Impact of Flooding Attack on Wireless Sensor Networks: Framework and Toolkit

Rakan AlGhofaili , Hussah Albinali , Farag Ahmed Azzedin *

Posted Date: 18 December 2023

doi: 10.20944/preprints202312.1356.v1

Keywords: RPL, Tool, Flooding attack, DoS, WSN, IoT

Article

Impact of Flooding Attack on Wireless Sensor Networks: Framework and Toolkit

Rakan Alghofaili [1,2], Hussah Albinali [1,3] and Farag Azzedin [1,4,*]

[1] Information & Computer Science Department, College of Computing & Mathematics King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia; g200747850@kfupm.edu.sa; g201906710@kfupm.edu.sa
[2] Technical Services, Saudi Aramco, Dhahran 31311, Saudi Arabia; rakan.ghofaily@aramco.com
[3] Networks and Communication Department, College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, Dammam 32214, Saudi Arabia; halbinali@iau.edu.sa
[4] Interdisciplinary Research Center for Intelligent Secure Systems, King Fahd University of Petroleum and Minerals
* Correspondence: fazzedin@kfupm.edu.sa

Abstract: IoTs and WSNs utilize their connectivity to enable solutions supporting a spectrum of industries in different and volatile environments. To effectively enhance the security and quality of service of networks, empirical research should take into account a variety of factors and be structured in a replicable manner. This will not only ensure scalability but also enable verification of conclusions, leading to more constructive outcomes. Cooja offers limited performance analysis capabilities of simulations, which are often extracted and calculated manually. In this paper, we introduce a framework and a toolkit that enable researchers to conduct structured and conclusive experiments considering different factors and metrics, experiment design to results analysis. Besides, this tool summarizes multiple network metrics and allows multiple scenarios. As proof of concept, this paper studies the flooding attacks on IoT and illustrates their impact on the network, leveraging the proposed framework and tools.

Keywords: RPL; framework; tool; flooding; cybersecurity; DoS; WSN; IoT

1. Introduction

Wireless Sensor Network (WSN) is a network consisting of sensing devices of different sizes, computational, and sensing abilities. These sensors collaborate to sense, collect, and process raw information in the sensing area and transmit the processed information to the observers [1]. Internet of Things (IoT) refers to a network of physical devices, vehicles, and appliances embedded with sensors, software, and network connectivity. These smart devices have the ability to collect and share data and communicate with each other and with other internet-enabled devices such as smartphones and gateways [2,3]. The Internet of Things (IoT) is considered the foundation for the fourth Industrial Revolution (IR 4.0) [4].

Protecting WSNs and IoTs from security threats and attacks is one of the primary research areas. Performing impact analysis enables identifying vulnerabilities and developing effective solutions. As a matter of fact, IoT devices are candidate targets of cyber attacks, including Zero-day [5–7] and Denial of Service (DoS) attacks. DoS attacks aim of bring the service provided by the network to a halt [3]. An example of DoS attack is a flooding attack, where a malicious node in the network generates a large number of packets that disrupts its neighbors' ability to process further packets, rendering a portion of the network unavailable.

WSNs are particularly vulnerable to such attacks due to their inherent limitations and protocol implementations. Accordingly, the performance of these networks and their resilience against threat actors are continuously evaluated by scholars. Authors in [8–12] are examples of studies that include studying the impact of certain attacks (including flooding) on IoT, as well as the overhead associated with deploying mitigation for these attacks. To maximize the potential of WSNs and IoTs, and to

 doi:10.20944/preprints202312.1356.v1

ensure their reliability and performance; it is vital to consider carefully and tailor network topologies and physical specifications to the unique requirements of each scenario.

Empirical studies require the ability to be replicated in order to verify findings and confirm conclusions. In WSN context, replicating experiments is necessary to reflect the impact of any change in the network configuration, report different behaviors, and, consequentially, draw different conclusions [13]. Therefore, manual placement and configuration for a relatively large number of sensor nodes might be tedious, time-consuming, and lacking accuracy [14]. For instance, Cooja, the simulation tool provided by Contiki, offers limited choices of automatic placement (either linear or ellipse) besides manual and random placement. In addition, Cooja doesn't provide network performance metrics for the entire network, which need to be extracted and calculated. Therefore, frameworks are fundamental to allow different research studies in a specific field to converge in terms of findings and through building on shared foundations.

It is crucial to analyze the network under various attacker positions, network topologies, and software/hardware implementations. Random configurations and placements may be utilized in experiments, but they have limited generalization, are difficult to replicate, and may be biased. If not controlled in a research experiment, these elements pose as a potential confounding factor that hider the conclusiveness and generalization of findings. Accordingly, this paper proposes a framework including systems and methods that provide a controlled environment under predefined parameters and allow for replicable research, and evaluate different hypotheses.

This framework can be generalized to allow for faster replication of empirical studies on further attacks on IoT and improve the quality of published research in the field. To demonstrate the effectiveness of our framework and tools, we utilize them to analyze flooding attacks on WSNs and their propagation and impact on different network topologies. The rest of the article is organized as follows. For clarity and completeness purposes, Section 2 covers related research on frameworks and tools related to WSN and IoT experiments, as well as works in the literature that studied the impact Routing Protocol for Low-Power and Lossy Networks (RPL) attacks under different network topologies and attacker positions. In Section 3, we present our proposed framework for conducting replicable experiments in a structured approach, including the tools that would help at each stage. As a demonstration of the effectiveness of utilizing this framework, Section 4 provides an empirical analysis and evaluation of the impact of flooding attacks on a WSN considering different topologies. Finally, Section 5 concludes the manuscript and envisions new directions.

2. Related Work

In this section, we'll delve into studies that consider different network topologies when they analyze the impact of RPL attacks as the proposed case study highlights the flooding attacks under different topologies. We will also highlight various frameworks designed for the purpose of extracting and managing data from wireless sensor networks.

2.1. Performance Analysis considering Attack Scenarios, Placements and Network Layouts

The literature includes wealth of studies on the effect of different treatments and attacks on WSNs. Unfortunately, few of which considered a structured approach and limited potential confounding factors to have more comprehensive insights. For example, the study presented in [11] drew conclusions on the differences between blackhole and grayhole attacks based on a network of random distribution, without considering the effect of other factors. Ramya and Vamsi [15] conducted an experiment to study the black hole attack in various network sizes and attacker placements. They found that the impact of the attack on packet delivery was inversely proportional to the number of nodes and directly proportional to the number of attackers. Several studies have investigated the impact of various attacks on different topologies. For instance, [9] performed an evaluation study of rank attacks on a grid topology network. The adversary nodes were dispersed at random over the grid in each multiple-attacker scenario. The analysis takes into account how the assault affected

 doi:10.20944/preprints202312.1356.v1

various network nodes. The findings show that an attack may have a significant negative effect on the network's performance, particularly if it is carried out in an area with a high forwarding load or involves several attackers [9]. Version attack has been studied by multiple studies to investigate the topology impact on the attack. Authors in[16,17] studied version attack detection in the context of cluster-based, random, and grid topologies. Their findings show that the scalability of their detection solution was reduced in randomly generated topologies compared to the grid topology. However, more realistic cluster-based topologies exhibit performance that is comparable to grid topologies. Moreover, authors in [18] employed grid and random node placement techniques in their work to study how the performance of the mitigation changes with the topological properties. Under version attack, the authors noted that the amount of control messages approximately doubled for grid corner network, and it tripled for random network [18]. The rationale being that a grid topology had a more consistent node densities than random placement, which resulted in topologies with varying node densities. Due to having longer links than others, grid topology showed the highest average power consumption values for the attack-free condition [18].
Hachemi et al. [19] conducted a study to investigate the impact of sinkhole attacks on a network consisting of 10 nodes in a tree layout with one attacker. The study was conducted in three different scenarios, and the authors observed that the DIO messages within the network increased significantly as the attacker was placed further in the network and closer to the sink node. Although the authors did not measure the quality of service (QoS) metrics of the network, their findings suggest that the network became less stable, which is evident in the frequent DODAG formations. This study provides valuable insights into the potential vulnerabilities of networks to sinkhole attacks and highlights the need for further research to develop effective countermeasures to mitigate such attacks.

In a study conducted by Anthuan Le et al. [20], researchers investigated the impact of rank attacks on wireless sensor networks (WSN). The findings revealed that such attacks can considerably reduce the packets delivery ratio and increase the end-to-end delay. The study also highlighted that WSN topology may face disruptions and bottlenecks, even when no actively malicious nodes are present to drop or delay packets. This can be particularly concerning in areas with high activity where malicious nodes could be located [20].

Rai and Asawa [21] studied the implications of a reduced rank attack on a simulated network in a grid pattern. Although the authors stated that they assessed the network with different attack placements around the sink, they did not explore the variations in the aftermath. Nonetheless, their observations show that the attack had a more significant impact when placed in an active network area [21].

Alternatively, in [22], the study tackled node reset attack, which includes both local repair and version attacks under a binary tree and mesh topologies. In the binary tree topology, each node only has one way to the sink. As a result, removing of each node causes the graph to split. In the mesh topology, each node has many nearby nodes, and the removal of any node will not lead to a partition of the graph. The results in [22] illustrate that mesh topology increases overall energy consumption but marginally less than that of binary tree topology. Due to the mesh topology, when a node reset happens, some nodes may already be utilizing alternate paths. Affected nodes might adapt their routes to avoid missing nodes. To the best of our knowledge, version, local repair, and rank attacks are the only attacks have been studied under different topologies. PyFUNS [23] is a framework that allows for rapid development of WSN-based applications through Python and CoAP APIs without requiring specialized expertise in embedded systems development. It abstracts networking and native codes, allowing the developer to focus more on the application development, [23]. When evaluating the framework's performance, it is noted that authors considered evaluating it in different topologies (star, tree and mesh), running different applications, and different placements. Moreover, authors in [24] introduced a multi-protocol Software-Defined Networking platform for IoT called MINOS. This platform utilizes appropriate interfaces for centralized network control of diverse and resource-constraint IoT environments. In addition, the introduced graphical user interface provides a

 doi:10.20944/preprints202312.1356.v1

bespoke dashboard and a real-time visualization tool. The main focus of this tool is handling mobility and heterogeneity in the experimentation setup. The calculated metric in this platform is PDR and control overhead.

2.2. Approaches and Tools for Analyzing WSNs Performance

Several researchers developed tools to replicate nodes' places and types systematically in RPL networks. Besides, other tools have been designed to collect network performance parameters.

Authors in [25] introduced Multi-Trace, which is an extension to the Cooja simulator that offers multi-level tracing capabilities. These capabilities allow for data logging at varying levels while keeping track of a collective time. The proposed system also includes customized scripts to expand a simulation into multiple ones with different ones with varying sizes, distributions, and logical implementations. Since all generated simulations are reflecting the same scenario, implying that all simulations are following the same timeline, the generated logs and results are can be tracked through a global timestamp. However, this tool does not provide additional abilities to select other topologies for placing the network's nodes.

Additionally, authors in [26] introduced ASSET – an IDS for RPL that addresses 13 different types of attacks, such as blackhole, flooding, replay, and rank attacks. The system uses diverse profiles to combat these issues. The application plane offers a user-friendly interface for real-time visualization and monitoring of the IoT topology. It also identifies potential IoT nodes that may act as attackers. It's important to note that this tool doesn't automatically place network nodes, and the results are presented at the node level rather than the network level. Besides, authors in [27] paper presented a software built on top of Cooja called ViTool-BC, which allows a real time visualization of the network construction and connection behavior. In addition, ViTool-BC offers a heat map of energy consumption traces and battery depletion. Therefore, this tool helps researchers monitor and analysis of the available routing protocols in Cooja. It's important to note that this tool doesn't provide the capability to replicate nodes' placement in the network. Also, other metrics like PDR and E2E delay are not visually presented in a heatmap. Also, it lacks consolidation of network's performance results.

3. Framework Overview

In this section, we introduce a framework that enables researchers to conduct replicable experiments under predefined conditions. The framework provides a systematic approach that allows researchers to conduct conclusive experiments comparing multiple scenarios, different network layouts and shuffled node placements. In the case of studying the impact of malicious attacks on WSNs, a reasearcher can leverage this framework to conduct structured experiments comparing different attack secenarios, on different network topologies and different attacker placements. The attack scenarios may scale in breadth by covering different attacks, or in depth by considering multiple attackers carrying out the same attack. Figure 1 provides an illustration of considerations of this framework. Further, we propose set of tools that allow for conducting experiments according to this framework through precise positioning of network nodes in the network, identifying their types, and extract and summarize network metrics.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 December 2023 doi:10.20944/preprints202312.1356.v1

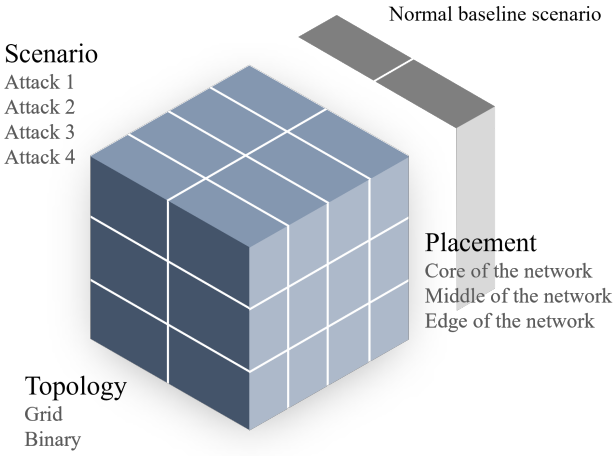

Figure 1. Visual representation of proposed framework.

Contiki provides a Java-based simulator called Cooja, which facilitates network debugging and performance analysis, including power tracing and profiling. However, each scenario needs to be configured manually by specifying nodes' types and their location, run separately and analyzed accordingly. The results collected during each simulation period provide detailed information about each node in the network. These results are often fragmented, limited and hard to consolidate either at a holistic level or at individual level. Analyzing both simultaneously are expected to provide further insights on how the overall network behave during experiments. The framework offers structure and tools that are built upon three main components as shown in Figure 2.

The **Builder** allows for flexible and precise generation of Cooja automated simulation files. It allows the definition of different classes of nodes, precise placement of each node to form consistent topologies with different scenarios. The **Launcher** allows running simulation files produced by the builder in masses, each with and properly labeling and storing output log files. The **Consolidator** consolidates all output files into one master worksheet leveraging Power Query. It analyzes and calculates metrics and statistics of each scenario. These metrics can be interpreted through tables and charts and can also be exported to CSV files. These statistics can also be used to generate heatmaps for further analysis of the network's state using the builder. we will discuss in detail how each component works, what are its inputs, and outputs.

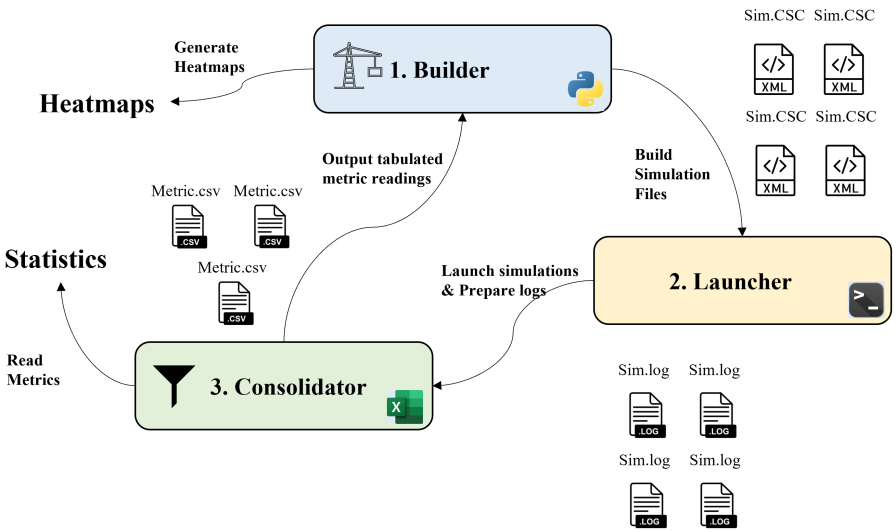

Figure 2. Process flow of interactions for proposed tools to execute and analyze simulations.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 December 2023 doi:10.20944/preprints202312.1356.v1

3.1. Builder

The builder facilitates the experiment setup by allowing the definition of different node/mote types and their position for the simulation to ensure providing customized and replicable experiments. To achieve that, the builder utilizes CSC files saved by Cooja simulation (which includes XML descriptors of the simulation environment). Each simulation file contains full configuration data such as required Cooja extensions, simulation parameters, and information about each mote such as type and position Figure 3. In addition, the simulation file includes pointers to Contiki process source files that run for each mote type, allowing the builder to build the simulation with the correct source files for each type of node.

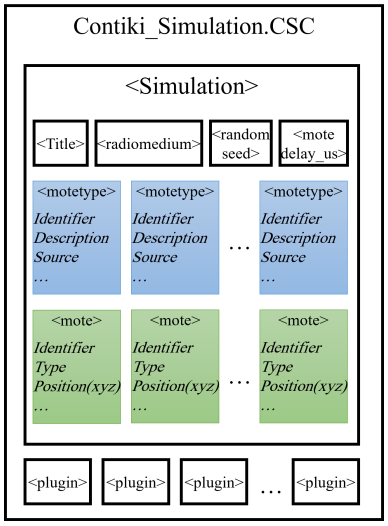

Figure 3. CSC XML structure highlighting fragments generated by our builder.

The CSC file also includes exact coordinates for each mote as it is placed in the simulation hyperplane. The builder tool can be used either to programmatically position each node, or to use pre-built methods to build two (2) classes of layouts which are: 1) binary layout, where each node would extend the network's coverage to a maximum of two additional nodes, and 2) grid layout where nodes are structured such that nodes are arranged in columns and rows.Figure 4 illustrates how the builder These layouts were implemented as default, but the builder can still be scaled to accommodate different layouts and topologies. Both layouts offer contradicting trade-offs as the binary layouts favors covering a larger area with finite set of devices, whereas grid maximizes redundancy as each node would have significantly more paths to the sink, at the cost of area covered. The aim is to obtain a more comprehensive insights of our experiment by expanding its scope to cover two contrasting layouts that may show different behavior and results.

Figure 4 shows how our builder determines the coordinates of the next nodes based on existing ones through the illustrative arrows. In our implementation, nodes in binary layout are characterized by having a maximum of one (1) potential parent. Distance between nodes (d) is fixed as the maximum distance for transmission range , and the angle where nodes branch is a constant $90\circ$. The builder tool is capable of generating such networks recursively. Each node branches with 2 new child nodes, noting the angle of the branch vector. Coordinates for child nodes created at each branch are calculated algebraically based on their parent's coordinates and the angle of the parents' branch vector. Before introducing a new node to the network, the pre-built graph is traversed to ensure that the node to be added is not close to any other node (besides its parent) to avoid cycles and enforce the binary layout rule. If another node is found in proximity, the new node is not added, leaving its parent with only one child.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 December 2023 doi:10.20944/preprints202312.1356.v1

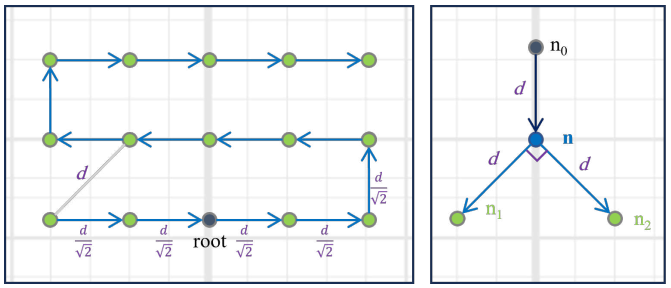

Figure 4. Branching approach as implemented by the builder for the grid layout (left) and the binary layout (right).

On the other hand, grid layout is characterized by its grid-like distribution. In our approach, central nodes will have up to eight neighbors and potential parents. Our builder can have the node at the edge of the layout or position it towards the center. Provided with the number of rows and columns, the builder will build the network by generating coordinates for each node until the network is fully built. It is worth noting that the distance is shorter among vertically and horizontally adjacent nodes compared to diagonally adjacent nodes. Since the distance between a node and its diagonal neighbors would be equal to d (the maximum distance), the builder uses basic trigonometry to calculate the distance between a node to its other neighbors which is $d/\sqrt{2}$. Both layouts can be adjusted programmatically with manually specified coordinates. Further, the user may use direct access to functions to allow branching at specific points, allowing for more hybrid topologies to be created and merged.

Further, the user can specify which nodes run which source file. Once all parameters are specified, the user can generate the XML excerpt for the simulation parameters, which can be inserted between $< simulation >$ tags on any existing Cooja simulation file. The user can use the builder to generate as many Cooja simulation files, each with its own parameters according to the experiment design. The simulation files can be run by the launcher to conduct experiments, and collect datapoints for analysis. The builder can be leveraged once more to build powerful heatmaps to analyze network's performance in each simulation, after once simulations are run and and data points are collected. It capitalizes on its prior knowledge of the positioning of nodes in the network, combined with the provided performance data collected ; the builder tool can generate a heatmap of the network for each collected metric. The heatmap serves as a powerful visual aid to spot the changes in the collected metrics.

3.2. Launcher

We designed a launcher, which is a bash script that allows the user to run simulations on Cooja while labeling output logs and packet captures for future use and analysis. Once the builder generates simulation files, the launcher runs each simulation independently without loading the Cooja GUI. Thus, the launcher offloads resources on the host machine and allows experiments to run more efficiently. This feature proved to be vital for sizable simulations consisting of a large population of nodes, as well as simulations where expensive processes are run (such as flooding attacks), which would overcome memory issues.

Specifically, the launcher is a bash script that interfaces with Cooja through commands. It requires the simulation name, in the format "$< simulation\_name > .csc$", as input. As the simulation concludes, it locates the newly generated packet capture and captured logs and properly labels and save them with the same name as the simulation. Both files are stored properly, ensuring that it's not overwritten in the next simulation. If the simulations are run in a virtual machine, it is considering to store the results in a shared folder between the guest and host machines to enable faster access and allow analyzing each scenario's results while the next simulation is running.

 doi:10.20944/preprints202312.1356.v1

3.3. Consolidator

The consolidator consolidates all output files into one master workbook leveraging Power Query. The consolidator relies on the logs generated by the simulations to calculate different metrics. This implies that the source code for network's nodes is programmed to output readings of desired metrics beforehand. Accordingly, log outputs include timestamps of these readings and events. Readings for each metric are collected by a specific query designed to parse for it. Readings for each metric are then stored separately to allow for further analysis. The consolidator pre-processes all the log files resulting from simulations and record individual messages by simulation, timestamp, node ID and the output message. Each message would be checked against a match of pre-defined signature of each metric.

Our existing implementation collects three popular metrics for measuring the performance of the network; namely Packet Delivery Ratio (PDR), End-to-End Delay (E2E), and Power Consumption (PC). The consolidator can be expanded to accommodate additional metrics with proper configuration. PDR is a key indicator for the quality of message transmission and reception, and by extension the availability of the service provided by the network. End-to-End Delay, on the other hand, looks at the degradation of the service resulting from delayed packets. Both measures rely on the nodes logging timestamped messages of sending and delivering data packets, which would be read by the consolidator for each metric. Increased power consumption is detrimental to the lifetime of the network, as it would quickly deplete the nodes' batteries. We estimated power consumption using PowerTrace [28] and ENERGEST module. Both are used to track - at the node level during run time - how long each hardware component has been on (e.g. cpu and transmitter). If we know the rate of energy consumption for each individual component when its used, then we can use both to calculate an estimated power consumed at a specific interval. These rates are often documented as part of the hardware specification for individual node types and can vary among different manufacturers and architectures. Table 1 illustrates the specification factors used for the Zolertia Z1 motes according to the datasheet [29]. Readings of power consumption rates are collected by reading logged messages by the nodes. The consolidator allows easy modification of these factors to scale to different hardware.

At the core of the consolidator, would be master lists that include all simulations and their average readings for all nodes for each metric. In addition, the consolidator includes one worksheet that includes holistic average readings for all metrics for all simulations, as a summary for experiments results as illustrated in Figure 5. The consolidator also outputs separate sheets (one for each metric) that include node-level average readings for each experiment. These can be exported to CSV files for further analysis. Since the layouts of these networks were already built using the aforementioned builder, the same tool can be fed with resulting CSV results to help generate visual heatmaps. For example, if an experiment included 6 simulations (1 for each combination of scenario-placement-topology) to collect 3 metrics, we would have 18 heatmaps each showing average readings of each node in each simulation. This proved to be a strong analysis tool and allowed us to visually trace the impact of each attack or treatment throughout the network structure.

Table 1. Simulation environment Parameters.

#	Parameter	Value
1	Number of motes	34 + 1 Sink
2	Malicious motes	0 or 1
3	Mote type	Zolertia Z1
4	Mote distribution	Binary or Grid
5	Tx & Rx success ratios	1.0
6	Duration	30 Minutes

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 December 2023 doi:10.20944/preprints202312.1356.v1

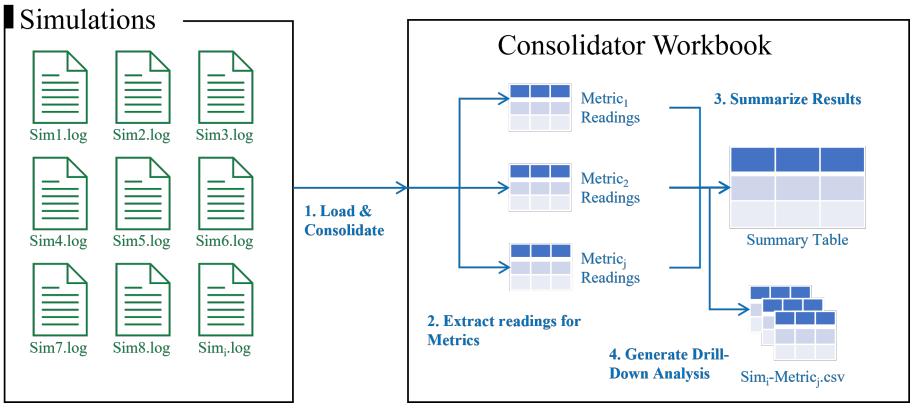

Figure 5. Consolidator process map.

4. Case Study: Analyzing Flooding Attack

In this section, we will examine the impact of flooding attack on RPL networks considering different topologies, leveraging the framework and tools proposed in Section 3.

4.1. Flooding Attack

Flooding attack is a form of Denial-of-Service (DoS) attacks that aims to degrade or completely bring the service provided by the network to a halt. There are many forms and implementations for carrying out a flooding attack that exploit weaknesses in the communication protocols stack. One such attack is DODAG Information Solicitation (DIS) flooding attack. As per RPL specifications, the actual topology of a network is built dynamically as nodes in the network exchange information with each other. DIS messages are sent (usually as multicast) by new nodes to solicit information on available networks and candidate parents. Adjacent nodes that are already in the network would respond by sending DIO messages to advertise their information [30]. After receiving DIO messages from neighbor nodes, the new node would stop sending further solicitation messages and view all senders as prospective parents [31]. However, in DIS Flooding attack scenario, a malicious node would keep sending DIS messages excessively, overwhelming other nodes in the network and forcing them to generate additional DIO messages which disrupts their ability to handle their roles in passing benign traffic, leading to congestions across existing links [32]. In our attack, the malicious node is assumed to have infiltrated the network and is acting normally as other nodes until the attack is triggered. Flooding attacks are a classic example of DoS attacks. Their footprint on RPL networks has been studied widely and was shown to cause significant disruption to the networks in Packet Delivery Ratio, throughput, and power consumption [33,34]. Almomani and Alkasasbeh [35] have compared flooding attacks to other DoS attacks (namely blackhole, grayhole, and scheduling attacks) and have demonstrated in a simulated experiment that it caused more severe damage to the target network's lifetime and service availability.

4.2. Simulation Environment

We will carryout the flooding attack in a simulated environment using Cooja and Contiki, leveraging our aforementioned framework tools to build the network and analyze our simulations. Our virtual simulation consists of two (2) layouts (Binary and Grid) as discussed in the Section 3, and two (2) scenarios (with an attacker, and without an attacker as a baseline scenario); making a total of four (4) simulations. Each simulation is consists of 35 nodes (34 sensors and one sink). Each sensor node record some observation will send its supposed readings to the sink, which in a real application would analyze the different readings and act accordingly. The position of the attacker is not directly adjacent to the root node, nor at the edge of the network as illustrated in Figure 6. Further details on specifications of our setup are summarized in Table 1. After we had setup the environment, we ran all

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 December 2023 doi:10.20944/preprints202312.1356.v1

of our 4 scenarios using the launcher script. Afterward, we compared the performance of our network when attacked to our original baseline. This experiment aims to compare the effect of a treatment (applying flooding attack), while controlling one potential confounding variable (network topology).

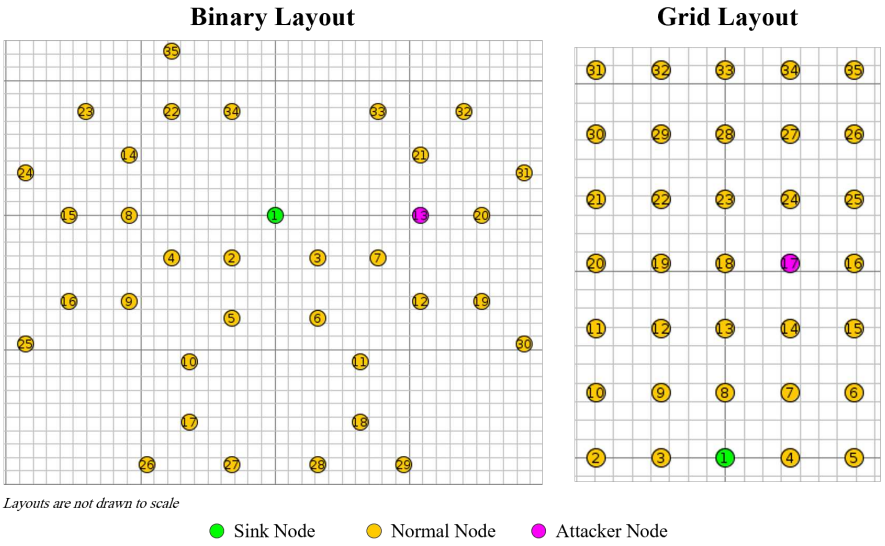

Figure 6. Layouts in our experiments highlighting the sink and attacker's positions.

4.3. Results and Analysis

The outcomes of our experiment are summarized in Table 2, where we compared the results of each attack scenario to the baseline for the respective network layout.

Table 2. Experiments results as shown by the consolidator.

Scenario	Binary			Grid		
	Avg PDR(%)	Avg E2E(*ms*)	Avg PC(*mW*)	Avg PDR(%)	Avg E2E(*ms*)	Avg PC(*mW*)
Normal	98%	1,027	0.52	96%	1,385	0.76
Flooding Attack	79%	1,177	0.80	54%	2,389	1.67
Impact (%)	-19%	+15%	+55%	-44%	+72%	+120%

We will distinguish both layouts in the baseline scenario. We note that both performed differently, where the binary topology being the most efficient across all metrics. Starting with PDR, we note that on average it was marginally better in the binary layout compared to the grid layout. After a closer look at Figure 7, we can determine that the marginal difference is mainly driven by the extended time needed to build the grid layout and its DODAG, as both layouts performed virtually the same after minute 3. For the average E2E delay, we noted significant delays in the grid layout equal to 35% more compared to the binary layout. The delays were particularly higher at the first 3 minutes, but rapidly decreased thereafter and slightly fluctuated throughout the rest of the simulation as visible in Figure 8. However the grid layout was still experiencing more delays more often than not average compared to the binary network.

Lastly, we can see that binary network showed more efficient power consumption compared to the grid layout, throughout the experiment as illustrated in Figure 9. In fact, there were periods were the power consumption for the Grid layout was double and triple that for the binary (e.g. minutes 1, 6, and 21. Over the duration of the experiment, the grid layout had a 46% higher power consumption rate compared to the binary. We attribute this to the increased exchange of control messages in the grid layout due to its increased density.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 December 2023 doi:10.20944/preprints202312.1356.v1

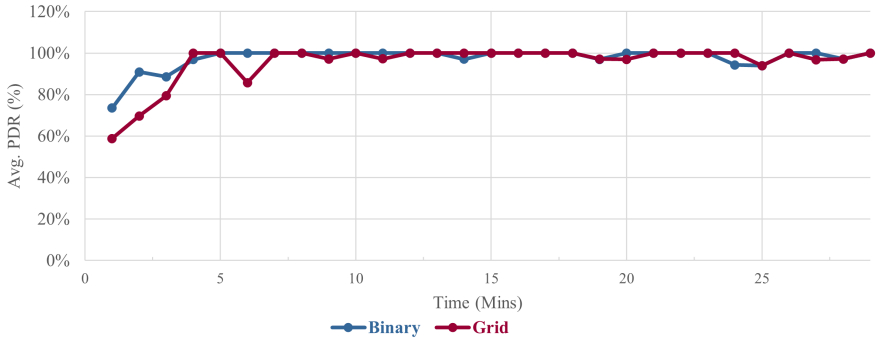

Figure 7. PDR performance for grid and normal topologies under normal conditions.

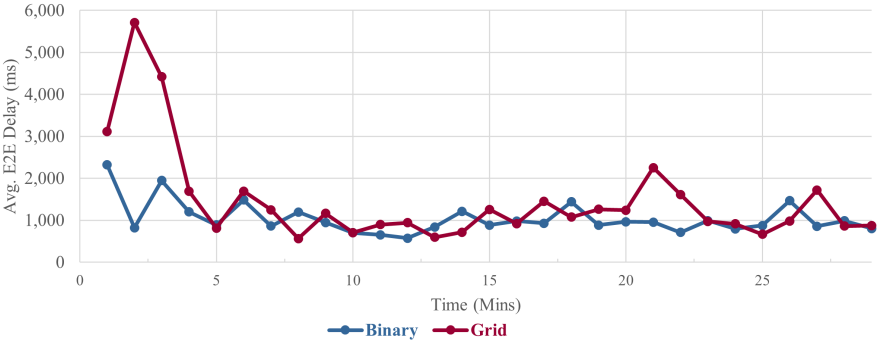

Figure 8. E2E performance for grid and normal topologies under normal conditions.

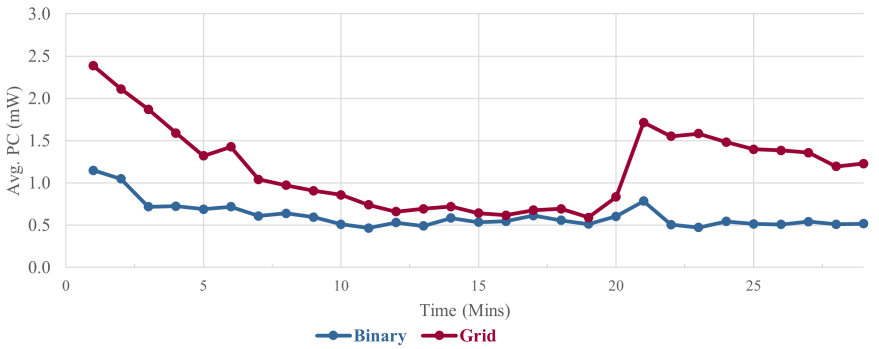

Figure 9. Power Consumption (PC) for grid and normal topologies under normal conditions.

The results consolidated by our consolidator (shown in Table 2) shows that both layouts were impacted when exposed to flooding attacks to a varying degree. To facilitate quick analysis particularly for larger experiments, the background color of the cells gets redder as the score gets worse for each metric. We note that the grid layout was more impacted with the attack when compared to binary layout, implying that having more dense networks where a node has more neighbors helps the propagation of the flooding messages to a larger population of the network.

Looking at the network's population to analyze impact on individual nodes would help us better understand the attack's propagation and consider it when distributing our sensors planning our mitigation approach. To that end, we will analyze the changes in each metric for each topology in more detail, leveraging the heatmaps generated by our framework and toolkit. There will be one heatmap for each metric-topology-attack combination, where each heatmap plots all nodes as colored circles in their corresponding coordinates and highlights potential links as lines. The sink node is colored in light green, whereas the attacker node is highlighted in pink color. Other nodes are labeled with their node number, as well as the change in the analyzed metric compared to the normal baseline. The

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 December 2023 doi:10.20944/preprints202312.1356.v1

nodes themselves are also colored based on the severity of that change for easier interpretation. If there is no change, the node will be transparent. As the change gets more significant, the more saturated the color of the node gets. If the change reflects an improvement, the color will change to green. If the change reflects a deterioration, the color would change towards red. Thresholds for the significance and polarity of interpreting these changes are defined for each metric and can be customized. If there are no readings for that specific node, the node will be labeled with "No data" and colored in gray.

4.3.1. Impact on Packet Delivery Ratio

Table 2 shows that PDR for the binary layout network has decreased by 19% following the flooding attack, compared to a 44% decrease in the grid layout. We will utilize the heatmaps generated by our tools to analyze how individual nodes performed during each simulation, which will give us insights on how the attack propagates. Figure 10 shows us a heatmap for changes in PDR across the binary network nodes after the attack compared to the normal baseline scenario (without the attack). Figure 11 shows the same comparasion for grid layout network.

Starting with the binary layout, Figure 10 demonstrations that the child nodes of the attacker failed to deliver any messages to the sink. The attacker's parent, however was not impacted, but the average PDR of its other branch (nodes 12, 19, and 30) was notably decreased by 23% on average. The same can also be noted - to a lesser degree - for the further outer branch (nodes 6, 11, 18, 28, 29), which showed a decrease in PDR by an average of 11.8%. The other half of the binary layout showed no significant changes in PDR.

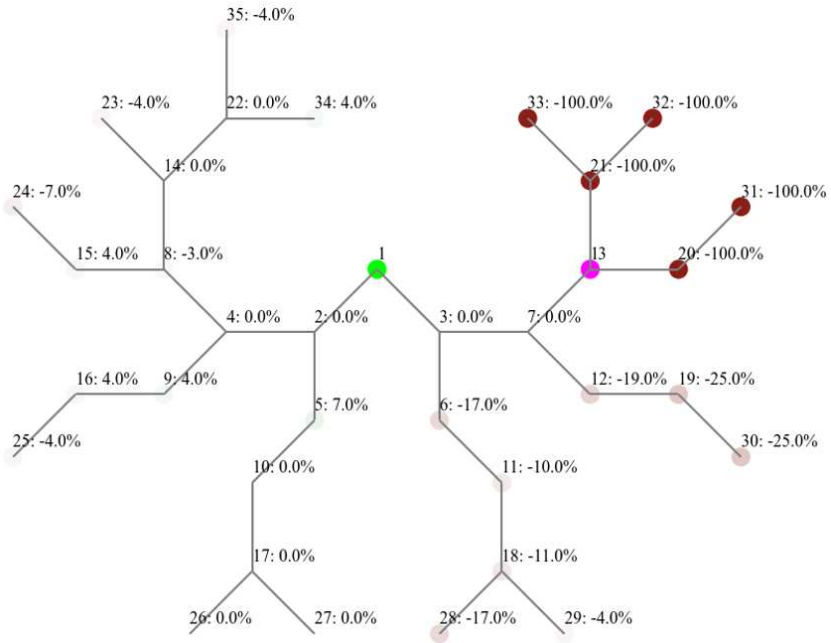

Figure 10. Heatmap showing the impact of flooding attack in terms of percentage change in average PDR on the nodes of the binary layout, compared to the normal baseline.

As for the grid layout as Figure 11 reveals, we can see that 13 out of 33 benign nodes (excluding the sink and the attacker) had a decrease in PDR by 50% or more. The most impacted nodes are those closer to the attacker and further away from the sink. We notice nodes that are adjacent to the attacker but closer to the sink (nodes 13, 14, and 15) are significantly less impacted compared to those that are further from the sink (nodes 24, 23, 25). Apparently, node 18 changed its parent as the experiment progressed and lost all subsequent packets. The effect of the attack in this layout propagated to a larger and further population compared to the binary layout. For example, it reached node 5, which is one hop away from the sink and two hops away from the attacker.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 December 2023 doi:10.20944/preprints202312.1356.v1

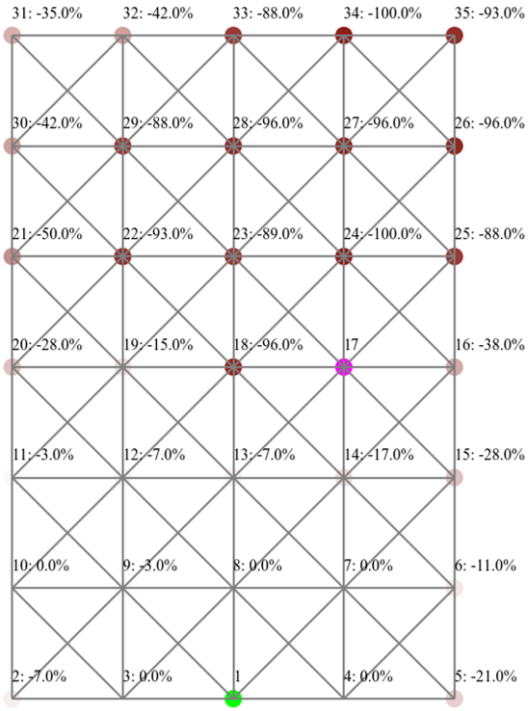

Figure 11. Heatmap showing the impact of flooding attack in terms of average PDR percentage on the nodes of the grid layout, compared to the normal baseline.

4.3.2. Impact on End-to-End Delay

The average E2E delay for binary layout networks, as demonstrated in Table 2, following the flooding attack has increased by 15% compared to the normal scenario, unlike the grid layout where delays in message delivery have increased by 72%. A closer look would provide further insights into how each network behaved under the attack, as illustrated in Figure 12 and Figure 13 for binary and grid layouts.

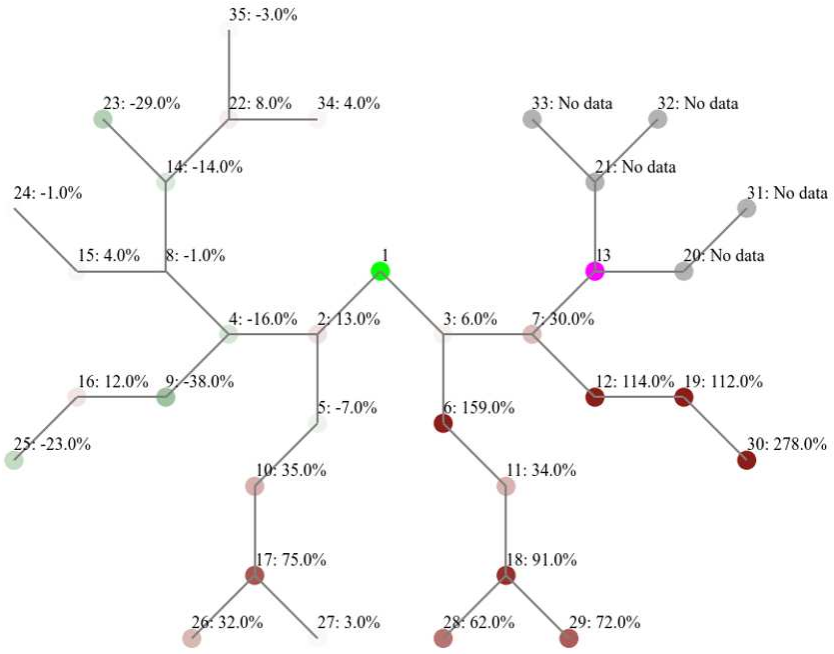

Figure 12. Heatmap showing the impact of flooding attack in terms of percentage change in average E2E delay on the nodes of the binary layout, compared to the normal baseline.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 December 2023 doi:10.20944/preprints202312.1356.v1

Since the child nodes of the attacker in binary layout had a PDR of 0% there were no messages to be delivered and, consequentially, no delay to be measured as presented in Figure 12. Although the attacker's parent witnessed a 30% additional delays, its other branch (nodes 12, 19, and 30) were the most impacted. Across the network, node 30 recorded the largest average delay of 4,101 ms, followed by node 19 with a delay of 2,477 ms, which corresponds to an increase of 278% and 119%, respectively, from their baseline levels. It's noted the impact of E2E delays spilled over to other branches and impacted a larger proportion of the network. On the other hand, some nodes in further branches recorded notable improvements (e.g., nodes 9, 23, 25, and 4), which slightly offset the average delay of the overall network. Here we can see how the heatmap help us decompose the impact on the network and obtain insights that would not have been visible if we have measured the performance at an overall.

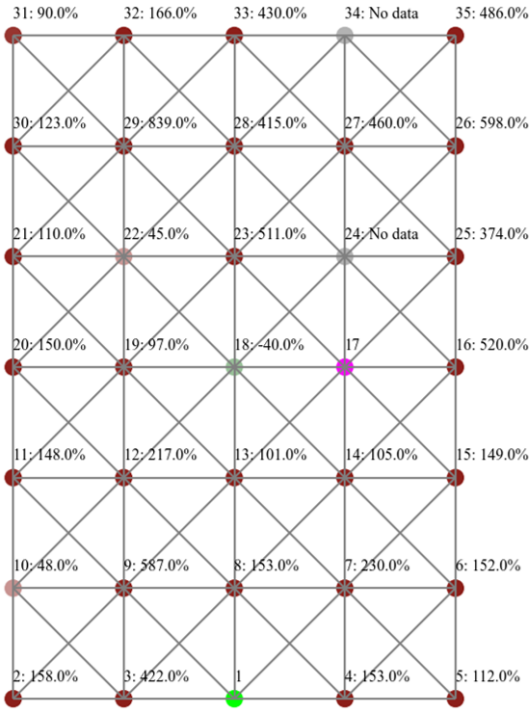

Figure 13. Heatmap showing the impact of flooding attack in terms of percentage change in average E2E delay on the nodes of the grid layout, compared to the normal baseline.

As for the grid layout, shown in Figure 13, we notice an increased spread of impacted nodes across the network, demonstrating the ripple effect of the flooding attack. Twenty-six (26) nodes witnessed increased delays by 2-folds (200% increase) as a result of the attack. Almost all nodes suffered extensive delays including some that are close to the sink. For example, node 3 is adjacent to the sink and recorded a 422% increase in delays. Although, attacker's direct neighbors where significantly impacted, node 18 shows a 40% improvement in E2E delay. After further examination, this is a measurement of the only packet that was delivered as previously analyzed in PDR heatmap (4% PDR).

4.3.3. Impact on Power Consumption

The flooding attack has increased the power consumption in binary networks by 55% as presented in Table 2. At the same time, the same metric for grid layout has increased by 120%. A drill-down into the performance of individual nodes is summarized in Figure 14 for binary layout, and Figure 15 for grid layout.

Although both binary and grid layouts have been impacted to varying degrees, we notice a common pattern that is in both layouts the nodes closer to the attacker are more affected, and thus

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 December 2023 doi:10.20944/preprints202312.1356.v1

have shorter lifetime. In the case of binary layout, the children nodes of the attacker (nodes 20 and 21) had the most increase in terms of power consumption (by 846% and 505%, respectively). Similarly for the grid layout, adjacent nodes to the attacker showed the highest increase in power consumption across the network. This can be seen through nodes 23, 18, and 24; which increased by 388%, 320% and 274%.

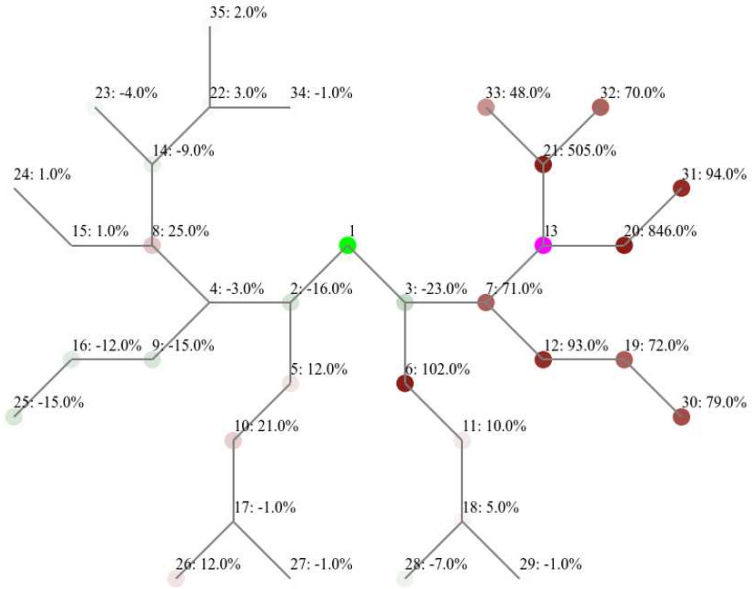

Figure 14. Heatmap showing the impact of flooding attack in terms of percentage change in average power consumption on the nodes of the binary layout, compared to the normal baseline.

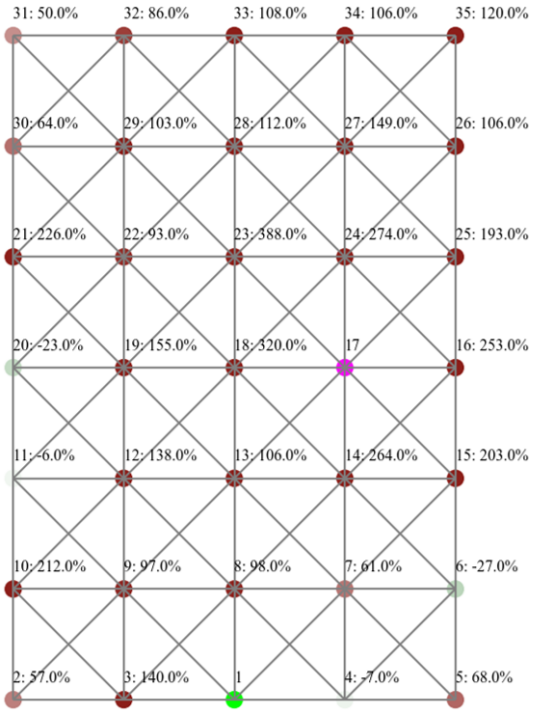

Figure 15. Heatmap showing the impact of flooding attack in terms of percentage change in average power consumption on the nodes of the grid layout, compared to the normal baseline.

Additionally, we also notice that some nodes that are further away from the attacker have showed a slight decrease in power consumption. This is applicable to both binary layout (e.g. nodes 2 and 3),

 doi:10.20944/preprints202312.1356.v1

and grid layout (e.g. nodes 6 and 23). However, one major distinction between the two layouts is that the effect of power consumption was more contained in the binary layout compared to that of the grid layout. In binary layout, Figure 14 points out that the average power consumption for the nodes in the right-hand branch where the attacker is located has increased by 130%, whereas that of the left-hand branch away from the attacker did not show visible changes. The same can't be said for the grid layout, Figure 15, as some nodes closer to the sink and further from the attacker have demonstrated a significant increase in power consumption, such as node 10 which showed an increase by 212%. The lifetime of the sink its surrounding nodes are critical for the lifetime of the overall network, as these serve the only link for further nodes to get their messages transmitted. Accordingly, we believe that a flooding attack would have a devastating impact shall the attacker is positioned closer to the sink.

5. Conclusion & Future Works

In this paper we presented a structured framework for conducting replicable simulations to study WSNs. It calls for considering different factors when assessing the network performance, such as different nodes' distribution, source codes (including attack scenarios), and node placements. The framework is complemented with tools that help in conducting experiments according to it by considering all stages, from building each simulation, to carrying out the experiments until consolidating the results and analyzing its outcomes. We demonstrated the adoption of such a framework when studying the impact of flooding attacks on two contrasting network topologies (grid and binary) and showed different impacts among both topologies and among nodes positioned differently within each network. As a future work, we intend to scale our tools to consider further network topologies (such as Linear, Ring, and Random). Further, we aim to embed applicable capabilities of our tools to interface directly with Cooja to allow easier and faster conduction of experiments by building on the architecture of ViTool [27], which is modular, scalable, and interfaced with Cooja stack.

References

1. Lata, S.; Mehfuz, S.; Urooj, S. Secure and Reliable WSN for Internet of Things: Challenges and Enabling Technologies. *IEEE Access* **2021**, *9*, 161103–161128. doi:10.1109/ACCESS.2021.3131367.
2. Muzammal, S.M.; Murugesan, R.K.; Jhanjhi, N.Z. A Comprehensive Review on Secure Routing in Internet of Things: Mitigation Methods and Trust-based Approaches. *IEEE Internet of Things Journal* **2020**, *4662*, 1–1. doi:10.1109/jiot.2020.3031162.
3. Swessi, D.; Idoudi, H. A Survey on Internet-of-Things Security: Threats and Emerging Countermeasures. *Wireless Pers Commun* **2022**, *124*, 1557–1592. doi:10.1007/s11277-021-09420-0.
4. Saravanan, G.; Parkhe, S.S.; Thakar, C.M.; Kulkarni, V.V.; Mishra, H.G.; Gulothungan, G. Implementation of IoT in production and manufacturing: An Industry 4.0 approach. *Materials Today: Proceedings* **2022**, *51*, 2427–2430. doi:10.1016/j.matpr.2021.11.604.
5. Azzedin, F.; Suwad, H.; Alyafeai, Z. Countermeasureing zero day attacks: asset-based approach. 2017 International Conference on High Performance Computing & Simulation (HPCS). IEEE, 2017, pp. 854–857.
6. Suwad, H.I.M.; Azzedin, F.A.M. Asset-based security systems and methods, 2022. US Patent 11,347,843.
7. Azzedin, F.; Suwad, H.; Rahman, M.M. An Asset-Based Approach to Mitigate Zero-Day Ransomware Attacks. *Computers, Materials & Continua* **2022**, *73*.
8. Azzedin, F.; Albinali, H. Security in internet of things: Rpl attacks taxonomy. The 5th International Conference on Future Networks & Distributed Systems, 2021, pp. 820–825.
9. Le, A.; Loo, J.; Lasebae, A.; Vinel, A.; Chen, Y.; Chai, M. The impact of rank attack on network topology of routing protocol for low-power and lossy networks. *IEEE Sensors Journal* **2013**, *13*, 3685–3692.
10. Panda, N.; Supriya, M. Blackhole Attack Impact Analysis on Low Power Lossy Networks. 2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT), pp. 1–5. doi:10.1109/GCAT55367.2022.9971814.

11. Tripathi, M.; Gaur, M.S.; Laxmi, V. Comparing the impact of black hole and gray hole attack on LEACH in WSN. *Procedia Computer Science* **2013**, *19*, 1101–1107.
12. Iqbal, M.M.; Ahmed, A.; Khadam, U. Sinkhole Attack in Multi-sink Paradigm: Detection and Performance Evaluation in RPL based IoT. 2020 International Conference on Computing and Information Technology (ICCIT-1441); IEEE: Tabuk, Saudi Arabia, 2020; pp. 1–5. doi:10.1109/ICCIT-144147971.2020.9213797.
13. Sun, X.; Wang, G.; Xu, L.; Yuan, H. Data replication techniques in the Internet of Things: a systematic literature review. *Library Hi Tech* **2021**, *39*, 1121–1136.
14. Argota Sánchez-Vaquerizo, J. Getting real: the challenge of building and validating a large-scale digital twin of Barcelona's traffic with empirical data. *ISPRS International Journal of Geo-Information* **2022**, *11*, 24.
15. Ramya, P.; Sairamvamsi, T., Impact Analysis of Blackhole, Flooding, and Grayhole Attacks and Security Enhancements in Mobile Ad Hoc Networks Using SHA3 Algorithm. In *Lecture Notes in Electrical Engineering*; Springer Singapore, 2018; pp. 639–647. doi:10.1007/978-981-10-7329-8_65.
16. Mayzaud, A.; Badonnel, R.; Chrisment, I. A distributed monitoring strategy for detecting version number attacks in RPL-based networks. *IEEE transactions on network and service management* **2017**, *14*, 472–486.
17. Mayzaud, A. Monitoring and Security for the RPL-based Internet of Things **2016**. p. 165.
18. Arış, A.; Örs Yalçın, S.B.; Oktuğ, S.F. New lightweight mitigation techniques for RPL version number attacks. *Ad Hoc Networks* **2019**, *85*, 81–91. doi:10.1016/j.adhoc.2018.10.022.
19. Hachemi, F.E.; Mana, M.; Bensaber, B.A. Study of the Impact of Sinkhole Attack in IoT Using Shewhart Control Charts. GLOBECOM 2020 - 2020 IEEE Global Communications Conference, 2020, pp. 1–5. doi:10.1109/GLOBECOM42002.2020.9322603.
20. Le, A.; Loo, J.; Lasebae, A.; Vinel, A.; Chen, Y.; Chai, M. The Impact of Rank Attack on Network Topology of Routing Protocol for Low-Power and Lossy Networks. *IEEE Sensors Journal* **2013**, *13*, 3685–3692. doi:10.1109/JSEN.2013.2266399.
21. Rai, K.K.; Asawa, K. Impact analysis of rank attack with spoofed IP on routing in 6LoWPAN network. 2017 Tenth International Conference on Contemporary Computing (IC3), 2017, pp. 1–5. doi:10.1109/IC3.2017.8284340.
22. Kulau, U.; Müller, S.; Schildt, S.; Büsching, F.; Wolf, L. Investigation & Mitigation of the Energy Efficiency Impact of Node Resets in RPL. *Ad Hoc Networks* **2021**, *114*. doi:10.1016/j.adhoc.2021.102417.
23. Bocchino, S.; Fedor, S.; Petracca, M. Pyfuns: A python framework for ubiquitous networked sensors. European Conference on Wireless Sensor Networks. Springer, 2015, pp. 1–18.
24. Theodorou, T.; Violettas, G.; Valsamas, P.; Petridou, S.; Mamatas, L. A Multi-Protocol Software-Defined Networking Solution for the Internet of Things. *IEEE Commun. Mag.* **2019**, *57*, 42–48. doi:10.1109/MCOM.001.1900056.
25. Finne, N.; Eriksson, J.; Voigt, T.; Suciu, G.; Sachian, M.A.; Ko, J.; Keipour, H. Multi-trace: multi-level data trace generation with the cooja simulator. 2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS). IEEE, 2021, pp. 390–395.
26. Violettas, G.; Simoglou, G.; Petridou, S.; Mamatas, L. A Softwarized Intrusion Detection System for the RPL-based Internet of Things networks. *Future Generation Computer Systems* **2021**, *125*, 698–714. doi:10.1016/j.future.2021.07.013.
27. Jabba, D.; Acevedo, P. ViTool-BC: Visualization Tool Based on Cooja Simulator for WSN. *Applied Sciences* **2021**, *11*, 7665. doi:10.3390/app11167665.
28. Dunkels, A.; Osterlind, F.; Tsiftes, N.; He, Z. Software-Based on-Line Energy Estimation for Sensor Nodes. Proceedings of the 4th Workshop on Embedded Networked Sensors; Association for Computing Machinery: New York, NY, USA, 2007; EmNets '07, p. 28–32. doi:10.1145/1278972.1278979.
29. Zolertia. Z1 Datasheet **2010**. pp. 1–20.
30. Gaddour, O.; Koubâa, A. RPL in a nutshell: A survey. *Computer Networks* **2012**, *56*, 3163–3178. doi:10.1016/j.comnet.2012.06.016.
31. Alexander, R.; Brandt, A.; Vasseur, J.P.; Hui, J.; Pister, K.; Thubert, P.; Levis, P.; Struik, R.; Kelsey, R.; Winter, T. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks **2012**. doi:10.17487/RFC6550.
32. Medjek, F.; Tandjaoui, D.; Djedjig, N.; Romdhani, I. Multicast DIS attack mitigation in RPL-based IoT-LLNs. *Journal of Information Security and Applications* **2021**, *61*, 102939.

33. Bokka, R.; Sadasivam, T. DIS flooding attack Impact on the Performance of RPL Based Internet of Things Networks: Analysis. 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC), 2021, pp. 1017–1022. doi:10.1109/ICESC51422.2021.9532901.
34. Banga, S.; Arora, H.; Sankhla, S.; Sharma, G.; Jain, B. Performance Analysis of Hello Flood Attack in WSN. Proceedings of International Conference on Communication and Computational Technologies; Purohit, S.D.; Singh Jat, D.; Poonia, R.C.; Kumar, S.; Hiranwal, S., Eds.; Springer Singapore: Singapore, 2021; pp. 335–342.
35. Almomani, I.; Al-Kasasbeh, B. Performance analysis of LEACH protocol under Denial of Service attacks. 2015 6th International Conference on Information and Communication Systems (ICICS), 2015, pp. 292–297. doi:10.1109/IACS.2015.7103191.