

Creative Commons Attribution 4.0 International (CC BY 4.0)

<https://creativecommons.org/licenses/by/4.0/>

Access to this work was provided by the University of Maryland, Baltimore County (UMBC) ScholarWorks@UMBC digital repository on the Maryland Shared Open Access (MD-SOAR) platform.

**Please provide feedback**

Please support the ScholarWorks@UMBC repository by emailing [scholarworks-group@umbc.edu](mailto:scholarworks-group@umbc.edu) and telling us what having access to this work means to you and why it's important to you. Thank you.

Received 16 August 2023, accepted 27 October 2023, date of publication 6 November 2023, date of current version 16 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3330688

## APPLIED RESEARCH

# A Networked System Dependability Validation Framework Using Physical and Virtual Nodes

SUHEE SANJANA MEHJABIN<sup>1</sup>, (Graduate Student Member, IEEE),  
ALI TEKEOGLU<sup>2</sup>, (Member, IEEE), MOHAMED YOUNIS<sup>1</sup>, (Fellow, IEEE),  
MOHAMMAD EBRAHIMABADI<sup>1</sup>, (Graduate Student Member, IEEE), RAHUL CHANDRAN<sup>3</sup>,  
TAMIM SOOKOOR<sup>3</sup>, (Member, IEEE), AND NAGHMEH KARIMI<sup>1</sup>, (Senior Member, IEEE)

<sup>1</sup>Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, Baltimore, MD 21250, USA

<sup>2</sup>Johns Hopkins University Whiting School of Engineering & Leidos Innovations Center, Baltimore, MD 21218, USA

<sup>3</sup>Johns Hopkins University Applied Physics Laboratory, Laurel, MD 20723, USA

Corresponding author: Suhee Sanjana Mehjabin (suheesm1@umbc.edu)

This work was supported by the University of Maryland Baltimore County from the Johns Hopkins University Applied Physics Laboratory under Award 011412.

**ABSTRACT** Emerging applications in the context of smart cities pursue a decentralized design that often involves numerous networked components. To validate such a design, the scientific community has resorted to software based simulators to find a way around the complexity of building large scale physical network test-beds. Network Simulator-3 (ns-3) is one of the most popular platforms for this purpose where communication-related performance metrics, e.g., latency and throughput, can be evaluated. Yet, concerns exist about the viability of such a simulated approach when assessing dependability metrics, e.g., trust and resilience to cyberattacks, since the misbehavior is mainly defined by the evaluator. Incorporating physical nodes within the simulated network would be advantageous in that regard. Advances have been made to connect network simulators, e.g., ns-3, to virtual machines to emulate communication with real devices. However, all efforts in the literature so far have been limited to a single physical host. This paper presents a framework where many external physical devices can act as a part of the ns-3 simulator and interact seamlessly with the nodes within the simulated network via Docker containers. Hence, our framework enables scalable and cost effective experimentation to validate dependability metrics like fault-tolerance and attack resilience. We demonstrate the utility of the proposed framework in evaluating performance under a set of attack scenarios.

**INDEX TERMS** Hardware-in-loop experimentation, dependability evaluation, ns-3, docker.

## I. INTRODUCTION

The major advances in communication and computing technologies in recent years have fueled interest in unconventional applications that combine sensing, processing, and actuation. Numerous examples can be noted in the context of smart cities [1], such as cooperative driving [2], power distribution management [3], [4], etc. Decentralization and close coordination reflect the main characteristics of the design paradigm of these applications, where an application is realized as a set of networked components. The notions of Internet of Things (IoT), and Cyber-physical systems (CPS)

The associate editor coordinating the review of this manuscript and approving it for publication was Xujie Li<sup>1</sup>.

are often used to capture how the design fundamentally differs from the traditionally centralized approach.

## A. MOTIVATION

Due to the scale and complexity of the aforementioned applications, the process of validation and testing becomes significantly challenging. Generally, validation of an application design involves functional and robustness attributes at both the module (component) level and the network (integrated system) level. The functional aspect reflects the correctness of the output for a given input, and can be systematically checked using simulated, prototype or full implementation. The networking aspect, on the other hand,

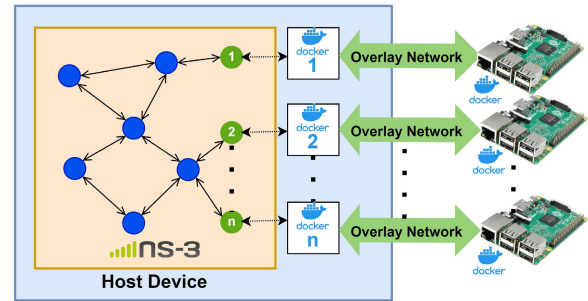
is harder to validate where the cost of a full network implementation deems it to be an impracticable option. Hence, simulation based validation has been the popular means for assessing network performance. Ns-3 [5] is one of the most widely-used network simulation environment where nodes executing applications can be instantiated with several communication protocols. It enables relatively high fidelity simulation.

However, as demonstrated in [6] and [7], it is impossible to fully characterize a network, in particular a wireless network; therefore, the gold standard for experimentation is still a physical environment. In addition, simulation is good for handling scale rather than diversity in terms of module-level functional behavior. Moreover, assessing dependability attributes, such as fault tolerance, would be better characterized using physical devices that can be independently controlled. For example, to gauge resilience to cyberattacks, it is better to separate the definition of the attacker's behavior from the simulated network and possibly let it be determined by someone other than the designer of the attack countermeasure.

## B. TECHNICAL GAP

Existing frameworks for validating networked systems differ based on the modeling fidelity of the various components. Simulation-based frameworks, such as [8], pursue an abstract (coarse level) modeling of all or some of the components. Researchers frequently rely on simulations as the preferred approach at the outset of their research and for initial validation of their ideas. Although simulations are often extensively used in the development phase, it is not always feasible to completely capture the behavior of the real system, as the effect of environment and the operation of communication and commutation circuits may be unpredictable or difficult to model in advance. Emulation-based frameworks, e.g., [9], elevate the validation of results to a higher level through the provision of manageable and reproducible environments, effectively reducing the reliance on real-world application experiments. However, emulation, much like simulation, fails to offer the opportunity to fully test a system under practical conditions; such a limitation hampers the ability to accurately reflect real performance.

Overall, existing frameworks are geared for assessing scalability and limit the support for heterogeneity whereas provisioning certain behavior requires predefinition. We argue that these capabilities are not sufficient for validating the system's robustness under unexpected operating conditions where the system's response is not provisioned at the design phase. This particularly is very crucial for mission critical applications where dependability and resilience to cyberattacks are primary objectives. To illustrate, many of the existing IoT routing protocols have been evaluated in a simulated setup without studying the effect of what an intelligent attacker will do in practice to cripple the performance and elude security protection.



**FIGURE 1.** Illustrating the proposed network-level dependability validation framework with multiple external physical nodes. The ns-3 environment is marked as orange, where as the host device is marked as light blue color.

## C. CONTRIBUTION

To fill the technical gap, we propose a novel framework that enables incorporation of physical nodes in conventional network simulations. We use a prominent tool, specifically, ns-3, as the underlying network simulator, and extend its capabilities to allow physical devices to be part of the network along with the virtual (model-based) nodes. Such added features are enabled and supported by docker containers, overlay networks, and Linux bridges. Using docker containers on embedded devices allows the impact of external factors, such as environmental noise, to be studied across multiple devices. In fact, the physical devices can be individually and independently controlled to enable fault injection, link quality variations, and for launching attacks, as means to gauge the network performance and dependability metrics under stress. Existing approaches that use Linux containers virtualize nodes on a single machine and thus would not allow diverse and independently controlled devices to be part of the network. A high-level architecture of our dependability validation framework is depicted in Figure 1. In such an extendable, modular setup one or more external devices could be configured to communicate with a simulated ns-3 network. Docker containers are used on the external nodes to create an overlay network between corresponding docker containers on the host. Ghost nodes, represented as green nodes in Figure 1, are a digital twin of the external physical devices inside the simulated ns-3 network. Hence, the proposed framework enables large scale experimentation to be conducted where a subset of physical nodes can be used to investigate the impact of phenomena that may not be captured in the simulation. The contribution of the paper can be summarized as follows:

- Develop a novel framework for validating a networked system that is composed of virtual and physical nodes.
- Demonstrate the viability of the proposed framework through use-cases for tolerating node failures and assessing resilience to cyberattacks.

The rest of this paper is organized as follows. In Section II we cover related work and provide a detailed comparison to most relevant previous research in the literature. To provide the necessary background, Section III presents a brief overview of ns-3, docker containers, and Linux network

bridges. Section IV describes the proposed framework in detail. Section V discusses the experimental setup and the validation results. Finally, Section VI concludes the paper and discusses future directions.

## II. RELATED WORK

Based on the way underlying network nodes are modeled, the existing frameworks for studying the network performance can be categorized into; (1) *simulation*, (2) *emulation* and (3) *hybrid* and (4) *hardware-based prototypes*. These categories fundamentally differ in how the behavior of network nodes are captured. In simulation-based frameworks, a node is abstracted using a coarse-grained behavior model with limited diversity among the network nodes. Meanwhile, the emulation category employs a significantly finer node model that can even be defined using the actual code that a node runs in practice; yet the effect of the environment and the underlying hardware is not factored in. Moreover, scalability is a major shortcoming for emulation-based methods. The last category of frameworks combines simulation and emulation models to achieve hybrid networks.

On the other hand, validation of dependability metrics is commonly conducted through fault injection experiments and formal analysis. For systems with centralized control or management, the experiments often involve a prototype implementation. Yet, such an approach is unsuited for distributed systems involving numerous networked components; in such a case simulation/emulation becomes unavoidable. In the balance of this section we discuss prior work, grouped based on the three categories, and highlight the shortcomings of each cited framework. A comparative summary of the cited work is provided in Table 1.

### A. SIMULATION-BASED VALIDATION

Researchers in industry and academia have developed and relied upon network simulations and models as virtual test-beds to evaluate new algorithms and networking protocols. Numerous tools are available including commercial ones such as OPNET [28] and QualNet [29], as well as open-source tools such as ns-3, OMNeT++ [30], and J-Sim [31]. As noted earlier, these tools focus more on the communication aspects and support only high-level definitions of node behavior models with limited diversity among the nodes. In default, none of these tools support the capability of incorporating physical nodes in the simulated network topology.

To support applications involving Cyber-physical systems, researchers have tried to augment the capabilities of conventional network simulators by integrating other tools. For example, the research effort detailed in [10] proposes a hybrid simulation environment combining different data acquisition, data processing and component modeling tools within MATLAB for intelligent transportation systems. Similarly, Liu et al. combined vehicular traffic simulator with ns-3 [32]. In the context of remote health monitoring applications, vital sign sensors are incorporated to validate the performance of

the underlying body area network [33]. However, CPS-based simulators opt to model the physical aspect of the application rather than engaging actual devices in the simulation.

### B. EMULATION-BASED SCHEMES

To make the validation research and experiments more realistic the concept of network emulation was introduced to replicate real-world environments closely. Zhang et al. have created an experimental platform for mobile networks where an image of Mobile Operating System (MOS) is created in a Virtualbox virtual machine where ns-3 is deployed. All network testing is done in simulation so that the code can be directly installed into a real mobile device [8]. To reduce the setup time and complexity involved in experimentation, To et al. presented a tool that allows researchers to emulate physical networks bridging between ns-3 and Linux containers through Docker [11]. Petersen et al. have built on this work by leaving out Linux containers and providing a more streamlined and user friendly version [12]. However, none of the frameworks accommodate the notion of using independently controlled external devices and thus cannot analyze the effects external nodes might have on the network.

Similarly, quite a few emulation tools support both coarse and fine grained nodes models while providing their own or incorporating open-source network simulators [13], [15], [16], [17], [18], [19], [20]. Yet, nodes external to the simulator are virtual in the same host machine. The notion of independently controlled external devices and their integration with the simulator is absent. On the other hand, the concept of Colosseum as a digital twin presented by Villa et al. also only mimics a real network rather than creating a bridge between simulated and physical networks [9].

### C. HYBRID APPROACHES

This line of research has further included hardware-in-the-loop test-beds where the ease of scalability and configurability offered by virtualization is complemented by the fidelity of external physical networks. Yet, many current solutions are not sufficiently flexible to easily enable the trade off between scale and fidelity. While tools like [21] and [22] have introduced a hybrid component in the network by connecting two or more simulated networks via physical link, they have not explored integrating an independent physical network with the virtual one. Although the design of [23], [24], and [25] incorporates Linux containers with a network simulator, physical nodes are part of the same host machine, which defies the objective of assessing dependability attributes with independently controlled physical devices.

### D. HARDWARE-BASED PROTOTYPES

Furthermore, researchers have turned to genuine hardware testbeds for testing and experimentation. Most notable efforts were in the realm of wireless sensor networks (WSN) to take into account realistic environmental conditions. SDN Wisebed is one of such testbeds where one can

**TABLE 1. Comparison of this work with previous research in the literature.**

Ref.	Contribution	Used Tools	Methodology	Limitations
[8]	Incorporates a device operating system in a virtual environment with ns-3 for network testing and direct device installation.	Android-x86, ns-3, Docker	Virtual	Lack of independently controlled external devices.
[9]	This accounts for simulating a digital twin of a real network rather than combining a simulated and physical network.	Colosseum	Virtual	Lack of independently controlled external devices.
[10]	Supports systems with data acquisition and allows integration and adaptation of new components.	Matlab, LabVIEW, Multisim	Virtual	Completely simulation based; the notion of "hybrid" is used to refer to a combination of different tools.
[11]	Emulating wireless networks in a virtual environment rather than connecting physical and virtual nodes.	ns-3, Linux containers	Virtual	The frameworks work with isolated containers in a single host which makes it impossible to fully characterize the network in real world scenarios.
[12]	Upgraded streamlined code and installation process for the [11].	ns-3, docker containers	Virtual	An emulation tool that lacks support for independently-controlled external devices.
[13]	Still employs coarse-grained node models but attempts to generate a realistic mobility scenario to simulate real networks.	GTNetS [14]	Virtual	Does not support independently-controlled external devices.
[15]	An emulator that creates virtual wireless networks at lower layers and enables the real code to run at higher layers.	Scripting language based (Tcl/Tk)	Virtual	All virtual nodes run on the same physical machine; no support for independently controlled external devices.
[16]	An emulation framework to run native applications in a simulated network by utilizing ns-3 and lightweight operation system-level virtualization	ns-3	Virtual	All virtual nodes run on the same physical machine; also external devices cannot be Incorporated.
[17]	A simulation system that executes discrete event simulations by embedding simulation semantics directly into the Java execution model.	Java	Virtual	Lack of independently controlled external nodes.
[18]	A network emulator that facilitates protocol level testing and debugging of performance issues in mesh networks.	Vde-switch, Virtual box	Virtual	Lack of independently controlled external nodes.
[19]	A lightweight network emulator based on User-Mode Linux; it allows users to experiment with a large number of network technologies.	User-mode Linux kernel	Virtual	Lack of support for independently-controlled external nodes.
[20]	Emulates Wide-area networks using a LAN-based test environment.	Linux OS platform	Virtual	Emulation tool. No support for independently-controlled external nodes.
[21]	Develops an emulation environment where Ethernet is used to connect different machines, each simulating a different wired/wireless network.	FreeBSD with patched kernel, Tcl/Tk	Hybrid	Does not allow the incorporation of physical nodes.
[22]	Develops a remotely accessible testbed in a physical setup, a simulated environment, or a hybrid scenario connecting two simulated networks on separate physical devices.	ns-3, Rpis, Switches	Physical/ Virtual/ Hybrid	Allows the use of a physical device to interconnect two simulated networks.
[23]	Uses Linux-based Docker Containers with ns-3 to conceptualize a haptic systems.	ns-3, docker	Hybrid (Same host)	Independently-controlled external devices are not supported
[24]	Combines LTE and ad-hoc simulated networks at the same time for a data transfer.	ns-3, docker	Hybrid (Same host)	Lack of independently-controlled external devices.
[25]	Enables simulation of wireless sensor networks using fine-grained node and link models; the models can include hardware specific features and enable code auto-generation for actual sensor nodes.	MathWorks	Hybrid (Same host)	Limited scope to sensor networks.
[26]	Analyzes the impact of security attacks on the performance of the AODV routing protocol.	ns-2	Virtual	Does not allow independently controlling the attacker's behavior.
[27]	Studies the effects of flooding attacks on the performance of the AODV routing protocol.	ns-3	Virtual	Does not allow external control of the attacker's behavior.

integrate, test, and evaluate the performance of routing and application [34]. A study by Ketata et al. also employed a hardware based platform to efficiently optimize performance and validation of the WSN operation [35]. Another study aims to provide a wired/wireless WSN hardware testbed in a real smart-grid environment to evaluate different performance metrics [36]. However, pursuing hardware-based testbeds hinders scalability assessment and increases the overall expense associated with the implementation using off the shelf physical devices.

#### E. NETWORK DEPENDABILITY ASSESSMENT

Dependability is a term that covers quite a few metrics, some of which relate to network performance. For example, packet drop, and bit error rates are popular metrics that reflect the robustness of communication links within a network. Yet, in simulation these conventional communication-related metrics are governed by the underlying link quality and traffic generation models and do not allow externally-controlled deviation and/or manipulation. In addition, introducing node failure is often synchronized and pre-scheduled. The



incorporation of external devices in the network enables better assessment of the network response to transient and sudden failure that is introduced asynchronously and unpredictably.

Moreover, work on network security has provisioned means to mimic attacks. For example, ns-3 has been used to validate resilience to cyberattacks on Mobile Ad Hoc Networks (MANETs) that employ dynamic data routing protocols, such as Ad hoc On-demand Distance Vector (AODV), which are vulnerable to a number of attacks including Black-hole, Flooding [27], and Rushing attacks [26]. As another example in the context of a MANET, trust assessment and management have been evaluated using ns-2 [37]. Our proposed framework provides more effective means for evaluating robustness to failure and cyberattacks by enabling the abnormal behavior to be injected and controlled independently by external devices, i.e., decoupling the designer from the attacker. Section V demonstrates the utility of our proposed framework in conducting such evaluations. Specifically, we have enabled multiple external physical nodes to be incorporated to act as attackers for a virtual ns-3 network.

### III. DESIGN CHALLENGES AND PRELIMINARIES

This section provides an overview of the key technologies used to create our framework.

#### A. DESIGN ISSUES AND APPROACH OVERVIEW

Researchers opt to achieve specific objectives while creating any network system dependability validation framework. Their design goals include making use of low-cost equipment and user-friendly software, supporting large-scale experiments, combining simulation and/or emulation with actual devices to display heterogeneity, and having the ability to implement and validate dynamic network topologies in a modular, efficient, and effective manner. The proposed framework aims to achieve the aforementioned design goals effectively. The implementation uses off-the-shelf devices and open source network simulator, namely ns-3. As a matter of fact, our framework incentivizes the implementation of large-scale testbeds since the overall application validation becomes cost-effective. On the other hand, open source network simulators such as ns-3, have a large library of readily-available modules and protocols which expedites the validation process and improves the user-friendly experience.

Several of the studies discussed in Section II aiming to achieve hybrid framework, typically utilizes virtual machines (VM) which are provisioned with resource allocators called hypervisors on top of a host system. In some of the studies VMs are in fact replaced by Linux containers which are standalone packages and run only the application needed, not a full OS. However, it still possesses certain limitations such as degraded scalability and node virtualization only on a single machine, which constitute hindrance for achieving the design objective. To address these issues, our framework uses Docker containers, which: (1) have lightweight images

that do not require guest OS, and (2) are standalone packages that can accommodate diverse and independently controlled devices to be part of any simulated network through overlays and Linux bridges, thus demonstrating heterogeneity in our design. This also allows us to study any impact of completely independent external entity infiltration in any network. Our framework also allows integration of hardware based security primitives in an otherwise virtual only network simulation.

#### B. PRELIMINARIES

This subsection provides brief background coverage of the key technologies and tools used in the proposed framework.

##### 1) NETWORK SIMULATOR-3

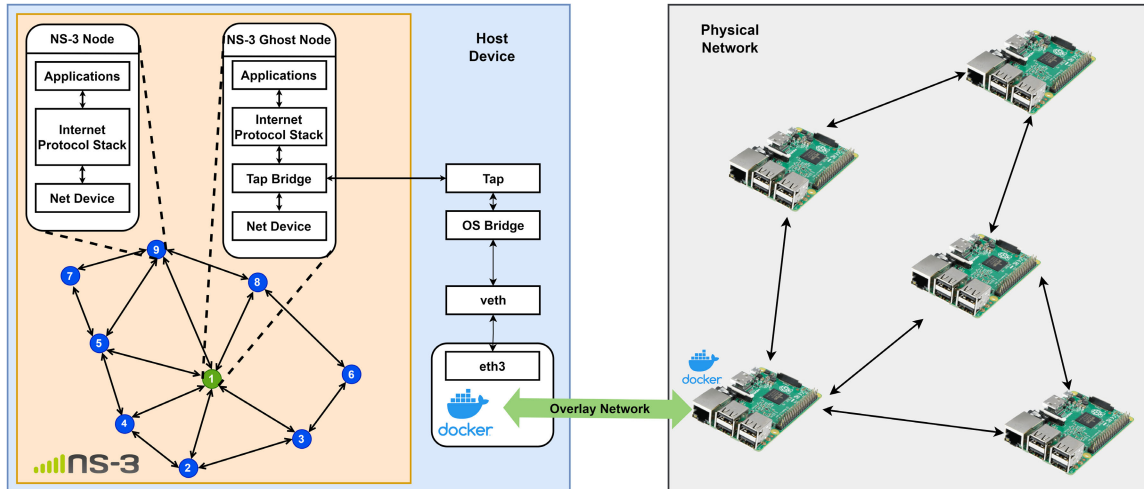
ns-3 is a discrete-event network simulator that fulfills the research, development, and educational needs of exploring modern computer networks [38]. The key advantage of ns-3 is that it enables realizing the entire network protocol stack and hence helps in developing realistic simulation models and enables the validation of different protocols and network typologies. ns-3, and its predecessor ns-2, are the most popular simulation software for network research. Most published ns-3 based studies involve wireless/IP simulations, which use models for Wi-Fi, WiMAX, or LTE for layers 1 & 2 and a range of static or dynamic routing protocols for IP-based applications, such as OLSR and AODV. ns-3 provides versatile modules like Tap NetDevice, which when installed in a node, accommodates communication with the real world through virtual Ethernet bridges.

##### 2) DOCKER

A container is a standard unit of full-machine virtualization containing software packages and its dependencies for efficient and reliable execution between multiple computing environments. Docker allows such containerization by capturing applications in isolation from their host [39]. It facilitates groups of processes operating on the same kernel to have an independent view of the underlying operating system including a unique network IP, usernames, and network interfaces. Docker also comes with a provision of creating a network and communicating among the existing containers in the same Linux host as well as in a different device; this feature makes Docker particularly advantageous in applications where integration of multiple devices is necessary.

##### 3) LINUX NETWORK BRIDGES

A Linux bridge behaves like a virtual network switch to transfer data packets between the interfaces that it connects to, in a protocol-independent way. Unless a Linux bridge is bound to real devices, it cannot receive or transmit anything [40]. Packets are forwarded based on Ethernet addresses at layer 2 which ensures transparent movement of all protocols through a bridge. A Tap bridge is one such bridge provided



**FIGURE 2.** Details of networking stack inside ns-3 nodes, the host OS as well as the external node in the proposed validation framework. Here, the ns-3 environment is marked as orange, where as the host device is marked as light blue and the physical network is represented as grey in color.

by ns-3 for transferring data packets between the simulator and the host OS.

#### IV. FRAMEWORK DESCRIPTION

The proposed framework enables the formation of a network using virtual and physical nodes. The virtual nodes are supported by a network simulator; ns-3 in our case. The physical nodes are external to the simulator, yet integrated in a seamless manner. This section describes our framework design in detail. The presentation first covers the architecture before discussing the flow of the various steps.

##### A. ARCHITECTURE

Our dependability validation framework consists of virtual and physical parts. The virtual part reflects the simulated network and resides on a host machine. The physical part refers to external nodes that can be individually interfaced to the host machine (virtual network) or interconnected to form a physical network that is linked to the virtual network through a gateway. Figure 2 shows a sample configuration. The individual components are described in the balance of this subsection.

##### 1) NETWORK SIMULATOR

The virtual part of the proposed framework has been implemented using ns-3, which is a versatile simulator that provides the means for network scenario generation. The simulation environment consists of multiple nodes. The position and mobility pattern of these nodes can be defined by the user or randomly generated. Each node in ns-3 can contain an Internet Protocol Stack, a list of NetDevices, and Applications. The application module provides a fundamental abstraction for an activity, such as generating traffic across the network, to be simulated whereas the NetDevices are installed in ns-3 virtual nodes

to establish communication over the channel [5]. The nodes can communicate with one another using wired or wireless connections or both. A variety of routing protocols can be applied. Common routing protocols are provided in ns-3 as libraries which can be used independently or combined with a different routing protocol in the same simulation setup. There is also a provision for implementing a user-defined routing protocol.

The entire simulation network has been developed in ns-3 consisting of virtual and physical nodes. The physical nodes are defined in the ns-3 environment as *Ghost Node*, which is a virtual representation of a physical node. There is a provision of *Ghost Nodes* communicating with the real work using network bridges.

##### 2) HOST MACHINE

The computer that runs the ns-3 simulator is called the host machine. This is the component that physical nodes are connected to in order to integrate with the network simulator. Specifically, a primary role of the host machine in the framework is to route the packets in and out of the ns-3 simulator towards their respective external devices (physical nodes). In our framework, this is accomplished using Linux bridges. The host machine resorts to containerization to ensure the packets are sent to the correct physical devices and that all the physical devices connected to the ns-3 environment are independent and cannot communicate with each other through the host machine itself.

##### 3) PHYSICAL SUB-NETWORK INTERFACE

The physical counterpart of the framework has been implemented using docker. By being computationally lightweight, docker can be installed in devices with limited processing power. Since multiple physical devices are connected to the same host machine, it is absolutely necessary to provide

isolation within the host machine to ensure that they only communicate by means of the routing protocol that is being applied. The provision for containerization in docker achieves such an isolation. Docker also has provisions for creating local Ethernet or wireless networks, through which one docker container can ceaselessly communicate with another even if they are deployed on different devices. Thus, docker provides a simple and lightweight networking solution for integrating physical networks, supporting module isolation in the host machine and exchanging packets between the host machine and the external devices.

#### 4) PHYSICAL NODE

A physical node in the context of the framework is any external device that is to be a part of the network being emulated. External physical nodes are connected to each other through some form of a network, which could be an Ethernet, WiFi, Zigbee or Bluetooth, as long as they have IP/MAC addresses and can communicate with each other. Some of the physical nodes are directly connected to the *Ghost Node* of ns-3 simulation framework. Those nodes have a docker container dedicated to them in the host machine via which they communicate with ns-3. Finally, we note that the host machine itself may be a physical node, i.e., the ns-3 simulator is running on one of the physical network nodes.

### B. NETWORK VALIDATION STEPS

The proposed framework accommodates integration of real physical network with virtual simulated network where nodes in both networks can communicate ceaselessly via the deployed routing protocol. The steps are as follows:

#### 1) DESIGNING THE VIRTUAL SUB-NETWORK

The simulated part of the network is handled in ns-3. Such a part which is referred to as the virtual sub-network may pursue wired or wireless links depending on the application specification. The virtual sub-network is also set to apply a routing protocol of choice. The involved virtual nodes are to have predefined positions and mobility models. A *Ghost Node* is introduced to the network for every physical node at one-hop distance from any node in the network. The simulated sub-network communicates with the physical sub-network through the *Ghost Nodes* and the physical devices associated with them. A *Ghost Node* is depicted as node # 1 and marked in green in Figure 2 while the other nodes marked in blue are virtual nodes.

#### 2) SETTING THE PHYSICAL SUB-NETWORK

The physical sub-network comprises multiple devices each with their own processing capabilities. It is imperative for each physical device to have the ability of docker being installed on them. The docker provides both wired and wireless networking opportunities among the devices. For connecting physical devices in different locations, docker has provisions for creating overlay network that enables one docker container to communicate with a remote docker

container. The physical devices located at one-hop distance from a virtual node are connected to isolated docker containers in the host machine which eases their communication with the host machine, which in turn can initiate communication with the simulation environment via *Ghost Nodes*. Figure 1 shows the architecture when multiple external physical nodes are attached to the ns-3 simulated network over Docker containers on both ends of the overlay networks which in turn are carried over via *Ghost Nodes* into the ns-3 network.

#### 3) BRIDGING THE VIRTUAL NETWORK TO HOST MACHINE

In order for the ns-3 simulation environment to actually communicate with physical nodes, the exchanged data packets need first to be brought in and out of the ns-3 environment to the host machine. This is achieved using Linux bridges. There is a provision in ns-3 to install a Linux bridge called the *Tap Bridge* which can map a node interface to a *Tap Device* installed in the host machine. *Tap Bridge* is installed in each *Ghost Nodes* and separate *Tap Devices* are installed in the host machine for each *Ghost Node*. The ns-3 environment can then communicate with the host device through the tap interface.

#### 4) BRIDGING THE HOST MACHINE TO DOCKER CONTAINER

The *Tap Device* in the host machine act as a portal where data is sent to and from the simulation environment. If there are multiple *Tap Devices* and they are directly connected to physical nodes, there is a possibility of traffic at different physical devices interfering with each other. In such cases interference could affect the entire simulation which makes it essential to isolate the communication process. For that purpose, a dedicated docker container is set up for each *Ghost Node*. In order to connect *Tap Bridge* with the docker container, Linux Ethernet bridges are set up. Thus, when the simulated node sends a data packet to a physical node, it send the packet first to the *Ghost Node*. The *Tap Bridge* of the *Ghost Node* then brings the packet out of the ns-3 simulation environment. The Linux bridges transfers those packets into the respective docker container from where the packet reaches the physical node via docker network.

#### 5) STATIC ROUTING

One of the challenges in integrating the virtual sub-network with the physical one to work as a single network is that the simulation environment does not know the existence of any physical node which is not connected to it via *Ghost Node*. Again, the physical nodes too do not know about the existence of any virtual nodes which are more than one-hop distance from them. Hence, it is necessary to define a gateway through which virtual nodes communicate with real nodes. For this purpose, static routes are created from each physical node to virtual node and vice-versa which define the *Ghost Nodes*, or the devices connected to them as gateway. It is done to ensure that while sending a data packet from a virtual to a physical sub-network or vice-versa, each node can know where the next-hop should be.



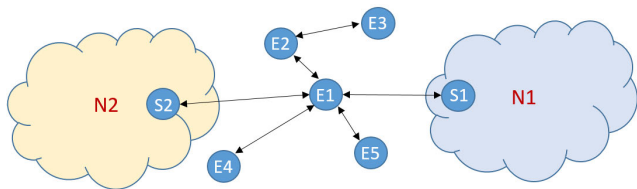
## V. EXPERIMENTAL EVALUATION

We have implemented the proposed framework and assessed its viability through a number of use cases. Details can be found in the balance of this section. The source code for the validation framework and experimentation have been made publicly available.<sup>1</sup>

### A. IMPLEMENTATION AND EXPERIMENT SETUP

The proposed framework is evaluated using ns-3 version 3.32 on a Ubuntu 20.04 host operating system. The latest version of Docker is installed on a host Ubuntu machine and Raspberry PI-4 (RPI-4) devices. The ns-3 network established for experimental evaluation contains 9 nodes, as depicted in Figure 2. Node 1 is the ghost node while the others are simulated nodes. Packets are routed among the nodes using the AODV protocol [41]. The ghost node is installed with a Tap bridge. A Linux bridge is created in the host device. The Linux bridge in combination with the ghost node enables exchanging packets with ns-3 as described in [42] and depicted in Figure 2.

Docker has been installed in the host device and a swarm is created within the docker, where that Docker engine on the host acts as the swarm manager. A physical IoT node, a RPI-4 in our case, can join the same swarm as a worker. The Docker container running on the RPI-4 and the Docker container running on the host device are connected with a docker overlay network through which they can communicate. Static routes to nodes inside the ns-3 network have been provided to the RPI-4 to enable it to communicate with the ns-3 environment using the overlay network, Tap bridge, and external Linux bridges.



**FIGURE 3.** Network topology for ns-3 and external nodes, running fault recovery algorithms. In such a configuration, two network segments N1 and N2 (represented using orange and light blue) are interconnected through node E1.

### B. EVALUATION EXPERIMENTS

To demonstrate the application of our framework, we have developed scenarios where an ns-3 simulated network cannot solely achieve what a hybrid physical and virtual network can. One of the scenarios opts to show how our framework enables the validation of fault-tolerance metrics by introducing node failures and gauging the performance of recovery procedures. Specifically, we apply the *Distributed Actor Recovery Algorithm (DARA)* [43], which assesses the criticality of the failed node with respect to network connectivity and, if needed, pursues node repositioning to interconnect disjoint network partitions that could have resulted from the failure.

The other scenarios are geared to highlight the agility in launching cyberattacks by physical nodes against a simulated network. For this purpose, a trust management system is employed and implemented as an application in ns-3. Such an application is installed on all virtual nodes to track the trust score of a node to its neighbors. The trust manager captures the behavior of a node based on the AODV routing protocol taking the number of Route Request (RREQ) and Route Reply (RREP) packets into account. The trust evaluation scheme has been derived based on the work of Niroshan and Pecorella [44], and is calculated as follows:

$$T_x^y = \frac{\#RREPPackets}{\#RREQPackets + \#RREPPackets} \quad (1)$$

where  $T_x^y$  defines the Trust Score of neighbouring node-y from the perspective of node-x. #RREP packets are the total of all route replies coming from node-y to node-x or to another originator node that has a route through node-x, while #RREQ packets count the number of route requests going to node-y from node-x. When node-x receives a packet it determines the immediate source from which the packet came through its IPV4 header and from the AODV header it determines the type of the packet (i.e., RREP/RREQ).

### C. NODE FAILURE SCENARIOS

In many IoT and CPS applications, nodes operate in a harsh environment. Upon their deployment/activation, they are expected to form a network in order to share data and coordinate their action when participating in the execution of a task. To enable such interaction, nodes need to stay reachable to each other. Given the importance of the inter-node connectivity to the effectiveness of the application, tolerance of a node failure is necessary to prevent the network topology from becoming divided into isolated segments [46], [47], [48]. In addition, it may be desired in some application setups to federate (link) multiple standalone networks to achieve a particular mission [49].

Validating failure recovery and network federation algorithms is quite challenging. The fundamental issue is how to accurately mimic environmental effects, ensure convergence and assess efficiency while factoring in decentralized control. Reliance on simulation alone would fall short of pointing out anomalies and cases for which the algorithm diverges or leads to unstable topologies. Our framework would be invaluable for these cases since the physical nodes can be controlled and operate in what could be an actual setting. To demonstrate the viability of our framework, we have considered the network configuration in Figure 3. Obviously, E1 is a critical node (cut-vertex in a graph) since its failure will not only disconnect N1 and N2, but also isolates nodes E4, E5 and the branch of E2 and E3.

DARA [43] is an algorithm for detecting the failure of critical nodes and restoring connectivity by relocating some of the healthy nodes. DARA assumes that each node is aware of its one-hop and two-hop neighbors. In the context of Figure 3, DARA will rely on the direct neighbors of E1,

<sup>1</sup><https://github.com/SS-Mehjabin/Dependability-Validation-Framework>

**TABLE 2.** Experiment scenarios showcasing failure and recovery of network via DARA procedure implemented on RPI controlled iRobot<sup>®</sup> Create and ns-3 simulation.

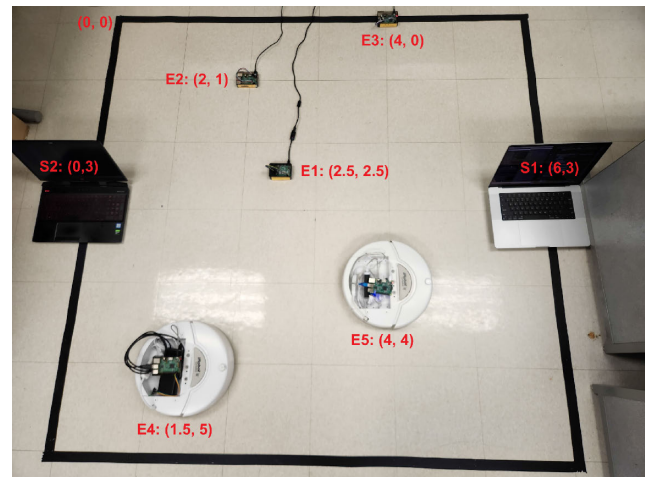
Exp.	Scenario	Implementation	Observation	Conclusion
(1)	Top recovery candidate ( $E5$ ) cannot reach the position of failed node.	$E5$ experiences impassable obstacles on its travel path.	The neighbors of $E1$ assumes that $E5$ is dead and selects the 2 <sup>nd</sup> best candidate for recovery ( $E4$ ).	The travel route should be factored in the selection.
(2)	The top candidate ( $E5$ ) takes a long time to reach $E1$ position due to travelling around an obstacle or over a rough road.	$E5$ is turned off in order to force $E4$ to time out; then $E5$ is turned on again.	The 2 <sup>nd</sup> top candidate, namely $E4$ , times out progress and unnecessarily relocates to the position of $E1$ (unless they are in a collision course), where $E5$ actually has moved to.	Keep broadcasting during travel to prevent the 2 <sup>nd</sup> top candidate ( $E4$ ) from going all the way to discover being redundant and return back to its pre-failure position.
(3)	$E1$ fails due to a hazard in the vicinity; when the top candidate $E5$ replaces $E1$ , it fails immediately as well.	Turn $E5$ off as soon as arriving at the position of $E1$ , forcing $E4$ to move. Upon arriving, then $E4$ is turned off.	Discovering the issue may be late (after both $E4$ and $E5$ fail due to hazard).	Consider other recovery algorithms such as RIM [46], which does not require placing a substitute at the failed node position.
(4)	The failed node is part of a cycle.	Create a cycle by adding a link between $E5$ and $S1$ . We also add a link between $E4$ and $S2$ so that $E5$ continues to be the best candidate based on node degree and proximity.	Since $E5$ died trying to replace $E1$ , $S1$ is tempted to replace $E5$ ; yet when $E5$ arrives at the position $E1$ , $S1$ seizes motion.	Apply the local flooding procedure (DARA-LF) [44] or use a recovery algorithm that factors in dominating sets in the topology.
(5)	Having simultaneous failure of multiple collocated nodes.	Turn both $E1$ and $E2$ off to mimic failure.	Both $E3$ and $E5$ move to replace both $E2$ and $E1$ , respectively.	Sometimes DARA can handle multi-node failure.

namely,  $E2$ ,  $E4$ ,  $E5$ ,  $S1$ , and  $S2$ , to individually detect the failure of  $E1$ , e.g., due to missing heartbeat messages. Each of these one-hop neighbors will apply DARA and conclude whether  $E1$  is critical and which node should move to replace it. The selection criteria favor the neighbor with the least node degree, then the least proximity. Based on the topology, nodes  $E4$  and  $E5$  have the least degree, out of which  $E5$  is the best candidate since it is closer to  $E1$  than  $E4$ .

We have implemented the topology in Figure 3. A RPI-4 was attached to an iRobot<sup>®</sup> Create to form a mobile physical node for mimicking  $E1$ ,  $E2$ ,  $E3$ ,  $E4$ , and  $E5$ . The DARA algorithm is implemented on the RPI-4.  $N1$  and  $N2$  have been implemented as two instances of ns-3 on two different hosts; we also tested running both simulations on the same host. To enable recovery, we created a docker link between each of  $E2$ ,  $E4$ , and  $E5$ , and  $S1$ , and  $S2$ ; yet we kept these docker links inactive and allowed them to be activated depending on the recovery process.

First, we considered the baseline scenario of mimicking the loss of connectivity from its neighbors to  $E1$  by turning RPI-4 off (or killing the python script running the experiment code on  $E1$ 's RPI-4). Such an experiment reflects when all goes as anticipated by DARA and the network indeed could successfully recover from the failure. We further ran experiments to factor in:

- (1) Presence of an impassable obstacle in the path of  $E5$  towards  $E1$ ,
- (2) Slow motion of  $E5$  (by powering the iRobot<sup>®</sup> on and off),
- (3) Immediate failure of any node that is present at the position of  $E1$  (by turning the corresponding RPI-4 off),
- (4) Making the failed node part of cycle (adding a link between  $E5$  and  $S1$ ), and
- (5) Failure of both  $E1$  and  $E2$  simultaneously, i.e., powering the corresponding RPI-4 off.

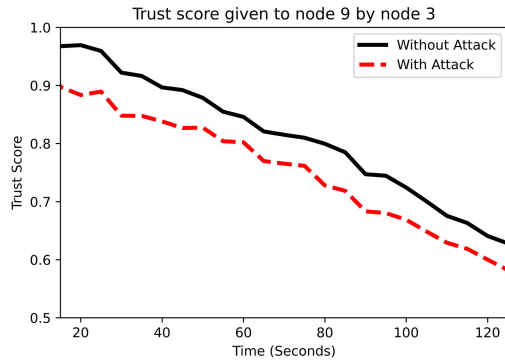


**FIGURE 4.** ns-3 simulated networks run in the laptops labelled as  $S1$  and  $S2$ . The Raspberry-Pi nodes labeled  $E1$  through  $E5$  are connected in an ad-hoc manner to form the topology shown in Figure 3. The origin is marked as  $(0, 0)$  at the top left corner. They execute the DARA-1C [43] algorithm to detect failures and decide collectively on a recovery plan. Each floor tile is used as a unit distance. The python script, used to control the iRobot<sup>®</sup>, is calibrated to cover the distances per unit time according to this setting.

Implementation of each scenario in the test-bed, observations and conclusions are summarized in Table 2. Figure 4 contains a snapshot of our implementation of the testbed in the lab. In summary, using our framework for testing these example scenarios points out practical issues beyond that insight that a network simulator would provide.

#### D. ATTACK SCENARIOS

In this subsection we show the utility of our framework through implementation of 3 different cyberattacks on the testbed. We demonstrate an attack on a routing protocol (AODV), then experiment with impersonation and collusion attack, and finally study DDoS attacks using our framework.



**FIGURE 5.** Trust score for an internal node with and without the impersonation attack, launched by an external attacker using an RPI-4 node.

### 1) ATTACKING THE TRUST SCORES

Eq. 1 defines how each node assesses the trustworthiness of their neighbours. AODV uses *Route Request* and *Route Reply* packets to find the best routes from a given node to the destination. Route Requests are broadcast packets that originate from the data source to everyone in the network. Only the destination node responds to these queries. According to Eq 1, if a node X forwards/sends much more Route Requests (RREQ) than it receives Route Replies (RREP) from its neighbour Y, then the trust score of node-Y from node-X's perspective does *decrease*. Knowing this fact, an external (or internal) attacker might spoof RREQ packets as if they are sent by the victim node, and broadcast these packets to the whole ns-3 network. The neighbours of the victim node, which are calculating trust for it, would eventually compute lesser scores for the target, resulting in less traffic being routed through it.

We implemented this attack in our framework using the Python's Scapy library [50] and recorded legitimate RREQs in the ns-3 simulation. Then, inside Scapy we modified the IP header's source field with the victim's IP, and sent this packet as a broadcast from the external attacker's RPI node to the internal ns-3 network. After the attack, all the neighbouring nodes to the victim had decreased trust scores for it, as could be seen in Figure 5. In this figure, the victim node was node #9, and node #3 was its neighbour. The black line shows the AODV trust score without any attack, and the red (dotted) line shows the decreased trust score after the external attacker sends spoofed RREQs impersonating the victim. The Scapy script's pseudo-code is given in Algorithm 1. We note that our framework facilitates the realization of such an attack where different strategies could be pursued by the adversary (physical node) to gauge the robustness of the network's defense. In fact, the attacker could be part of a red team who is not involved in the network design, to better assess resilience.

### 2) IMPERSONATION AND COLLUSIVE ATTACKS

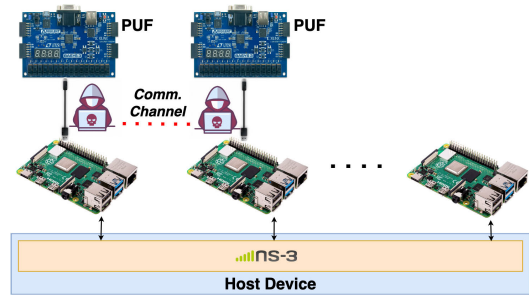
Our proposed framework would be most useful in scenarios where the attack is conducted by independent actors rather than nodes within the simulation. To demonstrate such capability, we have implemented an example scenario involving

### Algorithm 1 Attacker Sending AODV Packets to Decrease Trust Score of a Target ns-3 Node

```

Require:  $target\_ip \leftarrow argv[1]$   $\triangleright$  victim ns-3 node
Require:  $dest\_ip \leftarrow argv[2]$   $\triangleright$  broadcast to ns-3 subnet
Require:  $iface \leftarrow argv[3]$   $\triangleright$  interface on attacker's docker
Require:  $total\_pkts \leftarrow argv[4]$ 
Require:  $interval \leftarrow argv[5]$   $\triangleright$  seconds btw pkts
Ensure:  $len(argv) = 5$ 
     $spoofed\_pkt \leftarrow read("original.pcap")$ 
     $spoofed\_pkt.IP.src \leftarrow target$ 
     $spoofed\_pkt.IP.dst \leftarrow dest$ 
     $pkt\_counter \leftarrow 0$ 
    while  $pkt\_counter \leq total\_pkts$  do
         $send(spoofed\_pkt, iface)$ 
         $pkt\_counter \leftarrow pkt\_counter + 1$ 
         $sleep(interval)$ 
    end while

```



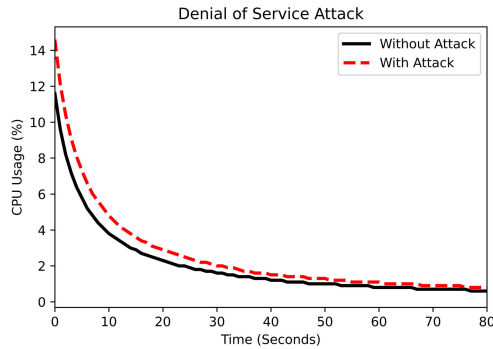
**FIGURE 6.** Collaborative malicious nodes armed with FPGA-based PUFs. The nodes mutually authenticate each other through PUFs prior to unleashing DDoS attacks towards internal ns-3 nodes. ns-3 environment is marked in orange within the host device, and host is depicted in light blue.

hardware security primitives. As shown in Figure 6, we have integrated three physical nodes two of which incorporate hardware-based fingerprints, specifically using a Physically Unclonable Functions (PUF). The PUF is implemented using a Field-Programmable-Gate-Array (FPGA) that is interfaced with the RPI-4 board. In this scenario we tested impersonation attacks launched by the third board that does not have a PUF. We also implemented the aforementioned trust score attack by two of the physical nodes in a coordinated manner, which mimics the case of coordinated cyberattacks by collusive actors. Clearly such a capability cannot be implemented using ns-3 and enable better dependability assessment since the external node behavior could be adjusted overtime to capture the effect of adaptive attack strategies.

### 3) LARGE SCALE DISTRIBUTED DENIAL OF SERVICE (DDOS)

When studying DDoS through simulation, an internal ns-3 node(s) would generate the packets which could cause the ns-3 itself to crash. By incorporating physical nodes in the process, such an attack can be effectively evaluated. In this experiment, we wanted to overload the network via sending a large volume of spoofed AODV RREQ packets. We used 2 RPI-4 devices, each targeting a specific internal ns-3 node.





**FIGURE 7.** Distributed-DoS attack, flooding the n-3 nodes with AODV RREQs, sourced from 2 external attacker controlled RPIs. CPU usage of ns-3 process in time, with & without the attack.

They sent the maximum number of packets possible based on the bandwidth of the connection, specifically, 350 packets per second per device. Each RREQ packet is 88 bytes, which adds up to 26.4 MBytes/second throughput for the DoS attack. We have also tracked the CPU usage of the ns-3 process through *ps* utility in Ubuntu. As can be seen in Figure 7, we were able to launch major DDoS without pushing CPU usage up much, where only little leap is observed compared to the same simulation without an attack. The DDoS attack scope can be further increased by:

- (i) Involving significantly more than 2 external physical attacker nodes,
- (ii) Sending RREQ packets with unknown destination address in AODV header, and maximum IP Time-to-Live to keep the attack packets traversing the network for maximum amount of hops,
- (iii) Generating RREQ packets carrying payload with as many bytes as possible for each packet, e.g., with maximum transmission unit (MTU) which equals to 1518 bytes in ns-3, or
- (iv) Appending irrelevant bytes at the AODV layer to increase the processing time per packet at each ns-3 node.

With these modifications, a set of external attackers can overwhelm the network without crashing ns-3, a scenario that is not possible without our hybrid validation platform. One could easily test DDoS detection algorithms utilizing our testbed.

## VI. CONCLUSION AND FUTURE WORK

Existing validation frameworks for large networked systems often pursue simulation to support scalability and rely on modeling nodes at various levels of abstraction. This paper has argued about the value of incorporating physical nodes when dependability attributes are of interest; something that existing frameworks fall short of supporting. To fill the technical gap, this paper has presented a flexible, modular, extendable framework for assessing dependability attributes of a networked system. Our framework is based on ns-3 and seamlessly interconnects virtual and physical nodes in the network. The framework allows a variety of valuable

experiments for testing networking protocols and studying the effect of cyberattacks in a cost efficient way. The utility of the proposed framework has been demonstrated through a variety of use cases. Future extensions include supporting fault-tolerance through redundancy management at a combined physical and virtual levels, and facilitating the configuration of networking rules needed for communication using Docker and ghost nodes in ns-3.

## APPENDIX A

### RECOVERY EXPERIMENT VIDEOS

Recorded videos for various experiments on the physical test-bed discussed in Section V-C can be watched at this link: <https://www.youtube.com/watch?v=...>

## APPENDIX B

### RECOVERY EXPERIMENT CODE

Implementation of various experiments on the physical test-bed discussed in Section V-C can be reached from here: <https://github.com/...>

## REFERENCES

- [1] R. Walczak, K. Koszewski, R. Olszewski, K. Ejsmont, and A. Kálmán, "Acceptance of IoT edge-computing-based sensors in smart cities for universal design purposes," *Energies*, vol. 16, no. 3, p. 1024, Jan. 2023. [Online]. Available: <https://www.mdpi.com/1996-1073/16/3/1024>
- [2] Y. Inagaki and A. Nakao, "Multi-layer edge computing for cooperative driving control optimization in smart cities," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2023, pp. 1–8.
- [3] P. Pandiyan, S. Saravanan, K. Usha, R. Kannadasan, M. H. Alsharif, and M.-K. Kim, "Technological advancements toward smart energy management in smart cities," *Energy Rep.*, vol. 10, pp. 648–677, Nov. 2023. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2352484723010995>
- [4] Q. Xin, M. Alazab, V. G. Díaz, C. E. Montenegro-Marin, and R. G. Crespo, "A deep learning architecture for power management in smart cities," *Energy Rep.*, vol. 8, pp. 1568–1577, Nov. 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S235248472101492X>
- [5] *ns-3 Tutorial*. Accessed: Aug. 1, 2023. [Online]. Available: <https://www.nsnam.org/docs/tutorial/html/>
- [6] K. Whitehouse, C. Karlof, A. Woo, F. Jiang, and D. Culler, "The effects of ranging noise on multihop localization: An empirical study," in *Proc. 4th Int. Symp. Inf. Process. Sensor Netw. (IPSN)*, 2005, pp. 73–80.
- [7] B. Dezfouli, M. Radi, K. Whitehouse, S. A. Razak, and H.-P. Tan, "CAMA: Efficient modeling of the capture effect for low-power wireless networks," *ACM Trans. Sensor Netw.*, vol. 11, no. 1, pp. 1–43, Aug. 2014.
- [8] T. Zhang, S. Zhao, B. Cheng, B. Ren, and J. Chen, "FEP: High fidelity experiment platform for mobile networks," *IEEE Access*, vol. 6, pp. 3858–3871, 2018.
- [9] D. Villa, M. Tehrani-Moayyed, C. Paul Robinson, L. Bonati, P. Johari, M. Polese, S. Basagni, and T. Melodia, "Colosseum as a digital twin: Bridging real-world experimentation and wireless network emulation," 2023, *arXiv:2303.17063*.
- [10] F. Nemtanu, I. Moise, and C. Banica, "Hybrid simulation in intelligent transportation systems," in *Proc. 36th Int. Spring Seminar Electron. Technol. (ISSE)*, 2013, pp. 106–107.
- [11] M. A. To, M. Cano, and P. Biba, "DOCKEMU—A network emulation tool," in *Proc. IEEE 29th Int. Conf. Adv. Inf. Netw. Appl. Workshops*, Gwangju, South Korea, Mar. 2015, pp. 593–598, doi: [10.1109/WAINA.2015.107](https://doi.org/10.1109/WAINA.2015.107).
- [12] E. Petersen, G. Cotto, and M. A. To, "Dockemu 2.0: Evolution of a network emulation tool," in *Proc. IEEE 39th Central Amer. Panama Conv. (CONCAPAN XXXIX)*, Panama, Nov. 2019, pp. 1–6.
- [13] K. Maeda, T. Umedu, H. Yamaguchi, K. Yasumoto, and T. H. Higashino, "MobiREAL: Scenario generation and toolset for MANET simulation with realistic node mobility," in *Proc. 7th Int. Conf. Mobile Data Manage. (MDM)*, 2006, p. 55.

- [14] G. F. Riley, "Large-scale network simulations with GTNetS," in *Proc. Winter Simulation Conf.*, vol. 1, 2003, pp. 676–684.
- [15] M. Pužar and T. Plagemann, "NEMAN: A network emulator for mobile ad-hoc networks," in *Proc. 8th Int. Conf. Telecommun. (ConTEL)*, vol. 1, 2005, pp. 155–161.
- [16] J. Zhang and Z. Qin, "TapRouter: An emulating framework to run real applications on simulated mobile ad hoc network," in *Proc. 44th Annu. Simulation Symp.*, San Diego, CA, USA, 2011, pp. 39–46.
- [17] R. Barr, Z. J. Haas, and R. van Renesse, "JiST: An efficient approach to simulation using virtual machines," *Softw., Pract. Exper.*, vol. 35, no. 6, pp. 539–576, 2005.
- [18] J. D. Britos, S. E. Arias, N. Echániz, G. Iribarren, L. Aimaretto, and G. Hirschfeld, "BATMAN adv. Mesh network emulator," in *Proc. 21st Congreso Argentino de Ciencias de la Computación (Junin)*, 2015, pp. 3–7.
- [19] M. Pizzonia and M. Rimondini, "Netkit: Easy emulation of complex networks on inexpensive hardware," in *Proc. 4th Int. ICST Conf. Testbeds Res. Infrastruct. Develop. Netw. Communities (TRIDENTCOM)*, May 2010, pp. 1–3.
- [20] H. K. Kalitay and M. K. Nambiar, "Designing WANem: A wide area network emulator tool," in *Proc. 3rd Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2011, pp. 1–4.
- [21] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim, "CORE: A real-time network emulator," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Nov. 2008, pp. 1–7.
- [22] A. Attaby, N. Osman, M. Elnainay, and M. Youssef, "Wipi: A low-cost large-scale remotely-accessible network testbed," *IEEE Access*, vol. 7, pp. 167795–167814, 2019.
- [23] I. Abdullah and W. Monnet, "Emulation framework for haptic data transmission using real-time transport protocol," *Int. J. Online Biomed. Eng.*, vol. 19, no. 7, pp. 142–159, Jun. 2023.
- [24] V. S. Hapanchak and A. D. Costa, "Emulation of multi-connectivity in hybrid vehicular networks," in *Proc. Int. Conf. Electr., Comput. Energy Technol. (ICECET)*, Jul. 2022, pp. 1–6.
- [25] Z. Y. Song, M. Mostafizur, R. Mozumdar, M. Tranchero, L. Lavagno, R. Tomasi, and S. Olivieri, "Hy-sim: Model based hybrid simulation framework for WSN application development," in *Proc. 3rd Int. ICST Conf. Simulation Tools Techn.*, 2012, pp. 3–6.
- [26] H. Moudni, M. Er-Rouidi, H. Mouncif, and B. El Hadadi, "Attacks against AODV routing protocol in mobile ad-hoc networks," in *Proc. 13th Int. Conf. Comput. Graph., Imag. Vis. (CGiV)*, Mar. 2016, pp. 385–389.
- [27] A. Bandyopadhyay, S. Vuppala, and P. Choudhury, "A simulation analysis of flooding attack in MANET using NS-3," in *Proc. 2nd Int. Conf. Wireless Commun., Veh. Technol., Inf. Theory Aerosp. Electron. Syst. Technol. (Wireless VITAE)*, Feb. 2011, pp. 1–5.
- [28] *OPNET Modeler*. Accessed: Aug. 1, 2023. [Online]. Available: <http://www.opnet.com/>
- [29] *QualNet Simulator*. Accessed: Aug. 1, 2023. [Online]. Available: <http://www.scalable-networks.com/products>
- [30] *OMNeT++ Simulator*. [Online]. Available: <http://www.omnetpp.org/>
- [31] *J-Sim Simulator*. Accessed: Aug. 1, 2023. [Online]. Available: <http://www.j-sim.org/>
- [32] W. Liu, X. Wang, W. Zhang, L. Yang, and C. Peng, "Coordinative simulation with SUMO and NS3 for vehicular ad hoc networks," in *Proc. 22nd Asia-Pacific Conf. Commun. (APCC)*, Aug. 2016, pp. 337–341.
- [33] S. Jain, P. Jain, P. K. Upadhyay, J. M. Moualeu, and A. Srivastava, "An energy efficient health monitoring approach with wireless body area networks," *ACM Trans. Comput. Healthcare*, vol. 3, no. 3, pp. 1–22, Apr. 2022.
- [34] J. Schaerer, Z. Zhao, J. Carrera, S. Zumburn, and T. Braun, "SDN Wisebed: A software-defined WSN testbed," in *Proc. 18th Int. Conf. Ad-Hoc Netw. Wireless (ADHOC-NOW)*, Luxembourg, 2019, pp. 317–329.
- [35] I. Ketata, S. Bdiri, N. Derbel, and F. Derbel, "Hardware based simulation platform for wireless sensor networks," in *Proc. 16th Int. Multi-Conf. Syst., Signals Devices (SSD)*, Mar. 2019, pp. 339–344.
- [36] F. A. Khalek, J. Nassar, V. Lefevre, F. Gosselin, and N. Gouvy, "A smart grid WSN research testbed," in *Proc. 16th Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Oct. 2020, pp. 388–391.
- [37] A. Pushpa, "Trust based secure routing in AODV routing protocol," in *Proc. IEEE Int. Conf. Internet Multimedia Services Archit. Appl. (IMSAA)*, Dec. 2009, pp. 1–6.
- [38] G. F. Riley and T. R. Henderson, *The ns-3 Network Simulator*. Berlin, Germany: Springer, 2010, pp. 15–34.
- [39] D. Merkel, "Docker: Lightweight Linux containers for consistent development and deployment," *Linux J.*, vol. 2014, no. 239, pp. 1–3, Mar. 2014.
- [40] C. Benvenuti, *Understanding Linux Network Internals*. Springfield, MO, USA: O'Reilly, 2006.
- [41] C. Perkins, E. Belding-Royer, and S. Das, *Ad Hoc On-Demand Distance Vector (AODV) Routing*, document RFC 3561, USA, 2003.
- [42] (2010). *Howto Make ns-3 Interact With the Real World*. [Online]. Available: [https://www.nsnam.org/wiki/HOWTO\\_make\\_ns-3\\_interact\\_with\\_the\\_real\\_world](https://www.nsnam.org/wiki/HOWTO_make_ns-3_interact_with_the_real_world)
- [43] A. A. Abbasi, M. Younis, and K. Akkaya, "Movement-assisted connectivity restoration in wireless sensor and actor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 9, pp. 1366–1379, Sep. 2009.
- [44] J. Niroshan and T. Pecorella. (2018). *Trust-Based Routing Protocols Framework*. [Online]. Available: [https://github.com/JudeNiroshan/ns-3-dev-git/tree/gsoc2018\\_final](https://github.com/JudeNiroshan/ns-3-dev-git/tree/gsoc2018_final)
- [45] M. Younis, S. Lee, and A. A. Abbasi, "A localized algorithm for restoring internode connectivity in networks of moveable sensors," *IEEE Trans. Comput.*, vol. 59, no. 12, pp. 1669–1682, Dec. 2010.
- [46] U. Baroudi, M. Aldarwbi, and M. Younis, "Energy-aware connectivity restoration mechanism for cyber-physical systems of networked sensors and robots," *IEEE Syst. J.*, vol. 14, no. 3, pp. 3093–3104, Sep. 2020.
- [47] W. Lalouani, M. Younis, and N. Badache, "Optimized repair of a partitioned network topology," *Comput. Netw.*, vol. 128, pp. 63–77, Dec. 2017.
- [48] M. Younis, I. F. Senturk, K. Akkaya, S. Lee, and F. Senel, "Topology management techniques for tolerating node failures in wireless sensor networks: A survey," *Comput. Netw.*, vol. 58, pp. 254–283, Jan. 2014.
- [49] S. Lee, M. Younis, M. Alsolami, and M. Lee, "LOAF: Load and resource aware federation of multiple sensor sub-networks," *IEEE Access*, vol. 8, pp. 179466–179485, 2020.
- [50] P. Biondi and The Scapy Community. *Scapy—A Powerful Interactive Packet Manipulation Library*. Accessed: Jul. 27, 2023. [Online]. Available: <https://scapy.net/>



**SUHEE SANJANA MEHJABIN** (Graduate Student Member, IEEE) received the B.Sc. degree in electrical and electronic engineering from the Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh, in 2021. She is currently pursuing the Ph.D. degree in computer engineering with the Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, MD, USA. She is a member of the Embedded Systems

and Networks Laboratory (ESNet), where her research focuses on cyber-physical security and secure and trusted communication with hardware in the loop.



**ALI TEKEOGLU** (Member, IEEE) received the B.Sc. degree in computer engineering from Bilkent University, Ankara, Turkey, in 2007, and the M.Sc. and Ph.D. degrees in computer science from The University of Texas at San Antonio. He was a Faculty Member with The State University of New York Polytechnic Institute and University of New Brunswick, Canada, while collaborating with Air Force Research Laboratory (AFRL) and Canadian Institute for Cybersecurity (CIC). He was a Cybersecurity Researcher with the Johns Hopkins University Applied Physics Laboratory, Critical Infrastructure Protection Group. He is currently with the Leidos Innovations Center (LIInC) as a Cyber/AI Research Scientist. His research interests include the security of cyber-physical systems, critical infrastructure, and the application of machine learning to cybersecurity problems.





**MOHAMED YOUNIS** (Fellow, IEEE) is currently a Professor with the Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County (UMBC). Before joining UMBC, he was with the Advanced Systems Technology Group, an Aerospace Electronic Systems Research and Development Organization of Honeywell International Inc. While at Honeywell, he led multiple projects for building integrated fault tolerant avionics and dependable computing infrastructure. He also participated in the development of the Redundancy Management System, which is a key component of the Vehicle and Mission Computer for NASA's X-33 space launch vehicle. His research interests include network architectures and protocols, the Internet of Things, fault tolerant computing, secure communication, and cyber-physical systems. He has published over 350 technical papers in refereed conferences and journals. He has five granted and two pending patents. In addition, he serves/served on the editorial board of multiple journals and the organizing and technical program committees of numerous conferences. He is a fellow of the IEEE Communications Society.



**TAMIM SOOKOOR** (Member, IEEE) is currently a Researcher with the Johns Hopkins University Applied Physics Laboratory (JHU/APL), where his research interests include cyber-physical systems, cyber-security, the Internet of Things, and machine learning. He is leading the development of new technologies at the frontier of secure CPS and resilient smart cities. He teaches graduate courses in JHU's Engineering for Professionals programs, where he is the Coordinator of the assured autonomy track. He is a member of ACM and the Secretary for the Baltimore Area Chapter of ACM.



**MOHAMMAD EBRAHIMABADI** (Graduate Student Member, IEEE) received the B.Sc. degree in electrical engineering from Zanjan University, Iran, in 2008, and the M.Sc. degree in electrical engineering from the Sharif University of Technology, Iran, in 2011. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, University of Maryland at Baltimore County (UMBC), USA. He is a member of the Secure, Reliable and Trusted Systems Research Laboratory, UMBC. His current research interests include hardware security, and in particular side-channel analysis and fault injection attacks and countermeasures, sensor-assisted secure and reliable design, and hardware-assisted authentication and secure communication protocols. He has published 25 papers in refereed conference proceedings and archival journals.



**RAHUL CHANDRAN** received the B.S. degree in computer science from The Pennsylvania State University, in 2020, and the M.S. degree in computer science from the University of Pennsylvania, in 2022. He is currently with the Johns Hopkins University Applied Physics Laboratory as a Software Engineer on the critical infrastructure protection team. He mainly works on software related to PLCs and SCADA systems.



**NAGHMEH KARIMI** (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in computer engineering from the University of Tehran, Iran, in 2002, and 2010, respectively. She is currently an Associate Professor with the Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County (UMBC). She was a Visiting Researcher with Yale University (2007–2009), a Postdoctoral Researcher with Duke University (2011–2012), and a Visiting Assistant Professor with New York University (2012–2014) and Rutgers University (2014–2016). She joined UMBC, in 2017, where she leads the SECure, Reliable and Trusted Systems (SECRETS) Research Laboratory. She has published three book chapters and over 80 papers in refereed conference proceedings and journal manuscripts. Her current research interests include hardware security, VLSI testing, design-for-trust, design-for-testability, and design-for-reliability. She is a recipient of the National Science Foundation CAREER Award, in 2020. She serves as an Associate Editor for the *Journal of Electronic Testing: Theory and Applications* (JETTA) (Springer) and IEEE DESIGN AND TEST journal.

...