A HYBRID DEEP LEARNING APPROACH FOR BOTTLENECK DETECTION IN IOT

1.Title :HYBRID DEEP LEARNING APPROACH FOR BOTTLENECK DETECTION IN IOT

2.Abstract:

Cloud computing is possibly the most attractive breakthrough in the current scenario. It offers a cost-effective solution by lowering the significant upfront cost of purchasing equipment foundations and processing power. Fog computing adds to cloud infrastructure by utilising a portion of the less-registered work at the edge devices, lowering end client reaction time, for example, IoT. However, most IoT devices have limited resources, and there are several gadgets that cyber assaults could target. Cyber-attacks such as bottleneck, Dos, DDoS, and botnets continue to pose substantial risks in the IoT context. Botnets are currently the most serious internet menace.

A botnet is a collection of infected systems that are connected online and commanded by an adversary to do harmful acts without authorization or authentication. A botnet can infiltrate the system and take data.

It can also carry out attacks such as phishing and spamming. To address this essential issue, we present a novel botnet attack detection approach that might be used in fog computing scenarios to avoid the assault by using the programmability of the software-defined network (SDN) environment. We thoroughly tested our suggested approach, standard and extended performance evaluation measures, and current DL models on the most recent dataset. Our findings are cross-validated to demonstrate overall performance.

Cloud computing is perhaps the most enticing innovation in the present figuring situation. It gives an expense-effective arrangement by diminishing the enormous forthright expense of purchasing equipment foundations and processing power. Fog computing is an additional help to cloud infrastructure by utilizing a portion of the less-registered undertaking at the edge devices, reducing the end client's reaction time, such as IoT. However, most of the IoT devices are resource-constrained, and there are many devices that cyber attacks could target. Cyber-attacks such as bottleneck, Dos, DDoS, and botnets are still significant threats in the IoT environment. Botnets are currently the most significant threat on the internet. A set of infected systems connected online and directed by an adversary to carry out malicious actions without authorization or authentication is known as a botnet. A botnet can compromise the system and steal the data. It can also perform attacks, like Phishing, spamming, and more. To overcome the critical issue, we exhibit a novel botnet attack detection approach that could be utilized in fog computing situations to dispense with the attack using the programmable nature of the software-defined network (SDN) environment.We carefully tested the most recent dataset for our proposed technique, standard and extended performance evaluation measures, and current

DL models. To further illustrate overall performance, our findings are cross-validated. The proposed method performs better than previous ones in correctly identifying 99.98% of multi-variant sophisticated bot attacks. Additionally, the time of our suggested method is 0.022(ms), indicating good speed efficiency results.

a. Brief Introduction:

Security is one of the most important challenges for the network system to be efficient and dependable when conducting transactions over the IoT .

Surveillance, healthcare, transportation, manufacturing, education, and other areas advocate protecting IoT infrastructure to boost performance. Previously, IoT devices generated data via various types of sensors, and Ahmed M. Elmisery was the associate editor in charge of coordinating the review of this paper and authorising it for publication.

It becomes tidy for the cloud servers to efficiently handle or process these transactions. Fog computing is one of the recently proposed strategies that might be used to add desired functionalities to IoT infrastructure .

As a result, among the primary challenges that can affect the operation of fog computing are system security and protection [12]. In this sense, availability is one of the most important security requirements for providing services to actual consumer applications based on their needs. However, adversaries are constantly putting this to the test by initiating various sorts of assaults, such as DoS or DDoS strikes [13]. These attacks can be carried out by an individual or a group. If a group does it, it is referred to as a "botnet," but if an individual does it, it is referred to as a "bot-master." [14]. The bot-master is the attacker node that can conduct a variety of server-side attacks such as phishing, spam, click fraud, and others.

A botnet is remotely controlled by a command-and-control channel.

The command-and-control channel is a system that the adversary employs to control a compromised system by transmitting messages and commands. Through these commands, the adversary can steal data and influence the infected network [8].

A botnet assault occurs when a 'n' number of infected nodes are controlled by a bot-master and launch an attack on the server from various compromised systems.

Security remains a difficult job in the fog computing paradigm, and many security solutions are proposed to make it resilient to flaws. However, the majority of the solutions emphasise the flexibility and continual monitoring of the fog server. At fog servers, software-defined networking (SDN) is employed to address flexibility and continuous monitoring difficulties [15].

SDN is a new networking paradigm that helps to make networks more flexible by assisting in network management, traffic analysis, and routing control structures. [16], [17] because a separate control plan allows a flexible device management approach. As a result, an SDN-based fog computing environment gives the fog computing system centralised control. The following are the characteristics of the SDN-based fog computing system:

• SDN can handle the secure connection for thousands of devices communicating across the fog.

• SDN can give real-time monitoring and awareness with minimal latency due to its flexible design. • SDN can dynamically balance the load due to its programmable nature. [18].

b. Existing System

Several researchers are focusing on detecting botnet attacks these days [28]_[30]. The main requirement in botnet detection is identifying the infected devices before they can exploit the network by initiating malicious activity. Authors propose numerous methods that claim to secure the network against botnet attacks. These approaches focus on anomaly detection schemes using artificial intelligence, primarily ML and DL algorithms. In various research approaches, authors [21]_[23] used ML and hybrid ML techniques for botnet detection such as BayesNet (BN), Support Vector Machine (SVM), J48, Decision Tree (DT), and Naive Bayes (NB). Furthermore, Machine Learning methods are categorized as the supervised, the unsupervised, or the semi-supervised learning.

Parakash *et al.* performed experiments using three well-known machine learning algorithms to detect DDoS packets: K-Nearest Neighbors algorithm (KNN), SVM, and NB. The findings show that the KNN performs better in detecting DDoS attacks having 97% accuracy, while SVM and NB algorithms achieve 82% and 83% accuracy, respectively [33]. In [34], the authors proposed a detection scheme that uses the SVM algorithm with their own proposed idle timeout adjustment algorithm (IA). They demonstrated the way their proposed methodology outperforms and achieves better results. In another work, [35] uses, NB, SVM and neural network. Results show that the neural network and NB models performed outclass and achieved 100% accuracy, while the SVM model was at 95% accuracy. Ye *et al.* [36] also used the SVM algorithm and achieved an average accuracy of 95.24%. In [37], authors performed experiments using various algorithms such as Naive Bayesian and decision tree classifier algorithms. They achieved a 99.6% detection accuracy rate.

DL algorithms are the subset of ML. That can deal with large datasets and unstructured data. ML algorithms do not provide better results for extensive data produced by IoT devices and unstructured data [38]. Hence DL algorithms are preferable for IoT compared to traditional ML algorithms such as KNN, SVM, NB, and others. Different DL and hybrid DL approaches are applied for detecting various kinds of malware in IoT devices [39]_[41]. In [42], the authors described a technique for defending the IoT environment against malware and cyber attacks, such as DDoS, brute force, bot, and infiltration. This strategy makes use of DL in SDN.

Disadvantages

- ➢ An existing system is not hybrid deep learning detection policy to improve the efficiency and effectiveness of the SDN-based fog computing architecture. Results show that the proposed scheme works better and provides a better detection rate.
- ➢ can't customize the policies and applications dues to its programmable nature.

c. Proposed System

- ➢ The system suggests an efficient deep learning framework for detecting Botnet attacks in an SDN-based fog computing environment.
- ➢ The practical experiment is performed on N_BaIoT Dataset, which comprises both Botnet attack and benign samples.
- ➢ The proposed technique is evaluated against well-known performance evaluation metrics of the machine and deep learning algorithms known as precision, F1-score, recall, accuracy, and so forth.
- ➢ For unbiased results, we also applied the technique of 10-fold-cross-validation.

Advantages

System can manage the secure connection for thousands of devices connected over the fog for data transmission.

System can provide real-time monitoring and awareness with low latency.

System can dynamically balance the load with its flexible architecture.

SYSTEM REQUIREMENTS

- **H/W System Configuration:-**

➢ Processor	- Pentium –IV
➢ RAM	- 4 GB (min)
➢ Hard Disk	- 20 GB
➢ Key Board	- Standard Windows Keyboard
➢ Mouse	- Two or Three Button Mouse
➢ Monitor	- SVGA

<u>SOFTWARE REQUIREMENTS:</u>

❖ **Operating system**	**:** Windows 7 Ultimate.
❖ **Coding Language**	**:** Python.
❖ **Front-End**	**:** Python.
❖ **Back-End**	**:** Django-ORM
❖ **Designing**	**:** Html, css, javascript.
❖ **Data Base**	**:** MySQL (WAMP Server).

3. Introduction

One of the most significant issues for the network system to be efficient and reliable while doing transactions over the IoT is security [1]. The tremendous growth of IoT in different fields, i.e., surveillance, healthcare, transportation, manufacturing industry, education, and others, encourages securing IoT infrastructure to improve its performance. Earlier IoT devices generate data through various types of sensors, and it becomes tidy for the cloud servers to handle or process these transactions efficiently. Fog computing is among the newly proposed schemes that could be utilized to add preferred features to the IoT infrastructure [2]. Fog computing is competent in doing some regional analysis of information [3] before communicating the aggregated data to the cloud server. It helps in keeping the latency constraints in some time compelled real-time issues, making them appropriate for IoT-based applications such as vehicular ad-hoc networks (VANETs) [4]_[11]. These advancements towards using fog servers in IoT infrastructure motivate the adversaries to target the fog server with malicious intent to lower its performance. Hence, security and protection of the system are among the major issues that can affect the performance of fog computing [12]. In this regard, availability is among the core security requirements for offering services to the actual customer applications according to their interest. However, this is constantly tested by the adversaries by launching different types of attacks, such as DoS or DDoS attacks [13]. An individual or a group can perform these attacks. If a group performs it, it is named ``botnet," while if an individual launches it, it is known as ``bot-master." [14]. The bot-master is the attacker node that can launch several types of attacks on the server, such as Phishing, spam, Click fraud, and others. A command-and-control channel remotely controls a botnet. The command-and-control channel is a system the adversary uses to control by sending messages and commands to a compromised system. The adversary can steal the data through these commands and manipulate the infected network [8]. In a botnet attack, some `n' number of compromised nodes are controlled by a bot-master, and they launch an attack on the server from different compromised systems.

In the fog computing paradigm security is still challenging task, and various security schemes are proposed to make it resilient against vulnerabilities. However, most of the schemes focus on flexibility and continuous monitoring of the fog server. Software-denied networking (SDN) is used at fog servers to address flexibility, and continuous monitoring issues [15]. SDN is an emerging networking paradigm that assists in making the network more flexible that can help in managing the network, analyzing the traffic, and assisting in the routing control architectures [16], [17] as there is a separate control plan that provides a flexible device management policy. Hence, an SDN-based fog computing environment provides centralized control to the fog computing system. The characteristics of the SDN based fog computing system are discussed below V

_ SDN can manage the secure connection for thousands of devices connected over the fog for data transmission.

_ SDN can provide real-time monitoring and awareness with low latency.

_ SDN can dynamically balance the load with its flexible architecture. _ SDN can customize the policies and applications dues to its programmable nature. [18]

The software-denied network plays a vital role as its network control architecture can be directly programmable through the command requests. SDN based fog computing architecture can assist in analyzing and managing IoT devices. The motivation behind SDN is to give consistency to network management through partitioning the network into the data plane and the control plane. SDN can add programmability, adaptability, and versatility to the fog computing system. In high-speed networks, discovering the botnet attack is a significant concern [19]. The proposed work shows the methodology through which the botnet attack is identified with a high detection rate which can be used in SDN to enhance the security of fog computing. Deep learning (DL) based detection approach in the SDN-based fog computing application can be a better counterattack to improve the overall performance of the system [20]. DL strategy is adaptable to conditions to recognize the abnormal behavior of the network. We proposed a hybrid deep learning detection policy to improve the efciency and effectiveness of the SDN-based fog computing architecture. Results show that the proposed scheme works better and provides a better detection rate.

4. Literature Survey

C. Xie, H. Hu and Y. Liu[1], proposed that congestion control based on shared bottleneck detection not only improves throughput in the Non-Shared Bottlenecks (NSB) scenario, but also maintains fairness to other connections in the Shared Bottlenecks (SB) situation for multipath transmission. However, because of the large difference in post-path delay (DPPD) in high latency satellite networks, the majority of existing shared bottleneck identification schemes misuse irrelevant data, resulting in poor performance. First, the relationships of OWDs between two subflows are examined in this work. Then, in high latency satellite networks, we offer a robust schema based on difference estimate of post-path delay (DEPPD) to find shared bottlenecks for multipath transmission. Our design looks for a continuously matching sequence between the elements of two congestion event arrays and uses the mean intervals of matched congestion events as an estimate of DPPD to determine whether the next pair of congestion events matches. If the tail element of one array can be matched in some way, the two subflows share a bottleneck. Otherwise, no shared bottleneck exists. When compared to other strategies, the schema is implemented in multipath quic (MPQUIC), and simulations show that DEPPD can achieve superior accuracy in our case and remain stable with variations of DPPD.

X. Xia, R. Togneri, F. Sohel and D. Huang[2] proposed that acoustic event detection (AED) using random forest regression and merging acoustic and bottleneck characteristics (BN). In acoustic signal processing, bottleneck characteristics have a high reputation for being intrinsically discriminative. To handle with unstructured and complex real-world acoustic events, an acoustic event detection system is built using bottleneck and acoustic properties. The UPC-TALP and ITC-Irst databases, which contain highly varied acoustic occurrences, were used for the evaluations. The experimental results show that the low-dimensional and discriminative bottleneck characteristics are useful, with error rates decreasing by 5.33% and 5.51%, respectively.

T. T. Nguyen, S. C. Calvert, H. L. Vu and H. van Lint,[3] proposed that the majority of traffic congestion is caused by highway bottlenecks. Although the topic of bottleneck identification is not new, current solutions have not entirely solved the problem in terms of bottleneck locations, activation time, and related congestion tracking. These characteristics are critical for detecting and characterising a bottleneck. This research presents a complete framework for recognising and extracting these motorway bottleneck characteristics from traffic data. We are especially interested in whether a bottleneck is the primary source of congestion or whether it is activated as a result of congestion induced by another downstream bottleneck.

G. -H. Kim, Y. -J. Song, C. -H. Park, W. -J. Eom, J. -K. Kim and Y. -Z. Cho [4] proposed that Multipath TCP (MPTCP) should outperform single path TCP in non-shared bottleneck (NSB) links while maintaining the same throughput in shared bottleneck (SB) networks. The effort of MPTCP congestion control, which does not impair single-path TCP, rather impedes achieving perfect throughput in NSB networks. In this research, we present a method for detecting the bottleneck bottleneck-type of an MPTCP connection and, as a result, BALIA for NSB (NSB-BALIA) to improve throughput in NSB links. We confirmed through emulation that NSB-BALIA enhanced throughput in the SB link without affecting single-path TCP and obtained an optimal throughput ratio in the NSB links compared to the conventional MPTCP congestion control techniques.

B. Abolhasanzadeh[5] proposed that the reason for the integration of our lives and information systems is the constant advancement of technology. As a result, the relevance of security in these systems grows. As a result, the use of intrusion detection systems as security solutions is expanding year after year. These systems (IDSs) are thought to provide defence against cyber-attacks. However, processing massive data is one of the key issues of intrusion detection systems and is the cause of these systems' poor performance in terms of time and spatial complexity. To address these issues, we proposed a method for reducing complexity. Our method is based on dimensionality reduction, and neural network bottleneck feature extraction is the primary method in this study.

W. Wei, Y. Wang, K. Xue, D. S. L. Wei, J. Han and P. Hong[6] proposed that the primary technological challenge of bottleneck fairness-based multipath congestion control is determining whether two flows share a single bottleneck. Previous techniques perform poorly when the time between the two paths from the shared bottleneck to the common recipient changes greatly (a phenomenon known as path lag). We propose and implement a shared bottleneck detection system based on congestion interval variance measurement (SBDV) in this letter. To determine whether two flows share a bottleneck, we employ just one-way delay measurements within each flow. If the time interval variance between two flows suffering congestion is less than a threshold established by the duration of congestion, the two flows can be deemed to have a common bottleneck.

H. Wang, H. Zhang and N. Ray[7] proposed that Under-segmentation of a multi-object image is a typical issue in image segmentation algorithms. This work describes a novel method for breaking clumps generated by several objects as a result of under-segmentation. The method consists of two steps: selecting a pair of points for clump separation and joining the selected pair of points. The initial stage is to discover a pair of points for splitting using a bottleneck rule, assuming that the desired objects have an approximately convex form. The selected pair of splitting locations is then linked in the second stage by determining the ideal splitting line between them based on minimising image energy. This method's performance is evaluated using photos from diverse applications.

H. Chen, H. Yuan, H. Qin and X. Mu[8] proposed that Emergency rescue is a key security safeguard for maritime transportation and engineering operations. Under-water robots could detect drowning persons automatically as an important piece of equipment for the emergency rescue effort, effectively improving operational efficiency. Because of the significant attenuation of light underwater, optical imaging systems such as cameras are unable to gather information from long-distance settings. Based on forward-looking sonar pictures, this paper developed an object detection network. Due to the limits of the processer in the robots, the proposed network is lightweight and relies on the Bottleneck Transformer and Feature Pyramid Networks to detect drowning persons underwater rather than big networks. The experimental findings suggest that the proposed network can achieve pretty acceptable accuracy while detecting objects quickly.

H. Gou and Y. Yoo[9] proposed that Wireless sensor networks (WSNs) have been identified as a possible tool for monitoring both civic and military surroundings under hazardous or dangerous conditions. Because of the unique property and distinction from standard wireless networks, the longevity of the entire network is the most significant consideration. Bottleneck nodes are common in WSNs and reduce the network's overall lifetime. The traditional centralised bottleneck detection method MINCUT has been offered as a solution for WSNs in order to identify bottleneck nodes. They are, however, impracticable for networks with a large number of nodes. This work first presents a distributed approach that can reduce algorithm complexity and quickly identify bottleneck nodes.

Y. Zhai, S. Sun, J. Wang and M. Wang[10] proposed that A new bottleneck definition is proposed from the standpoint of the manufacturing system's goal. According to the definition, a distinct bottleneck detection method based on orthogonal experiment (BD-OE) for job shop is proposed. The method takes the objective of the manufacturing system as estimated index, carries out typical and finite number of experiments by combining different dispatching rules, to detect bottleneck machine which has the greatest effect on the estimated index. The method can detect bottleneck before the system running, and to guide the following production planning and preparation of necessity in advance for the improvement of the system's performance.

J. Parekh, G. Jung, G. Swint, C. Pu and A. Sahai[11] proposed that The performance of multiple machine learning classifiers for bottleneck detection in corporate, multi-tier applications driven by service level objectives is presented in this paper. In particular, this paper shows the effectiveness of three classifiers, a tree-augmented Naive Bayesian network, a J48 decision tree, and LogitBoost, using our bottleneck detection process, which delves into a new area of performance analysis based on metric trends (first order derivative) rather than metric value itself. Furthermore, the efficiency of each classifier is displayed by assessing the convergence speed or the number of staging trials required to provide favourable findings. Finally, the effectiveness of the bottleneck detection classifiers, as each classifier strongly detects the enterprise system bottleneck.

C. Wang, S. Wang and Y. Wei[12] proposed that The bottleneck consumes more energy than other nodes, which may result in network partitioning in wireless sensor networks. For multi-hop wireless sensor networks, we proposed a coarse-grained bottleneck identification method based on the data flow to save energy. Furthermore, a data buffer method was built around the bottleneck to reduce control frames in the media access control layer. Experiments revealed that our technique can extend network lifetime while also causing a slight delay.

5. Proposed System

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures:	interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

6.1 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

6.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

6.3 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

SYSTEM TESTING

TESTING METHODOLOGIES

The following are the Testing Methodologies:

- **Unit Testing.**
- **Integration Testing.**
- **User Acceptance Testing.**
- **Output Testing.**
- **Validation Testing.**

Unit Testing

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module’s control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All important processing path are tested for the expected results. All error handling paths are also tested.

Integration Testing

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

The following are the types of Integration Testing:

1)Top Down Integration

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

2. Bottom-up Integration

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

- The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.
- A driver (i.e.) the control program for testing is written to coordinate test case input and output.
- The cluster is tested.
- Drivers are removed and clusters are combined moving upward in the program structure

The bottom up approaches tests each module individually and then each module is module is integrated with a main module and tested for functionality.

OTHER TESTING METHODOLOGIES

User Acceptance Testing

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

Validation Checking

Validation checks are performed on the following fields.

Text Field:

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

Numeric Field:

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error messages. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

Preparation of Test Data

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

Using Live Test Data:

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true systems test and in fact ignores the cases most likely to cause system failure.

Using Artificial Test Data:

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

The package “Virtual Private Network” has satisfied all the requirements specified as per software requirement specification and was accepted.

USER TRAINING

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

MAINTAINENCE

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user's requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing is simple and easy to understand which will make maintenance easier.

TESTING STRATEGY :

A strategy for system testing integrates system test cases and design techniques into a well planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation .A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software. Thus, a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

SYSTEM TESTING:

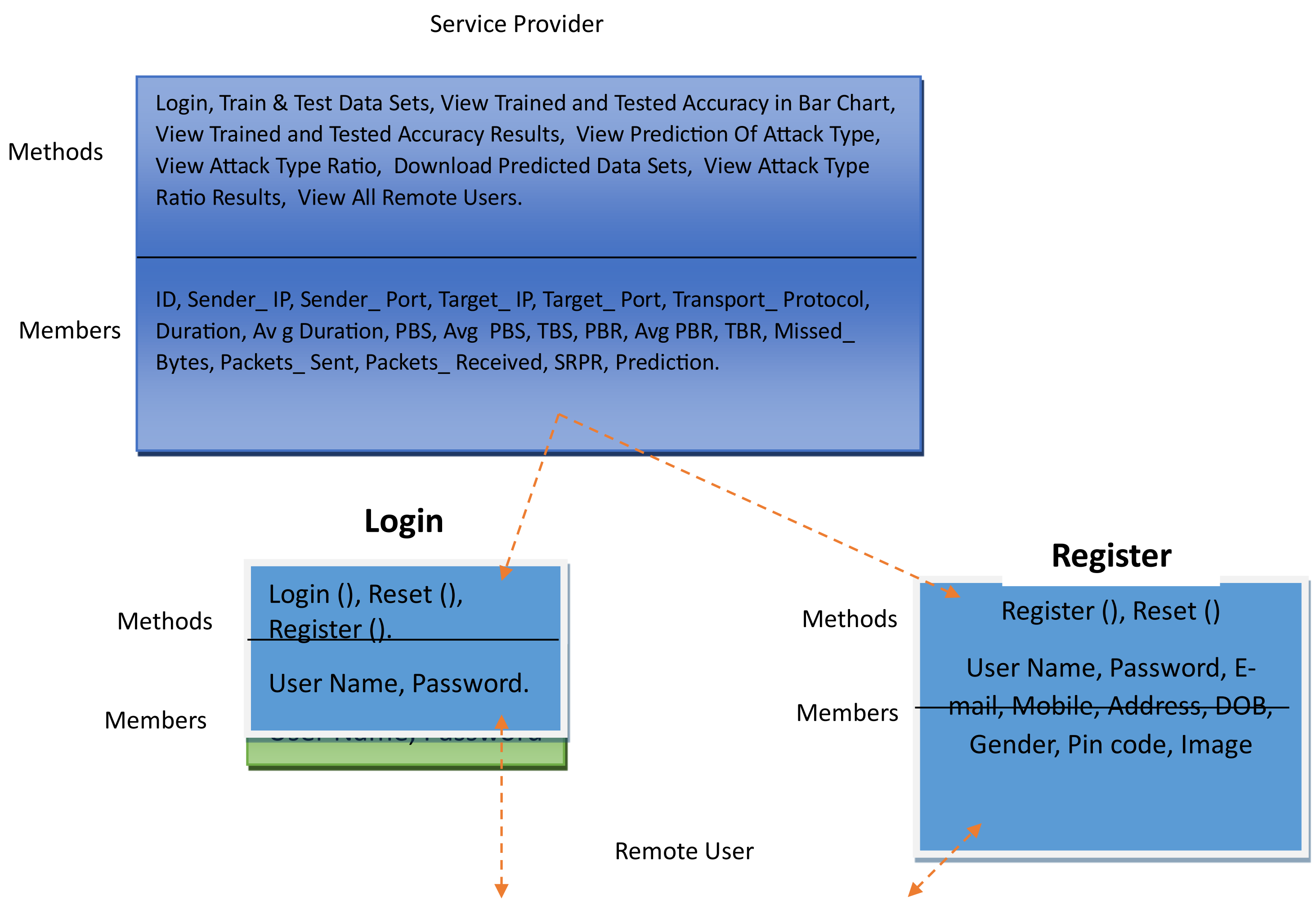
Software once validated must be combined with other system elements (e.g. Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

UNIT TESTING:

In unit testing different are modules are tested against the specifications produced during the design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goals to test the internal logic of the modules. Using the detailed design description as a guide, important Conrail paths are tested to uncover errors within the boundary of the modules. This testing is carried out during the programming stage itself. In this type of testing step, each module was found to be working satisfactorily as regards to the expected output from the module.

In Due Course, latest technology advancements will be taken into consideration. As part of technical build-up many components of the networking system will be generic in nature so that future projects can either use or interact with this. The future holds a lot to offer to the development and refinement of this project.

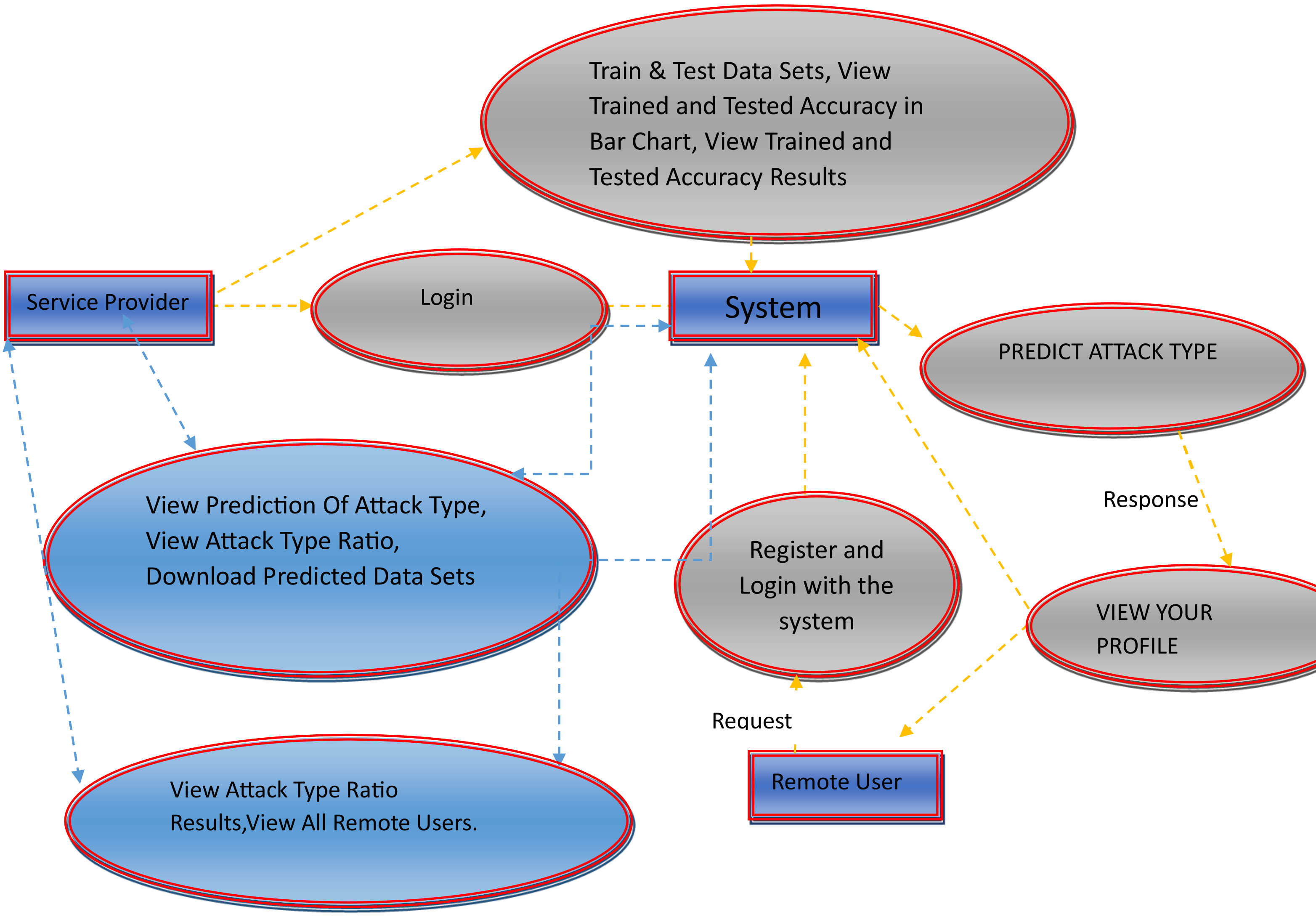
Class Diagram :



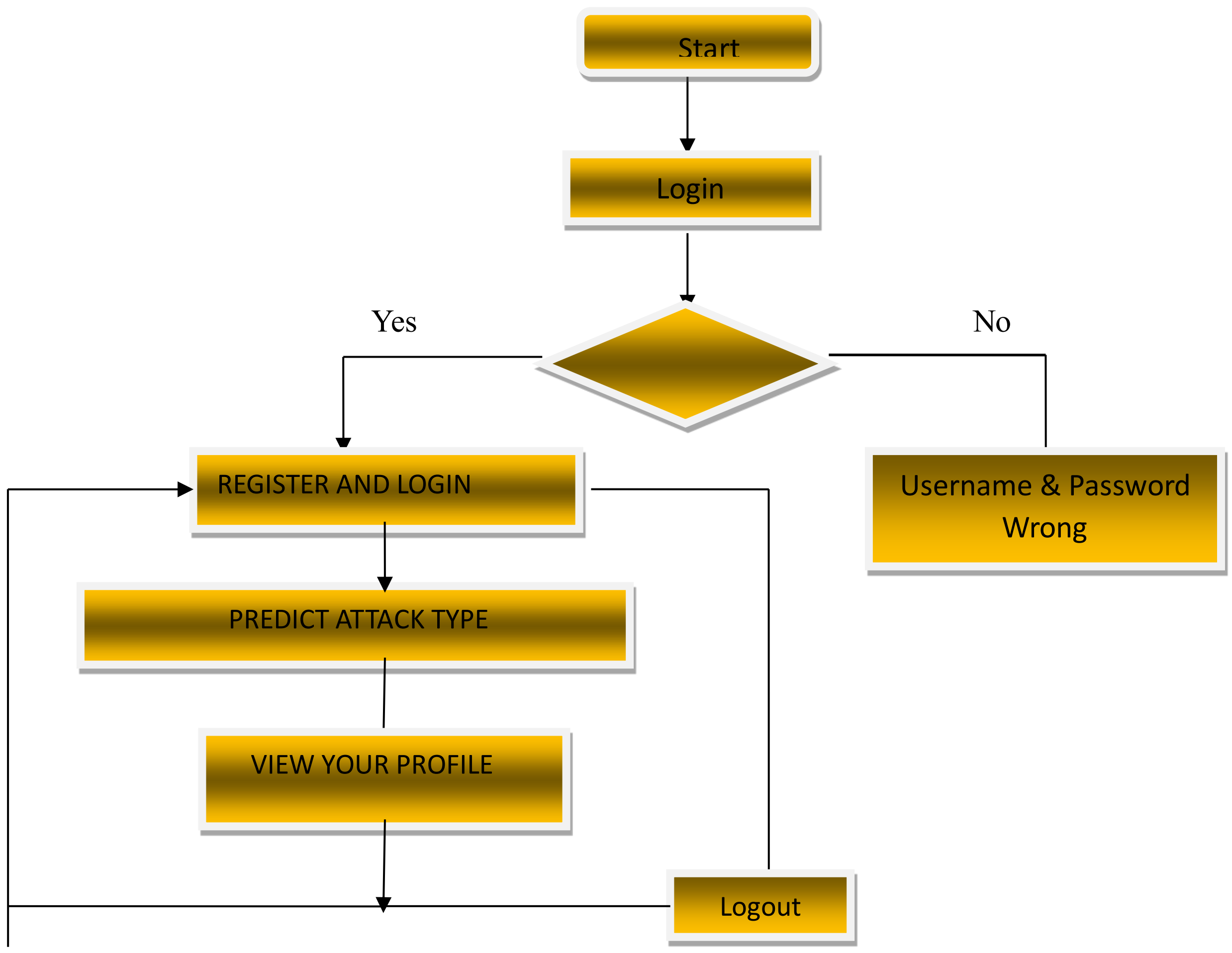
Service Provider
Methods
Login, Train & Test Data Sets, View Trained and Tested Accuracy in Bar Chart, View Trained and Tested Accuracy Results, View Prediction Of Attack Type, View Attack Type Ratio, Download Predicted Data Sets, View Attack Type Ratio Results, View All Remote Users.
Members
ID, Sender_ IP, Sender_ Port, Target_ IP, Target_ Port, Transport_ Protocol, Duration, Av g Duration, PBS, Avg PBS, TBS, PBR, Avg PBR, TBR, Missed_ Bytes, Packets_ Sent, Packets_ Received, SRPR, Prediction.
Login
Methods
Login (), Reset (), Register ().
Members
User Name, Password.
Register
Methods
Register (), Reset ()
Members
User Name, Password, E-mail, Mobile, Address, DOB, Gender, Pin code, Image
Remote User

Methods	REGISTER AND LOGIN, PREDICT ATTACK TYPE, VIEW YOUR PROFILE.
Members	ID, Sender_ IP, Sender_ Port, Target_ IP, Target_ Port, Transport_ Protocol, Duration, Av g Duration, PBS, Avg PBS, TBS, PBR, Avg PBR, TBR, Missed_ Bytes, Packets_ Sent, Packets_ Received, SRPR, Prediction.

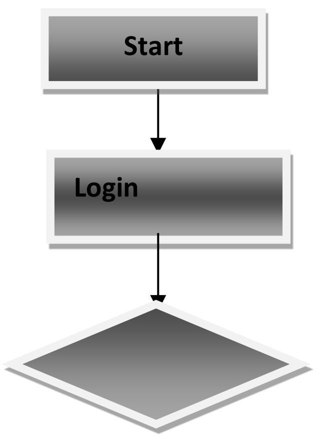
Data Flow Diagram :

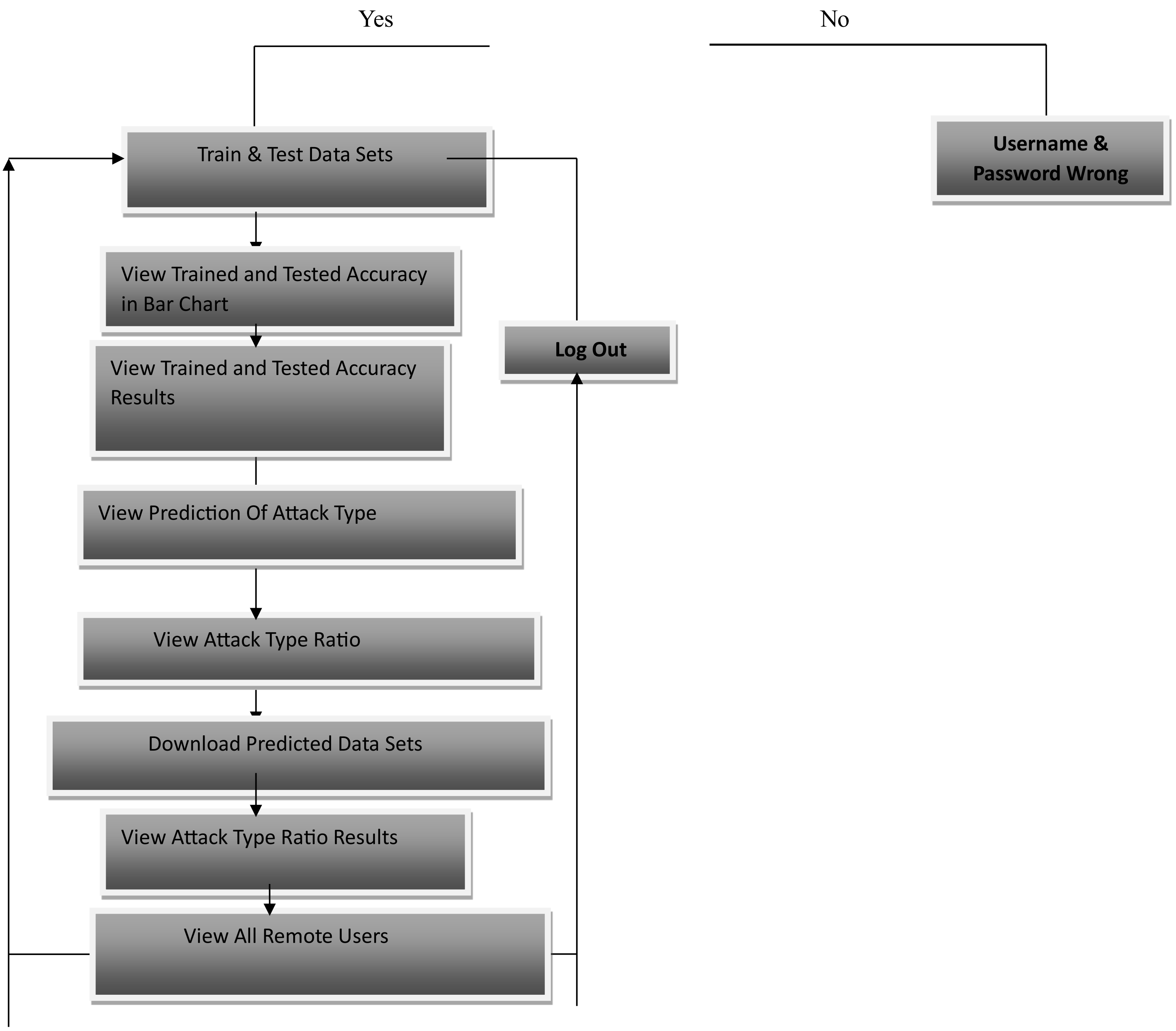

Train & Test Data Sets, View Trained and Tested Accuracy in Bar Chart, View Trained and Tested Accuracy Results
Service Provider
Login
System
PREDICT ATTACK TYPE
Response
View Prediction Of Attack Type, View Attack Type Ratio, Download Predicted Data Sets
Register and Login with the system
VIEW YOUR PROFILE
Request
Remote User
View Attack Type Ratio Results,View All Remote Users.

➢ **Flow Chart : Remote User**



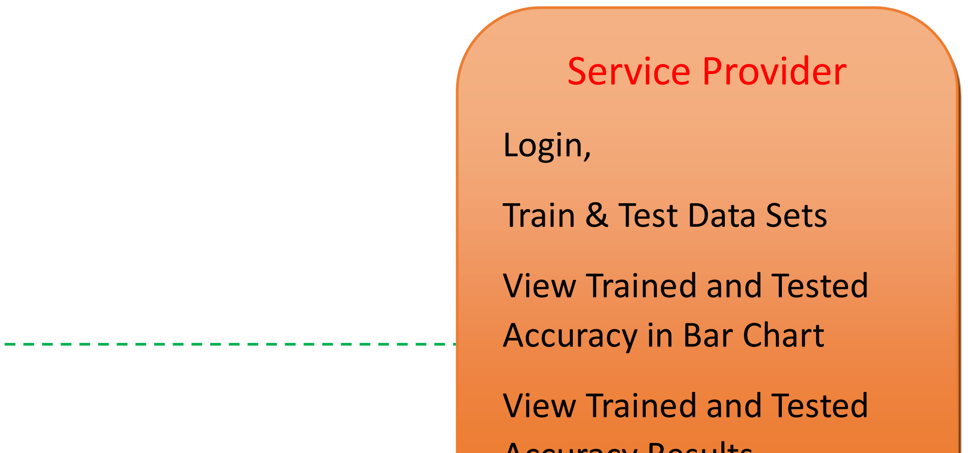
➢ **Flow Chart : Service Provider**

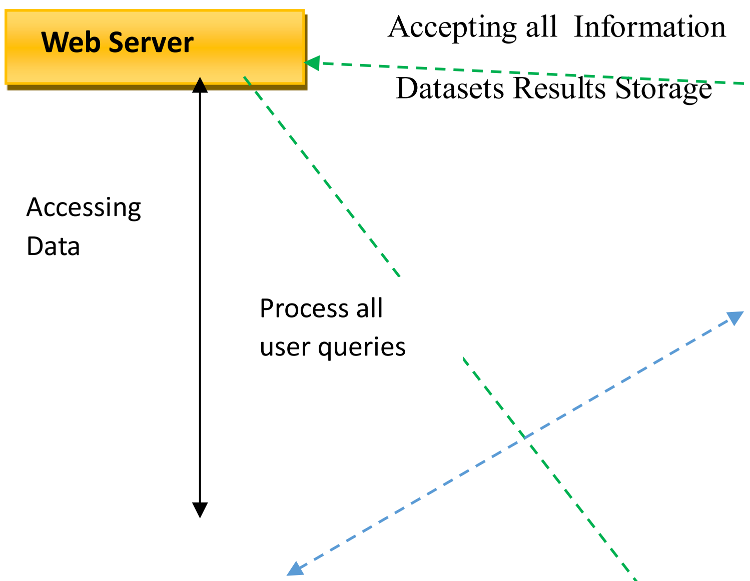



Yes
No
Train & Test Data Sets
Username & Password Wrong
View Trained and Tested Accuracy in Bar Chart
Log Out
View Trained and Tested Accuracy Results
View Prediction Of Attack Type
View Attack Type Ratio
Download Predicted Data Sets
View Attack Type Ratio Results
View All Remote Users

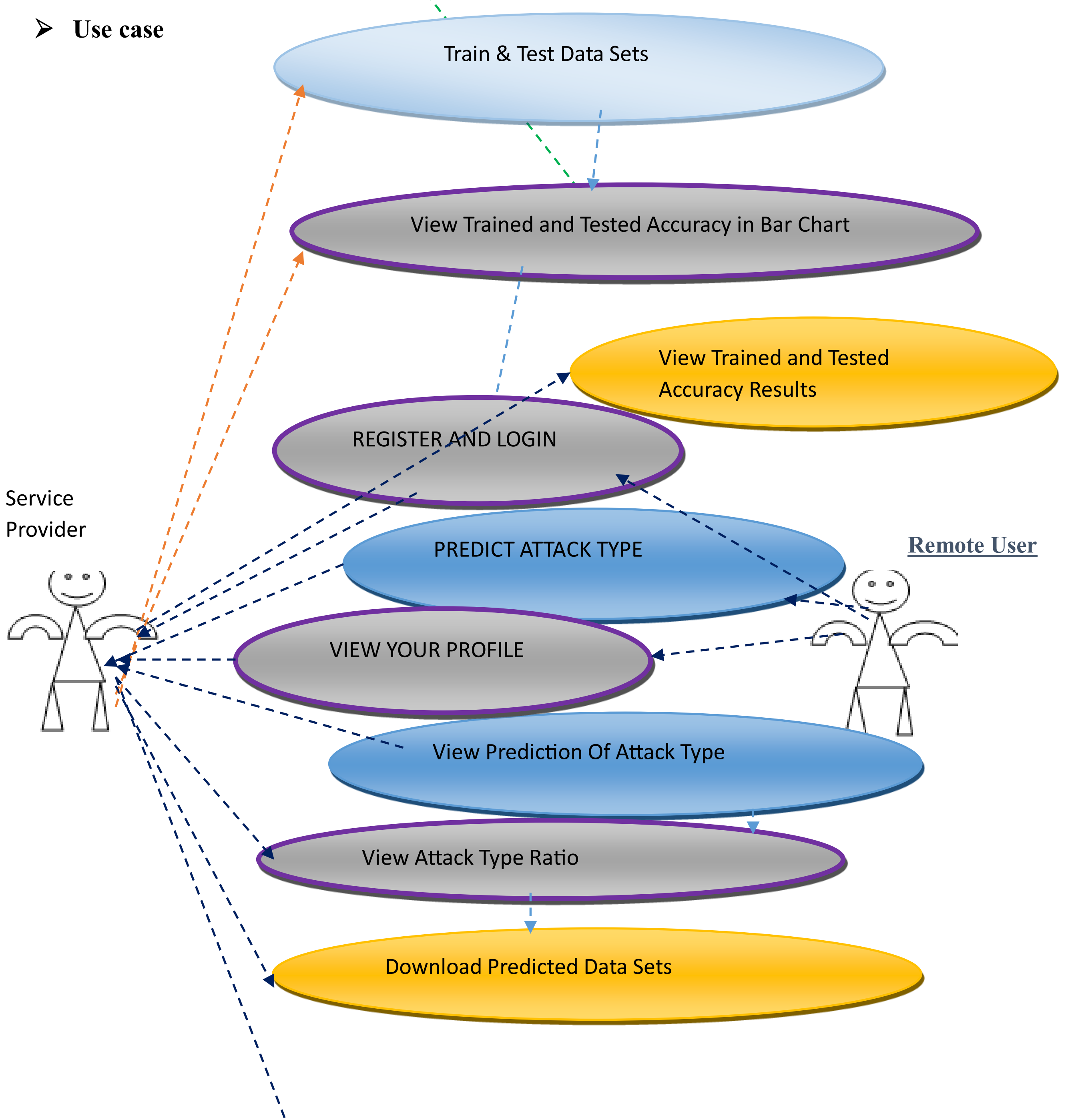
6. Results

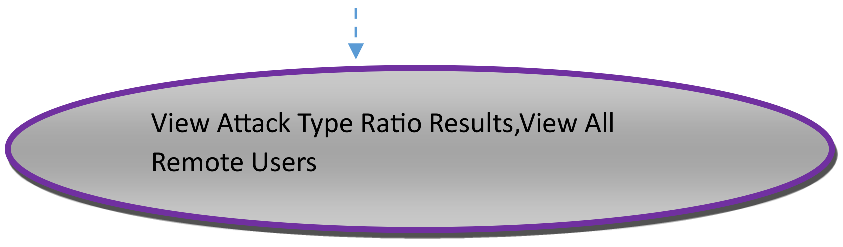
Architecture Diagram




Web Server
Accepting all Information
Datasets Results Storage
Accessing Data
Process all user queries

➢ **Use case**


Train & Test Data Sets
View Trained and Tested Accuracy in Bar Chart
View Trained and Tested Accuracy Results
REGISTER AND LOGIN
Service Provider
PREDICT ATTACK TYPE
Remote User
VIEW YOUR PROFILE
View Prediction Of Attack Type
View Attack Type Ratio
Download Predicted Data Sets



7. Conclusion

SDN-based fog computing architectures are the trending networking paradigms for several applications based on the IoT infrastructure. Fog computing systems are vulnerable to various types of Botnet attacks. Hence, there is a need to integrate a security framework that empowers the SDN to monitor the network anomalies against the Botnet attacks. DL algorithms are considered more effective for the IoT-based infrastructures that work on unstructured and large amounts of data. DL based intrusion detection schemes can detect Botnet attacks in the SDN-enabled fog computing IoT system.

We created a framework that utilizes a hybrid DL detection scheme to identify the IoT botnet attacks. It is trained against the dataset that contains normal and malicious data, and then we used this framework to identify botnet attacks that targeted different IoT devices. Our methodology comprises a botnet dataset, a botnet training paradigm, and a botnet detection paradigm.

Our botnet dataset was built using the N_BaIoT dataset, which was produced by driving botnet attacks from the Gafgyt and Mirai botnets into six distinct types of IoT devices. Five attack types, including UDP, TCP, and ACK, are included in both Gafgyt and Mirai attacks. We developed a botnet detection based on three hybrid models_ DNN-LSTM, CNN2D-LSTM, and CNN2D-CNN3D. Using this training model as a foundation, we developed a botnet detection paradigm that can recognise significant botnet attacks. The botnet detection approach is part of a multiclass classification model that can distinguish between the sub-attacks and innocuous data. The fact-finding analysis showed that our hybrid framework DNN-LSTM model had the highest accuracy of 99.98% at identifying the gafgyt and Mirai botnets in the N_BaIoT environment. In 2014 and 2016, the gafgyt and Mirai botnets essentially targeted home routers and IP cameras. The NBaIoT dataset we used for our experiments revealed that rather than the type of IoT devices, the type of training models has a more significant impact on botnet

detection performance. We think creating DNN-LSTM-based IoT botnet detection models would be an
excellent strategy to enhance botnet identification for different IoT devices.

In the future, we have in mind to compare the performance of the proposed hybrid algorithm to that of other IoT datasets with a more considerable number of nodes. Further, there is a need to test more combinations of DL algorithms and traditional machine learning algorithms.

8. References

[1] C. Xie, H. Hu and Y. Liu, "Shared Bottleneck Detection for Multipath Transmission in High Latency Satellite Network," *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*, Dalian, China, 2019, pp. 38-42.

[2] X. Xia, R. Togneri, F. Sohel and D. Huang, "Random forest regression based acoustic event detection with bottleneck features," *2017 IEEE International Conference on Multimedia and Expo (ICME)*, Hong Kong, China, 2017, pp. 157-162.

[3] T. T. Nguyen, S. C. Calvert, H. L. Vu and H. van Lint, "An Automated Detection Framework for Multiple Highway Bottleneck Activations," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 5678-5692, June 2022.

[4] G. -H. Kim, Y. -J. Song, C. -H. Park, W. -J. Eom, J. -K. Kim and Y. -Z. Cho, "Non-Shared Bottleneck Links of MPTCP: Bottleneck detection and congestion control," *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju, Korea (South), 2020, pp. 1013-1015.

[5] B. Abolhasanzadeh, "Nonlinear dimensionality reduction for intrusion detection using auto-encoder bottleneck features," *2015 7th Conference on Information and Knowledge Technology (IKT)*, Urmia, Iran, 2015, pp. 1-5.

[6] W. Wei, Y. Wang, K. Xue, D. S. L. Wei, J. Han and P. Hong, "Shared Bottleneck Detection Based on Congestion Interval Variance Measurement," in *IEEE Communications Letters*, vol. 22, no. 12, pp. 2467-2470, Dec. 2018.

[7] H. Wang, H. Zhang and N. Ray, "Clump splitting via bottleneck detection," *2011 18th IEEE International Conference on Image Processing*, Brussels, Belgium, 2011, pp.

[8] H. Chen, H. Yuan, H. Qin and X. Mu, "Underwater Drowning People Detection Based On Bottleneck Transformer And Feature Pyramid Network," *2022 IEEE International Conference on Unmanned Systems (ICUS)*, Guangzhou, China, 2022, pp. 1145-1150.

[9] H. Gou and Y. Yoo, "Distributed Bottleneck Node Detection in Wireless Sensor Network," *2010 10th IEEE International Conference on Computer and Information Technology*, Bradford, UK, 2010, pp. 218-224.

[10] Y. Zhai, S. Sun, J. Wang and M. Wang, "An Effective Bottleneck Detection Method for Job Shop," *2010 International Conference on Computing, Control and Industrial Engineering*, Wuhan, China, 2010, pp. 198-201.

[11] J. Parekh, G. Jung, G. Swint, C. Pu and A. Sahai, "Issues in Bottleneck Detection in Multi-Tier Enterprise Applications," *200614th IEEE International Workshop on Quality of Service*, New Haven, CT, USA, 2006, pp. 302-303.

[12] C. Wang, S. Wang and Y. Wei, "A coarse-grained bottleneck detection method with data buffer mechanism for wireless sensor networks," *Proceedings of 2012 2nd International Conference on Computer Science and Network Technology*, Changchun, China, 2012, pp. 2015-2018.