$ , to maximize evading detection. On the other hand, in Fig. 3 (b), the same malicious nodes, at time  $> t$ , have decided to attack the high capability target. In this scenario in Fig. 3 (b), both malicious nodes have voted to evict the good high capability target by voting “no”, and a good node has misidentified the target as malicious based on its host IDS and voted “no”, thus resulting in a majority calling for evicting the high capability target node.



**FIGURE 3. Selective attack during IDS voting by colluding nodes j and k: (a) Intentionally disregarding an opportunity to attack a low capability target node at time  $t$  and evading detection by IDS, (b) Colluding to attack another high capability target node at a later time  $> t$ .**

The attacker’s benefit of evicting high capability nodes is twofold; first high capability nodes have the capability to accomplish more tasks and fulfill the application requirements such that evicting them increases the probability of application failure to the system. Second, high capability nodes have better capability to cast accurate votes when participating in the IDS (they typically have a lower host-

level false positive probability and a lower false negative probability) thus detecting malicious voters with a higher probability than lower capability nodes. This strategy of mischievously voting for weaker opponents in order to be able to cause failure to stronger opponents later on, exhibits similarities to tactical voting (or strategic voting) strategies used in political elections where the voter may, at the time of voting, vote for its less preferable choice with the aim of getting an overall better gain later on by influencing the overall outcome [33, 34]. From the attacker's perspective, this means more damage or higher failure probability to the system.

### C. DEFENSE STRATEGIES

We list the defense strategies used by all good nodes (as dictated by the defense system) during IDS majority voting below. The defense strength can be controlled by adjusting the following two parameters:

- The number of voters ( $m$ ) selected from a target node's location for executing IDS majority voting. Higher  $m$  means higher detection strength.
- The intrusion detection interval ( $T_{IDS}$ ) to control the detection frequency at which IDS voting is performed. Smaller  $T_{IDS}$  means higher detection frequency.

## IV. MODELING AND ANALYSIS

In this section, we develop an analytical model to describe the IDS attack-defense dynamics and analyze the effect of attack/defense strategies executed by attackers/defenders on the security property and consequently the system lifetime. We also develop an iterative computational procedure to make it computationally feasible for a large ADIoT consisting of a large number of IoT devices (See Table II for the list of parameters used by the analytical model).

Our analytical model provides the following two pieces of information to facilitate modeling of attack/defense dynamics:

1. Location: we like to know the probability that node  $i$  is located in area  $l$  at time  $t$ , denoted by  $P_{i,l}^l(t)$ . By inspecting  $P_{i,l}^l(t)$  and  $P_{j,l}^l(t)$ , we will know if node  $i$  and node  $j$  are in the same location at time  $t$ .
2. Good/Bad/Evicted status: we like to know the probability that node  $i$  is good, bad, or evicted at time  $t$ , denoted by  $P_i^g(t)$ ,  $P_i^b(t)$  and  $P_i^e(t)$ , respectively, with  $P_i^g(t) + P_i^b(t) + P_i^e(t) = 1$ . By inspecting  $P_i^g(t)$ ,  $P_i^b(t)$  and  $P_i^e(t)$  for node  $i$ ,  $P_j^g(t)$ ,  $P_j^b(t)$  and  $P_j^e(t)$  for node  $j$ ,  $P_k^g(t)$ ,  $P_k^b(t)$  and  $P_k^e(t)$  for node  $k$ , and so forth, we know the attack/defense strength at time  $t$ . If a good target node is surrounded by many bad nodes, then there is a high probability that the good target node will be misidentified as a bad node (thus causing a false positive) and a bad target node will be misidentified as a good node (thus causing a false negative).

We use Stochastic Petri Net (SPN) modeling techniques to provide us the above two pieces of information. We utilize a tool called SPNP [1] to define and evaluate SPN node models describing node attack-defense behaviors and status, to measure the system security metrics for security analysis.

An SPN model [19] comprises 4 entities: (a) transitions to represent occurrences of events, (b) places to represent conditions or states; (c) arcs to connect transitions to places and specify the directions of transitions; and (d) tokens to represent jobs or nodes that can flow from input places into output places based on transitions, indicating changes of status.

Fig. 4 shows the SPN node model for node  $i$  for modeling the location and status of node  $i$  over time. It consists of a location subnet (top left) providing the location information of node  $i$  at time  $t$ , a timer/energy subnet (top right) providing the energy status of node  $i$ , and a compromise undetected/detected status subnet (bottom) keeping track of if node  $i$  has been compromised at time  $t$  and if the compromise has been detected. These subnets are described in more detail in the following subsections. Each node in the system is separately modeled by an SPN node model. Therefore, there will be many SPN node models in the system (i.e., one for each node), but each can be run and evaluated separately with our hierarchical modeling technique.

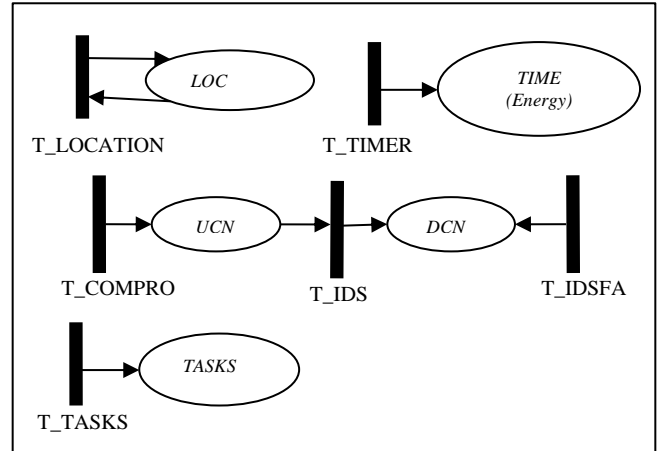


FIGURE 4. Node SPN Model.

### A. MODELING NODE STATUS

The location subnet (at the top left of Fig. 4) for node  $i$  provides us information about  $P_{i,l}^l(t)$ . The id of the current location of node  $i$  is indicated by the number of tokens in place  $LOC$ . The autonomous distributed IoT environment can be modeled as an  $M \times M$  location grid, with the unit length equal to the wireless radio range ( $R$ ) and each location is labeled with a unique location id. We allow each node to have its own mobility pattern specified by a sequence of time-ordered (location id, residence time) tuples, meaning that the IoT device stays at a location with the location id so indicated for this much time with the residence time so indicated. The mobility pattern can be generated by simulating the movement of a node following a mobility model such as the random movement model or the social SWIM mobility model [35]. The transition  $T\_LOCATION$  is triggered when node  $i$  moves from its current location to the next location with the transition rate calculated as  $1/RT$  where  $RT$  is the residence



time in the current location. Depending on the next location, the number of tokens in place LOC is adjusted to reflect the id of the location it resides under (after the movement is made), so by looking at the number of tokens in place LOC at time  $t$  we know the location of node  $i$  at time  $t$ .

The compromise undetected/detected status subnet (at the middle of Fig. 4) for node  $i$  gives us information about  $P_i^g(t)$ ,  $P_i^b(t)$  and  $P_i^e(t)$ . The status of node  $i$  is indicated by a token which flows from one place to another. Place UCN indicates that node  $i$  is compromised. A node is compromised when transition T\_COMPRO with rate  $\lambda_{com}$  fires where  $\lambda_{com}$  is the per-node capture rate. The transition T\_COMPRO is enabled if the node is not yet compromised or evicted. When node  $i$  is compromised, a token goes to UCN, meaning that node  $i$  is now a malicious node not yet detected by IDS, so it may perform persistent, random, or opportunistic attacks. Place DCN means that node  $i$  is evicted. An eviction can occur in two ways. The first way is that node  $i$  was compromised (i.e., the token was in place UCN) and is correctly identified by the system IDS, causing the token to flow from into DCN and node  $i$  to be evicted immediately. The transition rate of T\_IDS is  $(1 - P_{fn}^{IDS})/T_{IDS}$  where  $P_{fn}^{IDS}$  (derived in Equation 1 below) is the false negative probability of the system IDS and  $T_{IDS}$  is the IDS detection interval. The second way is that node  $i$  was a good node but is misidentified as a bad node by the system IDS, causing the token to be deposited in place DCN and node  $i$  to be evicted immediately. The transition rate of T\_IDSFA is  $P_{fp}^{IDS}/T_{IDS}$  where  $P_{fp}^{IDS}$  (derived in Equation 1 below) is the false positive probability of the system IDS.

The timer subnet (at the top right of Figure 4) keeps track of elapsed time in the node SPN model. After  $T_{IDS}$  is elapsed, T\_TIMER fires and a token is added to place TIME. T\_TIMER is disabled when the node is evicted (i.e., when a token is in place DCN). By looking at the number of tokens in place TIME, one can tell the current time. This information allows  $P_{fp}^{IDS}$  and  $P_{fn}^{IDS}$  to be updated in increments of  $T_{IDS}$  dynamically to reflect the effect of IDS attacker/defense dynamics on  $P_{fp}^{IDS}$  and  $P_{fn}^{IDS}$ . We also use the timer subnet as the energy subnet with each token deposited in place TIME

$$\begin{aligned}
P_{fp}^{IDS}(t, l) \text{ or } P_{fn}^{IDS}(t, l) = & \\
& \sum_{i=0}^{m-m_{maj}} \left[ \frac{C \binom{n_{bad}^a}{m_{maj} + i} \times C \binom{n_{good} + n_{bad}^i}{m - (m_{maj} + i)}}{C \binom{n_{bad}^a + n_{bad}^i + n_{good}}{m}} \right] \\
& + \sum_{i=0}^{m-m_{maj}} \left[ \frac{C \binom{n_{bad}^a}{i} \times \sum_{j=m_{maj}-i}^{m-i} \left[ C \binom{n_{good} + n_{bad}^i}{j} \times \omega^j \times C \binom{n_{good} + n_{bad}^i - j}{m - i - j} \times (1 - \omega)^{m-i-j} \right]}{C \binom{n_{bad}^a + n_{bad}^i + n_{good}}{m}} \right]
\end{aligned} \tag{1}$$

indicating the amount of energy spent by node  $i$  in an intrusion detection cycle. By knowing the number of IDS cycles elapsed (from place TIME) and the percentage of energy spent by node  $i$  per cycle for executing monitoring,

reporting, task execution, and performing IDS functions, denoted by  $P_e$ , we can estimate the remaining energy of node  $i$  at time  $t$ .

The task subnet for node  $i$  (at the bottom of Fig. 4) is to keep track of the tasks completed by node  $i$ . The transition T\_TASKS is triggered periodically with rate  $1/ET$  where  $ET$  is the execution time. Thus, in every  $ET$  interval, a unit of tasks (a TEU) is completed, and tokens representing this unit will be deposited into place TASKS. While a low capability node may deposit a small number of tokens, high capability nodes may have the capability to complete a larger amount of work within the  $ET$  interval, resulting in a larger number of tokens being deposited in the same execution time duration. We denote the task tokens deposited for low capability nodes and high capability nodes over the  $ET$  interval by  $tt_{lc}^{ET}$  and  $tt_{hc}^{ET}$  respectively. As a result, we expect that  $tt_{lc}^{ET} > tt_{hc}^{ET}$ . A node may execute tasks of different types including monitoring, actuating, or computation, each represented by a different number of tokens and executed as requested by the system. In this work, for simplicity, we do not differentiate between different task types and consider that nodes with similar capability will execute the same number of tasks within the same time duration (i.e.,  $ET$ ).

## B. MODELING ATTACKER/DEFENDER STRATEGIES

An attacker can perform persistent, random, or opportunistic attacks while participating in the majority voting IDS function. The attack strategy chosen affects the system IDS security measured by the false negative probability ( $P_{fn}^{IDS}$ ) and the false positive probability ( $P_{fp}^{IDS}$ ).

We derive the false positive probability ( $P_{fp}^{IDS}(t, l)$ ) and false negative probability ( $P_{fn}^{IDS}(t, l)$ ) for diagnosing a target node at location  $l$  and time  $t$  surrounded by  $n_{good}(t, l)$  good nodes and  $n_{bad}(t, l)$  bad nodes. Henceforth, the notation  $(t, l)$  at the end of a symbol is omitted for brevity.

Equation 1 gives a closed-form solution for  $P_{fp}^{IDS}$  and  $P_{fn}^{IDS}$  under random attack behavior where  $C \binom{a}{b}$  is the # of combinations to select  $a$  from  $b$ ,  $n_{bad}^a$  and  $n_{bad}^i$  are the numbers of ‘‘active’’ and ‘‘inactive’’ bad nodes, given by  $n_{bad} \times p_a$  and  $n_{bad} \times (1 - p_a)$ , respectively;  $m_{maj}$  is the minimum majority of  $m$ , e.g., 3 is the minimum majority of

5; and  $\omega$  is  $H_{pfp}$  for calculating  $P_{fp}^{IDS}$  and  $H_{pfn}$  for calculating  $P_{fn}^{IDS}$ . Here  $H_{pfp}$  and  $H_{pfn}$  are the host-level false positive probability and false negative probability,

respectively, as a result of each node executing host-level IDS duties monitoring behavior or misbehaving of a neighbor node as described earlier. They are given as input at the system start-up time.

Here we note that persistent attack is a special case of random attack with  $p_a = 1$ . Equation 1 can also be used to model opportunistic attack behavior such that  $p_a = 1$  when during IDS voting, more than one half of the nodes selected for IDS voting are bad nodes, thus resulting in  $P_{fp}^{IDS} = 1$  and  $P_{fn}^{IDS} = 1$ . If more than one half of the nodes selected for IDS voting are good nodes, an opportunistic attacker would simply fall back to random attack behavior because there is still a chance good nodes can still vote to evict a good target node (with probability  $H_{pfp}$ ), or retain a bad target node (with probability  $H_{pfn}$ ).

Under selective attack, attackers selectively prioritize high capability nodes during an attack. When bad nodes are a majority, the attackers always vote against good target nodes and vote for bad target nodes as in opportunistic attack, irrelevant of its capability. If bad nodes are less than a majority, the attackers only attack with probability  $p_a$  randomly. However, they give priority to selectively attack high capability nodes over lower capability nodes in order to achieve an application failure. In effect, attackers collude to evict the same expected number of target nodes under random attack with probability  $p_a$ .

### C. COMPUTATIONAL PROCEDURE

The underlying model of a node SPN model as shown in Fig. 4 is a continuous-time semi-Markov process with 5 state components, LOC, TIME, UCN, DCN, and TASKS describing the behavior of a node as time progresses.

One could put all node SPN models into one big SPN model and run it in SPNP [1] to yield the system mean time to failure (MTTF) as the security metric. However, the computational complexity is  $O(c^n)$  where  $c = 5$  is the number of state components (LOC, TIME, UCN, DCN, TASKS) and  $n$  is the number of nodes in the ADIoTS. It is computationally infeasible for a large  $n$  because of the state explosion problem as the underlying Markov model needs to consider the number of nodes in the system, the components for each node, and the states per component.

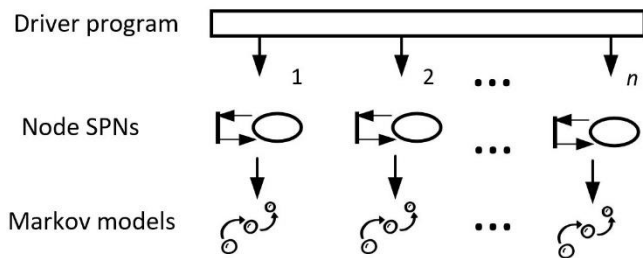


FIGURE 5. Flow of SPN Model Execution.

We develop an iterative computational procedure with linear complexity of  $O(n)$  to make it computationally feasible for a large ADIoTS. As illustrated in Fig. 5, the driver program will invoke SPNP [1] to run and evaluate the node SPN model  $n$  times, one for each distinct node, and then

integrate their outputs together to yield the system lifetime as output. Since SPNP is invoked only  $n$  times, the complexity is  $O(n)$  where  $n$  is the number of nodes in the ADIoTS.

The basic idea of our iterative computational procedure is to update the false positive probability  $P_{fp}^{IDS}(t)$  and false negative probability  $P_{fn}^{IDS}(t)$  iteratively until convergence, as follows:

The driver runs each node SPN model for node  $i$  to completion using SPNP [1] until node  $i$  is in an absorbing state, i.e., until node  $i$  is evicted (i.e., a token is in place DCN) or until energy is exhausted (i.e., maximum tokens are in place TIME). Initially we set  $P_{fp}^{IDS}(t)$  and  $P_{fn}^{IDS}(t)$  to 5% in the first iteration. We then reset them to the new values computed in step 3 in subsequent iterations.

For each node SPN model for node  $i$ , generate the output  $P_{i,l}^L(t)$ ,  $P_i^g(t)$ ,  $P_i^b(t)$ , and  $P_i^e(t)$  in increment of  $T_{IDS}$ .

Based on node status probabilities reported by all nodes (in previous step 2), compute the false positive probability  $P_{fp}^{IDS}(t)$  and false negative probability  $P_{fn}^{IDS}(t)$  for node  $i$  (in increment of  $T_{IDS}$ ). The time  $t$  at which the computation is performed can be looked up by inspecting the number of tokens in place TIME. Specifically,

$$P_{fp}^{IDS}(t) = \sum_i P_{i,l}^L(t) P_{fp}^{IDS}(t, l) \quad (2)$$

where  $P_{fp}^{IDS}(t, l)$  is computed based on Equation 1 with  $n_{bad}(t, l) = \sum_{k \neq i} P_{k,l}^L(t) P_k^b(t)$  and  $n_{good}(t, l) = \sum_{k \neq i} P_{k,l}^L(t) P_k^g(t)$ .

Check if the Mean Percentage Difference (MPD) of an important parameter  $X_i(t)$  of node  $i$  (such as  $P_{fn}^{IDS}(t)$ ) in iteration  $j$  and iteration  $j+1$  is less than the minimum threshold (set at 1%), i.e.,  $|X_i^{j+1}(t) - X_i^j(t)| / X_i^j(t) < 1\%$ . If no, go to step 1 to continue the iterative computational process. If yes, compute the MTTF of the system based on the failure conditions and exit. For attrition failure, MTTF can be identified by first sorting the mean time to bad/evicted status for all nodes and then the first time at which the number of good nodes falls below the system allowable minimum threshold ( $n_{good}^{TH}$ ) is the MTTF. For Byzantine failure, the first time at which the number of bad nodes is equal to or greater than 1/3 of the total number of good and bad nodes is the MTTF. For energy depletion failure, the first time at which the number of nodes with adequate energy falls below a threshold ( $E^{TH}$ ) is the MTTF. A nodes energy resource is indicated by the number of tokens in place TIME in the timer subnet and when it reaches a maximum allowable it indicates that the IoT device is too depleted. For application failure, the first time when the task completion rate of nodes (computed by dividing the number of tasks completed as indicated by the number of tokens in place TASKS in the task subnet, by the current time as indicated by the number of tokens in place TIME in the timer subnet) collectively falls below the system allowable minimum threshold ( $tc^{TH}$ ).

We list the computational procedure below:

#### 1: Driver Execution:

2: **while** not all nodes converged  
3: increment iteration counter  
4: run SPN model for each node in the iteration (line 10)  
5: **if** all nodes converge **then**  
6: find MTF (line 19) and exit  
7: **else**  
8: continue next iteration; pass saved values of this iteration to the next iteration; go to line 3  
9:  
10: **Running node SPN model (current iteration):**  
11: **while** not reach maximum IDS executions  
12: Find time  $t$  for the current IDS execution  
13: Retrieve previous iteration's expected values at time  $t$  (unless first iteration where it equals 1)  
14: Use retrieved nodes values to find  $P_{fp}^{IDS} / P_{fn}^{IDS}$  for the current iteration (use 5% if it equals 1)  
15: Calculate new expected values and store for the next iteration  
16: **If** convergence condition met **then**  
17: mark this node as converged  
18:  
19: **Find MTF:**  
20: **while** no failure condition **AND** not reach maximum IDS executions  
21: Find time  $t$  for the current ids execution  
22: find system  $P_{fp}^{IDS} / P_{fn}^{IDS}$  for all nodes  
23: execute IDS for all nodes  
24: **if** attrition **or** Byzantine **or** resource **or** application failure **then**  
25: no failure = false  
26:  $MTTF = t$

## V. APPLYING OPTIMAL DEFENSE SETTINGS FOR LIFETIME MAXIMIZATION

Our analytical results identify optimal defense settings in terms of the best  $(T_{IDS}, m)$  combination under which the ADIoTS lifetime is maximized. This includes best defense settings for sophisticated collusion-based attacks by inside attackers such as Random, Opportunistic, and Selective attacks. To apply the findings in this paper, the mission commander can apply the best defense settings in terms of  $(T_{IDS}, m)$  dynamically based on the current ADIoTS operational and environmental conditions sensed at runtime to maximize the ADIoTS lifetime. This is depicted in Fig. 6 where optimal defense settings are generated offline and stored in the form of a lookup table based on the analytical results obtained in the paper (top half of Fig. 6). When new ADIoTS operational and environmental conditions are sensed, a search is performed based on closest match or extrapolation techniques to find the best defense settings of  $(T_{IDS}, m)$  to apply so as to maximize the system lifetime (lower half of Fig. 6).

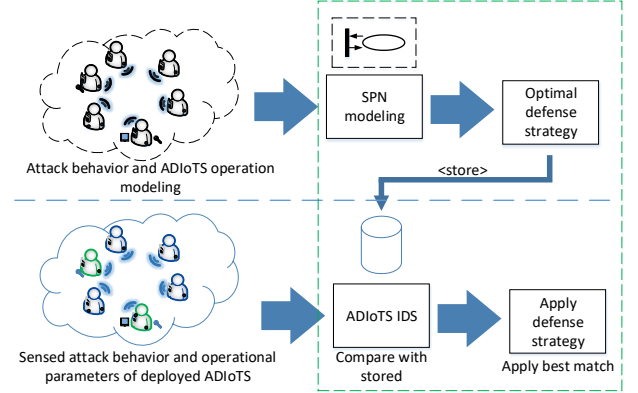


FIGURE 6. Flow of Determining Optimal Defense Settings for Lifetime Maximization.

## VI. EVALUATION

In this section, we use the stochastic Petri net package (SPNP) [1] to define and analytically solve the SPN model developed to yield the system lifetime as output, when given a set of parameter values characterizing the operational and environmental conditions as listed in Table IV as input. All parameters except the number of voters ( $m$ ) and the IDS detection interval ( $T_{IDS}$ ) have their values derived from an ADIoTS described in [36] comprising 128 sensor-carrying mobile nodes. The number of voters ( $m$ ) and the IDS detection interval ( $T_{IDS}$ ) are design parameters whose values are to be identified and applied at runtime to maximize the system lifetime.

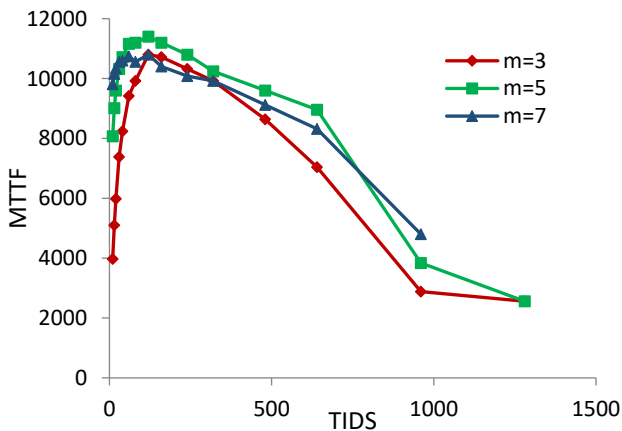
TABLE IV  
PARAMETERS FOR AN ADIoTS

Symbol	Meaning	Value
$n$	Number of nodes	128
$n_{good}^{TH}$	Minimum threshold for attrition failure	32,51
$E^{TH}$	Resource depletion threshold	5%
$t_C^{TH}$	Task execution rate threshold	25%
$n_{hc}$	Percentage of high capability nodes	30%
$H_{pfn}$	Host IDS false negative probability	[2.5,5,7.5]%
$H_{pfp}$	Host IDS false positive probability	[2.5,5,7.5]%
$\lambda_{com}$	Per-node capture rate	1/1800-5400
$m$	Number of voters per IDS voting	3,5,7
$T_{IDS}$	IDS interval	0-1400
$P_e$	Percentage of energy spent per $T_{IDS}$	0.01%
$P_a$	Random attack probability	[0, 1]
$MxM$	Operation area	64x64 m <sup>2</sup>
$R$	Radio range	100 m

The 128 sensor-carrying mobile IoT devices are randomly deployed in a 64x64 m<sup>2</sup> operational area, each following the SWIM mobility model [35] after deployment. The radio range is 100 m for peer-to-peer communication for the 128 nodes. When there are fewer than 32 devices in the system, the system is not able to perform its intended function, leading to an attrition failure. At the host level, each device monitors its immediate neighbors with a false negative probability

$H_{pfn}$  ranging in 2.5%-7.5% and a false positive probability  $H_{pfp}$  ranging in 2.5%-7.5%. Such values are assumed to be known before each device is released to operation by software engineering testing. IoT devices are compromised due to capture attacks by which a good device that is being captured is converted into a bad device. The per-node capture rate  $\lambda_{com}$  ranges from 1/5400 to 1/1800, meaning that on average after 1800-5400 (seconds, minutes, hours, or days depending on the system under consideration) is elapsed, a node would likely be captured and turned into malicious. Assume that the amount of energy consumed for each IoT device in an IDS period is 0.01%. The security metric is the system MTTF which is measured when the system fails due to Byzantine, attrition, application, or energy depletion failure.

Fig. 7 shows the system MTTF (s) vs  $T_{IDS}$  (s) for the ADIoTS in the case in which the attack strategy is persistent attack ( $P_a = 1$ ) to quickly fail the system. The defense strategies considered are the number of voters ( $m$ ) in majority voting IDS and the IDS detection interval ( $T_{IDS}$ ). With the persistent attack strategy in place, an attacker always performs ballot-stuffing (saying a bad node is a good node) and bad-mouthing attacks (saying a good node is a bad node) whenever it has a chance, to cause Byzantine and attrition failures at the fastest pace. Under this attacker strategy, there exists an optimal  $T_{IDS}$  under which the system lifetime is maximized. This is due to the following reasons: When  $T_{IDS}$  is too low, the frequency of performing intrusion detection is high, thus causing energy depletion failures to happen early on. When  $T_{IDS}$  is too high, it does not perform intrusion detection often enough to detect and remove bad nodes from the system. As a result, many bad nodes remain undetected in the system. This also results in a short lifetime, due to both Byzantine failure (when at least one third of the nodes are bad nodes) and attrition failure (when the number of good nodes falls below  $n_{good}^{TH}$ ).

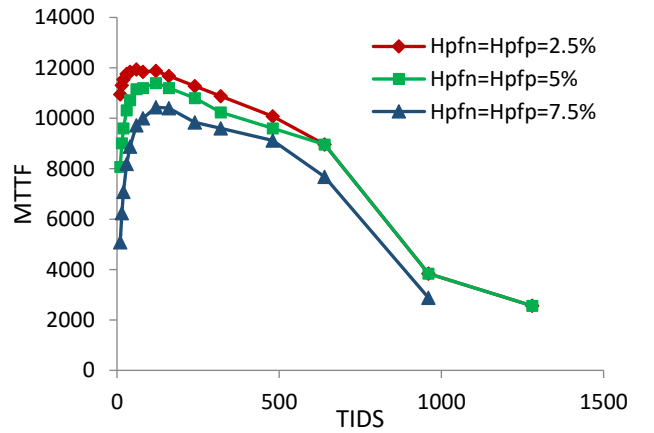


**FIGURE 7.** Optimal defense settings of ( $T_{IDS}$ ,  $m$ ) for maximizing MTTF of an ADIoTS as defined by Table IV, with  $\lambda_{com}=1/3600$  and  $H_{pfn}=H_{pfp}=5\%$ .

The effect of the number of voters ( $m$ ) is clearly demonstrated in Fig. 7. We observe that the optimal  $T_{IDS}$  depends on  $m$  and  $m = 5$  is the best choice of this defense strategy for maximizing the system lifetime for the example ADIoTS. The reason is that when  $m$  is high, it tends to deplete

energy early on thus causing resource depletion failure. When  $m$  is low, it tends to leave too many bad nodes undetected in the system, thus causing Byzantine or attrition failure. Consequently,  $m = 5$  can best balance resource depletion failure versus Byzantine or attrition failure to maximize the system lifetime. The most striking observation is that an optimal defense strategy exists in terms of the best ( $T_{IDS}$ ,  $m$ ) combination that will maximize the system MTTF, when the attack strategy is persistent attack ( $P_a = 1$ ).

The effect of per-host defense capability in terms of intrusion detection accuracy, represented by the host IDS false negative probability  $H_{pfn}$  and the host false positive probability  $H_{pfp}$ , on the system lifetime is demonstrated in Fig. 8. We first observe that the system lifetime is higher when the system has better defense capability, i.e., when  $H_{pfn}$  and  $H_{pfp}$  are lower.



**FIGURE 8.** Effect of defense capability in terms of ( $H_{pfn}$ ,  $H_{pfp}$ ) on MTTF of an ADIoTS as defined by Table IV, with  $m = 5$  and  $\lambda_{com}=1/3600$ .

We also observe that the optimal  $T_{IDS}$  at which the system MTTF is maximized strongly depends on the defense capability. That is, the optimal  $T_{IDS}$  that maximizes MTTF increases as  $H_{pfn}$  and  $H_{pfp}$  increase. The reason is that when the defense capability becomes weaker (meaning  $H_{pfn}$  and  $H_{pfp}$  have higher values at 7.5% in Fig. 8), many malicious nodes may be undetected and remained the system while many good nodes may be misidentified as malicious and evicted from the system, thus resulting in Byzantine or attrition failures. This happens more often when the detection interval is smaller. Consequently, when  $H_{pfn}$  and  $H_{pfp}$  are high, the system is better off using a large optimal  $T_{IDS}$  value. Fig. 8 demonstrates this trend, i.e., when  $H_{pfn}$  and  $H_{pfp}$  are higher at 7.5% the optimal  $T_{IDS}$  is 120 while when  $H_{pfn}$  and  $H_{pfp}$  are lower at 2.5% the optimal  $T_{IDS}$  is 60. The results reveal that the per-node defense capability affects not only the system lifetime but also the optimal detection interval  $T_{IDS}$  (a defense strategy) under which the system lifetime is maximized.

The effect of attacker capability in terms of per-node compromise rate  $\lambda_{com}$  on the system lifetime is demonstrated in Fig. 9. We first observe that the system lifetime is lower when the attacker capability is high, i.e., when  $\lambda_{com}$  is higher. We also observe that the optimal  $T_{IDS}$  at which the system



MTTF is maximized strongly depends on the attacker capability. That is, the optimal  $T_{IDS}$  that maximizes MTTF decreases as  $\lambda_{com}$  increases. The reason is that when the attacker capability is higher (meaning  $\lambda_{com}$  is higher at 1/1800 in Fig. 9), many good nodes may be compromised and turned into malicious in which case the system is better off by running intrusion detection more often by making  $T_{IDS}$  smaller to catch and evict malicious nodes from the system to prevent Byzantine failure from occurring. Fig. 9 demonstrates this trend, i.e., when  $\lambda_{com}$  is higher at 1/1800 the optimal  $T_{IDS}$  is 40 while when  $\lambda_{com}$  is lower at 1/5400 the optimal  $T_{IDS}$  is 160. The results reveal that the attacker capability also affects the optimal detection interval  $T_{IDS}$  (a defense strategy) under which the system lifetime is maximized.

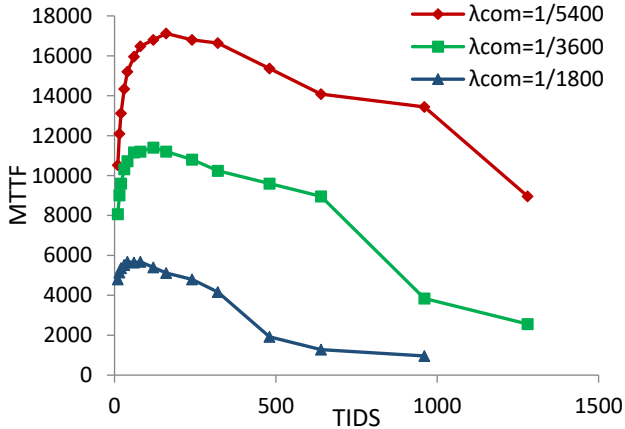


FIGURE 9. Effect of attack capability in terms of  $\lambda_{com}$  on MTTF of an ADIoT<sub>S</sub> as defined by Table IV, with  $m = 5$  and  $H_{pfn} = H_{pfp} = 5\%$ .

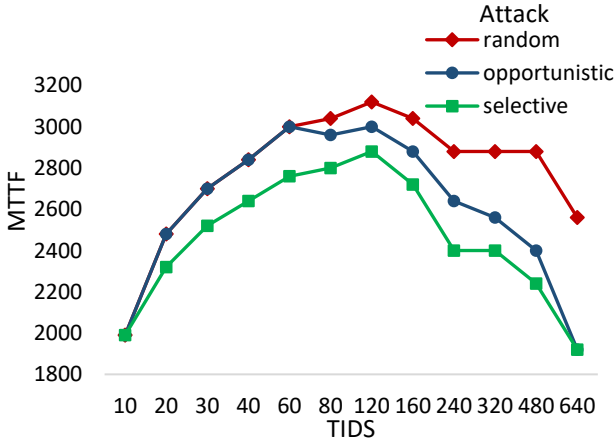


FIGURE 10. Effect of attack strategy on system lifetime under varying  $T_{IDS}$ .

Unlike defense capability, attacker capability is not a choice of the defense system. However, when learning the attacker capability is strong (e.g., from experiences), the results suggest that the system should shorten the detection interval to maximize the system lifetime. The optimal detection interval  $T_{IDS}$  of course depends on the operational setting represented by the set of parameters defined in Table IV. Given the operational setting, the methodology proposed

in the paper helps identify the optimal  $(T_{IDS}, m)$  for maximizing the system lifetime.

The security analysis thus far considers a homogenous system where all nodes are of similar capability. To illustrate the effects of selective attacks and application failures, we consider below a heterogeneous ADIoT system consisting of both high and low capability nodes, as discussed in Section III. Fig. 10 shows the effect of attack strategy on system failure conditions, under varying  $T_{IDS}$  values. For clarity, we list the system failure condition triggered for Fig. 10 results separately in Table V. We show the effect of random, opportunistic, and selective attacks on system failures. As a persistent attack is a special case of a random attack with  $P_a = 1$ , we omit persistent attack for brevity. We consider the system failure types as discussed in Section III.A, namely, Byzantine, resource depletion, attrition, and application failures. We consider that of the deployed nodes 30% are of high capability (i.e.,  $n_{hc} = 30\%$ ), where they execute 4 TEUs (Task Execution Units) as opposed to 1 TEU by lower capability nodes, hence contributing more towards task completion.

First, from Fig. 10 we again observe that there exists an optimal  $T_{IDS}$  that maximizes the system lifetime in response to various attack strategies. We observed this for persistent attacks earlier in Figures 7-9. Now we also observe it for random, opportunistic, and selective attacks.

Second, we find that in all attack strategies, using a very high intrusion detection frequency (small  $T_{IDS}$ ) results in rapid node energy consumption causing a resource depletion failure before other failure conditions can occur (e.g., when  $T_{IDS} = 10$ , all failures under all attack strategies are due to resource depletion). Conversely, using a very low intrusion detection frequency (high  $T_{IDS}$ ) results in a Byzantine failure occurring first, as IDS bad node eviction cannot cope with the compromise rate thus resulting in bad nodes  $> 1/3$  good nodes (e.g., when  $T_{IDS} = 640$ ). This is further illustrated in Fig. 11 (for the opportunistic attack case of Fig. 10), where the system good and bad node populations are shown as a function of time, as a result of node compromise and IDS execution (we do not show the evicted node population in Fig. 11 for brevity). Thus we observe that  $T_{IDS}$  greatly effects the system failure conditions (i.e., which system failure occurs first).

Third, we observe that the opportunistic attack results in lower system lifetime than random attack, since the opportunistic attack, in addition to attacking randomly, takes advantage of IDS voting occurrences where bad nodes form a majority in which case it always votes against good nodes and votes for bad nodes. Similarly, we observe that selective attack, in addition to attacking opportunistically, especially targets high capability nodes that are critical in meeting task execution rate. Thus, under the selective attack strategy, high capability nodes are chosen by colluding attackers as main targets. This has the effect of resulting in application failures (last column of Table V). Also, the colluding attackers still use the strategies of random and opportunistic attacks to result in Byzantine failures. As a result, the selective attack is the most effective attack strategy among all to minimize MTTF. However, as we observe from Fig. 10, the system designer can optimally adjust the  $T_{IDS}$  value to obtain the best



achievable MTTF (along with the best selection of  $m$  value although it is not shown in Fig. 10) against the selective attack strategy.

TABLE V  
FIRST FAILURE OCCURRENCE TYPE FOR VARYING ATTACKS AND UNDER VARYING  $T_{IDS}$

	Attack type		
	Random	Opportunistic	Selective
$T_{IDS}$ 10	Res. Dep.	Res. Dep.	Res. Dep.
40	Attrition	Attrition	Application
80	Attrition	Attrition	Application
240	Attrition	Attrition	Application
640	Byzantine	Byzantine	Byzantine

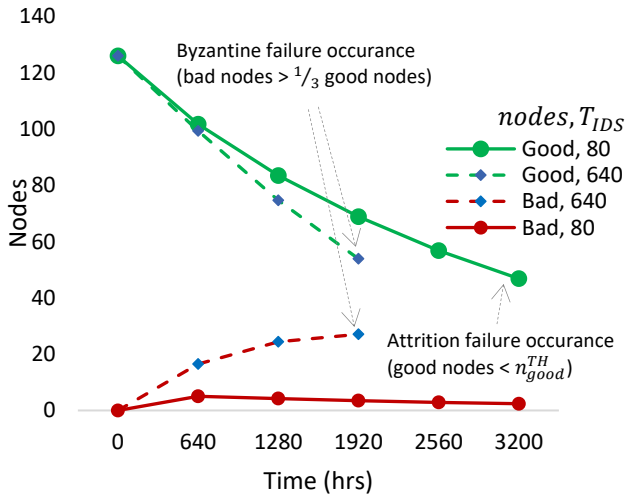


FIGURE 11. An illustration showing the occurrence of Byzantine and attrition failures under opportunistic attack for the two cases of  $T_{IDS}$  being 640 and 80, respectively.

Fig. 12 and Fig. 13 respectively compare two baseline IDS schemes against our proposed CAVBIDS scheme.

For the first baseline comparison, Fig. 12 shows the performance comparison of our proposed CAVBIDS scheme with a baseline IDS scheme that uses a fixed or static detection interval without changing the defense strength in terms of the detection interval length in response to attacker strength (i.e., compromise rate  $\lambda_{com}$ ). We observe that CAVBIDS outperforms the first baseline scheme using a large detection interval (i.e.  $T_{IDS} = 320$ ) as the attacker strength varies from high (e.g., compromise interval  $1/\lambda_{com} = 40$ ) to low (e.g., compromise interval  $1/\lambda_{com} = 100$ ). The first baseline scheme performs comparably with CAVBIDS only when the attacker compromise rate is low (e.g., compromise interval  $1/\lambda_{com} = 100$ ) at which point CAVBIDS also selects  $T_{IDS} = 320$  as the optimal defense strength.

For the second baseline comparison, in Fig. 13 shows the performance comparison of our CAVBIDS scheme with a baseline IDS scheme that uses a fixed number of host IDS voters for the ADIoT target voting, without changing the defense strength in terms of the number of voters in response to attacker strength (i.e., compromise rate  $\lambda_{com}$ ). We again

observe that CAVBIDS outperforms the second baseline scheme using a small number of voters (i.e.  $m = 3$ ) as the attacker strength varies from high (e.g., compromise interval  $1/\lambda_{com} = 40$ ) to low (e.g., compromise interval  $1/\lambda_{com} = 100$ ). The second baseline scheme performs comparably with CAVBIDS only when the attacker compromise rate is low (e.g., compromise interval  $1/\lambda_{com} = 90 - 100$ ) at which point CAVBIDS also selects  $m = 3$  as the optimal defense strength.

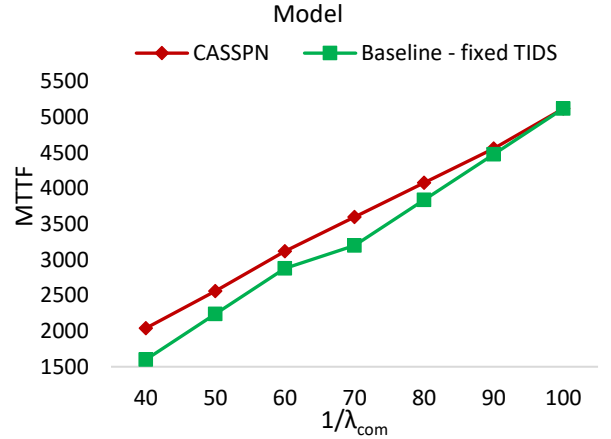


FIGURE 12. Comparing MTTF in a baseline IDS scheme where the detection interval is fixed ( $T_{IDS} = 320$ ) with our CAVBIDS scheme where  $T_{IDS}$  is adjusted based on expected compromise interval ( $1/\lambda_{com}$ ).

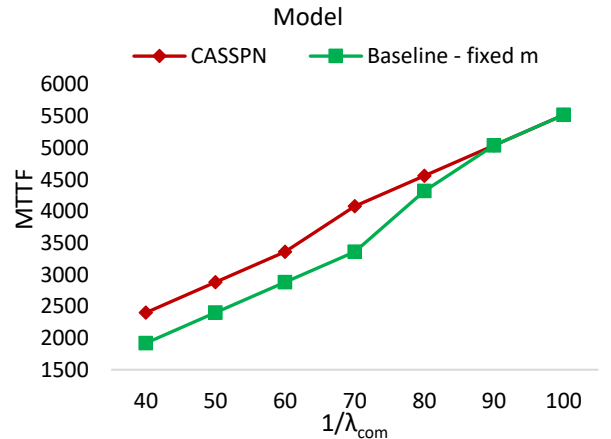


FIGURE 13. Comparing MTTF in a baseline IDS scheme where the number of Host IDS voters is fixed ( $m = 3$ ) with our CAVBIDS scheme where  $m$  is adjusted based on expected compromise interval ( $1/\lambda_{com}$ ).

All above results obtained in this section are based on analytical evaluation. That is, given a set of parameter values characterizing the operational and environmental conditions of the 128-node ADIoT as described in [36], we apply SPNP to run the 128 node SPN models, integrate the results from 128 outputs, and through assigning rewards with states of the system, identify the best defense settings of  $m$  and  $T_{IDS}$  under which the system lifetime is maximized. The obtained results can be further validated by building a testbed for the 128-node ADIoT to generate empirical results to match against the analytical results obtained in this paper. The practical implications of the obtained results are as follows: Our analytical results identify optimal defense settings in terms of

the best  $(T_{IDS}, m)$  combination under which the ADIoTS lifetime is maximized. This includes best defense settings for sophisticated collusion-based attacks by inside attackers such as Random, Opportunistic, and Selective attacks. To apply the findings in this paper, the mission commander can apply the best defense settings in terms of  $(T_{IDS}, m)$  dynamically based on the current ADIoTS operational and environmental conditions sensed at runtime to maximize the ADIoTS lifetime. This is depicted in Fig. 6 where optimal defense settings are generated offline and stored in the form of a lookup table based on the analytical results obtained in the paper (top half of Fig. 6). When new ADIoTS operational and environmental conditions are sensed, a search is performed based on closest match or extrapolation techniques to find the best defense settings of  $(T_{IDS}, m)$  to apply so as to maximize the system lifetime (lower half of Fig. 6).

## VII. CONCLUSION

In this work, we developed IDS duties that must be executed by every node of an autonomous distributed IoT system (ADIoTS) with the objective of maximizing the system MTTF. We developed SPN-based behavior models as well as a scalable iterative computational procedure with linear complexity in the number of nodes, allowing IDS attack/defense strategies for executing voting-based IDS functions to be specified and analyzed. We demonstrated the applicability with a selected set of attack-defense strategies and identified optimal defense settings in terms of the best  $(T_{IDS}, m)$  combination under which the ADIoTS lifetime is maximized. We also demonstrated that the per-node defense capability and the per-node attacker capability will affect not only the system lifetime but also the optimal detection interval  $T_{IDS}$  (a defense strategy) under which the system lifetime is maximized. We also analyzed the effect of attack strategies on system failure conditions and system lifetime, identified the most damaging attack strategy among all, and suggested defense strategies in terms of  $(T_{IDS}, m)$  for maximizing the system MTTF. In the future, we plan to extend this work to consider additional sophisticated collusion and strategic attacks, new IDS defense strategies, and more SPN-based modeling and complexity analysis for IoT system components. We plan to implement a testbed for the ADIoTS comprising 128 sensor-carrying mobile nodes as described in [36] using Raspberry Pi deployed nodes, each having a host IDS with lightweight detection techniques. By matching the analytical results obtained in the paper against the empirical results obtained from the testbed, we can validate the effectiveness of our collusion-aware voting-based IDS design proposed in this paper.

## REFERENCES

- [1] G. Ciardo, J. Muppala, and K. Trivedi, "SPNP: stochastic Petri net package," in *Proceedings of the Third International Workshop on Petri Nets and Performance Models, PNPM89*, 1989, pp. 142-151: IEEE.
- [2] E. Benkhalifa, T. Welsh, and W. Hamouda, "A critical review of practices and challenges in intrusion detection systems for IoT: Toward universal and resilient systems," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3496-3509, 2018.
- [3] C. Wu *et al.*, "A Hybrid Intrusion Detection System for IoT Applications with Constrained Resources," *International Journal of Digital Crime and Forensics (IJDCF)*, vol. 12, no. 1, pp. 109-130, 2020.
- [4] A. Sforzin, F. G. Mármol, M. Conti, and J. M. Bohli, "RPiDS: Raspberry Pi IDS—A fruitful intrusion detection system for IoT," in *Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress*, 2016, pp. 440-448: IEEE.
- [5] Y. N. Soe, Y. Feng, P. I. Santosa, R. Hartanto, and K. Sakurai, "Towards a Lightweight Detection System for Cyber Attacks in the IoT Environment Using Corresponding Features," *Electronics*, vol. 9, no. 1, p. 144, 2020.
- [6] M. Nobakht, V. Sivaraman, and R. Boreli, "A host-based intrusion detection and mitigation framework for smart home IoT using OpenFlow," in *11th International conference on availability, reliability and security (ARES)*, 2016, pp. 147-156: IEEE.
- [7] N. McKeown *et al.*, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69-74, 2008.
- [8] I. You, K. Yim, V. Sharma, G. Choudhary, R. Chen, and J.-H. Cho, "On IoT Misbehavior Detection in Cyber Physical Systems," in *2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC)*, 2018, pp. 189-190: IEEE.
- [9] B. Alotaibi and K. Elleithy, "A majority voting technique for wireless intrusion detection systems," in *IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, 2016, pp. 1-6: IEEE.
- [10] E. Anthi, L. Williams, and P. Burnap, "Pulse: an adaptive intrusion detection for the internet of things," in *IET Conference Proceedings*, 2018, pp. 35 (4 pp.)-35 (4 pp.): Institution of Engineering and Technology.
- [11] A. Amouri, V. T. Alaparthy, and S. D. Morgera, "A Machine Learning Based Intrusion Detection System for Mobile Internet of Things," *Sensors*, vol. 20, no. 2, p. 461, 2020.
- [12] M. Islabudeen and M. K. Devi, "A Smart Approach for Intrusion Detection and Prevention System in Mobile Ad Hoc Networks Against Security Attacks," *Wireless Personal Communications*, pp. 1-32, 2020.
- [13] Y. A. Qadri, R. Ali, A. Musaddiq, F. Al-Turjman, D. W. Kim, and S. W. Kim, "The limitations in the state-of-the-art counter-measures against the security threats in H-IoT," *Cluster Computing*, pp. 1-19, 2020.
- [14] N. K. Thanigaivelan, E. Nigussie, R. K. Kanth, S. Virtanen, and J. Isoaho, "Distributed internal anomaly detection system for Internet-of-Things," in *13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 2016, pp. 319-320: IEEE.
- [15] R. Mitchell and I. R. Chen, "Behavior rule specification-based intrusion detection for safety critical medical cyber physical systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 1, pp. 16-30, 2015.
- [16] H. Al-Hamadi and I. R. Chen, "Adaptive network defense management for countering smart attack and selective capture in wireless sensor networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 3, pp. 451-466, 2015.
- [17] A. Saeed, A. Ahmadinia, A. Javed, and H. Larjani, "Intelligent intrusion detection in low-power IoTs," *ACM Transactions on Internet Technology (TOIT)*, vol. 16, no. 4, pp. 1-25, 2016.
- [18] Z. A. Khan and P. Herrmann, "A trust based distributed intrusion detection mechanism for internet of things," in *IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, 2017, pp. 1169-1176: IEEE.
- [19] M. Ajmone Marsan, G. Conte, and G. Balbo, "A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems," *ACM Transactions on Computer Systems (TOCS)*, vol. 2, no. 2, pp. 93-122, 1984.
- [20] G. Cavone, M. Dotoli, and C. Seatzu, "A survey on Petri net models for freight logistics and transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 6, pp. 1795-1813, 2017.
- [21] K. M. Ng, M. B. I. Reaz, and M. A. M. Ali, "A review on the applications of Petri nets in modeling, analysis, and control of urban traffic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 858-870, 2013.

- [22]L. Zabala, R. Solozabal, A. Ferro, and B. Blanco, "Model of a Virtual Firewall Based on Stochastic Petri Nets," presented at the IEEE 17th International Symposium on Network Computing and Applications (NCA), 2018.
- [23]M. Ghazel, "Using stochastic Petri nets for level-crossing collision risk assessment," *IEEE transactions on intelligent transportation systems*, vol. 10, no. 4, pp. 668-677, 2009.
- [24]R. Zeng, Y. Jiang, C. Lin, and X. Shen, "Dependability analysis of control center networks in smart grid using stochastic petri nets," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 9, pp. 1721-1730, 2012.
- [25]R. Mitchell and I. R. Chen, "Modeling and analysis of attacks and counter defense mechanisms for cyber physical systems," *IEEE Transactions on Reliability*, vol. 65, no. 1, pp. 350-358, 2016.
- [26]R. Mitchell and I. R. Chen, "Effect of Intrusion Detection and Response on Reliability of Cyber Physical Systems," *IEEE Transactions on Reliability*, vol. 62, no. 1, pp. 199-210, 2013.
- [27]E. Andrade and B. Nogueira, "Dependability evaluation of a disaster recovery solution for IoT infrastructures," *The Journal of Supercomputing*, pp. 1-22, 2018.
- [28]W. C. Moody, H. Hu, and A. Apon, "Defensive maneuver cyber platform modeling with Stochastic Petri Nets," presented at the 10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing, 2014.
- [29]D. Miehle, B. Häckel, S. Pfossier, and J. Übelhör, "Modeling IT Availability Risks in Smart Factories: a Stochastic Petri Nets Approach," *Business & Information Systems Engineering*, 2019.
- [30]MQTT 3.1.1 specification. OASIS. December 10, 2015. Retrieved April 25, 2017.
- [31]Lightweight Machine to Machine Requirements: Version 1.1 – 10 Jul 2018 Open Mobile Alliance (OMA-RD-LightweightM2M-V1\_1-20180710-A).
- [32]L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Trans. Programming Languages and Systems*, vol. 4, no. 3, pp. 382-401, 1982.
- [33]R. M. Alvarez, F. J. Boehmke, and J. Nagler, "Strategic voting in British elections," *Electoral Studies*, vol. 25, no. 1, pp. 1-19, 2006.
- [34]J. H. Aldrich, A. Blais, and L. B. Stephenson, *The Many Faces of Strategic Voting: Tactical Behavior in Electoral Systems Around the World*. University of Michigan Press, 2018.
- [35]S. Kosta, A. Mei, and J. Stefa, "Large-scale synthetic social mobile networks with SWIM," *IEEE Transactions on Mobile Computing*, vol. 13, no. 1, pp. 116-129, 2012.
- [36]U.S. Department of Homeland Security, Geospatial Location Accountability and Navigation System for Emergency Responders (GLANSER) 2009.