DETECTION OF UDP FLOOD ATTACKS IN WIRELESS SENSOR NETWORKS BY VISUALIZATION ON PARALLEL COORDINATE PLOT

by

Neha Chukkayapally

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
February 2021

I owe my mother everything. It is because of her that I am where I am right now and who am in my life. Thank you AMMA for all eternity. It wouldn't have been possible to stay sane under so much stress and finish my thesis without the help of my best friend - Sarferoze. He has been and always will be the person I will be the most grateful for second only to my mother. Last but not the least, I would like to thank all of my friends for making my life so joyful. During these three years of studying and worrying, the fun I had with them is the only reason I didn't make a run for it!!

Contents

List of Tables

List of Figures

Abstract

Distributed Denial of Service (DDoS) attacks are one of the most destructive attacks that threaten network systems' security today. In this type of attack, the attacker tries to make a targeted computer or network unreachable to its intended users by incapacitating the server. Many detection and prevention tools, systems, and algorithms have been developed over the years to determine and restrain these intrusions. Many mechanisms that are available in the market try to visualize the after-effects of these breakthroughs. However, not many of them have explored the concept of using visualization itself for detecting these attacks. With technology growing day-by-day, Wireless Sensor Networks (WSNs) have gained popularity because these are very cost-effective and have other advantages over the wired networks. Like other technologies, these too are being harrowed by DDoS attacks. Therefore, there is much research interest in developing visualization-based detection models for WSNs. This field is still in its infant stage of development.

This thesis's framework is a detection model based on data visualization for User Datagram Packet (UDP) flood attacks on WSN. A dataset is generated using the Cooja simulator to identify the resource consumption attributes under regular and attack situations. The dataset is then separated into their corresponding groups (attack and non-attack) using k-means and mini-batch k-means clustering algorithms. The clustered data is then visualized onto a parallel Coordinate Map (PCM). The data points on PCM form a unique pattern, thus assisting the network administrator to quickly identify that an attack is taking place and can take necessary action. Performance comparison of both the clustering algorithms shows that mini-batch k-means exhibits an improvement of 10% over the k-means algorithm. The experiment's outcome conveys that this model is simple yet effective in detecting DDoS attacks in WSNs.

List of Abbreviations Used

6LowPAN	IPv6 over Low -Power Wireless Personal Area Networks
AnNetTe	Analyzing Network Technology
API	Application Programming Interface
BS	Base Station
CoAP	Constrained Application Protocol
CPU	Central Processing Unit
CSV	Comma Separated Values
DAO	Destination Advertisement Object
DAO-ACK	Destination Advertisement Object- Acknowledgement
DDoS	Distributed Denial of Service
DIO	DODAG Information Object
DIS	DODAG Information Solicitation
DNS	Domain Name System
DODAG	Destination oriented Directed Acyclic Graph
DoS	Denial of Service
EM	Expectation-Maximization
GUI	Graphical User Interface
HTTP	Hyper Text Transfer Protocol

ICMP	Internet Control Message Protocol
IDE	Integrated Development Environment
IDS	Intrusion Detection System
IoT	Internet of Things
IP	Internet Protocol
MAC	Media Access Control
MDS-VOW	Multi-Directional Scaling - Visualization Of Wormhole
NS2	Network Simulator2
OSI	Open System Interconnection
PCAP	Packet Capture
PCAV	Parallel Coordinates Attack Visualizer
PCM	Parallel Coordinate Map
PCTT	Parallel Coordinates version of Time-tunnel
PDE	Python Development Environment
PDR	Packet Delivery Ratio
PoD	Ping of Death
PSL	Python Standard Library
RPL	Routing Protocol for Low Power and Lossy Networks
SAVE	Sensor Anomaly Visualization Engine
SFC	Space Filling Curves

SNA	Sensor Network Analyzer
SSE	Sum of Squared Errors
TCP	Transmission Control Protocol
UDGM	Unit Disk Graph Medium
UDP	User Datagram Packet
WSN	Wireless Sensor Networks

Acknowledgements

First of all, I would like to thank my supervisor, Dr. Srinivas Sampalli, for guiding and supporting me in every stage of my thesis. Your kindness and understanding are some of the main reasons for me to move under your supervision when I thought I would not be able to continue any further. Your easy-going approach has helped me develop my thesis topic even though I am new to this area of research and finish it off successfully. Thank you for being there at all times.

I want to thank my committee members for accepting to be part of the committee. It is an honor to have such professional geniuses in my defense.

I want to thank all the lab members for being so helpful and giving me inputs and ideas when I needed them the most.

I am thankful to my family for supporting and not pressuring me when I took the time to get here.

Chapter 1

Introduction

1.1 Introduction to Data Visualization

Although data visualization has been in use since time immemorial, right since when cave dwellers used to draw them on their caves' walls to communicate, the term was coined only very recently. The most basic question of what data visualization is and how it has come to light and is now playing a significant role in today's fast-paced technological world, the roots can be found from our early ages themselves.

The word data visualization is self-explanatory. Put in simple words; it means the data, be it numbers, text, or any other form, is represented in the form of graphs or pictures. This representation can be thought of as a mapping between the original data and the graphical elements that are drawn[5]. When we look at large paragraphs of data or pages full of statistics, our minds become boggled and cannot perceive this massive amount of data. Instead, suppose the same data is put in the form of a picture or a graph. In that case, it is very easily understandable because our minds have been designed in such a way to understand data in the form of visualization quickly. Difficult concepts can be grasped, hidden patterns can be recognized, data can be analyzed, and decisions can be made with data visualization.

1.1.1 Types

Based on what type of information it represents, data visualization can be categorized as follows:

Change Over Time: Shift over time is one of the most popular and easiest data visualization forms. It is a visual aspect that reflects how a number in a certain sense has evolved. By looking at the results over time, one can see the progression of decided success and the future trend. Stream graph is an example of change over time visualization.

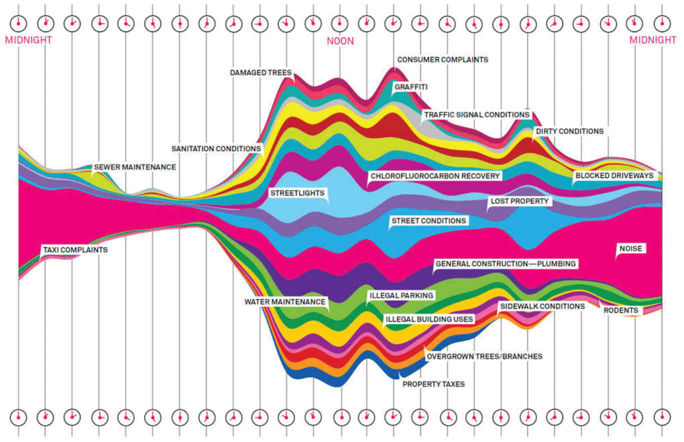


Figure 1.1: Example of a stream graph[12]

Category Comparison: In this type of visualization, several statistics are placed side-by-side, and the association between them is analyzed. With this visualization's help, we can understand how each statistic has contributed to a business or organization's success or failure. An example of this type of visualization is a bar graph.

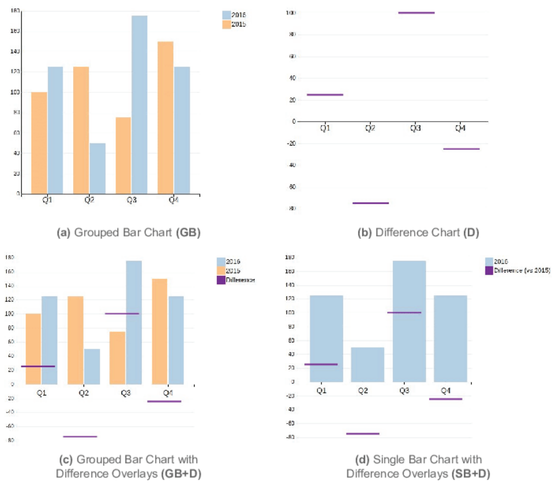


Figure 1.2: Example of a bar graph[57]

Part to the Whole: Sometimes, instead of comparing numbers amongst themselves, it makes more sense to understand their relative importance as part of a whole. This type of visualization is all about this concept. A treemap can be used to depict this type of data.

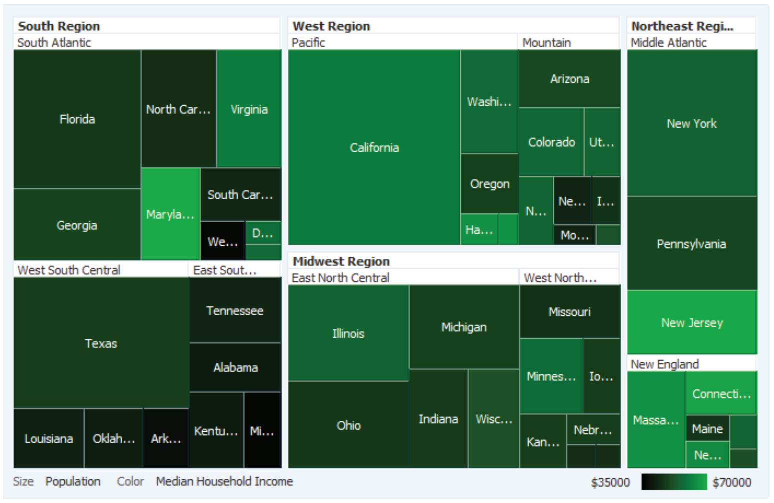


Figure 1.3: Example of a treemap[13]

Distribution: The incidence and volume are more relevant than direct results in distribution charts. They display how many times in a span a series of values exists. It is a little like part-to-all, but it is used mostly for numbers. A donut graph implements distribution.

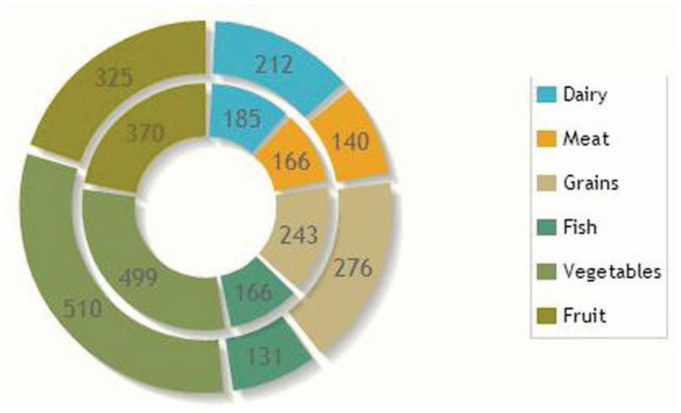


Figure 1.4: Example of a donut chart[68]

Flow: Flow maps are all about movement than volume. This type of visualization shows how a statistic or indicator has evolved and its evolution to other periods and points. Here, the main concern is about the connection of the indicator with all other aspects at various points of time. Like in the change over time type of visualization, stream graphs can also display the flow. Another type of visualization that presents this information is a line chart.

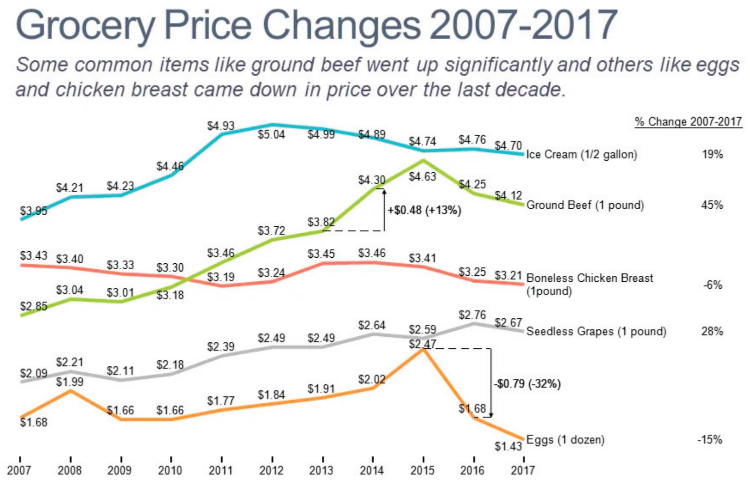


Figure 1.5: Example of a line chart[6]

1.1.2 Benefits

A good visualization can be defined as a balance between form and function and which can serve its main purpose successfully - to communicate information[5]. With good visualization, there are many advantages. Some of them are as follows:

- Promotes comprehension and interpretation
- Recognize relationships, patterns, and interdependencies
- Convenient to share
- Quick forecasts

1.2 Introduction to Wireless Sensor Networks

Wireless Sensor Networks (WSNs) are immensely gaining popularity because they have some unique features such as cost-effectiveness, being able to be set up without much infrastructure, accommodation of new devices at any time, etc., when dealing with real-world challenges. WSN is a wireless network without infrastructure. It comprises small devices that work together and gathers information. This set of spatially distributed and committed devices, also known as sensors, are used for tracking and documenting the physical conditions of the atmosphere and organizing the data gathered at a centralized spot[16]. The sensor nodes mainly have CPU, memory, transceiver (for sending and receiving signals from one node to another), and battery. The sensor nodes are connected to a Base Station (BS) that serves as the WSN device processing unit[39]. This base station is connected to other WSNs, which are part of the Internet of Things (IoT) system for information exchange.

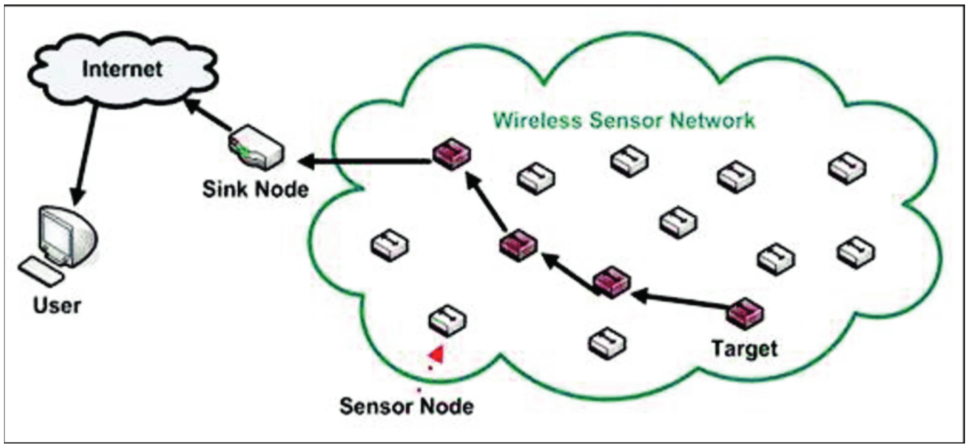


Figure 1.6: Wirelss Sensor Network (WSN)[51]

The main features[27] of WSNs are as follows:

- The devices WSN comprises of are autonomous.
- Each node exchanges knowledge regarding its neighborhood.
- It is ideal for distributed implementation.

- Knowledge is primarily distributed to a single node.
- Interconnecting devices like bridges, routers, etc., are not required.

1.2.1 Basic architecture

WSNs follow the Open System Interconnection (OSI) model. Accordingly, there are five layers in the sensor networks: the physical layer, data-link layer, network layer, transport layer, and application layer. There are three cross-over layer planes and the five layers: power management plane, mobility management plane, and task management plane. These layers are used for network control and for the sensors to function together to improve the overall network performance[18].

The main functions of each layer are[18]:

Physical Layer

This layer acts as an interface for transmitting a stream of bits and other functions like data encryption, signal detection, frequency selection.

Data Link Layer

Data frame recognition, Media Access Control (MAC) and error management, multiplexing data streaming, guaranteeing point-point or point-multi-point stability are the primary responsibilities of this layer.

Network Layer

The main function of this layer is routing. There are many routing protocols available in WSNs. These algorithms' primary responsibility is to find reliable paths for the message transmission, detect redundant paths, and work on them accordingly.

Transport Layer

The nature of WSN is multi-hop. So, there arises congestion at different points in the network at different times. However, the main congestion occurs when each sensor node in the sensor network captures some information and tries to pass it on to the sink node. The main responsibility of the transport layer is to reduce this congestion. So, many protocols are part of this layer in order to do this.

Application Layer

This layer is responsible for the management of traffic. The bitstream of data received from the previous layers is converted into an understandable form and sends out queries to other nodes or servers to obtain information.

1.3 Distributed Denial of Service (DDoS) Attacks

While the internet is trying to achieve the pinnacle of technological innovation that was unimaginable a decade or two ago, it is still being plagued by some of the deadliest attacks of which DDoS attacks are playing a significant role even though the strongest of cybersecurity measures are in place. DDoS attacks have been part of the criminal toolbox[34] for more than two decades now and are only becoming more assertive.

A Denial of Service (DoS) is a type of cyber attack where a targeted system or a targeted network is overloaded with a massive number of packets or requests to drain its capabilities and make it unavailable to legitimate users. DDoS attacks are a type of DoS attacks where more than one system is used to bring down the targeted system or network by sending malicious traffic. By bombarding a system from more than one location with attack packets, the attacker can easily disrupt the system and render it useless within no time. These types of attacks are more dangerous than DoS attacks since the origin of the attack cannot be pinpointed because many systems are being used at a time. The attacker launches a DDoS attack with remotely controlled slave computers known as zombies or bots. These bots form a network of connected devices, which is known as a botnet. The attacker manages this botnet with the help of a command and control server[38].

Some of the most commonly occurring DDoS attacks are:

- Ping of Death (PoD) or Internet Control Message Protocol (ICMP) Attacks
- UDP Flood
- SYN Flood
- Hyper Text Transfer Protocol (HTTP) Flood
- Slowloris
- Zero-Day Attacks
- Buffer OverFlow Attacks
- TearDrop Attacks

1.4 DDoS attacks in WSNs and security concerns

Similar to wired networks, every node in WSN is also vulnerable to many insider and outsider attacks[**?**]. Since WSNs could be part of IoT, which is spread over a vast area with a vast number of devices and objects connected as a network, the chances of security attacks occurring on them are very high. DDoS attacks in the area of WSNs can be categorized as active attacks and passive attacks.

Passive DDoS attacks are those in which the intruder or attacker is not harming the system in a physical sense. However, the aim is to steal important information from the nodes and network by secretly listening to the traffic. The attacker behaves like any other node, and thus it is challenging to catch the culprit because he/she does not leave any obvious traces that might track-back to him/her. Eavesdropping is one type of attack where the attacker spies on the packets traveling through the network to steal information. Whereas in active attacks, the intruder identifies some loopholes in the existing protocols and uses them to launch attacks on the system. These are fatal than passive attacks, but since abnormalities can be observed in the network traffic. In contrast, an attack occurs, it is easily identifiable that a culprit is working against the setup. An example of this type of attack is UDP flood attack. In this intrusion, the attacker floods the targeted system's ports by sending too many Internet Protocol (IP) packets containing user datagrams. UDP is a connectionless and sessionless networking protocol[14]. It does not require a three-way handshake to establish a connection. This feature has many advantages and, at the same time, makes the sensor network susceptible to attacks. When the server receives a UDP packet, it initially checks if any program or task is listening to this port. When it finds no program, it sends out an ICMP packet to let the sender know that the destination is unreachable[15]. The attacker sends out a huge number of these packets to flood the target. The receiver system now wastes its resources to respond with ICMP messages and thus becomes unresponsive after some time.

Some of the most common DDoS attacks that take place in WSNs are:

- Jamming
- Node Tempering
- Collision
- IP Spoofing
- Black holes and sink holes
- Sybil attacks
- Hello flood
- UDP flood
- SYN flood

1.5 Data Visualization for Network Traffic Analysis

Data visualization has been in use for quite some time now to show complex data clearly and effectively by representing through graphical means[63]. It has been used in many fields, and its usage in network security[33][64] has only started a few years ago. Network traffic can be defined as the amount of data or information that moves across a network at any given time. The data in the network traffic is usually made up of network packets. The network data (carried by the network packets) is very complicated and cannot be easily understood in large tables. Thus, the application of data visualization to understand the intricate details of network data has become immensely popular in recent times. It helps us to understand, analyze and find useful information from this mirage of data.

Just like in the case of wired networks, visualization tools have been developed for WSNs also. Some of the existing visualization tools are Surge, MoteView, Octopus, Sensor Network Analyzer (SNA). These display the operation of the network, the traffic between two sensor nodes, and provides network topologies and sensor details to users with live information so that the deployed sensor network could be debugged in real time[63]. Also, with the help of these tools, the sensor network's health and performance can be monitored, specific packets that the user is interested in can be viewed at any given time.

Even though these existing tools can monitor the sensor networks, they are still not equipped to visualize security-related data and events[63]. It means that when some unpredictable security attacks are launched against sensor network nodes, these tools cannot predict and understand the anomalies that occur while an attack is taking place. So, it is essential for a visualization system not only to be able to monitor the network but also to be able to differentiate between normal and abnormal activities[63]. Along with being able to differentiate between regular and aberrant actions, the administrator monitoring the network must also discern, categorize and assess the security-related incident by looking at the visually represented network data. Furthermore, data visualization has laid foundations to help reach all these goals with its powerful approach.

Many visualization-based attack detection methods have been proposed for wired networks. Since WSNs are relatively newer, this visualization-based approach to detect security attacks, mainly DDoS attacks, has not been used much in this area. Thus, there is a lot of potential for visualization based DDoS attack methods to successfully and efficiently detect DDoS attacks in WSNs.

1.6 Motivation and Scope

DDoS attacks are among the most prevalent cyber attacks that afflict the cyber world's security. Since network traffic data is very complex, it is not easy to understand it by looking at the packets that comprise the traffic. This complexity becomes manifold when the data being dealt with is Wireless Sensor Network data due to their dynamic nature. When the non-attack data itself in these networks is very complicated to comprehend, the user or the network administrator cannot conceive the irregularities that occur while an intrusion occurs. The main motivation for this research is the limited number of visualization based DDoS attack detection models available for WSNs.

The main changes that occur when a sensor network is in an attack situation are the increase in the systems' resource consumption levels. Many DDoS attack datasets are available on the internet, and other sources but none of them contain information about reserve dissipation like power, energy, packet loss, etc. To obtain these optimal features, a dataset is generated using a simulator. The dataset contains systems' use of sources under normal and attack conditions. In order to identify the abnormalities, these data points should be separated into their respective groups. To do this, clustering algorithms can be used since these work without any prior training. Nevertheless, the clusters that come as the output are still unclear since the network data is continuous and is thus humongous in size. Then, data visualization comes into the picture. Any form or level of compound data can be easily understood by representing it in a pictorial form. A parallel coordinate plot is handy to visualize high-dimensional data like network data. Thus, these clustered data points are plotted onto this plot. The points break out into two groups, and the graphical representation allows the user to identify an abnormal scene within moments. Therefore, a parallel plot presentation with the assistance of clustering algorithms can be useful in detecting DDoS attacks in WSNs.

1.7 Aims and Objectives

The main objectives of the proposed work are as follows:

- To create a dataset with the optimal features for detecting UDP flood attacks in a WSN using a simulator.
- To design a visualization based DDoS attack detection model that uses clustering algorithms and parallel plot in order to effectively detect UDP attacks in WSNs.
- To evaluate both the clustering algorithms and show that mini-batch k-means is better than k-means when the data is vast and complex.

1.8 Thesis Contribution

The key handouts of the research work are:

- The main contribution of this thesis is to develop a visualization based DDoS attack detection model for detecting UDP flood attacks in WSNs.
- The resource consumption features required for the detection of DDoS attacks like energy consumption, delay, packet loss, power consumption, idle time, and CPU time are not available in the datasets that can be accessed from various sources online or elsewhere. Thus, a dataset with these optimal features is generated with the help of a simulator called Cooja.

- Successful detection means to be able to identify which data is benign and which is malicious. This is done with the help of two clustering algorithms - k-means and mini-batch k-means.
- The clustering is done based on metrics like the systems' resource consumption under attack and non-attack conditions.
- The clustered data is visualized onto a parallel coordinate plot - a type of visualization technique used to represent high-dimensional data. The data branches out into two and forms a pattern. The data items belonging to different groups show up in different colors.
- Consequently, the visualization-based model is successful in detecting UDP attacks in WSNs by distinguishing between regular and irregular data in different colors.
- The two clustering algorithms are then compared based on evaluation metrics, and it is shown that mini-batch k-means clustering algorithm has high clustering accuracy than k-means algorithm when the data is enormous and is a better method for clustering when dealing with complex and high-dimensional data[65][31].

1.9 Organization of thesis

The rest of the report is organized as follows:

Chapter 2 surveys the existing DDoS attack detection mechanisms based on the visualization for traditional wired networks. It then discusses about the DDoS attacks that take place in WSNs. It provides a detailed literature survey about the mechanisms and visualization models that were developed for the detection of DDoS attacks in WSNs.

Chapter 3 outlines the proposed model for the thesis work. It provides the general approach's outline and gives details about the technologies, algorithms, and libraries used in the framework.

Chapter 4 gives details about how the experiment was carried out, the hardware and software requirements, the simulation setup, and the set parameters.
Chapter 5 defines the evaluation and performance metrics used in order to assess the system. It presents the results of the experiment and delivers a detailed analysis of the outcomes.
Chapter 6 concludes the thesis and describes the future directions.

Chapter 2

Related Works

This chapter surveys some of the vital research done in visualization models developed over the years for wired networks and sensor networks.

2.1 Research on visualization models for traditional wired networks

Chen et al. [28] have proposed a visualization tool named Analyzing Network Technology (AnNetTe). This tool aims to monitor the network and discover different attacks that take place in the network. AnNetTe has three visualizations with various levels of details - the timeline overview, ring graph, and connection river. There are three parts to the timeline overview itself. The first timeline shows details like CPU load, number of denial logs, and total bytes. The second timeline focuses on showing the anomaly detection timelines by using the entropy of the system. The third timeline is a zoom function. The ring graph displays the chosen time relations, structured and grouped by subnet[28]. Moreover, lastly, the connection river part of the visualization tool gives a parallel coordinate view. The axes are the IP addresses, connection time, and the ports used. With the help of this visualization system, DDoS attacks like port scans, host scans, UDP flood attacks, etc., can be detected even before the attacks fully take place.

Lee et al. [40] designed a real-time network security monitor called Visual-Firewall for detecting DoS attacks. It gives four views: real-time traffic, visual signature, statistics, and Intrusion Detection System (IDS) alarm. Each one of these views provides various detail, levels that help the network administrator detect any anomalies. In the real-time traffic view, the packets that enter and leave the firewall are represented using glyphs. This view aims to show what packets are going through the firewalls and which ones are being rejected. In the visual signature view, the packets' flows are represented as lines on a parallel coordinate plot. Different color codings are used in order to show what protocol a packet is using. The statistics view aims to show the network's overall throughput over time [40]. On a line chart, the throughput is represented in bytes/second. The IDS alert view displays IDS alerts in four-axis diagram. With these in-depth visualizations, even a beginner can understand which is normal and attack traffic.

Yang and Liu [66] have used parallel coordinates to detect DDoS attacks. Node resource consumption is expressed in this approach as the values of the axes. A fold line represents resource usage for each node. Since the network data is vast and is very complicated when all the data points are represented as fold lines on a parallel coordinate plot, it becomes very cluttered, and drawing patterns and conclusions from the visualization becomes impossible. In order to overcome this problem, the k-means clustering algorithm is used. One of the main disadvantages of using k-means is that the number of clusters (k) must be provided manually. Then, the cluster centers are selected randomly when the algorithm is being run. The manual selection of cluster centers reduces the performance of the algorithm drastically. So, an interactive k-means algorithm is proposed. In this algorithm, the dataset is initially illustrated on a parallel plot. With human cognition, the user manually picks out the number of clusters by observing the plot's intersection points. The user can also pick centers such that the number of iterations is reduced, so the performance is improved.

Zhang et al. [68] proposed an interactive visual system called DDoS Viewer for detecting DDoS attacks by surveying the visual patterns. The core host represents the main node. The gap between the center node and the other nodes is equal to the frequency of correspondence between them. In this representation, the node will stand out from the other nodes comparatively situated in the community while an abnormally regular contact occurs. Many concentrated clusters in varying colors then represent each node. If the number of ports of a node used in the network is high, the contrast would be greater. The degree of color in the concentric circles becomes higher. In the case of a DDoS attack, this method accurately identifies the suspicious nodes.

Blue et al. [26] created a real-time visualization tool called NetGrok to depict the computer network usage. It represents the data by using a graph and tree-map visualization. This work aims to use visualization techniques for real-time, streaming data [26]. The two visualizations capture: 1. hosts via IP, 2. the bandwidth use of hosts, 3. the interactions between the hosts. The visualization part has been implemented using the Prefuse visualization library. The system's data can be input as a pcap file or capture traces from a live network. In the graph representation, all the hosts are represented as nodes. The local hosts are inside a dashed ring, whereas the foreign nodes lie outside the dashed ring. Edges connect all the hosts. Based on the usage of the bandwidth, the hosts are colored accordingly. If the node's color is green, it means utilizing the optimal amount of bandwidth whereas, if the node is red, it uses bandwidth more than it should under ordinary circumstances. Whenever a DDoS attack occurs, it is easier to identify the irregularity from this graph because the bandwidth usage increases drastically and is visible as a large red blob. With hosts, the tree is grouped as leaf nodes and classes as internal nodes. In the tree-map, the host's size represents the number of connections, while the color denotes the host's bandwidth usage relative to other nodes. NetGrok also has other useful techniques embedded in it like overview, zoom and filter, details on demand.

Zhang and Huang [69] have implemented a visual methodology to detect flood attacks -a type of DDoS attack. In this paper, they came up with three coefficients to understand the flood attacks' behaviors and measure the network's performance under the attack situations. The three coefficients are a) the density of the attack that predicts the characteristics of the flood attack; b) the complexity of the system that reflects the capacity of the system to manage flood attacks; and c) the scalability to identify the extent of the effect of the flood attack at the victim's location. In visualization, each flood attack is represented as an assigned node containing its unique attributes. When the attack scale is more significant than 0.5, it means that the system has categorized the corresponding flood attacks as high-level DDoS attacks. The graph model describes floods, and their attributes are a classical undirected graph composed of nodes and edges. The nodes represent flood traffic attributes, and edges are calculated based on the attack scale factor.

Choi et al. [29] have presented a Parallel Coordinates Attack Visualizer (PCAV) to detect attacks like internet worms, scan attacks including DDoS attacks. As the axes of the parallel coordinate plane, the source IP address, destination IP address, destination port, and the packet's average length are set. The main thing that differentiates this paper from all other works is that instead of using packet data to show as the fold lines on the plot, flow data is represented. When put on the coordinate map, these groups of flow lines form a particular shape for each attack type. So, nine unique patterns for nine different types of attacks are identified. These novel signatures provide a simple and effective way for network administrators to identify the attacks and create a response plan. Flow data was used instead of packet data because it dramatically decreases the processing time and provides compatibility with famous routers that export flow data to Cisco routers, such as Net-Flow. They have also demonstrated their results with real-time traffic data, which shows the credibility of their work.

Okada [49] created a Parallel Coordinates version of Time-tunnel (PCTT) for the spotting of intrusion attacks. Time tunnel is a multi-dimensional data visualization tool, and the parallel coordinate version that was made use of in this approach has all the features of parallel coordinate visualization. It uses a combination of PCTT and 2D to 2D visualization to detect intrusion attacks [49]. 2D to 2D visualization uses four dimensions to map data onto a 3D plane. This 2D to 2D visualization feature has been added to the PCTT. Usually, the four attributes that help identify certain network attacks patterns are source IP address, destination IP address, source port number, and destination port number. 2D to 2D visualization also comes in handy for the identification of certain access formats.

Seo et al. [3] developed a system named CCSvis based on cylindrical coordinates to identify various types of network attacks including DDoS attacks. This is similar to the pattern recognition perceived by parallel coordinates. The approach mainly focuses on detecting botnet attacks in Domain Name System (DNS) traffic, one of the deadliest DDoS attacks. CCSvis uses a new method representing the botnets' behavior, also known as zombies, and defines a variety of threat patterns on cylindrical coordinates. For this grasp of the patterns, five attributes are extracted from the DNS packets, including the client IP address, the query type, the domain name, the time-stamp, and the flag, and are represented on cylindrical coordinates. The group of data forms a specific pattern for a different type of attack. Thus, five signatures are identified from these patterns, making the detection of the attacks easier for the network administrators.

Lu et al. [42] have derived a new method based on concentric circle visualization to represent multi-dimensional data related to networks. This method is an effective way to discern DDoS attacks because it can recognize visual patterns. On the parallel coordinate map, since the data is high-dimensional, there are many overlaps. To reduce the clutter that lessens the effectiveness of the visualization, concentric circles are used. They use poly curves to connect values (vertexes) rather than polylines used in parallel coordinate approaches. In the evaluation that was performed at the end, it is shown that this system based on the concentric circle approach reduces clutter by 15% when compared to the parallel coordinate approaches [13]. Using this new method they have devised, they have developed an integrated visual network scanning system called CCScanViewer. The new technique successfully realizes the threat features emerging from a range of networking trends, such as network scan features and DDoS attacks.

Samak et al. [55] represent the patterns in network data flow by using the Space Filling Curves (SFC) which render the collected information as images that display the perceivable patterns. In this paper, the network traffic is viewed as images or video clips sampled over short time intervals. The packet collection that corresponds to the same time slot is a single image frame and will be counted against the strength of the image pixels [55]. Each pixel in the picture corresponds to a particular value in a single (or vector) packet header region. The SFCs come into the picture while mapping the packets' flow properties onto the pixels' coordinates. Their approach to improved SFC clustering properties makes irregularities such as large scale DDoS attacks and scanning events readily detectable relative to other conventional techniques.

Zhang et al. [67] developed a network monitoring tool called NetViewer to detect DDoS attacks and other types of anomalous behavior and to represent them visually. For this purpose, the tool examines the packet headers. This header data is then represented as images or frames. For such a formulation, a set of samples can be seen as a sequence of frames or images. This representation facilitates image analysis and video coding techniques, including scene shift and motion prediction, to display exciting traffic properties on packet header details. The research brings together methodologies from image processing and video analysis to visualization to make real-time traffic patterns possible.

2.2 Research on DDoS attacks in WSN

Garg et al. [32] have done a brief survey about the security constraints that are part of the WSNs and listed out all the DDoS attacks that can be launched in the different layers of the WSN architecture model. The work's main aim is to give a short introduction to all the various types of attacks in each layer so that mechanisms can be developed to curb these attacks.

In [40], all the security attacks have been categorized into various types based on their impacts like data integrity and confidentiality, power consumption, service availability and bandwidth consumption, routing, identity, and privacy. DDoS attacks fall into some of these categories.

Qureshi et al. [53] have done a short survey that mainly focuses on attacks in the physical layer. The work's main aim is to group the attacks that occur in the physical layer, compare several properties like how the attacks take place, the schemes that are used to launch the attacks, the impacts these attacks have on the layer, and finally, the detection and defensive methods that are applied in order to contain these attacks.

Security attacks, including DDoS attacks, can be classified into various categories based on divergent indicators like the attackers' domain, the techniques these attackers use to launch an attack, etc. In [43], based on these indicators, the attacks are categorized into types like passive or active, internal or external, different protocol layer, stealthy or non-stealthy, cryptography or non-cryptography related attacks.

Bharat and Sahoo [25] have done a detailed study of WSNs and have come up with an approach that closely examines the threats and weaknesses in WSNs due to their dynamic nature. Elaborate work has been done to point out the challenges related to security, routing, and middleware. The paper then lists out all the security attacks that occur in each protocol layer and their counteractive measures proposed over the years by researchers to prevent these attacks. The approach also provides a study of data aggregation and energy-efficient routing protocols for WSNs [25].

[44] and [30] present all the DDoS attacks that occur at different layers of the Transmission Control Protocol (TCP) model of the WSNs. Then the defense mechanisms against all of these attacks have also been discussed. This study's main aim is to give a brief overview of the attacks and their counteractive procedures to guide future researchers to come up with better defense steps to provide better protection against DDoS attacks.

2.3 Research on visualization models for Wireless Sensor Networks

V. Nigam et al. [48] have proposed a profile based visualization model to detect DDoS attacks in WSNs. In their work, they continually visualize the network traffic in order to monitor the network. This helps to identify anomalous behavior and to monitor the performance and resource usage of each sensor node. In this scheme, each sensor node is closely monitored, and their corresponding messages and traffic generated in the network are looked upon. When a sensor node forwards too many packets or is creating huge traffic congestion or disturbance in the network, or is using up too much of the network's resources, it is categorized as an attack node. Its traffic is stopped from flowing into the network. Thus, a DDoS attack that consumes up the entire bandwidth, which leads to the ultimate break down of the network, is detected and prevented. The network's performance under other schemes and this profile-based scheme are compared, and the results prove that this model has a considerably better detection rate than others.

SecVizer [54] is a visualization tool capable of monitoring traffic generated from QualNet from wired and wireless networks. This tool visualizes the wireless sensor network's topology from a three-dimensional perspective to show the network's activities. It also uses a parallel coordinate plot to detect security attacks in their early stages itself effectively. The DDoS attacks that take place in WSNs form different patterns in both the topology visualization window and on a parallel coordinate map, which helps in the efficient detection of these deadly attacks.

In [61], the proposed Multi-Directional Scaling - Visualization Of Wormhole (MDS-VOW) is an attack detection visualization model that identifies the wormhole attacks that take place in sensor networks. In this scheme, the existing network is reestablished using multi-directional scaling. Since there will arise errors related to distance estimations [35] while scaling, a surface smoothing scheme is used. When the network is under normal conditions, the network topology is flat. Nevertheless, when there is a wormhole, the network looks like it is being pulled from both ends by the wormhole. Thus, the MDS-VOW method detects the wormhole by visually depicting the anomaly looking like a distortion caused by the wormhole on the renovated network.

Like in [61], the same authors have proposed a visualization-assisted detection of Sybil attacks in WSNs. A Sybil attack is an attack in which one sensor node gains possession of multiple identities and operates as these nodes. In this proposed method also, there is a security module as well as a visualization module. In the network security component, the network topology and its geometrical data [62] are used to identify any Sybil attacks. The visualization module then uses the results from the security module to identify anomalous patterns and fake identities.

Varsha et al. [37] have come up with an early detection of DDoS attacks in WSNs. In this method, the number of transmissions for each node is reduced based on the number of neighbor nodes in their transmission range to prevent packets' flooding by any one node. The network is grouped into grids, and each of these grids has an examiner node. This examiner node computes the number of neighbors for each node and their threshold value of the number of packets. If a huge number of packets are being sent by a node that exceeds its threshold value, the examiner node examines this with the Packet Delivery Ratio (PDR) of its neighbor nodes. If the node's PDR is abnormal, the network stops the node from further communicating with the other nodes, thus detecting the possibility of a DDoS attack in the early stages itself.

In this paper [36], the researchers have proposed a novel visualization model called SRNET for network security in WSN. This system can analyze and detect complicated patterns that occur in the sensor networks. It mainly focuses on selective forwarding attacks and jamming attacks by visualizing the network data onto three multi-coordinated views: a) multi-dimensional crossed view that shows the network topology and helps in monitoring the evolving status [35] of the two types of attacks b) crossed view which is combined with the track view in order to identify the source of the attacks c) track view tracks the source and identifies the pattern of a potential attack.

Lu et al. [41] established an integrated method for detecting Sybil attacks on mobile WSNs by visualizing and evaluating several topology trends. The automatic reordering and evaluation algorithms [36] that have been used in this approach help find the attacker nodes quickly and more efficiently in the network topology when an attack scenario appears. Here, time series analysis has also been used to keep track of the attack duration. This analysis is based on time histograms and a time segmental automated process.

Shi et al. [56] have come up with an attack detection mechanism known as Sensor Anomaly Visualization Engine (SAVE) in WSNs using visualization as a means. It makes use of the integrated approach by using the powerful visual analytic technology and works in three phases: a) data preprocessing, which includes cleaning, organization, and normalization of the data b) the abnormality identification of sensor nodes on a centroid by cluster-based temporal and spatial projection (data points that are not centroid are marked as outliers) c) multifaceted viewing like topology display of the sensor network routing, high-dimensional sensor properties, and correlation-based projection vision and projection dimension vision [35].

In [59], the proposed system is a visual assisted tool called EyeSim for identifying the DDoS attacks in IP-enabled WSNs. This tool is interactive and helps users monitor and identify wormhole links present in the sensor network. It can also be used to identify the anomalous nodes that are responsible for forming the wormhole link. The system can also perform network data analysis based on the routing information available and expose which nodes are working as the attacker nodes. The results prove that EyeSim can effectively detect various wormhole attacks in real-time and have good detection accuracy.

2.4 Summary

Chapter 3 has provided a detailed study of previous works that discuss the various visualization models and systems that the researchers have developed over the years to detect DDoS attacks in traditional wired networks. Then, the research related to DDoS attacks in WSNs has been outlined. Finally, the frameworks based on data visualization to detect DDoS attacks in WSNs have been provided.

Chapter 3

Methodology

This chapter describes the proposed approach. The suggested framework comprises two modules - the dataset generation module and the clustering-based data visualization module. The chapter gives a detailed overview of the tools, algorithms, and libraries used as part of the designed approach.

3.1 Approach

While network researchers surely went several steps ahead and built some of the most technologically advanced wireless networks like the Internet of Things (IoT) and Wireless Sensor Networks (WSNs), the attacker community is not far behind. WSNs are unique in their design and have many advantages over the traditional wired networks. In my opinion, the scope of this WSN is time-critical WSNs. For example, a WSN used to monitor the precipitation of an area of land becoming the victim to a DDoS attack is very rare because the attackers have nothing to gain by attacking this WSN. Another WSN is used to monitor the containers' temperatures inside a chemical factory or nuclear power plant. If this data is tampered with, it can wreak havoc in the form of human life loss and nuclear weapon loss. Since the attacker's main aim in flood type of attacks is to flood the system with useless packets and stop important information from reaching its destination, they target these time-critical systems. So, time-critical sensor networks are the scope of this work.

Because WSNs are infrastructure-less, the nodes in the sensor network tend to be very variable. Due to this dynamicity, the nodes have become a target for several DDoS attacks. DDoS attacks are intrusions where a select system or network is made unavailable to the actual users. These attacks have been a trouble to networks before wireless infrastructure and are now causing distress to the sensor networks. UDP flood attack is a type of DDoS attack that happens in both WSNs and wired networks where the attacker floods the targeted system's ports with UDP datagrams. In turn, the system sends an ICMP packet to the sender, replying that no port is listening to the request it has sent. Thus, the victim machine loses precious computation time and resources in sending these unnecessary packets while it has other important tasks to do. Applying data visualization for detecting DDoS attacks is a fairly new field that has gained attention in the past two decades. The main task at hand is to use this concept in securing WSNs against DDoS attacks, which is the main motivation for this research work. The procedure thus aims at building a visualization-based DDoS detection model in WSNs. This is achieved in two parts.

The first module generates a dataset with both UDP attack and non-attack data. There are very limited datasets available for DDoS attacks in WSNs. Some of them being WSN-DS[19], Captured Zigbee packets from commercial smart home devices [17] etc. In the papers encountered while doing this research, many of the works simulated their datasets using Network Simulator2 (NS2), Cooja, OMNET++, and similar simulating tools. However, the resource consumption features required for designing this model were not available in these datasets. To obtain this data, a dataset is generated by using the Cooja simulator. As mentioned earlier, even though other simulators are available for simulating WSN environments, Cooja was used because it has several advantages over other simulators. The problem with the current simulators is that they can simulate only a single system level at once. Because of this drawback, the developers cannot use the simulator for both high-level algorithm development and low-level development such as device-driver implementations[50]. Whereas with Cooja, cross-level or simulations at different levels like network, operating, and machine code levels can be built simultaneously[50]. So, Cooja is efficient, scalable, flexible, and extensible[50]. It is an open-source simulation software that runs on Contiki operating system. It is simple to use and can very effectively build a WSN environment based on RPL - a type of routing protocol for low-power networks like sensor networks.

The simulation captures the required attributes: energy consumption, power consumption, delay, packet loss, CPU time, and idle time of the sensor nodes when they are under normal conditions and when they are attacked. Sometimes, a task might be vast and time-consuming. Then the nodes take more energy to finish this task than they usually take. Just because the nodes consume more reserves, it cannot be classified as an attack situation. Thus the receiving rate attribute of the nodes further assists in detecting a breakthrough. The attacker mainly achieves his goal of flooding the targeted system by sending too many packets too frequently. That means the attacked nodes have too many incoming packets, i.e., the Received_Packets is very high. From the resultant dataset, only the features mentioned above are preprocessed because only those are useful for detecting attacks. The timers, parameters, conditions, and requirements set and needed to run the simulation are provided in the next chapter.

In the next step, k-means and mini-batch k-means clustering algorithms are applied to the dataset using Python programming language in Spyder Integrated Development Environment (IDE). The conditions for identifying an attacker are specified in the experiment chapter. Clustering is a process of grouping items, things, data, etc., based on some metrics. It is an unsupervised learning method in which the machine is expected to identify similarities and differences between the unsorted items of the dataset and group them into various categories based on these parallels and disparities[23]. Clustering is a continuous process of knowledge discovery[22] that involves judgment and loss. For the proposed approach, clustering techniques work better than classification techniques because of two main reasons[2]:

- The dataset to be dealt with is quite huge.
- The work's objective is to distinguish anomalies from regular data for which clustering is the most suitable choice.

K-means is the most commonly used and effective algorithm when cluster analysis is involved. But network data is enormous and complicated. When such data is provided to k-means, it has to do too much computation to arrive at the results. This is because it tries to work on the entire data at a time, which is not feasible and only leads to mis-clustering and too many outliers. Effective clustering is needed for this work because we are dealing with a time-sensitive and destructive attack situation. If the attacker is successful, it leads to the breakdown of the entire network. Thus, to overcome this limitation, the approach suggests using mini-batch k-means algorithm. This algorithm works similarly to k-means. However, instead of working on the entire data, it divides it into mini batches. By doing this, the algorithm is not under pressure to process everything at once. Thus, it works uniformly and arrives at better clustering results than k-means with less number of data points being wrongly clustered and with limited anomalies[65][31].

So now, all the data points have a label assigned to them, either attack or non-attack. The clustering results of both the algorithms are depicted on scatter plots - a type of visualization method which shows the relationship between the x and y-axis. In the next step, the mini-batch k-means based clustered results are plotted onto a parallel coordinate plot since it is already established that it has better performance over k-means. In the intrusion detection system, visualizing the information is crucial. The reasons why visualization is necessary are discussed as follows:

- Visualization effectively deals with highly noisy and heterogeneous data. The network traffic's complex environment is highly correlated with the features of source address, packet length, port number, TCP flag, and destination address. However, visualization assists in intuitively understanding the traffic scenario even under such complexity.

- It provides perceptual clues for identifying attacks even without understanding statistical algorithms and mathematical functions.

- It provides faster results compared to the anomaly detection models in identifying the attacks. The attack visualization employs pre-defined image patterns rather than training and comparing the data with historical patterns.

There are many types of visualizations like stream graphs, pie charts, line charts, etc., that could depict the sensor network data. For example, the stream graph discussed in the introduction section could depict this data since the number of features to be used is only 7. But there are many other features like Source ID, Destination ID, etc., in the generated dataset, which may help determine the attack if used in the graphical representation. But these features cannot be put all at once on the stream graph because its capacity is only ten features at a time. Similarly, other visualization methods have their shortcomings. The proposed methodology uses a parallel coordinate graph because it has several advantages over other methods which overcome the limitations mentioned above:

- The network data visualization must ensure scalability, which is assured by parallel coordinates with multiple dimensions. The maximum number of axes that could be shown at a time is 25. No other method has this capability.

- Parallel coordinates-based attack visualization makes it easy to understand the correlations, divergences, and trends present even in the most complex datasets.

- Moreover, it can handle continuous as well as categorical data.

The attack data points show up on one end of the cluster axis and the benign points at the other end in different colors on the parallel plot. The plot separates the data from one another, thus detecting the UDP attack. By observing this pattern, the network administrator or user can understand the abnormal conditions and promptly curb the attack. Figure 3.1 gives an overview flowchart of the presented design.

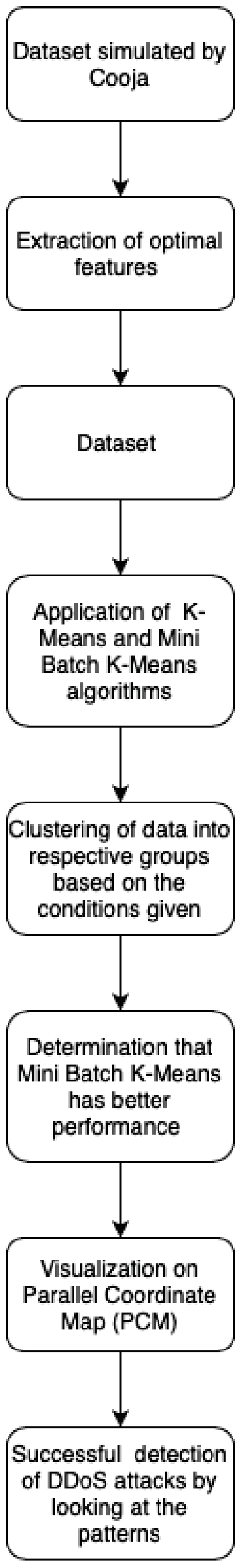

Dataset simulated by Cooja
Extraction of optimal features
Dataset
Application of K-Means and Mini Batch K-Means algorithms
Clustering of data into respective groups based on the conditions given
Determination that Mini Batch K-Means has better performance
Visualization on Parallel Coordinate Map (PCM)
Successful detection of DDoS attacks by looking at the patterns

Figure 3.1: Overview of the proposed methodology

3.2 Tools and Algorithms

This section provides the general functionality of the utilized simulating tool for generating the dataset, the clustering algorithms for categorizing data points, and the visualization method to graphically show the clustered dataset.

- Cooja Simulator
- K-means
- Mini-Batch K-means
- Parallel Coordinates Map (PCM)

3.2.1 Cooja Simulator

Cooja is a java-based network simulator designed for emulating real hardware platforms in WSNs. It is an application that runs on Contiki operating system. Contiki is an open-source software designed especially for devices that work with constrained resources like IoT and WSNs whose base code is written in C programming language. Contiki supports high standard IP protocols like TCP, UDP and HTTP as well as low power protocols like RPL, Constrained Application Protocol (CoAP), IPv6 over Low -Power Wireless Personal Area Networks (6LowPAN)[21].

In Cooja, the sensor nodes are known as motes. There are different types of motes like Wismote, Z1 mote, sky mote, ESB mote, etc. Each of these motes works similarly but differs only in the features like memory, CPU, clock speed, etc. Any type of mote has to be selected for the creation of the network setup. The motes can be placed randomly, in an elliptical manner, or a linear manner. The simulator interface comprises five windows:

- The network window shows the arrangement of the motes. The motes are represented in different colors. The green-colored one is the sink mote, the yellow ones are the sender motes, and the others can be colored according to the user's choice. When the cursor hovers on a mote in the network window, the green ring that appears represents the transmission range of the mote. Details like the radio information of the particular mote, its attributes, and the radio traffic flowing between the motes can also be observed in this window.

- Simulation control window allows the user to control the speed of the simulation, to start, pause and reload the current simulation[46].
- In the note window, the user can jot down critical information about the simulation and can save it to review afterward.
- The timeline window shows events like the communication between layer two and the motes over a period of time.
- The mote output window prints any data that the motes outputs. This is where the actual message communication that takes place between the motes is displayed. The data file can be saved, which will be the dataset generated by running the simulation.

Routing Protocol for Low Power and Lossy Networks (RPL)

The main objective of a routing protocol in any network is to keep track of all the routes available on the network so that any two systems can communicate, routing tables can be built, and decisions on the routing can be made. RPL is thus a routing protocol used in wireless networks which comprise of devices with low power consumption and use IPv6[8].

This protocol is based on the concept of Destination oriented Directed Acyclic Graph (DODAG) which means that the devices are connected so that no loops are formed. A single destination node is known as the root/sink node to which all the information is passed from the nodes present in the network. The formation of the loops is avoided by using the concept of ranks. Each node calculates its position in the network concerning its surrounding nodes, called rank or cost[46]. The sink node to which all the nodes send the information acts as an interface between the network and other sensor networks. Along with Contiki, several other operating systems implement RPL protocol like LiteOS, TinyOS, T-Kernel, EyeOS[8]. To build the topology and for communication to happen between the sensor nodes, the RPL protocol uses four types of control messages:

- DODAG Information Solicitation (DIS): This message is used to request information from other nodes. This is mainly sent in order to discover the neighboring nodes[8].
- DODAG Information Object (DIO): This message is sent as a response to the DIS message to share one node's information with the other node that has requested the information. It is also sent out at times in order to refresh the information of the nodes.
- Destination Advertisement Object (DAO): This message is sent to the parent node to update the information of the child nodes.
- Destination Advertisement Object- Acknowledgement (DAO-ACK): This is an acknowledgment message sent by the parent node to the child or leaf node.

3.2.2 K-Means

The k-means algorithm is the most commonly used clustering algorithm because of its simplicity and effectiveness. It works iteratively to partition the dataset into non-intersecting groups/clusters concerning the number of clusters (k) specified by the user. Each data point of the dataset belongs to only one cluster. The algorithm works so that the data points are assigned to a given cluster by ensuring that the Sum of Squared Errors (SSE) (shown as in Eq.1) between the cluster center and the data point is minimal. Thus, the intra-cluster distance, which is defined as the distance between the cluster center and the data points belonging to a cluster, is minimum. At the same time, the inter-cluster distance defined as the distance between two cluster centers is maximum. This way, all the data points that are similar to one another are grouped, and those that are different fall into different groups. The idea behind the k-means algorithm's working is to resolve the Expectation-Maximization (EM) problem. In E-step, each data point is assigned to the closest cluster. The cluster center is updated in the M-step after a data point has been assigned to the cluster.

$$E = \sum_{i=1}^{k} \sum_{p \in c_i} (p - m_i)^2 \tag{3.1}$$

with k clusters, C the set of objects in a cluster, m the center point of a cluster.

The working of the algorithm is presented below[4]:

Algorithm 1 K-Means clustering algorithm

```
1: Let X = x1,x2,x3,........,xn be the set of data points and V = v1,v2,.......,vc be the set of centres.
2: Randomly select 'c' cluster centers.
3: Calculate the distance between each data point and cluster centres with the Euclidean distance formula:
   $d(p,q) = \sqrt{(p1-q1)^2 + (p2-q2)^2}$   (3.2)
4: Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers.
5: Recalculate the new cluster centre using:
   $v_i = (1/c_i) \sum_{j=1}^{c_i} x_i$   (3.3)
6: Recalculate the distance between each data point and new obtained cluster centers.
7: If no data point was reassigned, then stop; otherwise, repeat from step 3.
```

3.2.3 Mini Batch K-Means

K-means is very popular because of its efficient performance. Nevertheless, when the size of the dataset increases by manifold, the computation that the k-means has to do also increases proportionally because the entire dataset has to be present in the memory while clustering is taking place, which in turn reduces its efficiency. To reduce the algorithm's temporal and spatial cost while performing the clustering[20], an alternative method has been proposed called mini-batch k-means. It is a slightly modified version of k-means.

This algorithm works very similarly to its parent. However, the main difference is that instead of storing the entire dataset in the memory, it stores a small batch of data of a fixed size selected randomly in the memory[20]. In each iteration, a new data point is selected randomly from the entire dataset and is used to update the clusters until convergence is reached. A convex combination of values of the prototypes[20] is used to update the clusters by each mini-batch. The learning curve of the algorithm decreases with each iteration. When the number of iterations increases, the randomly selected data point has little to no change on the clusters. Thus, with the learning rate that falls with the increase in the number of iterations, convergence is achieved. This leads to the saving of significant computational time[65][31].

The working of mini-batch k-means[65] is similar to that of k-means, and the pseudocode of the algorithm is provided below.

Algorithm 2 Mini Batch K-Means clustering algorithm

```
1: Let X = x1,x2,x3,........,xn be the set of data points and B = b1,b2,.......,bc be the number of batches the data points have been divided into.
2: Randomly select 'c' cluster centers.
3: Allocate small batch data to the nearest cluster center using Euclidean distance formula.
4: Update the cluster center iteratively until the cluster center does not change anymore.
```

3.2.4 PCM

A Parallel Coordinate Map (PCM) or simply parallel plot or graph is a type of visualization technique widely used to represent numerical data that is multi-faceted or multi-dimensional. It is beneficial for users when trying to find patterns or associations between the quantitative values. It so often happens that the various numerical values under observation or in a given dataset have different proportions and are measured using different units[63]. In these scenarios, a parallel plot is very appropriate. When the data is put on this graph, it is easy to understand the correlations, parallels, and groups in multi-dimensional datasets and whether these unraveled associations are good, bad, or have no unique relationships. So, the primary purpose of parallel coordinates is to visualize high-dimensional data, which is very difficult to achieve because the information is enormous to look at, and most of the visualization techniques used in day-to-day life will not be able to successfully depict this data without losing some or most of the important information. When most of the lines drawn between these axes are parallel to each other, there is a positive association between these two aspects. Whereas, when these lines criss-cross each other and form patterns similar to X, it means there is a negative relationship between them[7].

In a parallel plot, each feature or attribute is shown as an axis, and these axes are placed parallel to each other vertically with equal space between them. Each data entity is defined on each axis by connected line segments generated from a connected collection of points[63]. After drawing the graph, we get a collection of lines that shows the multi-attribute portrayal of all the data elements[63]. There is one problem to be solved before drawing the map. Since the data can have different units and magnitudes measurements, the data has to be normalized first. The scales of the dimensions have the same units, and now a meaningful comparison can be made between them. One of the most common normalization techniques is to range each axis between its minimum and maximum values[63]. Usually, a zero is placed at one end of the axis and a one at the other end, but they can be replaced with anything according to the users' convenience.

In order to better recognize the hidden relationships, three methods can be used[21]:

- Colouring: With this filter, a single line or multiple interest lines can be colored according to users' choice. This is normally helpful when one particular value has to be compared against others.

- Reordering: Even when the entire dataset is sitting in front of us in the most simplified version, it is still challenging to find what we were searching for with the way the visualization is depicted. Sometimes, by changing the axes' order beside each other, it becomes easier to find things. Also, since a parallel plot is a set of lines, it most often is cluttered. By changing the order of the axes, there is a high possibility of reducing this clutter. So, by reordering the axes, the map's understandability can be enhanced, and the hidden patterns can be identified very easily.

- Brushing: During the analysis of massive datasets, there is a chance of over-plotting, and thus, the analysis of such representation becomes nearly impossible[63]. To overcome this problem, brushing technique can be used. Here, specific data points can be shown while other elements are faded out. This helps highlight specific subsets of data, which reduces the clutter, and the representation becomes aesthetically more attractive, thus disclosing many hidden details.

3.3 Python libraries

Python libraries are pieces of code written in python that have some inbuilt functions. These libraries give access to installing the required modules or packages, which can be used to achieve specified tasks. The section gives a brief elucidation of the libraries that were used in the framework.

NumPy: NumPy short form for Numerical Python plays a significant role in supporting the multi-dimensional, large arrays and matrices, incorporating the numerous high-level mathematical functions for the arrays and matrices operations[60]. It is an extension of the python programming language, providing an efficient interface for storing and operating data buffers with high density. It helps process the arrays with the storage of values in the same data types and eases the arrays' math operations. It provides efficient data operations and storage with the increased size of the array. NumPy arrays are the core of data science tools in python.

Pandas: Pandas library dramatically assists in the analysis and manipulation of the data, especially time series and numerical tables with the offering of data operations and data structures[45]. Pandas package is built on top of NumPy. It supports the implementation of a data frame in an efficient manner. Pandas is based on two data structures, namely series and data frames. Series refers to a one-dimensional structure in the form of a list of items, and data frames refer to the two-dimensional structure in the form of a table with multiple columns. Data frames consist of row and column labels for homogeneous and heterogeneous data types with or without missing data. Pandas enables converting the data structures to data frame objects, missing data handling, histogram or box plotting, and column insertion or deletion[45].

Scikit-learn: Scikit-learn library supports learning and decision-making through the accumulation of different classification, clustering, and regression algorithms. It is a Python module that is built on top of the SciPy package for machine learning. Scikit-learn is an industry-standard package for Python-based data science projects. It provides a concise interface to the variety of machine learning algorithms, supports the handling of data mining and machine learning tasks such as classification, clustering, regression, dimensionality reduction, and model selection[52]. With the incorporation of the efficient versions of the multiple common algorithms, Scikit-learn greatly assists the data scientists in performing the machine learning projects.

Matplotlib: Matplotlib library is the extension of the numerical mathematics that is NumPy, which plays an essential role in embedding the plots into applications through the object-oriented Application Programming Interface (API) based on the general-purpose Graphical User Interface (GUI) toolkits[24]. Matplotlib helps visualize the data as two-dimensional graphs and diagrams, including scatter plots, histograms, and non-cartesian coordinates graphs. It greatly assists the visualization for the data science projects, which is compatible with the different graphics backends and operating systems. Matplotlib is also referred to as the everything-to-everyone and cross-platform approach with the advantage of supporting the dozens of output types and backends within the scientific Python world.

Scatter Plot: A Scatter plot is a type of mathematical diagram representing the data points between two different variables using dots, which is also called the scatter graph and scatter chart. In the scatter plot, each dot's location on the horizontal and vertical axis refers to an individual data point's values, which assists in observing the relationship between the numerical variables.

Plotly: Plotly is an interactive and open-source python plotting library, supporting 40 unique chart or graph types with the coverage of the financial, statistical, scientific, 3-dimensional, and geographic use-cases[58].

Chapter 4

Experiment

This chapter presents the experimental setup for the dataset generation in the Cooja simulator and the clustering-based visualization on a parallel plot. It provides the details about the simulation's hardware and software requirements, the set parameters and timers to run the simulation, the working of the attacker nodes, and presents the final dataset generated. The chapter also details the programming setup for clustering and parallel plot using python and the packages and functions used in the coding.

4.1 Simulation setup and parameters

To successfully run a simulation whose output will be used for designing the detection model, standard network connections with good quality and adequate memory are required. These two requirements can be met with the Cooja simulator, which is designed explicitly for emulating the environments of WSNs with all the security and quality requirements in place.

Contiki Operating System: The open-source operating system of Contiki is mainly designed for resource-constrained devices. Contiki supports a full stack IP network with different high-standard IP protocols and different low-power standard protocols.

Cooja Simulator: It models the proposed simulation system based on the consideration of the network connectivity and protocol performance with the advantage of provisioning the emulation of the real hardware platforms.

Table 4.1 and 4.2 show the hardware and software requirements for implementing the proposed simulation model.

Table 4.1: Hardware requirements

Processor	Pentium Dual-Core CPU E5800 @ 3.2GHZ
Monitor	SVGA
Memory	4096 MB RAM

Table 4.2: Software requirements

Operating System	Ubuntu 16.04 LTS 64bit
Networking tool	Contiki 3.0, Cooja Simulator
Machine learning tool	Scikit-learn
Language for networking	C
Machine learning	Python

4.1.1 Network environment setup

The network environment setup implements the dataset generation using the Cooja simulator on Contiki based on RPL routing protocol. Cooja enables the emulation of the wireless sensor network without the need for a particular mote.

It conducts the experiment on a 100 x 100 sq. meters grid of 40 nodes, including the sink nodes and client nodes. The Contiki motes' implementation is to be performed using the Wismote nodes, with 16KB RAM and 256KB flash memory. It builds the RPL network by considering the Wismote mode for both the static sink node and mobile client nodes. The simulation model assigns the total simulation time as 60 seconds and 50m as the nodes' communication range. To configure the transport layer, it utilizes UDP in which the transmission size of the data for every node is 127 bytes. The simulation model utilizes the Unit Disk Graph Medium (UDGM) with distance loss and packet loss due to the advantage of real-world emulation capability of the shared media collision and lossy links in the IoT network[46]. The MAC and the physical layer are modeled with the 802.15.4 MAC layer protocol and the UDGM propagation model.

Table 4.3 illustrates the simulation parameters for the network environment.

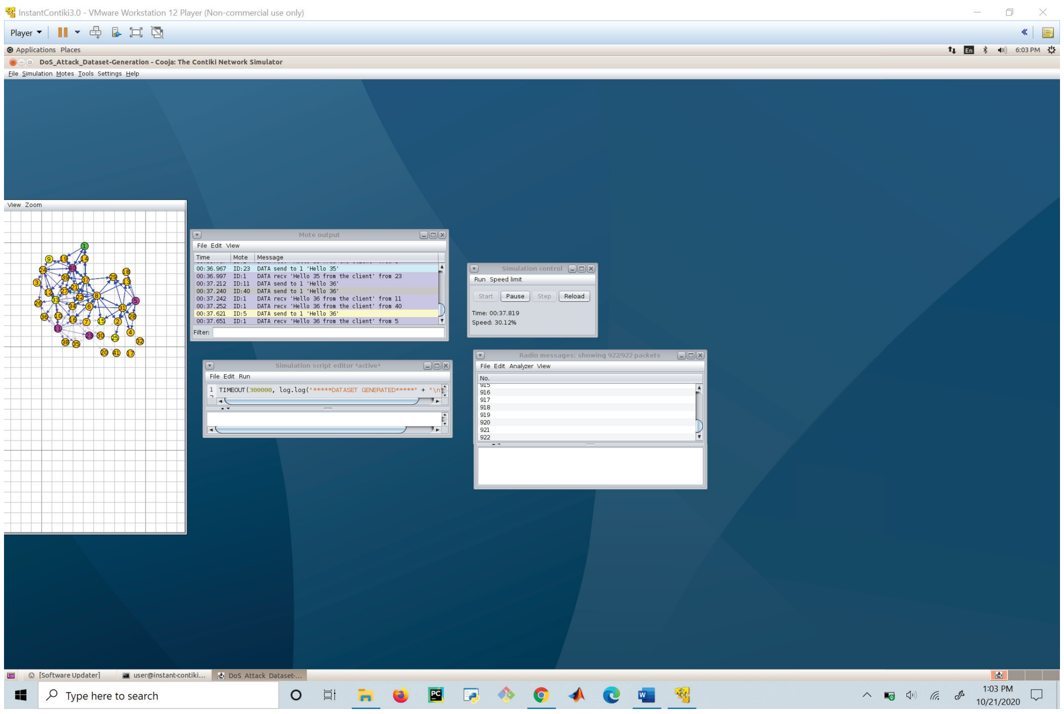
Table 4.3: Network simulation parameters

Number of Client Nodes	40
Number of Server Node	1
Sensor Type	wismote
Propagation Type	Unit Disk Graph Model
Transport Agent	UDP
RPL Mode of Operation	No_downward_route
Data-Transmission Interval	30 seconds
Attacker Interval	1 second
Data Set Generation Time	300 seconds

Cooja builds the WSN environment in the form of a graph (G) with nodes (V) and edges (E) for the generation of the dataset. In the modeled WSN, each edge (E) means that the connected two nodes are within the transmission range. Since Cooja works with RPL protocol, it builds the topology with the help of 4 control messages - DAO, DIS, DIO, and DAO-ACK thus forming a DODAG. Only DIO packets carry some data in this setup, which are messages from child nodes sending their updated information as a response to the DIS messages sent by their parents. Under normal working of the sensor network, the parent nodes send out DIS messages every 30 seconds and send the UDP messages every 30 seconds. This transmission does not create any congestion in the network, and thus the resource consumption by the systems is in control.

Nodes 5, 11, 23, and 40 were selected as the attacker nodes. To launch UDP flood attacks, the attacker node sends UDP packets to a benign node every 1 second, asking for an open port even though it has no use for it. It also sends out DIO messages to the targeted system at random times, sending its information even though the targeted node is not its parent and has not asked for it. The attacker node's transmission interval is very frequent than the time frame used by the benign nodes. So, the system becomes flooded with these useless messages so that resources are consumed in high values to process the packets. Therefore the attacker achieves his goal of exhausting valuable reserves of energy.

Figure 4.1 shows the screenshot of the Cooja simulator. The node green in color is the sink node, and those pink in color are the UDP attack nodes.


InstantContiki3.0 - VMware Workstation 12 Player (Non-commercial use only)
Player
Applications Places
DoS_Attack_Dataset-Generation - Cooja: The Contiki Network Simulator
File Simulation Motes Tools Settings Help
View Zoom
Mote output
File Edit View
Simulation control
Run Speed limit
Start Pause Step Reload
Simulation script editor *active*
File Edit Run
Radio messages: showing 922/922 packets
File Edit Analyzer View
Type here to search

```
Time       Mote   Message
00:36.967  ID:23  DATA send to 1 'Hello 35'
00:36.997  ID:1   DATA recv 'Hello 35 from the client' from 23
00:37.212  ID:11  DATA send to 1 'Hello 36'
00:37.240  ID:40  DATA send to 1 'Hello 36'
00:37.242  ID:1   DATA recv 'Hello 36 from the client' from 11
00:37.252  ID:1   DATA recv 'Hello 36 from the client' from 40
00:37.621  ID:5   DATA send to 1 'Hello 36'
00:37.651  ID:1   DATA recv 'Hello 36 from the client' from 5
```

Time: 00:37.819
Speed: 30.12%

```
TIMEOUT(300000, log.log("*****DATASET GENERATED*****" + "\n
```

```
No.
915
916
917
918
919
920
921
922
```

Figure 4.1: Screenshot of Cooja running the simulation with all the set parameters

4.1.2 Generated dataset

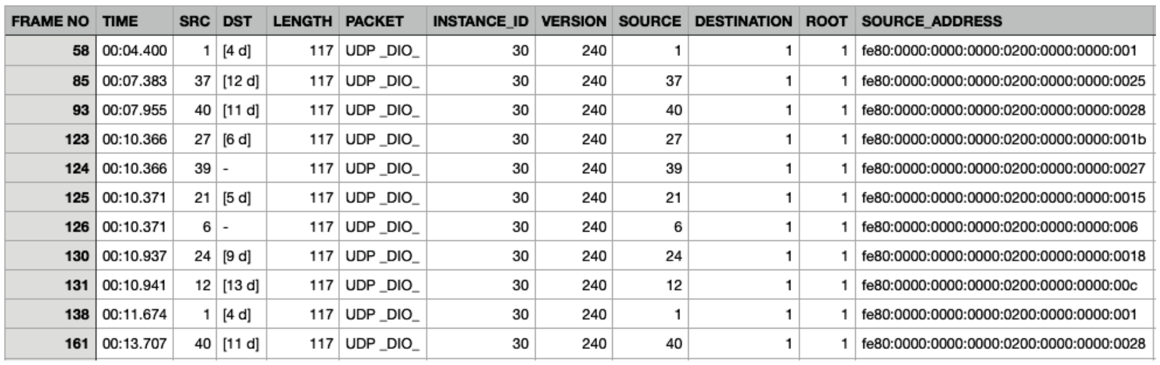
The generated dataset with legitimate nodes and UDP attacker nodes is shown in table 4.4. The simulation results are raw Packet Capture (PCAP) files saved as Comma Separated Values (CSV) file.

Table 4.4: Features of the generated dataset

S.No	Fields	Description
1	Frame No.	Order of message or serial no. of message
2	Time	Message arrival time
3	Source	source ID
4	Destination	Destination ID
5	Instance_ID	Unique identifier in the network (same for all nodes)
6	Version	Version no. of DODAG (same for all nodes)
7	Source_Address	IP address of the source node
8	Destination_Address	IP address of the destination node
9	Root	ID of the root node
10	Node_ID	Node identity
11	Sending_Rate	Data packet sending count of the node
12	Sending_Time	Data packet sending time of the node
13	Energy_Consumption	Energy consumed by the nodes to do a task
14	Power_Consumption	Energy consumed by the nodes per unit time
15	Idle_Time	Time taken to wait for the execution of a task
16	Packet_Loss	No. of failed packets of data while in transmission
17	Delay	Time taken by each packet to reach the server node
18	CPU_Time	CPU time taken for the execution of a task
19	Received_Packets	Received packet count at t time
20	Forwarded_Packets	Forward packet count at t time
21	Hops	Number of hops present in a path
22	Delta_ Time	Inter-packet arrival time
23	Rank	Node's topology level in the tree
25	MOP	Mode of Operation
26	DTSN	Destination Advertisement Trigger Sequence Number

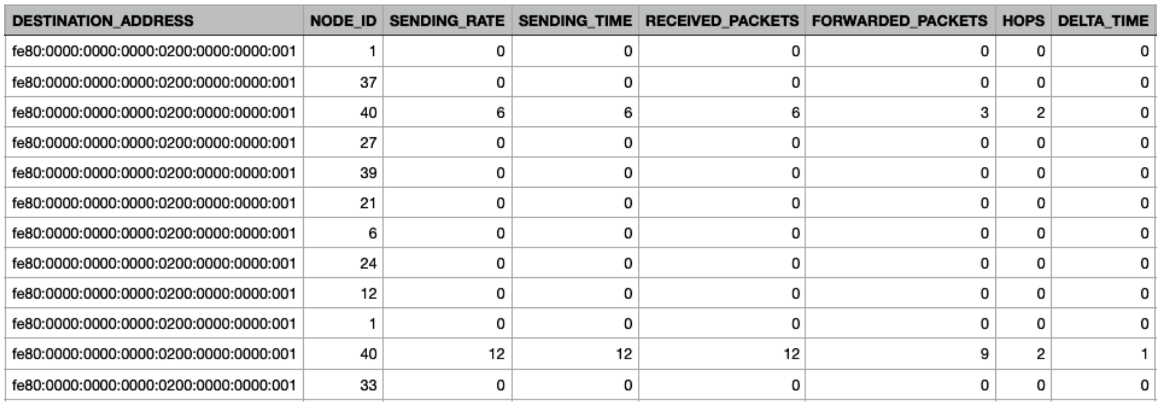
Figures 4.2, 4.3 and 4.4 are the screenshots of the resultant dataset.

From the resultant dataset, only the optimal features like power consumption, energy consumption, packet loss, delay, idle time, CPU time along with Received_Packets are preprocessed and separated into a CSV. The received_packets feature is useful for the categorization of attacker nodes. The resultant dataset after prepossessing is shown in table 4.5



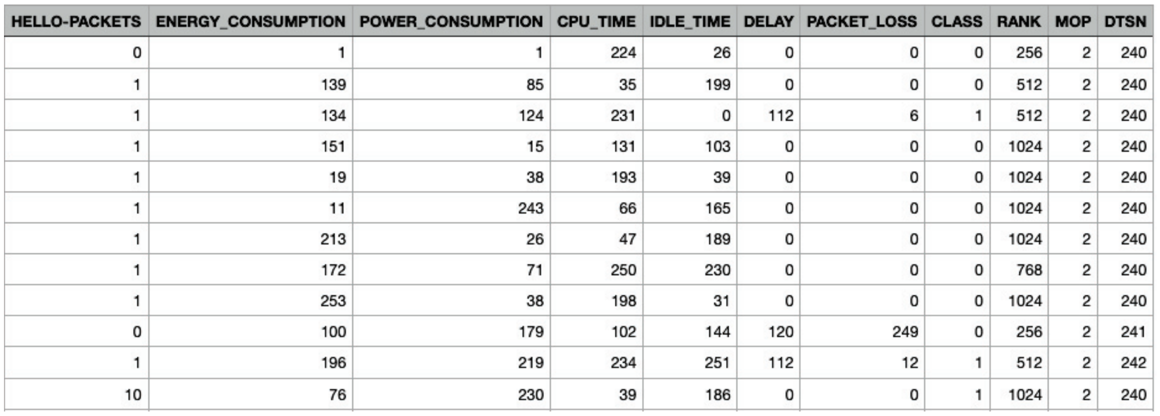
FRAME NO	TIME	SRC	DST	LENGTH	PACKET	INSTANCE_ID	VERSION	SOURCE	DESTINATION	ROOT	SOURCE_ADDRESS
58	00:04.400	1	[4 d]	117	UDP _DIO_	30	240	1	1	1	fe80:0000:0000:0000:0200:0000:0000:001
85	00:07.383	37	[12 d]	117	UDP _DIO_	30	240	37	1	1	fe80:0000:0000:0000:0200:0000:0000:0025
93	00:07.955	40	[11 d]	117	UDP _DIO_	30	240	40	1	1	fe80:0000:0000:0000:0200:0000:0000:0028
123	00:10.366	27	[6 d]	117	UDP _DIO_	30	240	27	1	1	fe80:0000:0000:0000:0200:0000:0000:001b
124	00:10.366	39	-	117	UDP _DIO_	30	240	39	1	1	fe80:0000:0000:0000:0200:0000:0000:0027
125	00:10.371	21	[5 d]	117	UDP _DIO_	30	240	21	1	1	fe80:0000:0000:0000:0200:0000:0000:0015
126	00:10.371	6	-	117	UDP _DIO_	30	240	6	1	1	fe80:0000:0000:0000:0200:0000:0000:006
130	00:10.937	24	[9 d]	117	UDP _DIO_	30	240	24	1	1	fe80:0000:0000:0000:0200:0000:0000:0018
131	00:10.941	12	[13 d]	117	UDP _DIO_	30	240	12	1	1	fe80:0000:0000:0000:0200:0000:0000:00c
138	00:11.674	1	[4 d]	117	UDP _DIO_	30	240	1	1	1	fe80:0000:0000:0000:0200:0000:0000:001
161	00:13.707	40	[11 d]	117	UDP _DIO_	30	240	40	1	1	fe80:0000:0000:0000:0200:0000:0000:0028

Figure 4.2: Screenshot of the resultant dataset



DESTINATION_ADDRESS	NODE_ID	SENDING_RATE	SENDING_TIME	RECEIVED_PACKETS	FORWARDED_PACKETS	HOPS	DELTA_TIME
fe80:0000:0000:0000:0200:0000:0000:001	1	0	0	0	0	0	0
fe80:0000:0000:0000:0200:0000:0000:001	37	0	0	0	0	0	0
fe80:0000:0000:0000:0200:0000:0000:001	40	6	6	6	3	2	0
fe80:0000:0000:0000:0200:0000:0000:001	27	0	0	0	0	0	0
fe80:0000:0000:0000:0200:0000:0000:001	39	0	0	0	0	0	0
fe80:0000:0000:0000:0200:0000:0000:001	21	0	0	0	0	0	0
fe80:0000:0000:0000:0200:0000:0000:001	6	0	0	0	0	0	0
fe80:0000:0000:0000:0200:0000:0000:001	24	0	0	0	0	0	0
fe80:0000:0000:0000:0200:0000:0000:001	12	0	0	0	0	0	0
fe80:0000:0000:0000:0200:0000:0000:001	1	0	0	0	0	0	0
fe80:0000:0000:0000:0200:0000:0000:001	40	12	12	12	9	2	1
fe80:0000:0000:0000:0200:0000:0000:001	33	0	0	0	0	0	0

Figure 4.3: Screenshot of the resultant dataset



HELLO-PACKETS	ENERGY_CONSUMPTION	POWER_CONSUMPTION	CPU_TIME	IDLE_TIME	DELAY	PACKET_LOSS	CLASS	RANK	MOP	DTSN
0	1	1	224	26	0	0	0	256	2	240
1	139	85	35	199	0	0	0	512	2	240
1	134	124	231	0	112	6	1	512	2	240
1	151	15	131	103	0	0	0	1024	2	240
1	19	38	193	39	0	0	0	1024	2	240
1	11	243	66	165	0	0	0	1024	2	240
1	213	26	47	189	0	0	0	1024	2	240
1	172	71	250	230	0	0	0	768	2	240
1	253	38	198	31	0	0	0	1024	2	240
0	100	179	102	144	120	249	0	256	2	241
1	196	219	234	251	112	12	1	512	2	242
10	76	230	39	186	0	0	1	1024	2	240

Figure 4.4: Screenshot of the resultant dataset

Table 4.5: Features of the preprocessed dataset

S.No	Fields	Description
1	Energy_Consumption	Energy consumed by the nodes to do a task
2	Power_Consumption	Energy consumed by the nodes per unit time
3	Idle_Time	Time taken to wait for the execution of a task
4	Packet_Loss	No. of failed packets of data while in transmission
5	Delay	Time taken by each packet to reach the server node
6	CPU_Time	CPU time taken for the execution of a task
7	Received_Packets	Data packet receiving count of the node

4.2 Programming setup

To perform clustering, it is necessary to install the Anaconda software that supports machine learning algorithms' execution using python language on the Spyder IDE.

Table 4.6: Clustering implementation parameters

Language	Python 3.8
Machine Learning Library	NumPy, Pandas, Scikit-learn
Visualization library	Matplotlib
Development Environment	Spyder IDE
Algorithms	K-means and mini-batch K-means

Python: Python is a high-level programming language, which is an interpreted language and more suitable for the development of desktop and web applications.

Anaconda: Anaconda is one of the python distributions that offer a python interpreter along with a set of python tools and packages. It incorporates both conda and python with the scientific computing-focused pre-installed packages. Anaconda python distribution also provides the Python Development Environment (PDE) - Spyder IDE with an editor[1].

Spyder IDE: To execute or run the code written in python programming language, the scientific Spyder IDE is widely used. This IDE has the features of interactive testing, advanced editing, introspection, and debugging. It incorporates useful features such as providing the IPython(Qt) console, enabling the snippets of code execution from the editor in the console, performing step-by-step execution, parsing the files in the editor continuously, and providing a visual warning for the errors[11].

To perform computational modeling and scientific computing, the experimental model necessitates additional libraries or packages instead of using the Python Standard Library (PSL). With the target of creating plots, operating matrices, and applying specialized numerical methods, additional python libraries have been used. The experimental model implements the proposed clustering algorithms to categorize the data points into their respective clusters for DDoS attack detection using additional python libraries such as NumPy and Pandas Scikit-Learn and Matplotlib.

sklearn.cluster.KMeans[9] was used to implement the K-Means clustering algorithm with the number of clusters set to 2 and the maximum number of iterations set to 50. The default value for the maximum number of iterations for the k-means algorithm is 10. The dataset to be clustered is large in size. So, when this default value was set, the algorithm hasn't reached the point of convergence, i.e., the point where all the points are placed in their respective clusters, and the centroids of the clusters do not change. The experiment was conducted by setting the iteration value to 20, 25, 30, 35, 40, 45, and 50. The algorithm did not reach the convergence for any of these values except for value 50. 50 was the minimum iterations required for this dataset to reach convergence.

learn.cluster.MiniBatchKMeans[10] was used to implement the mini-batch k-means algorithm with the number of clusters set to 2, the maximum number of iterations set to 50, and the batch size set to 500. Since mini-batch k-means and k-means are similar to each other, the value of maximum iterations was set to 50 like in k-means for the reason mentioned above. The default value for the batch size is set to 100 in sklearn. The mini-batch k-means algorithm was run with several batches sized like 250, 300, 600, 700, 1000, etc., the algorithm's performance is best when the batch size is set to 500 with the combination of 50 iterations.

pandas.plotting.parallel_coordinates was used to draw the parallel plot to represent the clustered dataset. The attack data belongs to cluster one, and the non-attack data belongs to cluster zero. An axis with zero at the bottom and one at the top in the parallel plot represents the cluster axis. The non-attack data is shown in green color, whereas the attack data is shown in red color.

4.3 Summary

This chapter explains how the experiment was done in detail. It specifies all the conditions under which this demonstration was performed and provided details about the classes, packages, libraries used, and their provided parameters.

Chapter 5

Evaluation Methodology and Result Analysis

This chapter discusses the definitions of the performance measures used to validate the RPL based WSN environment setup and the clustering algorithms. Initially, it provides a comparative performance evaluation of the WSN with and without the impact of UDP attack on the network. Secondly, it explains the clustering results on scatter plots, and their performance is compared against one another using a bar graph. Lastly, it presents the visualization on a parallel plot, which helps identify patterns and detect the UDP attack.

5.1 Definitions of evaluative metrics

To assess the sensor network's performance under attack and non-attack situations, the evaluation metrics such as delay, control overhead, energy consumption, power consumption, processing time, idle time, packet loss, and memory consumption are used. These metrics are later used to draw the axes of the parallel plot. To evaluate the performance of clustering algorithms, homogeneity, accuracy, and v-score are applied. The definitions of the metrics mentioned above are as follows:

Delay: The time taken by each packet to reach the server node in the network.

Control Overhead: The ratio between the number of control messages sent and the number of actual data received at each node.

Energy Consumption: The total amount of energy consumed by the nodes in the network.

Power Consumption: The total amount of energy consumed by the nodes per unit time in the network.

Processing Time: The total time spent by a node to process a network task.

Idle Time: The amount of time taken to wait for the execution of the nodes in the network.

Packet Loss: The total number of failed packets of data when traveling towards their destination in the network.

Memory Consumption: The total amount of memory utilized by the nodes throughout its execution in the network.

Homogeneity: It indicates that each generated cluster comprises only the members or data that belongs to a single class. The homogeneity value lies between zero and one, in which degree one refers to the optimal result of the cluster.

Completeness: It measures the clustering performance in the form of assigning the members belonging to a particular class in the same cluster. The range of completeness is between zero and one.

V-measure: It is the harmonic mean of homogeneity and completeness, which measures the satisfaction of the homogeneity and completeness in their criteria based on the entropy values.

5.2 Comparative performance of the attacker and non-attacker scenario in WSN environment

An assessment of the impact created by the UDP attack on the WSN environment set up by the Cooja simulator is explained with the metrics under attack and non-attack scenarios in this section.

5.2.1 Delay

Figure 5.1 illustrates the delay of the network with and without the attacker scenario. The delay in the network increases with the increase in the number of attackers in the network. The delay-influenced factors in the mobility-based RPL network are packet loss and network holes. The network without attacker scenario obtains 0.0521 seconds of delay, and with attacker attains 0.2564 seconds of delay.

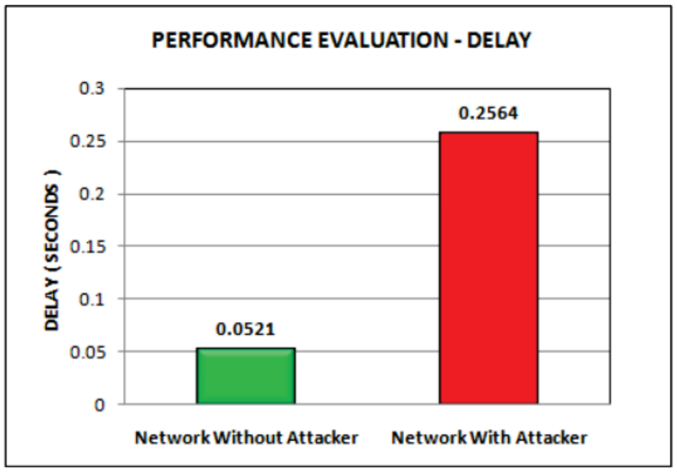


Figure 5.1: Delay in WSN

5.2.2 Control Overhead

Figure 5.2 shows the control overhead. While building the network to discover the routes, control messages are exchanged. The nodes' dynamic movement increases the possibility of the parent node's disconnection from the child nodes, which results in the increased control overhead. The sensor network has a control overhead of 369 packets when there are no attacks but results in an overhead value of 378 packets when the network is affected by the attack.

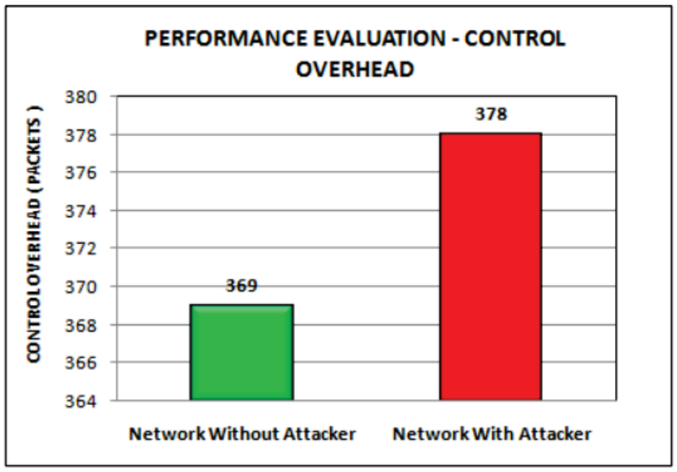


Figure 5.2: Overhead in the RPL based WSN

5.2.3 Energy Consumption

The energy consumption of the network for two scenarios is depicted in Figure 5.3. The consumption of energy in the network is greatly affected by malicious nodes, frequent topology changes, and mobility. The normal WSN consumes 72841.887 mJ of energy, whereas the UDP attack affected network consumes 75528.952 mJ of energy during the execution of the input data.

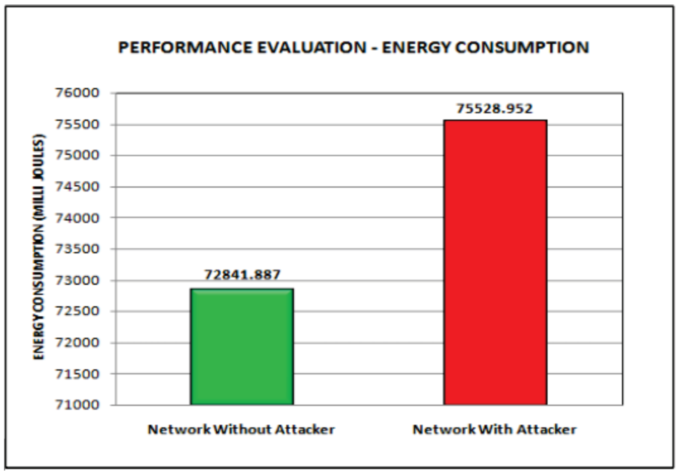


Figure 5.3: Energy consumption of the nodes in WSN

5.2.4 Power Consumption

Figure 5.4 demonstrates the power consumption. The consumption of power in the network is 1697.397 MW when the attacks are absent, whereas the consumption of the power increases to 1712.904 MW when the network is affected by the attack for the same set of data processing.

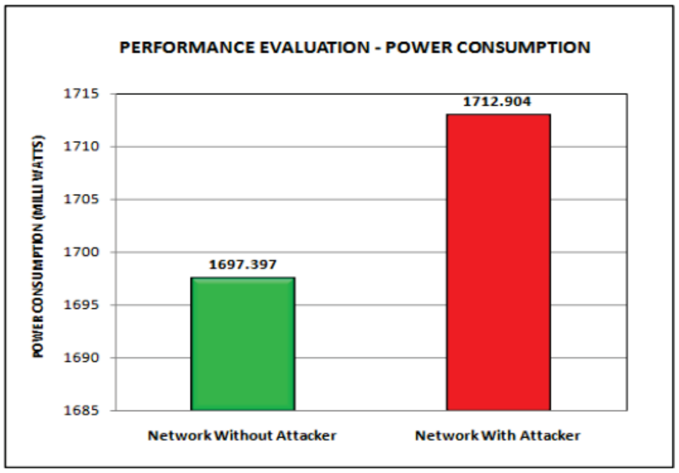


Figure 5.4: Power consumption of the nodes in WSN

5.2.5 Processing Time

Figure 5.5 illustrates the processing time in the RPL based network. The processing time plays a significant role in supporting the execution of time-critical IoT applications, which comprise WSNs. The network affected by the DDoS attack takes 4666226147 ticks to process the input dataset whereas, the normal IoT network spends 4442009067 ticks of processing time to process the same input dataset.

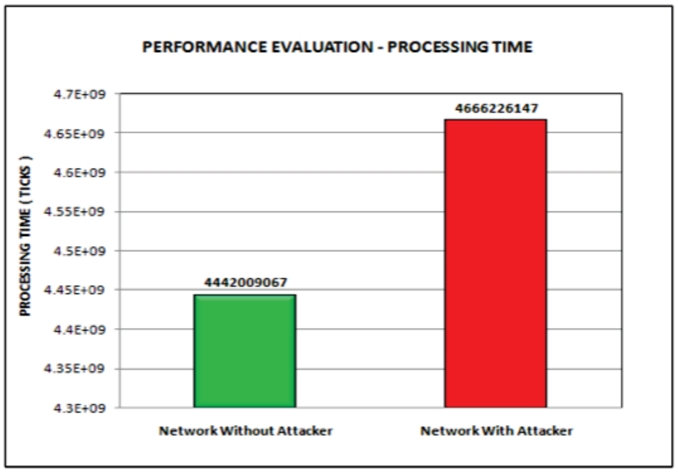


Figure 5.5: Processing time of the nodes

5.2.6 Packet Loss

Figure 5.6 illustrates the packet loss. The affected network exhibits an increased packet loss at 73. The packet loss causes performance degradation in terms of time efficiency and leads to the failure of IoT applications connected to the WSN.

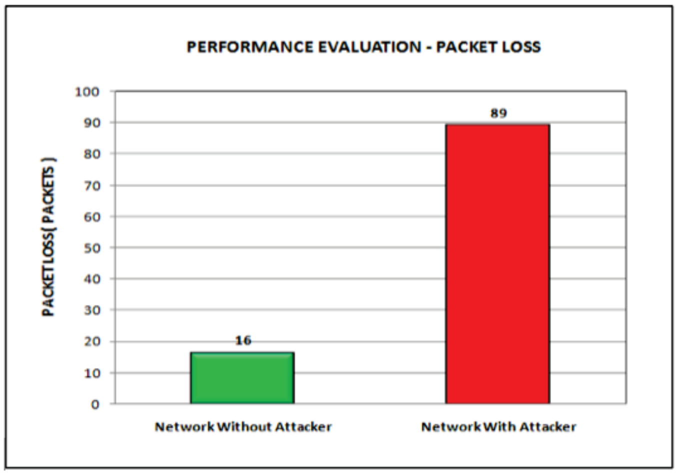


Figure 5.6: Packet loss in WSN

5.2.7 Idle Time

The idle time in the network is shown in Figure 5.7. The idle time in the attack-affected sensor network is 248680271 ticks higher than the typical network's idle time. This is because the attack increases the nodes' inactivity due to the sensor network's delayed data packet transmission.

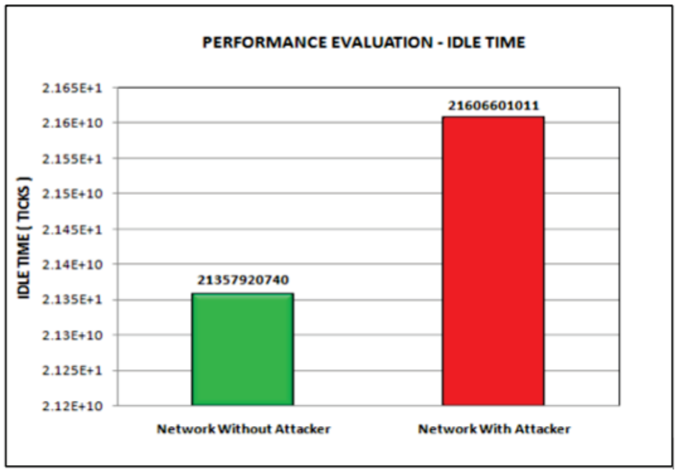


Figure 5.7: Idle time in WSN

5.3 Performance comparison of the clustering algorithms

The results of k-means and mini-batch k-means clustering are illustrated in the form of scatter plots, which show the relationships between the x-axis and y-axis. Their performance results are depicted using a bar graph.

5.3.1 K-means clustering results

Figure 5.8 depicts the scatter plot of the k-means clustering for the corresponding input data. As shown in the figure, k-means distinguishes the data points or nodes as the attacker (blue colored dots) and non-attacker (orange-colored dots) with increased false positives even after filtering or reducing the number of features in the input dataset. The orange dots are scattered, and it can be comprehended that a significant amount of nodes are wrongly clustered.

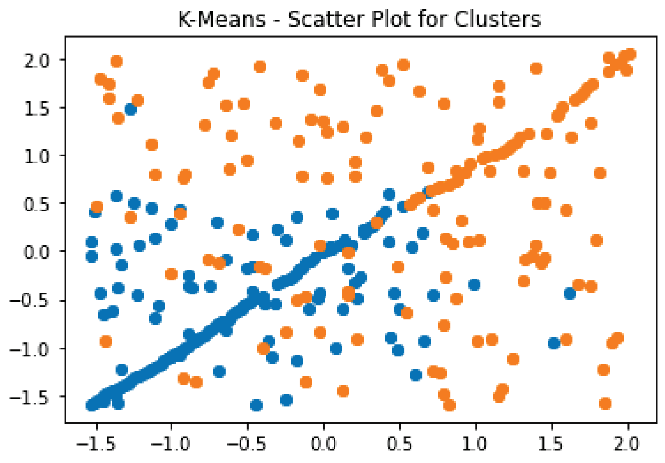


Figure 5.8: Scatter plot for K-means

5.3.2 Mini batch k-means clustering results

The results obtained by the mini-batch k-means clustering algorithm are illustrated in Figure 5.9. Compared to the k-means algorithm, the grouping effectively distinguishes the nodes' characteristics. The clustering is uniform. The attack nodes are shown by orange dots, whereas blue dots show the non-attack data points. (The color associated with the clusters is opposite in the case of k-means). There are very few outliers. Better clustering by mini batch k-means is accomplished by analyzing the mini batches' features rather than analyzing them all at a time.

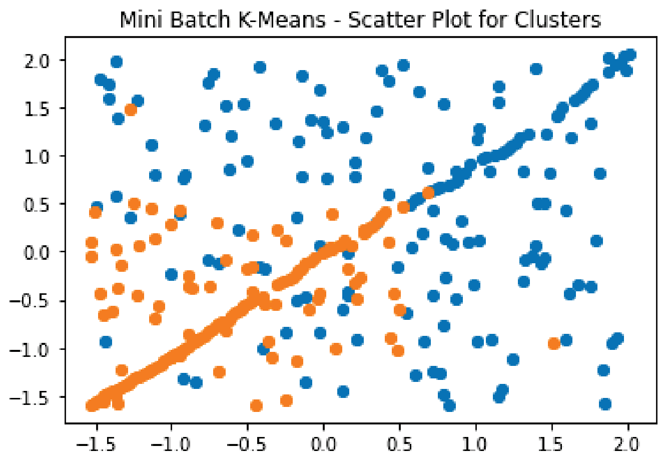


Figure 5.9: Scatter plot for Mini Batch K-Means

The performance evaluation of the clustering algorithms is illustrated in Figure 5.10. The attack detection model with mini-batch k-means achieves higher clustering performance, significantly detecting the UDP attacks. It effectively clusters them into a group with less mis-clustering than the k-means algorithm. Mini-batch k-means clustering yields the homogeneity, completeness, and v-score measure values of 0.0165, 0.0127, and 0.0142, respectively. This is a nearly 10% increase in the performance than k-means clustering whose values lie at 0.0104, 0.0082, and 0.0092 in the same order. It is because analyzing the attack behaviors in mini batches, i.e., a small set of data points, accurately distinguishes between the attack points and non-attack points, which reduces the outliers and mis-clustering. K-means tries to analyze the attack behaviors on the entire dataset at a time, which only increases its computation, degrading its performance and efficiency.

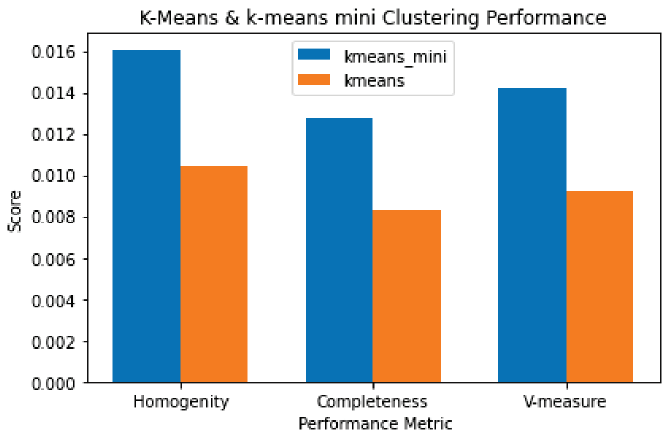
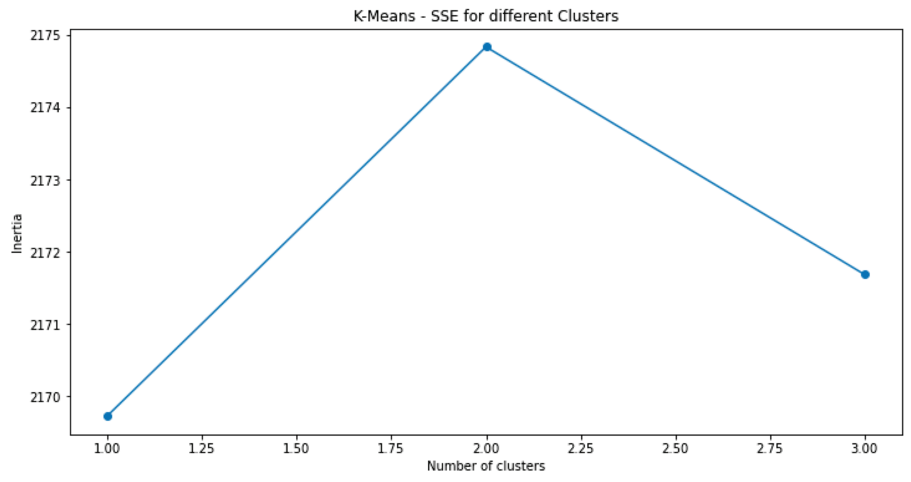


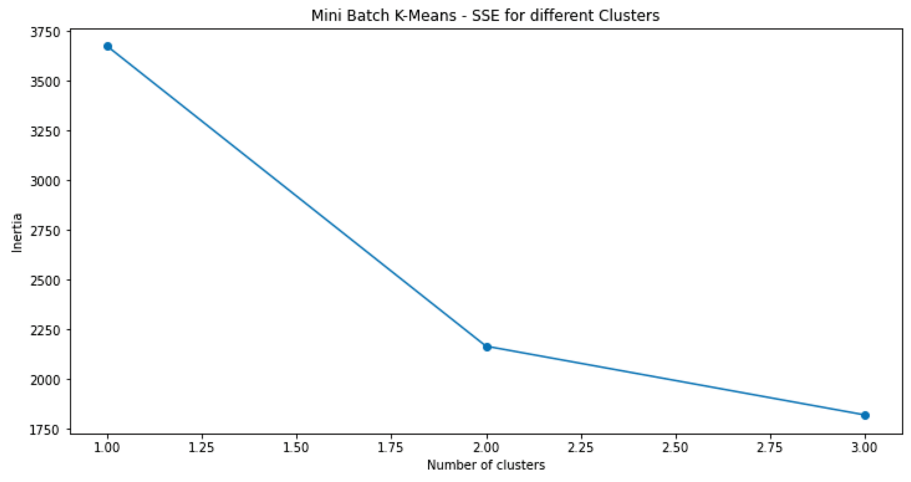
Figure 5.10: Comparative performance evaluation of the clustering algorithms

5.3.3 Observations made during clustering

Two clustering algorithms, namely, k-means and mini-batch k-means, have been applied to cluster the preprocessed dataset. The number of clusters(k) into which the dataset had to be divided was set to two because the distinction between the two sets is obvious. There should be two groups, attack, and non-attack data, after they are grouped. While experimenting, k was set to different numbers like 3, 5, and 10. During these experiments, the batch size was kept constant at 500. Figures 5.11, 5.12, and 5.13 show the plots between inertia or Sum of Squared Errors (SSEs) and the number of clusters - 3, 5, and 10, respectively. For each cluster, the sum of squared error is defined as inertia[47]. So, small inertia means the cluster is dense, or the data points are close to one another[47]. From these graphs, it can be observed that the inertia of mini-batch k-means decreases with the increase in the number of clusters.

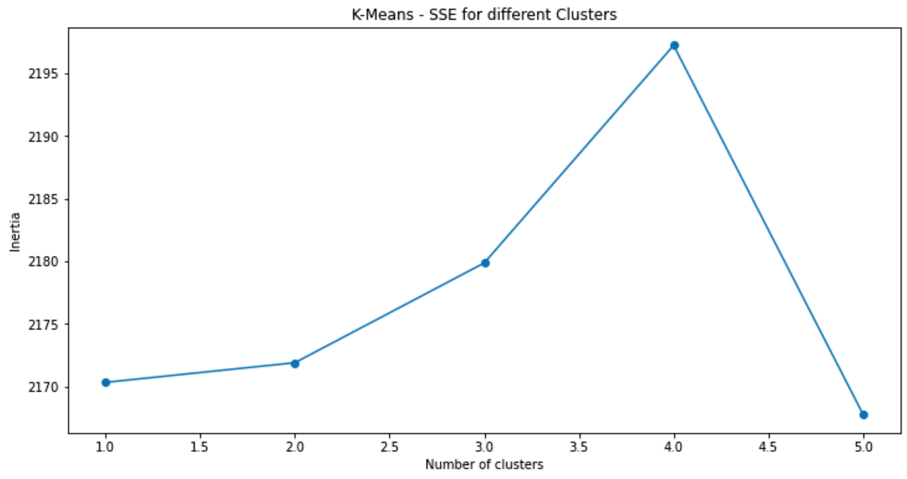


(a)

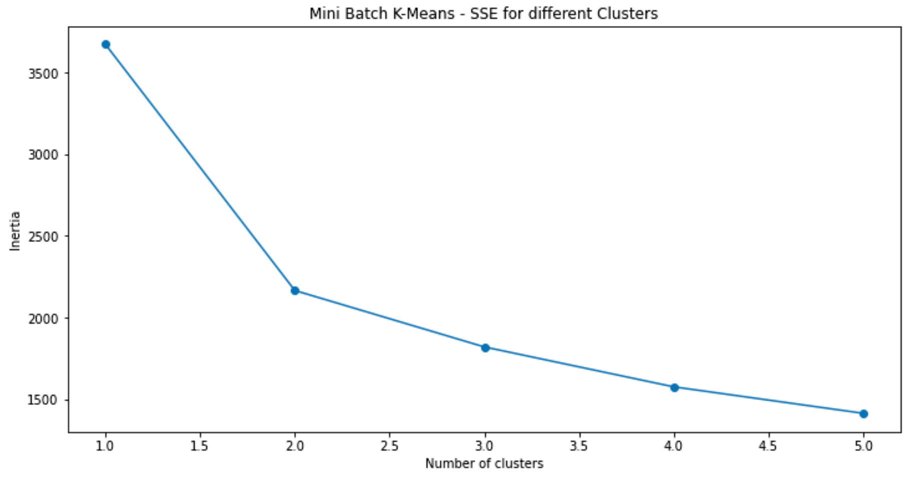


(b)

Figure 5.11: Inertia vs Number of clusters(3) for k-means and mini-batch k-means



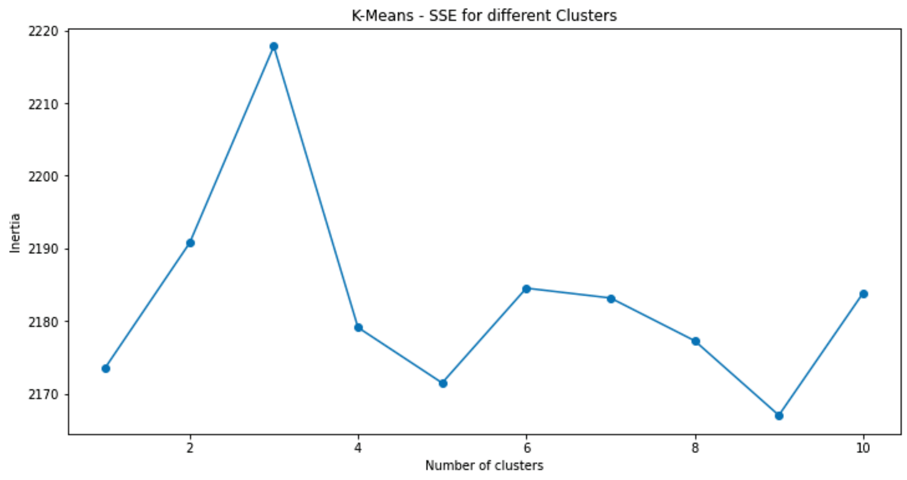
(a)



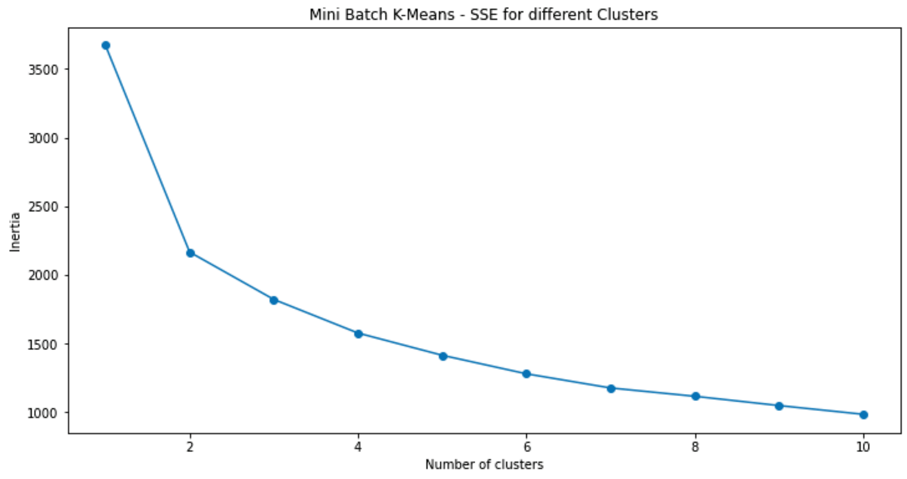
(b)

Figure 5.12: Inertia vs Number of clusters(5) for k-means and mini-batch k-means

The performance of the mini-batch k-means algorithm over k-means is higher by 10% in this experiment. This means their performance is almost at level to each other. This is because mini-batch k-means give better cluster results under some circumstances, and not all of them are met in this experiment. When the dataset is huge with many attributes or features, it is better than k-means because the computational cost reduces dramatically by storing only a part of the dataset in batches in the memory while performing clustering. However, the difference in the cost is huge when the number of clusters is large, and the batch size is small[20]. Nevertheless, in this experiment, the



(a)



(b)

Figure 5.13: Inertia vs Number of clusters(10) for k-means and mini-batch k-means

number of clusters is deficient, i.e., only 2. So, the difference is not very much. The clustering quality is better than that of k-means, but it improves further when the number of clusters increases.

5.4 Visualization on PCM

The parallel coordinates plot of the attack detection model based on mini-batch k-means clustering is depicted in Figure 5.14. It was already established that mini-batch k-means has better performance over k-means. Thus mini-batch k-means based clustered results are visualized on the parallel coordinate graph. The network performance metrics such as energy consumption, power consumption, processing time, CPU time, idle time, delay, and packet loss form the plot's axes. After clustering is applied, each data point is assigned a class, either zero (blue color) or one (red color). The datum that has label zero belongs to the non-attack set, and those that have label one belong to the attack set. The label is also appended to the parallel plot as an axis with zero at the bottom and one at the top. The clustered data points belonging to their respective groups branch out on the plot. Their representation in different colors clearly distinguishes them and makes it explicit for the user or network administrator to spot atypical situations like attacks. Since the dataset is quite large, the graph becomes cluttered by drawing all points at once. So, the metrics are restricted between 100 and 150 to distinguish the two groups effectively. In the following section, different graphs are drawn to understand better how the designed system detects attack scenarios. The cluster axis lies in the center, and the axes on the left and right side of this axis are changed to form various combinations.

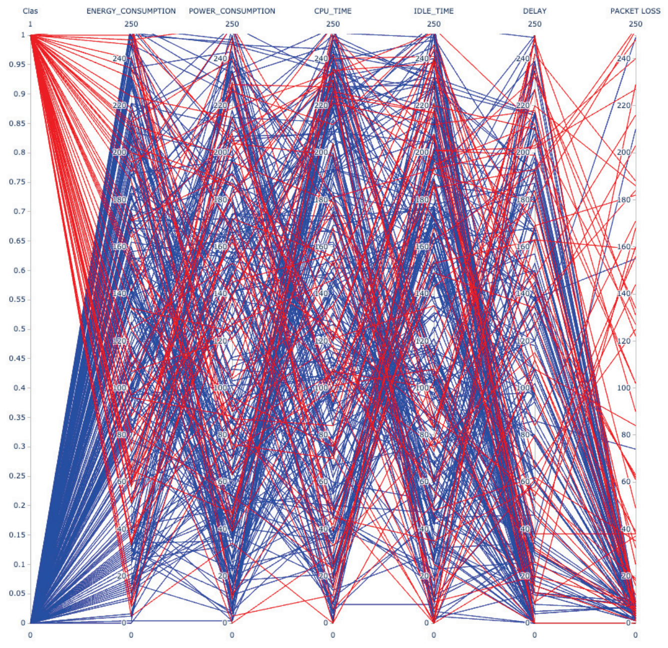


Figure 5.14: Visualization on Parallel Coordinates Plot

5.4.1 Energy Consumption Vs. Power Consumption

Figure 5.15 illustrates the performance of the RPL based WSN with the combination of energy consumption and power consumption while applying the attack detection model.

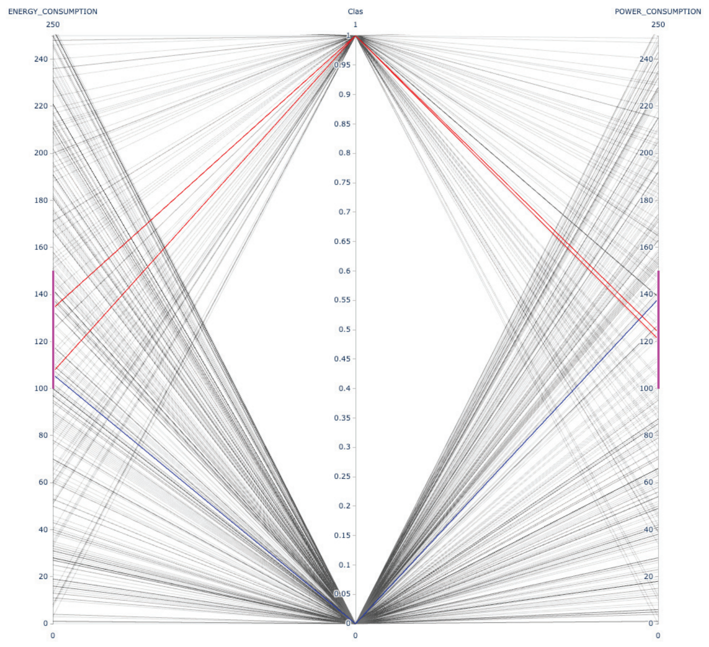


Figure 5.15: Energy Consumption Vs. Power Consumption

5.4.2 CPU time Vs. Idle time

The visualization of the CPU time and idle time performance in the sensor network with the result of the attack detection model is depicted in Figure 5.16.

5.4.3 Delay and Vs. Packet loss

Figure 5.17 visualizes the relationship between the delay and packet loss performance measures using the parallel coordinates model.

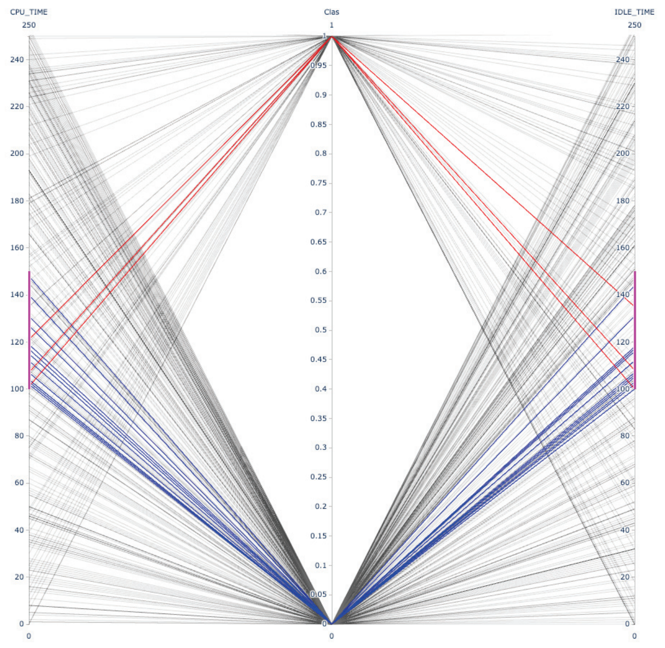


Figure 5.16: CPU time Vs. Idle time

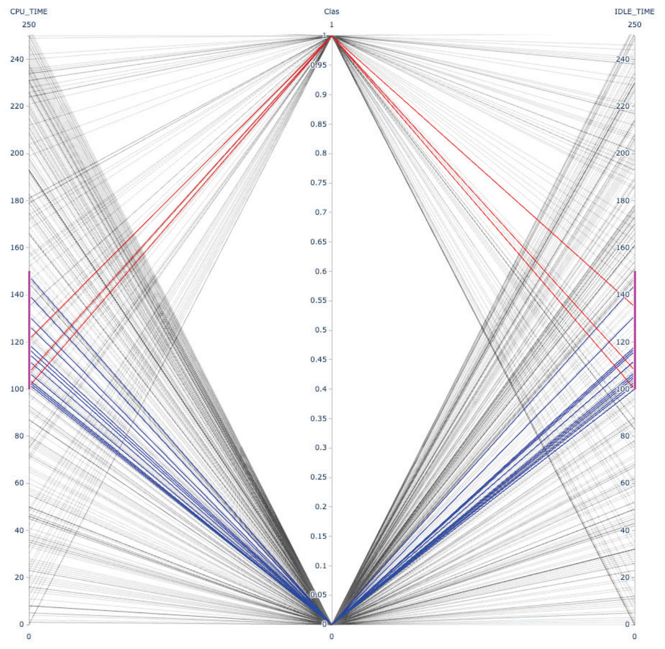


Figure 5.17: Delay Vs. Packet Loss

5.5 Summary

This chapter presented the result analysis of the impact created by the UDP attack on the sensor network. This chapter has shown the performance improvement by the mini-batch k-means clustering algorithm in the visualization-based attack detection model compared to the k-means clustering with the help of evaluation metrics such as homogeneity, completeness, and v-measure. Furthermore, the visualization of the attack scenario using parallel coordinates is illustrated.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

DDoS attacks have been, and more powerful versions of them are still being used as potential methods to disrupt the networks' security. Although several visualization-based detection models have been developed for network security of traditional wired networks, this idea has not been explored much in securing Wireless sensor Networks (WSNs), which predominate today's network systems across the globe. This is because these networks are dynamic, can be built with minimal resources, and are cost-effective. While researchers are trying to tap into the full potential of WSNs, the attackers have also taken advantage of these sensor networks' dynamic nature. They have launched many types of DDoS attacks. Thus, in this thesis work, a visualization-based DDoS attack detection model for WSNs has been proposed.

In this approach, a dataset containing non-attack and UDP attack data was generated to identify optimal features for attack detection like energy consumption, power consumption, packet loss, delay, CPU time, and idle time. Under normal conditions, the sensor nodes use the required amount of reserves to complete a task. However, under attack situations, they tend to consume higher than usual amounts of these resources. Not all conditions under which a sensor node uses more energy than usual can be suspected as attack conditions. Thus another feature called Received_Packets (number of packets received by a node per unit time) also helps distinguish between the attacker and non-attacker nodes. Then, k-means and mini-batch k-means algorithms are applied to the dataset to separate the data into two apparent groups with the parameters mentioned earlier. The clustered data points are then visualized onto a parallel coordinate graph to identify the unique patterns they form and discern attack points from non-attack points. The malicious points lie on one edge of the cluster axis, whereas the benign points lie on the other end. This branching out of the data points shows that the proposed model quickly identifies an attack situation, and anybody using it can tell so by merely looking at the graph. The comparative performance analysis of both the clustering algorithms determines that mini-batch k-means has an improvement of 10% over the k-means algorithm. This reveals that the developed system is accurate and straightforward in detecting UDP flood attacks in WSNs and has made a notable contribution to the field of network security in WSNs.

6.2 Future Work

The research work focuses on detecting flood types of DDoS attacks, namely UDP attacks. When a system in the sensor network is under a flood attack, the main change is that the targeted system consumes more resources than under regular conditions, creating a bottleneck in the entire sensor network. The work's main aim was to focus on the effects of flood attacks on the sensor nodes' resource consumption. In other types of DDoS attacks in WSNs like a wormhole, blackhole, or sinkhole attack, the main aim of the attacker is to either steal the information by attracting other nodes to send the information to it by showing the shortest path or to drop the packets so that important information can be stopped from reaching the destination. These attacks do not create noticeable congestion in the network by using the systems' resources or the network. So, in my opinion, this work would be best applicable to detect other types of flood DDoS attacks rather than non-flood attacks.

The possible future directions of this research work could be:

Usage of a different clustering algorithm: In this work, the k-means clustering algorithm was used as the benchmark over which the mini-batch k-means algorithm was used to improve the system's performance with better clustering results. Any other algorithm could be used as an alternative for better accuracy in clustering results.

Detection of different types of DDoS attacks: The proposed model is designed to detect only UDP flood attacks in WSNs. It can be improved further to detect different types of attacks and multiple intrusions instead of a single attack.

Bibliography

[1] Anaconda (Python distribution). https://en.wikipedia.org/wiki/Anaconda_(Python_distribution). [Online; accessed Jan-2021].

[2] Clustering — When You Should Use it and Avoid It. https://www.explorium.ai/blog/clustering-when-you-should-use-it-and-avoid-it/. [Online; accessed Jan-2021].

[3] Cylindrical coordinates security visualization for multiple domain command and control botnet detection. *Computers and Security*, pages 141–153.

[4] Data Clustering Algorithms - k-means clustering algorithm. https://sites.google.com/site/dataclusteringalgorithms/k-means-clustering-algorithm. [Online; accessed Jan-2021].

[5] Data visualization. https://en.wikipedia.org/wiki/Data_visualization. [Online; accessed Jan-2021].

[6] Line charts — mekko graphics. https://www.mekkographics.com/resources/charts-by-type/line/. [Online; accessed Jan-2021].

[7] Parallel coordinates. https://en.wikipedia.org/wiki/Parallel_coordinates#CITEREFInselberg1997. [Online; accessed Jan-2021].

[8] RPL (IPv6 Routing Protocol for LLNs). https://en.wikipedia.org/wiki/RPL_(IPv6_Routing_Protocol_for_LLNs). [Online; accessed Jan-2021].

[9] sklearn.cluster.KMeans. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html. [Online; accessed Jan-2021].

[10] sklearn.cluster.MiniBatchKMeans. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html. [Online; accessed Jan-2021].

[11] Spyder - Anaconda Documentation. https://docs.anaconda.com/anaconda/user-guide/tasks/integration/spyder/. [Online; accessed Jan-2021].

[12] Stream Graph — Data Viz Project. https://datavizproject.com/data-type/stream-graph/. [Online; accessed Jan-2021].

[13] Treemap and sunburst components. https://docs.oracle.com/middleware/1212/adf/ADFUI/dv_treemap.htm#ADFUI12939. [Online; accessed Jan-2021].

[14] UDP Flood. https://www.imperva.com/learn/ddos/udp-flood/. [Online; accessed Jan-2021].

[15] UDP Flood Attack. https://www.cloudflare.com/en-ca/learning/ddos/udp-flood-ddos-attack/. [Online; accessed Jan-2021].

[16] Wireless sensor networks. https://en.wikipedia.org/wiki/Wireless_sensor_network. [Online; accessed Jan-2021].

[17] Akestoridis, Dimitrios-Georgios and Harishankar, Madhumitha and Weber, Michael and Tague, Patrick. CRAWDAD dataset cmu/zigbee-smarthome (v.2020-05-26). http://www.crawdad.org/cmu/zigbee-smarthome/20200526/. [Online; accessed Jan-2021].

[18] Ahmad Abed Alhameed Alkhatib and Gurvinder Singh Baicher. Wireless sensor network architecture. In *International conference on computer networks and communication systems (ICNCS 2012)*, volume 35, pages 11–15, 2012.

[19] Iman Almomani, Bassam Kasasbeh, and Mousa AL-Akhras. Wsn-ds: A dataset for intrusion detection systems in wireless sensor networks. *Journal of Sensors*, 2016:1–16, 01 2016.

[20] Javier Béjar Alonso. K-means vs mini batch k-means: a comparison. 2013.

[21] Sobhan babu B., Lakshmi Padmaja P., Ramanjaneyulu T., Lakshmi Narayana I., and Srikanth K. Role of COOJA Simulator in IoT. *International Journal of Emerging Trends Technology in Computer Science (IJETTCS)*, Volume 6, Issue 2, March - April 2017:pp. 139–143.

[22] Bansal, Shubham. Cluster analysis. https://en.wikipedia.org/wiki/Cluster_analysis#Applications. [Online; accessed Jan-2021].

[23] Bansal, Shubham. Supervised and Unsupervised learning - GeeksforGeeks. https://www.geeksforgeeks.org/supervised-unsupervised-learning/. [Online; accessed Jan-2021].

[24] Paul Barrett, J. Hunter, J.T. Miller, J.-C Hsu, and P. Greenfield. matplotlib – a portable python plotting package. 12 2005.

[25] Bharat Bhushan and Gadadhar Sahoo. Recent advances in attacks, technical challenges, vulnerabilities and their countermeasures in wireless sensor networks. *Wireless Personal Communications*, 98, 01 2018.

[26] Ryan Blue, Cody Dunne, Adam Fuchs, Kyle King, and Aaron Schulman. Visualizing real-time network resource usage. In *In Proc. Visualization for Computer Security: 5th Int. Workshop, Vizsec 2008*, page 119. Springer, 2008.

[27] Miriam Carlos-Mancilla, Ernesto Lopez-Mellado, and Mario Siller. Wireless sensor networks formation: Approaches and techniques. *Journal of Sensors*, 2016:1–18, 03 2016.

[28] Siming Chen, Fabian Merkle, Hanna Schäfer, Cong Guo, Hongwei Ai, Xiaoru Yuan, and Thomas Ertl. Vast 2013 minichallenge 3: Annette collaboration oriented visualization of network data. 01 2013.

[29] Hyunsang Choi, Heejo Lee, and Hyogon Kim. Fast detection and visualization of network attacks on parallel coordinates. *Computers and Security*, 28(5):276–288, July 2009.

[30] Munish Dhar and Rajeshwar Singh. A review of security issues and denial of service attacks in wireless sensor networks. *International Journal of Computer Science and Information Technology Research*, 3(1):27–33, 2015.

[31] Ali Feizollah, Nor Anuar, Rosli Salleh, and Fairuz Amalina. Comparative study of k-means and mini batch k- means clustering algorithms in android malware detection using network traffic analysis. 08 2014.

[32] Sunil Ghildiyal, Amit Kumar Mishra, Ashish Gupta, and Neha Garg. Analysis of denial of service (dos) attacks in wireless sensor networks. *IJRET: International Journal of Research in Engineering and Technology*, 3:2319–1163, 2014.

[33] Nazrul Hoque, Dhruba K Bhattacharyya, and Jugal K Kalita. A novel measure for low-rate and high-rate ddos attack detection using multivariate data analysis. In *2016 8th International Conference on Communication Systems and Networks (COMSNETS)*, pages 1–2. IEEE, 2016.

[34] Hulme, George. DDoS explained: How distributed denial of service attacks are evolving. `https://www.csoonline.com/article/3222095/ddos-explained-how-denial-of-service-attacks-are-evolving.html`. [Online; accessed Jan-2021].

[35] E. Karapistoli and A. A. Economides. Wireless sensor network security visualization. In *2012 IV International Congress on Ultra Modern Telecommunications and Control Systems*, pages 850–856, 2012.

[36] Eirini Karapistoli, Panagiotis Sarigiannidis, and Anastasios Economides. Srnet: A real-time, cross-based anomaly detection and visualization system for wireless sensor networks. pages 49–56, 10 2013.

[37] Kanchan Kaushal and Varsha Sahni. Early detection of ddos attack in wsn. *International Journal of Computer Applications*, 134:14–18, 01 2016.

[38] Keary, Tim. DoS vs DDoS attacks: The Differences and How To Prevent Them. `https://www.comparitech.com/net-admin/dos-vs-ddos-attacks-differences-prevention/`. [Online; accessed Jan-2021].

[39] Kumar, Shivam. Wireless Sensor Network (WSN). `https://www.geeksforgeeks.org/wireless-sensor-network-wsn/`. [Online; accessed Jan-2021].

[40] C. P. Lee, J. Trost, N. Gibbs, Raheem Beyah, and J. A. Copeland. Visual firewall: real-time network security monitor. In *IEEE Workshop on Visualization for Computer Security, 2005. (VizSEC 05).*, pages 129–136, 2005.

[41] Aidong Lu, Weichao Wang, Abhishek Dnyate, and Xianlin Hu. Sybil attack detection through global topology pattern visualization. *Information Visualization*, 10:32–46, 01 2011.

[42] Liangfu Lu, Jiawan Zhang, Mao Huang, and Lei fu. A new concentric-circle visualization of multi-dimensional data and its application in network security. *J. Vis. Lang. Comput.*, 21:194–208, 08 2010.

[43] Teodor-Grigore Lupu. Main types of attacks in wireless sensor networks. In *Proceedings of the 9th WSEAS International Conference on Signal, Speech and Image Processing, and 9th WSEAS International Conference on Multimedia, Internet amp; Video Technologies*, SSIP '09/MIV'09, page 180–185, Stevens Point, Wisconsin, USA, 2009. World Scientific and Engineering Academy and Society (WSEAS).

[44] Djamel Mansouri, Lynda Mokddad, Jalel Ben-Othman, and Malika Ioualalen. Preventing denial of service attacks in wireless sensor networks. In *2015 IEEE International Conference on Communications (ICC)*, pages 3014–3019. IEEE, 2015.

[45] Wes Mckinney. pandas: a foundational python library for data analysis and statistics. *Python High Performance Science Computer*, 01 2011.

[46] Tayyab Mehmood. Cooja network simulator: Exploring the infinite possible ways to compute the performance metrics of iot based smart devices to understand the working of iot based compression routing protocols, 2017.

[47] Robert Miller. Clustering: Why to use it. `https://towardsdatascience.com/clustering-why-to-use-it-16d8e2fbafe`, 2017. [Online; accessed March-2021].

[48] V. Nigam, S. Jain, and K. Burse. Profile based scheme against ddos attack in wsn. In *2014 Fourth International Conference on Communication Systems and Network Technologies*, pages 112–116, 2014.

[49] Y. Okada. Network data visualization using parallel coordinates version of time-tunnel with 2dto2d visualization for intrusion detection. In *2013 27th International Conference on Advanced Information Networking and Applications Workshops*, pages 1088–1093, 2013.

[50] Fredrik Osterlind, Adam Dunkels, Joakim Eriksson, Niclas Finne, and Thiemo Voigt. Cross-level sensor network simulation with cooja. *Local Computer Networks, Annual IEEE Conference on*, 0:641–648, 11 2006.

[51] Fauzi Othman and Khairunnisa Shazali. Wireless sensor network applications: A study in environment monitoring system. *Journal of Procedia Engineering*, 41:1204 – 1210, 12 2012.

[52] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011.

[53] Waseem Raja, Majid Anwar, Anazida Bakhtiari, Abdul Zainal, Kashif Abdullah, and Kashif Qureshi. Security issues and attacks in wireless sensor network. 30:1224–1227, 06 2014.

[54] Giovani Rimon Abuaitah. *Trusted Querying over Wireless Sensor Networks and Network Security Visualization.* PhD thesis, Wright State University, 2009.

[55] Taghrid Samak, Adel El-Atawy, E. Al-Shear, and Mohamed Ismail. A novel visualization approach for efficient network-wide traffic monitoring. pages 1 – 7, 05 2007.

[56] Lei Shi, Qi Liao, Yuan He, Rui Li, Aaron Striegel, and Zhong Su. Save: Sensor anomaly visualization engine. pages 201–210, 10 2011.

[57] Arjun Srinivasan, Matthew Brehmer, Bongshin Lee, and Steven Drucker. What's the difference?: Evaluating variations of multi-series bar charts for visual comparison tasks. pages 1–12, 04 2018.

[58] I. Stančin and A. Jović. An overview and comparison of free python libraries for data mining and big data analysis. In *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 977–982, 2019.

[59] N. Tsitsiroudi, P. Sarigiannidis, E. Karapistoli, and A. A. Economides. Eyesim: A mobile application for visual-assisted wormhole attack detection in iot-enabled wsns. In *2016 9th IFIP Wireless and Mobile Networking Conference (WMNC)*, pages 103–109, 2016.

[60] Stéfan van der Walt, S. Colbert, and Gael Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science Engineering*, 13:22 – 30, 05 2011.

[61] Weichao Wang and Bharat Bhargava. Visualization of wormholes in sensor networks. In *Proceedings of the 3rd ACM Workshop on Wireless Security*, WiSe '04, page 51–60, New York, NY, USA, 2004. Association for Computing Machinery.

[62] Weichao Wang and Aidong Lu. Visualization assisted detection of sybil attacks in wireless networks. pages 51–60, 01 2006.

[63] Weitz, Darío. Parallel Coordinates Plots - Why & How, Storytelling with Parallels. `https://towardsdatascience.com/parallel-coordinates-plots-6fcfa066dcb3`. [Online; accessed Jan-2021].

[64] Chunyuan Wu, Shiying Sheng, and Xiaoju Dong. Research on visualization systems for ddos attack detection. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2986–2991. IEEE, 2018.

[65] B. Xiao, Z. Wang, Q. Liu, and X. Liu. Smk-means: An improved mini batch k-means algorithm based on mapreduce with big data. *Computers, Materials and Continua*, 56:365–379, 01 2018.

[66] Y. Yang and Y. Liu. A dos attack situation visualization method based on parallel coordinates. In *2012 IEEE 12th International Conference on Computer and Information Technology*, pages 340–344, 2012.

[67] Jiawan Zhang, Yang Peng, Liangfu Lu, and Chen Lei. Netviewer: A visualization tool for network security events. *Networks Security, Wireless Communications and Trusted Computing, International Conference on*, 1:434–437, 04 2009.

[68] Jiawan Zhang, Guoqiang Yang, Liangfu Lu, MaoLin Huang, and Ming Che. *A Novel Visualization Method for Detecting DDoS Network Attacks*, pages 185–194. 01 1970.

[69] Jinson Zhang and Mao Lin Huang. Visual analytics model for intrusion detection in flood attack. In *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pages 277–284. IEEE, 2013.