PUCRS

ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

FRANCISCO ASSIS MOREIRA DO NASCIMENTO

DECENTRALIZED FEDERATED LEARNING-BASED INTRUSION DETECTION IN IOT SYSTEMS

Porto Alegre
2023

PÓS-GRADUAÇÃO - *STRICTO SENSU*

Pontifícia Universidade Católica
do Rio Grande do Sul

PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
SCHOOL OF TECHNOLOGY
COMPUTER SCIENCE GRADUATE PROGRAM

DECENTRALIZED FEDERATED LEARNING-BASED INTRUSION DETECTION IN IOT SYSTEMS

FRANCISCO ASSIS MOREIRA DO NASCIMENTO

Doctoral Thesis submitted to the Pontifical Catholic University of Rio Grande do Sul in partial fulfillment of the requirements for the degree of Ph. D. in Computer Science.

Advisor: Prof. Fabiano Passuelo Hessel

Porto Alegre
2023

Ficha Catalográfica

N244d Nascimento, Francisco Assis Moreira do

Decentralized Federated Learning-Based Intrusion Detection in IoT Systems / Francisco Assis Moreira do Nascimento. – 2023.

129 p.

Tese (Doutorado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Fabiano Passuelo Hessel.

1. Federated Learning. 2. Internet of Things. 3. Cybersecurity. 4. Decentralized Computing. 5. Artificial Intelligence. I. Hessel, Fabiano Passuelo. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS com os dados fornecidos pelo(a) autor(a).

Bibliotecária responsável: Clarissa Jesinska Selbach CRB-10/2051

FRANCISCO ASSIS MOREIRA DO NASCIMENTO

DECENTRALIZED FEDERATED LEARNING-BASED INTRUSION DETECTION IN IOT SYSTEMS

This Doctoral Thesis has been submitted in partial fulfillment of the requirements for the degree of Ph. D. in Computer Science of the Computer Science Graduate Program, School of Technology of the Pontifical Catholic University of Rio Grande do Sul

Sanctioned on August 4, 2023.

COMMITTEE MEMBERS:

Prof. Dr. Jorge Luis Victória Barbosa (PPGCA/Unisinos)

Prof. Dr. Leonel Pablo Carvalho Tedesco (PPGSPI/UNISC)

Prof. Dr. Cesar Augusto Missio Marcon (PPGCC/PUCRS)

Prof. Fabiano Passuelo Hessel (PPGCC/PUCRS - Advisor)

To my loving wife, *Juliana*, who has been my steadfast partner through every step of this journey. Your support and belief in me have been invaluable.

To my daughter, *Ana Julia*, who brings joy and purpose to my days. Your spirit and curiosity inspire me to always strive for more.

In memoriam,

To my father, *João Neto*, who left when I was still young. His memory continues to guide me.

To my mother, *Judith*, whose influence remains a part of who I am.

May this work serve as a testament to the importance of perseverance, dedication, and the pursuit of knowledge.

“We can only see a short distance ahead, but we can see plenty there that needs to be done.” (Alan Turing)

ACKNOWLEDGMENTS

I owe an immense debt of gratitude to my advisor, Prof Dr. Fabiano Hessel, whose unparalleled support has been the bedrock of this work. His mentorship, continuous feedback, and guidance have been invaluable. His dedication to nurturing and listening to his students during this demanding journey is a testament to his exceptional character. The wisdom and experiences I've garnered from him will indelibly shape my professional and personal trajectory.

In my computer science Ph.D. journey, I'm genuinely thankful for the scientific tools and methods that guided my research. They were essential in helping me tackle challenges, make discoveries, and complete my thesis. These tools made it possible for me to contribute to our field meaningfully.

I extend my heartfelt appreciation to Prof. Dr. César Marcon, Prof. Dr. Leonel Tedesco, and Prof. Dr. Jorge Barbosa for their invaluable feedback, insights, and dedication. Their contributions have been pivotal in refining this dissertation.

Further, I wish to express my gratitude to the faculties of the Computer Science department of the Polytechnical School in the PUCRS for their efficient and effective support during my coursework.

To my family, friends, and well-wishers, your belief, and encouragement have been the emotional backbone of this journey. This accomplishment is as much yours as it is mine.

Lastly, I thank CAPES and HPE. Your belief in my potential and your financial support have been instrumental in bringing this work to fruition.

Thank you all once again for your support and encouragement.

APRENDIZAGEM FEDERADA DESCENTRALIZADA PARA DETECÇÃO DE INTRUSÃO EM SISTEMAS IOT

RESUMO

Os sistemas baseados na Internet das Coisas (IoT) são vulneráveis a diversos tipos de ataques, em grande parte devido à fragilidade dos dispositivos IoT, que possuem pouco poder computacional e de memória, necessário para a implementação de recursos de segurança mais sofisticados. Além disso, os sistemas IoT são sistemas distribuídos (dispositivos autônomos interconectados e colaborativos) e, portanto, herdam todos os problemas destes sistemas, relacionados à necessidade de garantir a confidencialidade, integridade, autenticidade e disponibilidade. Uma das estratégias tradicionais para lidar com alguns desses problemas envolve a detecção de intrusão e técnicas de prevenção. É comum implementá-los de forma centralizada, o que além de não ser escalável para sistemas IoT com um número crescente de componentes distribuídos, implica em um ponto único inaceitável de falha no sistema. Além disso, o envio de todos os dados coletados para um servidor centralizado na nuvem representa um grande risco para a privacidade das informações. Com o processamento de dados sendo executado na borda, esse problema também é minimizado em uma abordagem distribuída. Esta tese apresenta uma arquitetura de segurança descentralizada para suporte à detecção de intrusão em sistemas baseados em IoT, que é baseada em técnicas de aprendizado de máquina federado para detecção de intrusão, combinado com o uso de tecnologias de razão distribuída para a implementação de autenticação e autorização no acesso a recursos, permitindo obter um mecanismo eficaz e eficiente para minimizar os riscos de segurança em sistemas IoT. Para a avaliação da arquitetura descentralizada foi implementado um protótipo, permitindo a realização de vários experimentos, que comprovaram sua efetividade, com resultados similares aos obtidos com abordagem centralizada, mas com todas as vantagens oferecidas por uma arquitetura totalmente descentralizada.

Palavras-Chave: Aprendizagem Federada, Internet das Coisas, Cibersegurança, Aprendizagem de Máquina, Registro Distribuído.

DECENTRALIZED FEDERATED LEARNING-BASED INTRUSION DETECTION IN IOT SYSTEMS

ABSTRACT

Internet of Things (IoT) based systems are vulnerable to several types of attacks, mainly due to the weakness of IoT devices, which have reduced computational and memory power necessary to implement more sophisticated security features. In addition, IoT systems are distributed systems (interconnected and collaborative autonomous devices) and thus inherit all problems of these systems, which are related to the need to guarantee confidentiality, integrity, authenticity, and availability. One of the traditional strategies to deal with some of these problems involves intrusion detection and prevention techniques. It is usual to implement them in a centralized way, which in addition to not being scalable for IoT systems with an increasing number of distributed components, implies an unacceptable single point of failure in the system. Furthermore, sending all collected data to a centralized server in the cloud poses a significant risk to the privacy of information. With data processing at the edge, this problem is minimized in a distributed approach. This thesis presents a decentralized security architecture for supporting detecting intrusion in IoT-based systems, which is based on federated machine learning techniques for intrusion detection. It also combines the use of distributed ledger technologies for the implementation of authentication and authorization in the access to resources. Thus, the thesis approach allows one to obtain an effective and efficient mechanism to minimize security risks in IoT systems. A prototype was implemented to evaluate the decentralized architecture, allowing several experiments, which proved its effectiveness, with results similar to those obtained with a centralized approach but with all the advantages offered by a decentralized, federated learning-based architecture.

Keywords: Federated Learning, Internet of Things, Cybersecurity, Machine Learning, Distributed Ledgers.

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ACRONYMS

AI – Artificial Intelligence

API – Application Programming Interface

ANN – Artificial Neural Network

BC – BlockChain

BOT – BotneT

CDN – Content Distribution Network

CNN – Convolutional Neural Network

CPE – Customer-Premises Equipment

DARPA – Defense Advanced Research Projects Agency

DCD-FL – DeCentralized Federated Learning

DDOS – Distributed Denial of Service

DL – Deep Learning

DL – Distributed Ledger

DOS – Denial of Service

DR – Detection Rate

FAR – False Alarm Rate

FC – Fully Connected

FL – Federated Learning

FN – False Negative

FP – False Positive

FPR – False Positive Rate

GDPR – General Data Protection Regulation

GECA – Global Edge Computing Architecture

GRPC – g Remote Procedure Call

GRU – Gated Recurrent Unit

GSE – Grupo de Sistemas Embarcados

HTTP – HyperText Transfer Protocol

HTTPS – HyperText Transfer Protocol Secure

IDS – Intrusion Detection System

IIOT – Industrial Internet of Things

IOT – Internet of Things

IP – Internet Protocol

IPFS – InterPlanetary File System

JSON – JavaScript Object Notation

KDD – Knowledge Discovery in Databases

LGPD – Lei Geral de Proteção de Dados

LSTM – Long Short-Term Memory

ML – Machine Learning

MLP – Multi-Layer Perceptron

NIDS – Network Intrusion Detection Systems

NON-IID – non-Independent Identically Distributed

NSL-KDD – Network Security Laboratory-Knowledge Discovery in Databases

P2P – Peer-to-Peer

POS – Proof of Stack

POW – Proof of Work

PUCRS – Pontifícia Universidade Católica do Rio Grande do Sul

REST – Representational State Transfer

RNN – Recurrent Neural Network

RPC – Remote Procedure Call

SGD – Stochastic Gradient Descent

SOAP – Simple Object Access Protocol

SOC – System on Chip

SSL – Secure Sockets Layer

SVM – Support Vector Machine

TCP – Transmission Control Protocol

TLS – Transport Layer Security

TN – True Negative

TP – True Positive

UNSW-NB15 – University of New South Wales-Network Benchmark 15

VM – Virtual Machine

XML – eXtensible Markup Language

LIST OF ABBREVIATIONS

DApp. – Decentralized Application

FedAvg. – Federated Average

FedCS. – Federated Learning with Client Selection

N-BaIoT. – Network-Based Detection of IoT Botnet Attacks

LIST OF SYMBOLS

CONTENTS

1. INTRODUCTION

Internet of Things (IoT) consists of *the interconnection of machines and devices through the internet, enabling the creation of data that can yield analytical insights and support new operations [86]*. In this sense, IoT characteristics are similar to the ones of distributed systems: a set of independent devices, communicating to cooperate in order to perform some useful tasks [112]. Thus, IoT systems have problems similar to those presented in distributed systems: How to guarantee confidentiality, integrity, authenticity, non-repudiation, and high availability? How to tolerate computation and communication failures? [104]

These questions are becoming even more relevant with the growing demand for IoT-based systems in all areas. As illustrated by Figure 1.1 (extracted from [121]), it is expected that by 2025, the total of IoT-connected devices to be 75.44 billion worldwide, demanding new architectures and network protocols, which must be able to handle such a huge number of IoT devices and provide necessary scalability and flexibility levels. Thus, for instance, traditional architectures based on a centralized server in the cloud usually have high latency and many difficulties managing so many connected devices, which can generate an immense amount of data.

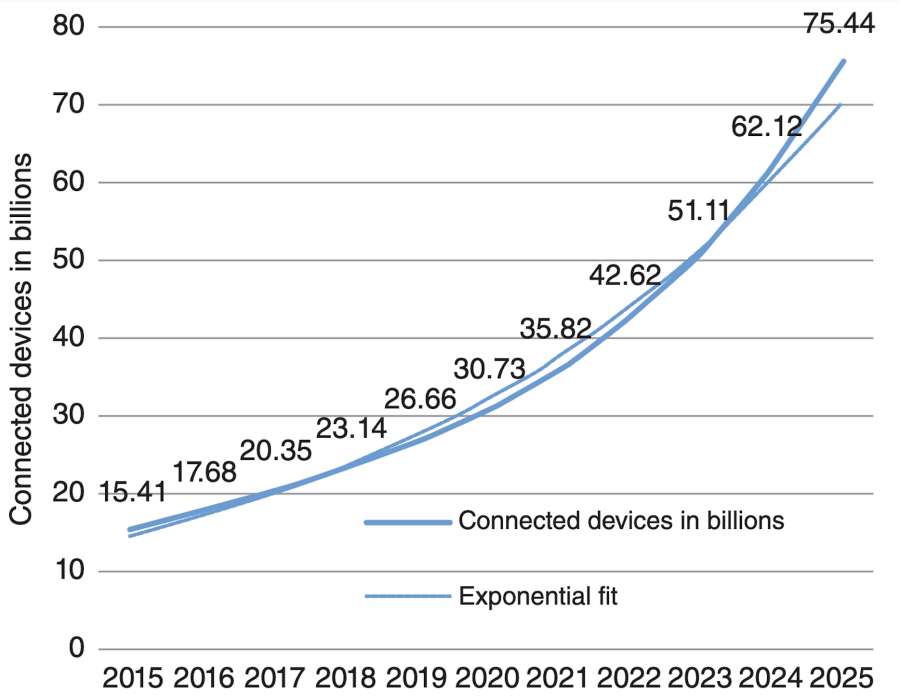

Figure 1.1 – Number of connected devices worldwide 2015–2025. [121]

There is a growing demand for IoT applications, for example, in the areas of: Healthcare (Pace et al. [90] describe BodyEdge, an IoT architecture oriented to support applications for the health industry); smart home (Marikyan et al. [62] present a detailed re-

view of smart home concepts and applications, and Heartfield et al. [40] present many of the cyber threats to smart home applications); smart cities (Yin et al. [122] offer an exhaustive literature survey on smarty cities), and logistics (Han et al. [38] describes an electronic pedigree system for food safety, tracking the processes in production, storage, transportation, and sale).

All these application areas, and many others [67] [49], demand a high level of data privacy and strong data access control and do not allow unauthorized access to information and available resources and services. One can obtain more details on these data security threats from Zhang et al. [125], and Hou et al. [42], which explains the main data security and privacy-preserving issues related to IoT applications and approaching IoT security issues from a data-driven perspective.

Moreover, Sfar et al. [107] give an overview of the IoT security roadmap based on a cognitive and systemic approach, in terms of relationships between individuals, processes, intelligent objects, and technological ecosystems, as basic components of any IoT system.

1.1 Motivation

The essential security requirements for IoT applications are challenging because many devices that can be part of an IoT-based system have low processing power and small memory capacity. Moreover, most of the devices are designed with no concerns about security issues and are vulnerable to many kinds of security threats and criminal attacks [123].

Some basic security principles to minimize risks are usually not taken into account: many IoT devices have weak passwords and also backdoors left by manufacturers to provide remote support for the device; security updates are not regularly installed in the IoT devices; and, usually, there is no protection, as firewalls, for vulnerable IoT devices, which do not have computational power either memory capacity to support more sophisticated security mechanisms [37].

In 2016, the Mirai botnet attack [50] infected a huge number of IoT devices and used them to launch one of the biggest Distributed Denial of Service (DDoS) attacks ever [47]. Since then, these kinds of security incidents are becoming very common, as the one reported by [51]: *On April 24, 2019, a botnet consisting of more than 400,000 IoT devices (home routers, modems, and other CPEs - Customer-Premises Equipments) launched a massive DDoS attack against a Content Distribution Network (CDN) company in the entertainment industry that lasted 13 days with peak attack flows reaching 292,000 HTTP GET/POST requests per second, making it one of the largest Layer 7 attacks handled by the DDoS mitigation firm, Imperva [110].*

Another security incident, reported by [29] on a German steel mill, caused serious damages and was very dangerous: *In 2015, the German Federal Office for Information Security issued a report confirming that some hackers breached a steel plant in their country, compromising numerous systems, including components of the production network. As a result, the personnel of the milling sector could not shut down a blast furnace when required, resulting in massive damage to the system*.

One of the traditional strategies to deal with these problems involves intrusion detection techniques [4], where the traffic network is monitored. Based on some specific rules, inadequate behaviors may indicate a security threat, leading to triggering preventive actions to minimize risks. Unfortunately, it is usual to implement them in a centralized way. In addition to not being scalable for IoT systems with an increasing number of distributed components, it implies an unacceptable single point of failure in the security of IoT systems.

Another very essential strategy to guarantee security and also privacy in IoT is the use of efficient and effective access control mechanisms to prevent unauthorized access to information (confidentiality), to avoid information modification without adequate authorization (integrity), and to guarantee access to information to authenticated users at any needed time (availability) [89].

It is also important to highlight the growing demand for IoT applications and a huge number of connected IoT devices generating a massive amount of data, the traditional Cloud-oriented model is not more effective in supporting these new kinds of applications. Since to be efficient, they demand the data be processed near where it is produced to avoid the high latency offered by the access to the centralized servers in the cloud [121]. This local processing strategy is also important to guarantee more privacy since data is no longer transiting in the network.

In 2012, Cisco researchers, taking into account this strategy, proposed a new network architecture they called Fog Computing [15], where some network elements are allocated near the edge of the network, providing computational power and memory and storage capacity near the users and their devices and applications. Figure 1.2 illustrates this concept.

As shown in Figure 1.2, the "things" (or edge devices: user gadgets, mobiles, smartphones, music players, wearables, game controllers, etc.) interact with the edge layer, which may consist of edge nodes (edge servers, routers, switches, base stations). The fog layers compose intermediary interconnection between the edge layer and the cloud layer, providing more powerful resources directly to the edge devices.

Typically, IoT applications are deployed at the edge layer and eventually communicate with fog and cloud nodes to execute some functions demanding more computational resources. In this way, edge nodes will also be exposed to the many kinds of cyber-attacks usual in fog and cloud nodes, such as DoS (Denial of Service), DDoS, Man in the Middle, botnets, privilege escalation, data leakage, etc.

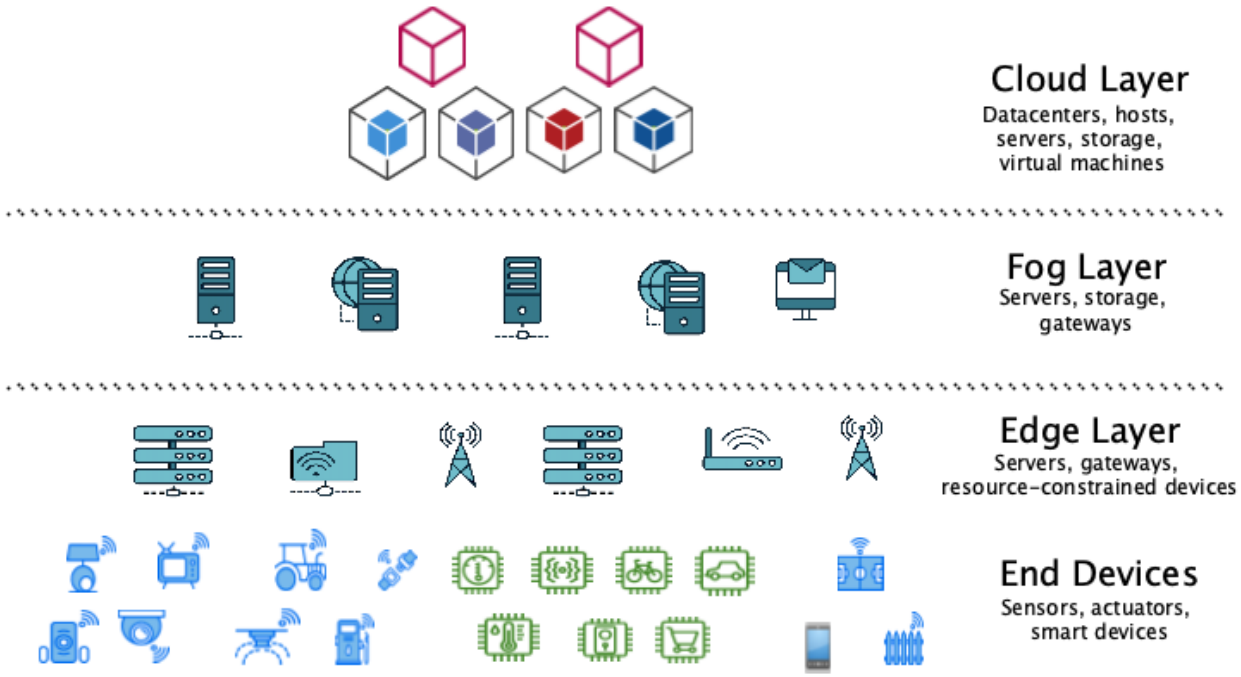

Figure 1.2 – Typical three-tier architecture for IoT

Intrusion Detection Systems (IDSs) have been used in IoT systems to cope with these security risks [93]. It has been usual to adopt machine learning techniques to implement IDS tools [115]. Since many of these techniques depend on manually selected features, they are becoming obsolete due to the many new IoT applications and the different kinds of network traffic they imply. Thus, Deep Learning algorithms are becoming a viable strategy for developing traffic classifiers, which can automatically extract features from complex traffic patterns, and also being able to handle encrypted traffic [3].

1.2 Research Problems

As IoT applications become present in the everyday life of everyone, efficient security mechanisms become the main concerns in developing IoT applications [85]. Since all these issues represent challenging problems, intense research is dedicated to them, but there are many open problems. To summarize all the issues mentioned above, the main problems to be considered in the present thesis include:

P.1. **Problem 1**: vulnerable and constrained devices should be protected and supported by any IoT security platform intended to be effective and efficient;

P.2. **Problem 2**: huge amount of generated data by devices should not be transmitted to fog and cloud to be processed by the intrusion detection system; instead, the IDS should be near where the data is produced;

P.3. **Problem 3**: traditional machine learning techniques for intrusion detection, demanding some manual feature engineering, are not enough anymore, given the very dynamic aspects of IoT, in terms of growing new kinds of devices and applications;

P.4. **Problem 4**: data privacy-preserving is becoming even more important since penalties for privacy violations become very severe, according to recently approved regulations (e.g., General Data Protection Regulation - GDPR in Europe and General Data Protection Law - LGPD in Brazil);

P.5. **Problem 5**: typical fog and cloud cyber threats will also affect the edge since they are all interconnected, so they must be carefully considered.

In this context, the present thesis investigates the following hypotheses:

H.1. **Hypothesis 1**: federated machine learning techniques for intrusion detection allows obtaining a decentralized, no single point of failure, effective mechanism to minimize security risks in IoT systems;

H.2. **Hypothesis 2**: distributed ledgers guarantee the adequate availability and integrity of digital identities and all necessary access control information to the authentication and authorization for resource uses in IoT systems;

H.3. **Hypothesis 3**: resulting security platform can be adopted in many kinds of IoT applications.

To validate these hypotheses, the explored research questions in this thesis included:

RQ.1. **Research Question 1**: What are the state-of-the-art of intrusion detection techniques, and can they be applied to IoT systems?

RQ.2. **Research Question 2**: Federated machine learning techniques can be effectively applied to intrusion detection in IoT systems?

RQ.3. **Research Question 3**: Can distributed ledgers be efficiently deployed on the edge in IoT systems to guarantee information integrity and availability in IoT systems?

RQ.4. **Research Question 4**: Is it possible to develop an IoT security platform based on the techniques mentioned above, which can be adopted in different kinds of IoT applications?

Given that the primary goal of this research is to develop an efficient, effective, and flexible IoT security platform, the specific objectives to be achieved included: (i) identify and report the state-of-the-art of intrusion detection approaches that can be applied to IoT systems; (ii) develop an efficient, effective, and flexible IoT security platform; (iii) explore the use of federated learning to implement decentralized machine learning algorithms for intrusion detection in IoT systems; (iv) evaluate the developed IoT security platform and compare its performance with results reported by other similar platforms; and, (v) report the research results by publishing them in scientific publications and presenting at academic events, as well as exploring the possibility of technology transfer to the industry.

1.3 Thesis Contributions

As part of the present thesis, it was developed a decentralized, federated learning-based security platform for IoT applications, mainly focused on intrusion detection, which allows for handling problems P.1, P.2, P.3, and P.4 since model training and prediction for anomalies/intrusion are performed locally at edge nodes, according to their available resources, and only machine learning models, not collected data, are transmitted to other edge nodes for global models generation.

Moreover, distributed ledgers, included as part of a decentralized IoT platform, guarantee the integrity and high availability of the necessary information for the security platform in a decentralized way. The developed platform also provides authentication, authorization, and auditing mechanisms, essential to treat problem P.5 by providing necessary security features in a distributed manner.

Thus, the main contributions of the present research work consist of:

C.1. **Contribution 1**: Efficient and effective intrusion detection mechanism based on decentralized machine learning techniques, which should be able to predict attacks based on previous incidents, as well on the behaviors of IoT devices and users;

C.2. **Contribution 2**: Efficient and effective use of distributed ledger technologies to provide security mechanisms for authentication, integrity, confidentiality, non-repudiation, and high availability of IoT-based systems;

C.3. **Contribution 3**: Security platform based on distributed ledger technologies and intrusion detection using machine learning techniques, which can be adopted in different IoT application areas.

1.4 Thesis Outline

This thesis is organized into the following chapters:

- In the next **Chapter 2**, basic concepts related to IoT and security for IoT-based systems are presented, as well the main topics of the thesis are introduced;
- In **Chapter 3**, the adopted approach in the research is described, including the main characteristics of the developed security platform for IoT-based systems, as well the methods, techniques, and tools, which have been used during the research work;
- **Chapter 4** presents the obtained results, describing the performed experiments, using a prototype of the security platform and how the developed approach compares to existing intrusion detection systems;
- **Chapter 5** discusses some relevant related work, mainly the ones that try to provide some security mechanisms for IoT-based systems using decentralized intrusion detection techniques;
- Finally, **Chapter 6** summarizes the research findings, discusses their implications, and suggests areas for future research.

2. BACKGROUND

This chapter introduces some basic concepts related to IoT and security for IoT-based systems, providing more detailed explanations of some ideas for readers who may need to become more familiar with them. Additionally, it discusses the existing methods and techniques used in the field and their limitations, which the present research addresses.

2.1 Internet of Things

The term Internet of Things is applied to a network of smart objects, which can intelligently interact with each other and cooperate for the IoT-based system to provide useful services [49].

According to Nord et al. [86], IoT applications are adopted in many areas, from personal to 'big' business. Also, IoT facilitates the development of a myriad of industry-oriented and user-specific IoT applications. Whereas devices and networks provide physical connectivity, IoT applications enable device-to-device and human-to-device interactions reliably and robustly.

Generally, the performed operations in IoT-based systems pass through a collection phase, transmission phase, and processing, management, and utilization phase [17]. The objective of the collection phase is to collect data about the physical environment employing devices, which are usually small and resource-constrained [52].

In the transmission phase, the data is sent to the IoT applications/users through a network interconnecting objects and users across longer distances. Commonly, gateways servers are adopted to interconnect the collecting devices with the Internet-based network components that act in the transmission phase [124].

In the processing, management, and utilization phase, IoT applications process collected data to obtain useful information about the environment and the behavior of the devices [17]. Based on this information, these IoT applications decide when and how to control the devices and actuate the physical environment. The integration and communication between very heterogeneous devices and multi-platform applications are usually implemented through middleware or APIs.

Transport Layer Security (TLS) [100] and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols designed to provide secure communication over a network. TLS/SSL [102] protocols ensure data confidentiality, integrity, and authentication between client-server applications.

In this context, gRPC [44], a high-performance, open-source framework for remote procedure calls (RPC), which supports various programming languages, has been used to

implement communication in distributed systems and use TLS/SSL to secure communications between clients and servers.

By utilizing TLS/SSL certificates, which act as digital identities using cryptographic keys, gRPC ensures secure and encrypted communication between clients and servers. It protects against eavesdropping, tampering, and impersonation attacks, establishing a trusted channel for data exchange.

2.1.1 Reference Architecture for IoT

Sitton-Candanedo et al. [112] propose, based on the existing reference architectures for IoT-based systems, a Global Edge Computing Architecture (GECA) consisting of three layers: IoT layer, Edge layer, and Business Solutions layer (see Figure 2.1).

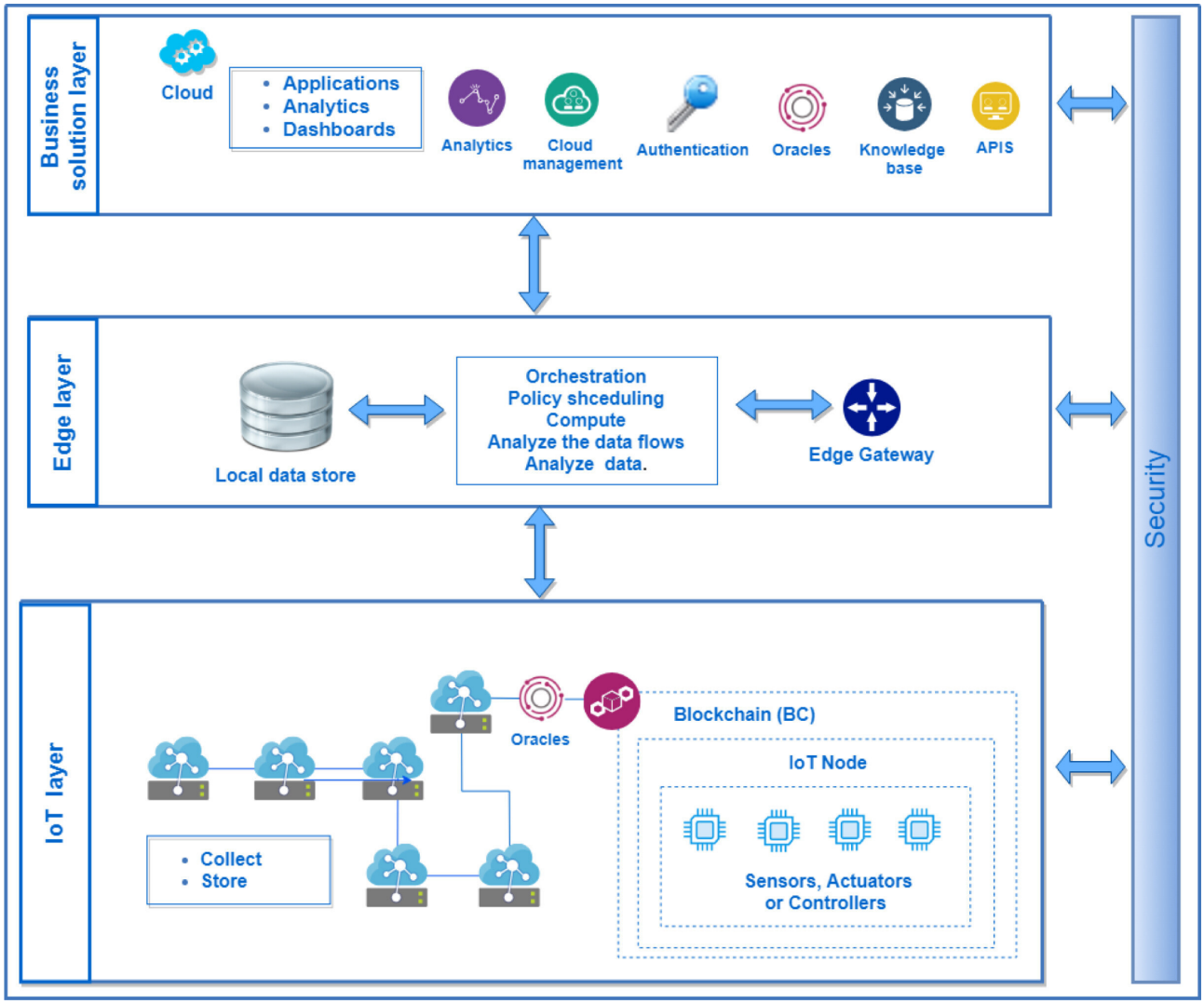

Figure 2.1 – Global Edge Computing Architecture - GECA. [112]

In this proposed reference architecture, the IoT layer comprises devices or physical objects that monitor services, activities, or equipment in operation. These devices include sensors, actuators, controllers, and gateways for IoT environments. Usually, they are based on technologies traditionally used for embedded systems development [63], as, for example, Systems-on-Chip (SoC) [68].

Also, in the IoT layer, to guarantee the integrity and validity of the data, a basic blockchain scheme (indicated as Blockchain (BC) in Figure 2.1) is adopted. When data is captured by sensors (inside of IoT Node in Figure 2.1), a hash is generated and stored in the blockchain, and then the data is sent to the edge nodes (blue bubbles in Figure 2.1) to be stored off-chain [112].

Thus, an edge node can verify data integrity at any time by generating a hash on the locally stored data and comparing it with the hash previously stored in the blockchain. The data is still valid if the hashes have the same value. Brokers called oracles in Figure 2.1 are responsible for this interaction between edge nodes and blockchain. This GECA´s IoT layer corresponds to the Edge layer in the generic three-tier architecture, shown in Figure 1.2.

The proposed Edge layer implements the orchestration of diverse technological assets, improving supply, monitoring, and updating existing resources by managing physical resources or analyzing large volumes of data in real time. This layer filters and pre-processes the obtained data from the IoT layer, usually implemented using micro-controllers with reduced computational and limited storage resources. This GECA´s Edge layer corresponds to the Fog layer in Figure 1.2.

The filtered and processed data is then transferred to the layers in the Cloud, known as the Business solution layer in the proposed GECA. The integrity of all the data obtained from the IoT layer is another responsibility of the Edge layer by using a blockchain implemented in the lowest layer of the architecture.

The Business Solution layer, corresponding to the Cloud layer in Figure 1.2, provides services and business applications in the Cloud. Each API call can be activated by executing more complex operations involving interactive interfaces and are part of the business applications ecosystem. The main components of this layer include analytics, cloud management, authentication, knowledge base, and API (Application Programming Interface), for example, REST, XML, and SOAP.

As already mentioned in the previous chapter (see Figure 1.2), there are proposed reference architectures for IoT that suggest some additional layers between this proposed GECA's Edge layer and the Business Solution layer [121]. More powerful computational, storage, and communication resources will compose these intermediate layers, which can interact directly with the resources in the above layer, the Business Solution layer [75].

So, many reported reference architectures for IoT systems are well covered in [99]. However, the decentralized architecture, elaborated in the present thesis, adopted the ter-

minology from this IoT architecture reference (named GECA) mentioned above since the main objective of the present thesis is to demonstrate the feasibility of distributed ledgers combined with edge computing concepts in a decentralized way, to implement intrusion detection in the network edge layer.

Moreover, following the concepts from the Peer-to-Peer (P2P) networking paradigm [108], the decentralized architecture developed in the present thesis establishes a P2P overlay on the components in the IoT and Edge layer of the GECA reference architecture.

Remember that P2P networks are decentralized distributed systems and enable computational systems to share and integrate their computing resources, data, and services [108]. In P2P networking, each peer node works as both client and server, requesting services from and providing services for other peer nodes. A P2P network can be seen as a logical overlay network over a physical infrastructure. Thus, as detailed in Chapter 3, the decentralized architecture developed in the present thesis defines a logical unstructured P2P overlay, where each P2P node corresponds to an IoT or Edge node in the GECA reference architecture, which can communicate between them by using cryptographed remote procedure calls.

2.1.2 IoT Security and Intrusion Detection

Given the large volume of generated data by most IoT applications, the edge computing paradigm [125], where processing is performed preferentially in devices at the network edge, is becoming predominant to data network traffic be minimized.

Even so, some IoT applications need powerful resources that usually are only available at the cloud layer or at least at the intermediate fog layer (e.g., big data analytics may need a huge amount of storage and computational power). Thus, any efficient and effective architecture for IoT security must consider all security threats inherent to all these three layers [128]. Some of the main threats to IoT are summarized in Table 2.1 and explained in the following.

All the threats listed in Table 2.1 can produce peculiar traces of network traffic, which usually characterize each possible security risk. The presence of these traces is one of the main arguments for using techniques based on network traffic analysis to detect possible anomalous behaviors in a system that can indicate some security issues.

Zarpelão et al. [124] claim that some ongoing projects for enhancing IoT security include methods for providing data confidentiality and authentication, access control within the IoT network, privacy and trust among users and things, and the enforcement of security and privacy policies. However, even with these mechanisms, IoT networks are vulnerable to

Table 2.1 – Main threats to IoT security. [52]

Threat	Description	Attack
Cloning of things	By manufacturing, a device copy is produced	Conventional
Substitution of thing	By installation, a device is replaced	Conventional
Firmware replacement	By operation, the device's firmware has security parameters extracted and altered	Conventional
Spoofing, modifying, or replaying routing information	Create routing loops, attract or repel network traffic, extend or shorten source routes, and so on	Routing
Man-in-the-middle	Modify communications from entity A to another entity B without both A and B noticing it	Routing
DoS, distributed DoS	By exhausting service provider resources and/or network bandwidth, make service unavailable	Denial of Service (DoS)

multiple attacks to disrupt the network. For this reason, another line of defense designed for detecting attackers is needed. Intrusion Detection Systems (IDSs) can fulfill this purpose.

Mitchell et al. [69] define *Intrusion Detection* as consisting of collecting data related to the actions being performed in a system and analyzing the collected data to identify the eventual presence of intruders in the system. The identified intruder actions, known as intrusions, usually try to obtain unauthorized access to the system.

Also, according to Michell et al. [69], there are two kinds of intruders: *internal* ones, which are users inside the network with some degree of legitimate access that attempt to raise their access privileges to misuse non-authorized privileges, and *external* ones, which are users outside the target network trying to gain unauthorized access to system information.

Mohammadi et al. [70] define *Intrusion Detection Systems* as systems designed to inspect all in bound and outbound traffic and identify the suspicious traffic and actions of various attackers in a timely and accurate manner. So, an IDS *monitors a host or a network and alerts the system administrator when it detects a security violation* [124].

Due to the previously mentioned characteristics of IoT-based systems, current solutions for IDS need to be adequate. In traditional networks, the IDS agents are deployed in nodes with higher computing capacity, and IoT networks are usually composed of nodes with minimal resources. Therefore, finding nodes supporting IDS agents is more challenging in IoT systems.

Moreover, as argued by Zarpelao et al. [124] *In traditional networks, end systems are directly connected to specific nodes (e.g., wireless access points, switches, and routers) responsible for forwarding the packets to the destination. IoT networks, on the other hand, are usually multi-hop. Then, regular nodes may simultaneously forward packets and work as end systems*.

Usually, an IDS consists of sensors (for collecting data), an analysis engine (to receive data from sensors and detect intrusion based on the investigation of the collected data), and a reporting system (to generate alerts when an intrusion is detected). In this context, Zaperlão et al. [124] propose a taxonomy of IDS for IoT, shown in Figure 2.2, divided into placement strategy, detection method, security threat, and validation strategy.

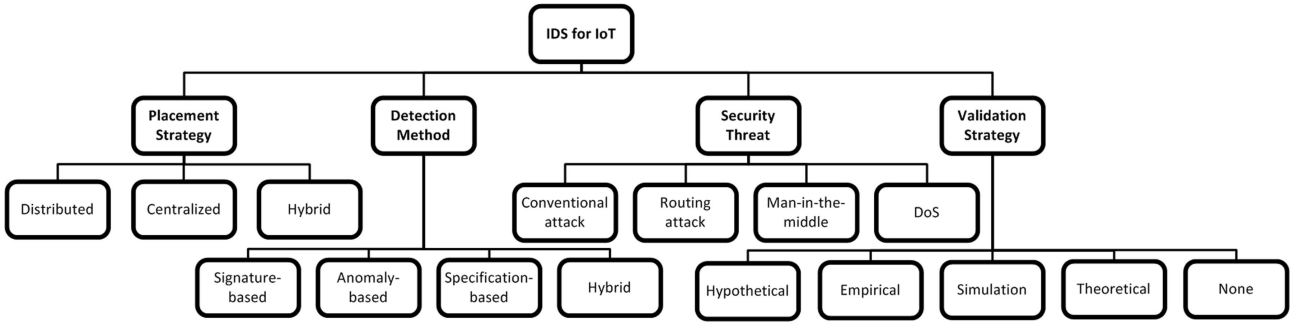

Figure 2.2 – Taxonomy of IDS for IoT. [124]

As shown in Figure 2.2, the *placement strategy* can be distributed (IDSs are placed in every physical object of the network), centralized (the IDS is set in a centralized component, for example, in the border router or a dedicated host.), or hybrid (combines concepts of centralized and distributed placement).

As *detection method*, there are four categories: signature-based (IDSs detect attacks when a system or network behavior matches an attack signature stored in the IDS internal databases), anomaly-based (compare the activities of a system at an instant against a normal behavior profile and generates the alert whenever a deviation from normal behavior exceeds a threshold), specification-based (detect intrusions when network behavior deviates from manually specified rules in a set of specification definitions), and hybrid (explores all the above methods trying to maximize their advantages and minimize the impact of their drawbacks) [14].

It is important to note that signature-based IDS approaches cannot deal with new attacks in which their signatures are unknown, and traffic is encrypted. However, anomaly-based IDS schemes operate according to the users' expected behavior profiles and can detect newly unleashed attacks.

The main types of *security threats* include conventional attacks, routing attacks, man-in-middle attacks, and Deny of Service (DoS) [124]. Cloning of things, the malicious substitution of things, firmware replacement, and extraction of security parameters can be organized according to the process phase in which the attacker acts - manufacturing (cloning of things), installing (malicious substitution of things), operation (firmware replacement and extraction of security parameters) or maintenance (firmware replacement).

Routing attacks consist of spoofing, modifying, or replaying routing information to create routing loops, attract or repel network traffic, extend or shorten source routes, and so

on. A man-in-the-middle attack is performed when an attacker node modifies communications from entity A to another entity B without both A and B noticing it [52].

Physical objects usually have tight memory and limited computation capacity, so they might be vulnerable to DoS attacks that can be launched in a traditional way, exhausting service provider resources, and network bandwidth or targeting the wireless communication infrastructure, jamming the communication channel [52].

Zarpelão et al. [124] state that *IDS validation consists of checking that the built model behaves with satisfactory accuracy within the study objectives*. Based on this validation concept, they investigated 18 works on intrusion detection. They identified the validation types listed in Figure 2.2: *hypothetical* (use of examples, having unclear relation to actual phenomena and degree of realism), *empirical* (for example, a systematic experimental gathering of data from operational settings), *simulation* (use of some simulator for IoT), *theoretical* (formal or precise arguments to support results), or *none* (no validation methods are employed).

As pointed out by Zarpelão et al. [124], only one of the 18 works performed empirical validation and compared the results with other IDS approaches. Since there is no standard benchmark for IDSs, they recommend as very important at least to perform validation with realistic data sets oriented to IoT systems, with realistic network configuration, realistic traffic, labeled dataset, full capture, and multiple attack scenarios. Unfortunately, many of these characteristics are not present in the existing data sets for IoT, so creating specific data sets must be the object of research.

Kumar et al. [52] claim that many of the difficulties in developing Intrusion detection algorithms for IoT-based systems are related to their dynamic characteristics: intrusive and normal behavior of users, applications, and networks is always changing over time.

These dynamic behaviors are mostly due to the many new IoT devices and applications constantly coming up and even due to new functionalities offered by current devices and applications. Thus, an IDS must constantly learn and adapt itself to cope with the changes to be effective and efficient. Machine learning-based Intrusion Detection is a possible approach to this issue, as suggested by [92].

Access control is the usual mechanism to grant permission to users and devices to access shared resources in an IoT network [96]. An access control process usually includes authentication, authorization, audit, and administration functions. Authentication verifies the identity of a user, process, or device. Authorization grants and denies specific user, process, or device requests. The audit function allows the analysis of records to assess the adequacy of the access control, determining if it complies with specified policies. And administration consists of creating, provisioning, and managing users, groups, roles, devices, and policies.

In the context of Internet of Things (IoT) applications, XACML and OAuth are the most commonly utilized technologies for access control to shared resources. XACML (eX-

tensible Access Control Markup Language) [89] includes an attribute-based access control policy language, an architecture, and a processing model to evaluate access requests. The Policy Enforcement Point (PEP) enforces policy application by intercepting the request and forwarding it to the Policy Decision Point (PDP) for evaluation. Based on the PDP's evaluation, the PEP decides whether to grant or deny access. OAuth [23] is a token-based authorization protocol, where permission to access a given resource is granted to a requester only if it possesses a token requested to and provided by the resource owner, which also determines the allowed access level.

Both technologies are unsuitable for IoT applications since they assume a central authorization server, which receives requests and evaluates them to decide if access is granted to the requester. This assumption restricts IoT scalability and also represents a single point of failure. Thus, many blockchain-based approaches to access control for IoT systems allow the decentralization of the process and handle these two issues [96].

2.2 Machine Learning-based Intrusion Detection

Machine Learning (ML) is a computational paradigm allowing machines to adjust their behavior based on helpful knowledge inferred from data sets. ML techniques have been used for classification, regression, and estimation tasks, which can be applied to solve problems in fraud detection, computer vision, natural language processing, and many other areas [43].

Intelligent IDS is a kind of IDS, which explores the use of machine learning methods and techniques to detect intrusion based on historical data about previous incidents and behaviors of the IoT devices, as well all the participating nodes of the IoT-based system [92]. Table 2.2 lists some ML algorithms and techniques for implementing IDSs.

An intelligent IDS can be categorized, according to data labels in the used datasets and the adopted machine learning method, as (i) supervised, (ii) semi-supervised, and (iii) unsupervised IDS. Supervised intelligent IDSs use a fully labeled dataset to train the model and usually achieve the highest accuracy in recognizing intrusions. Since finding a fully labeled dataset for intrusion in IoT systems is challenging, implementing this type of IDS takes work.

Unsupervised IDSs have no access to a labeled dataset and try to build a cluster of similar data samples. Any data point out of the cluster is considered an anomaly and eventually indicates an intrusion attempt. In general, this kind of intelligent IDS can generate a high False Alarm Rate (FAR), where benign behavior is classified as malign and identified as an intrusion. Semi-supervised intelligent IDSs use a partially labeled dataset to train the model for intrusion detection.

Table 2.2 – Machine Learning-based IDSs.

IDS	Su,Se,Un[a]	Description	ML	Dataset	H/N[b]	IoT
Sindhu et al. [111]	Su	Reduce KDD's redundance, perform feature selection, and combine decision tree and a neural network, to implement a multi-class classifier	DT[e], ANN[i]	KDD	N	No
Ammar et al. [9]	Su	Complement to a conventional IDS to classify traffic identified as suspicious	DT[e]	ISCX	N	No
Hodo et al. [41]	Su	A three layers feed-forward Neural Network is used offline to identify DoS attacks	ANN[i]	Script in C	N	Yes
Koroniotis et al. [48]	Su, Un	Present some intrusion detection approaches based on three ML techniques to evaluate the Bot-IoT dataset	SVM[h], RNN[g], LSTM[k]	Bot-IoT	N	Not only
Wu et al. [119]	Su	Combine ML techniques in a 5-class classifier (normal, DoS, Probe, R2L, U2R)	DBN[i], SVM[g]	NSL-KDD	N	Yes
Kumari et al. [54]	Sup, Se	Intrusion detection was conducted using both classifiers engines (SVM and FCM) to improve confidence	SVM[h], FCM[l]	NSL-KDD	N	No
Almiani et al. [6]	Su	A multi-layered recurrent neural network-based IDS to be deployed at the fog layer	RNN[g]	NSL-KDD	N	Yes
Mohammadi et al. [70]	Su	Present many IDS approaches based on SVM combined with various techniques	SVM[h] and others	KDD, NSL-KDD	N	Yes
Kumar et al. [53]	Su	Preprocessing data set to remove redundancy, perform feature selection, and combine four decision trees to implement an IDS as a 5-class classifier	DT[e]	UNSW-NB15	N	Yes
Alsaedi et al. [8]	Su, Un	Present many intrusion detection approaches based on various ML techniques to evaluate a data-driven dataset, instead of a usual network flow-driven one	LR[c], kNN[d], RF[f], SVM[h], LSTM[k]	TON_IoT	Both	Yes

[a]Supervised,Semi-supervised,Unsupervised [b]Host/Network, [c]Logistic Regression, [d]k-Nearest Neighbour, [e]Decision Tree, [f]Random Forest [g]Recurrent Neural Network, [h]Support Vector Machine, [i]Artificial Neural Network, [j]Deep Belief Network, [k]Long-Short Term Memory, [l]Fuzzy C-Means

In Table 2.2, there is an indication of this classification for each approach, as well as a brief description, adopted type of model learning, used dataset, if it is a host-based or network-based IDS, and if it has a focus on IoT-based systems.

2.2.1 Datasets for IoT

As previously mentioned, supervised machine learning depends essentially on the available data to be used during a model's training, which will be used to predict eventual anomalous behavior that can indicate an intrusion. Most of the available data sets are not representative of IoT systems [67]. Table 2.3 presents common data sets with reported results in the literature for intrusion detection approaches.

Table 2.3 – Common datasets.

Data set	Description	Year	IoT	Samples	Feat.	Classes
KDD99 [70]	Records of network traffic using tcpdump	1999	No	4,898,430	41	5
NSL-KDD [114]	Selected records from KDD99 by eliminating redundancy	2009	No	148,517	41	5
UNSW-NB15 [73]	Records of real benign network activities (87.35%) and synthetic attack scenarios (12.65%), captured with tcpdump	2012	No	2,540,440	49	10
ISCX [109]	Records of generated normal and attack traffic samples from profiles specific to some internet protocols	2015	No	2,450,324	9	1
BoT-IoT [48] [101]	Records of a realistic network environment, with normal (0.01%) and botnet traffic (99.99%)	2019	Yes	3,668,522	42	5
TON_IoT [8]	Records of a heterogeneous dataset, including telemetry data and network traffic (3.56% benign flows and (96.44%) attack samples	2020	Yes	22,339,021	44	1
N-BaIoT [66]	Records real network traffic from 9 IoT devices infected by two malware	2018	Yes	7,062,606	115	11
Edge IIoT [30]	Records real network traffic generated by more than ten types of IoT devices, with a subset specific for use in deep learning	2023	Yes	2,219,201	92	15
IoT23 [34]	Records real and labeled malware and benign traffic from 3 IoT smart home devices	2020	Yes	1,444,674	24	10

The KDD99 or KDDCup dataset is derived from the DARPA 98 dataset, which was generated by MIT'S Lincoln Lab for the evaluation of intrusion detection systems and simulated a small Air Force network connected to the internet [70]. The NSL-KDD dataset [114] is derived from the KDDCup, mainly by removing data redundancy. These DARPA-based datasets contain 41 input features with discrete or continuous values, including Transmission Control Protocol (TCP) connections data, duration, prototype, number of bytes from the source and destination IP (Internet Protocol) addresses, etc.

KDD99 and NSL-KDD datasets are labeled with five classes, one for normal traffic and four classes representing attack of: (i) Denial of Service - DoS (the attacker overloads the victim with many requests, exhausting its memory, the network interface, or other server resources); (ii) User to root - U2R (the intruder gains unauthorized access into the system as a normal user, which it then tries to modify as an admin-user exploiting vulnerabilities; (iii) Remote-to-Local - R2L (the intruder uses system vulnerability to act as a normal user; and, (iv) PROBE (the intruder scans the network to retrieve information about the victim's computers attached to a network) [21].

KDDCup and NSL-KDD are obsolete since they do not have more recent types of attack [19] and even so, are adopted in many recently published works [103] [115] [119] [97]. More recent and adequate datasets for IoT are also included in Table 2.3.

UNSW-NB15 [73] is a dataset developed at UNSW Canberra, containing 100 Giga bytes of benign and anomalous traffic. It consists of 49 features and flows labeled with ten classes: normal, fuzzers (feeding a program/network with randomly generated data to force its suspension), analysis (port scan, spam), backdoors (bypass system security to obtain unauthorized access), DoS (try to make a server or network resource unavailable for used by causing excessive overhead to a provided service), exploits (use of a known security problem to exploit system vulnerabilities), generic (attack applicable to all block-ciphers), reconnaissance (attacks to gather system information), shellcode (code as payload in exploiting system vulnerabilities), and worms (attacker self replicated to other systems).

ISCX [109] was produced at the Canadian Institute for Cybersecurity, using the concept of profiles that can characterize attacks in a network. From profiles generated for various network protocols (HTTP, FTP, SSH, etc.) and activities related to intrusion detection systems, network traffic was generated that was captured and used to build the dataset with 2,381,532 normal traffic samples and 68,792 attack samples.

BoT-IoT was created by Koroniotis et al. [48] at the Research Cyber Range lab of UNSW Canberra using both real and simulated IoT network traffic. This dataset consists of normal and botnet traffic, with 42 features, and containing 477 (0.01%) benign flows and 3,668,045(99.99%) attack ones, that is 3,668,522 flows in total. The testbed environment used to generate it includes (i) network services and platforms (legitimate and malicious virtual machines), (ii) a simulated IoT-based smart-home application (with thermostat, garage door, refrigerator, weather monitoring system, and lights), and (iii) a tool to perform forensics analytics (Argus tool). Bot-IoT was used to train a Recurrent Neural Network (RNN), a Support Vector Machine (SVM), and a Long-Short Term Memory (LSTM), and the SVM produced better results (accuracy of 100%). In the present thesis, this dataset was adopted in some of the experiments, using a subset of it and considering four types of attacks (DDoS, DoS, Information gathering, and Data theft) and normal traffic.

TON_IoT [16] includes telemetry data of IoT services, network traffic of IoT networks, and operating system logs, and contains 44 features, with 796,380 (3.56%) benign flows and 21,542,641 (96.44%) attack samples, that is, 22,339,021 flows in total. TON_IoT was collected from a realistic and large-scale network designed at the Cyber Range and IoT Labs, UNSW Canberra, at the Australian Defence Force Academy (ADFA), using a testbed of IoT and IIoT networks.

The N-BaIoT dataset was created by Meidan et al. [66], capturing actual network traffic samples from 9 IoT devices infected by two malware, Mirai and BASHLITE. N-BaIoT contains a total number of instances of 7,062,606, with 23 statistical features about the network traffic, which were computed for five different time windows, resulting in a total of 115 features. The normal traffic was captured after a new testbed installation to ensure no infected streams were injected. The malicious traffic consists of 10 attack types carried by

the two botnets (Mirai and BASHLITE). It is reported a TPR of 100% and mean FPR of 0.7%, using a fine-tuned autoencoder.

IoT23 [34] is a dataset that consists of IoT network traffic captured in the Stratosphere Laboratory at CTU University in the Czech Republic from 3 different smart home IoT devices (Amazon Echo, Philips HUE, and Somfy Door Lock). It consists of real and labeled IoT malware infections and benign traffic, including 23 captures: 20 malicious and three benign captures.

Edge Industrial IoT dataset [30] was generated using a testbed organized into seven layers, including the Cloud Computing Layer, Network Functions Virtualization Layer, Blockchain Network Layer, Fog Computing Layer, Software-Defined Networking Layer, Edge Computing Layer, and IoT and IIoT Perception Layer. Each layer incorporates technologies related to IoT and IIoT applications, such as the ThingsBoard IoT platform, OPNFV platform, Hyperledger Sawtooth, Digital twin, ONOS SDN controller, Mosquitto MQTT brokers, and Modbus TCP/IP. The IoT data was generated from various IoT devices, such as Low-cost digital sensors for sensing temperature and humidity, Ultrasonic sensors, Water level detection sensors, pH Sensor Meters, Soil Moisture sensors, Heart Rate Sensor, Flame Sensor, etc.), from which it was identified fourteen attacks related to IoT and IIoT. These are categorized into five threats: DoS/DDoS attacks, Information gathering, Man in the middle attacks, Injection attacks, and Malware attacks. This dataset was used to train different machine learning models, including a deep learning neural network, with 47 trainable parameters that presented 99.99%

Ring et al. [101] and Sarhan et al. [105] present an exhaustive list of datasets, giving many details of each one. Most datasets are oriented to intrusion detection in traditional networks and not specific to IoT.

2.2.2 Metrics

An intelligent IDS should be trained using some subset of a given dataset, called *training dataset*, and should be evaluated using another subset of the dataset, called *testing dataset*. The classifications produced by the intelligent IDS can be: (i) True Negatives (TN), normal traffic is correctly recognized as normal traffic; (ii) True Positives (TP), attacks, and malicious behaviors are correctly detected as intrusions; (iii) False Positives (FP), normal traffic is falsely detected as attacks; and, (iv) False Negatives (FN), attack traffic is falsely detected as normal traffic [67].

Usually, based on TN, TP, FP, and FN, common evaluation metrics for intelligent IDSs include:

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{2.1}$$

Accuracy is the fraction of correctly classified examples by the number total of examples in the dataset. This metric is adequate when the dataset is balanced, otherwise, it may not reflect the actual model quality.

$$Detection\ rate\ (DR) = \frac{TP}{TP + FN} \tag{2.2}$$

Detection Rate (or Recall) is the fraction of examples classified as positive, among the total number of positive examples. It is useful when False Negative is of high cost for Model quality, for example, Fraud Detection Model.

$$False\ Positive\ Rate\ (FPR) = \frac{FP}{FP + TN} \tag{2.3}$$

FPR is the fraction of examples classified as positive among the total number of negative examples. It is useful when False Positive is of high cost for Model quality, for example, email Spam Detection Model.

$$Precision = \frac{TP}{TP + FP} \tag{2.4}$$

Precision is the fraction of true positive examples among the examples that the model classified as positive. It is also useful when False Positive is of high cost for Model quality.

$$F1 - score = \frac{2}{\frac{1}{DR} + \frac{1}{Precision}} \tag{2.5}$$

F1-score is a combination of recall and precision, and when equal to 1 indicates perfect both precision and recall metrics. This is a metric of importance for an imbalanced dataset as equal importance is given to both Precision and Recall. An efficient intelligent IDS should have ideally 0% FPR and 100% DR.

2.3 Federated Learning

The term *Federated Learning* (FL) was proposed by Google in 2016 [65] to designate a machine learning technique where a group of client devices performs model training locally, using their private datasets, and a global model is generated by a central server, which only receives updates from local models and never the local datasets. The global model, generated by the central server, is then shared back to all the client devices.

2.3.1 Strategies for Model Training in FL

In conventional model training, a central server trains a model using data gathered from distributed end devices. The trained model is then shared back with the devices to be used in prediction or inference tasks. Figure 2.3 illustrates this approach.

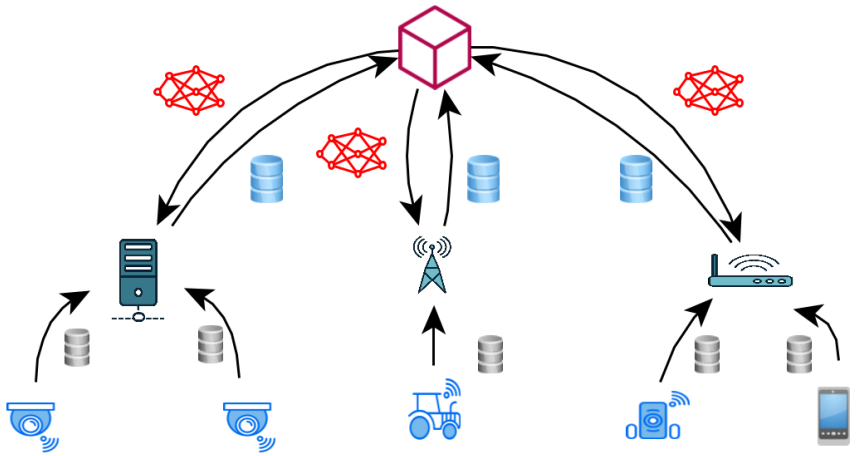

Figure 2.3 – Model training: Conventional.

In centralized FL (see Figure 2.4), local models are trained at devices based on available data; A server trains a global model by some aggregation applied on collected local models from devices. The trained global model is shared back to devices for updating their local models. As previously discussed, this approach corresponds to the centralized cloud-based one, which cannot support IoT peculiarities.

In the decentralized FL architecture (shown in Figure 2.5), each node can perform local model training based on local data and models shared by other nodes. It corresponds to the ideal concept of federated learning. But, most published approaches adopt a unique central server for aggregation implementation, leading to an undesirable single point of failure in the system. The decentralized FL architecture corresponds to the concept of Federated Learning, adopted in the present thesis.

It is important to note that, independent of the architecture, the model training objective is to minimize a training loss function, i.e., the difference between the learned values and the labeled data used to train the model.

In the decentralized FL architecture, the local training must converge to a consensus in the global shared model, depending on how its gradient will be computed [128].

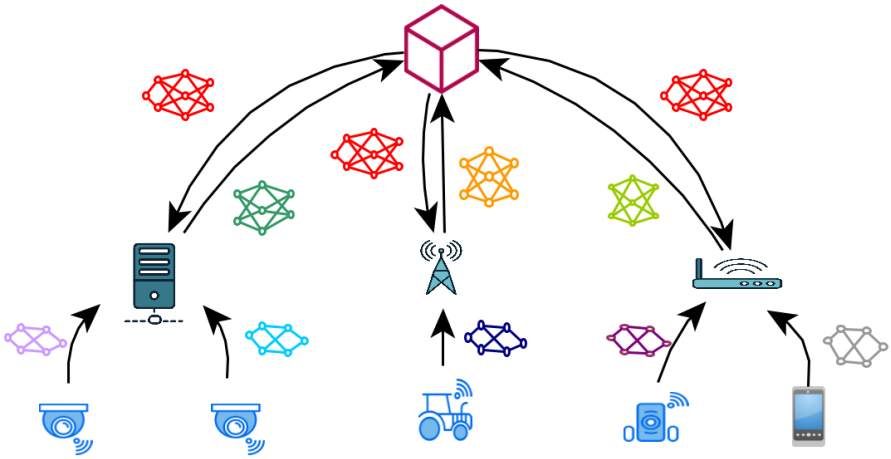

Figure 2.4 – Model training: Centralized FL

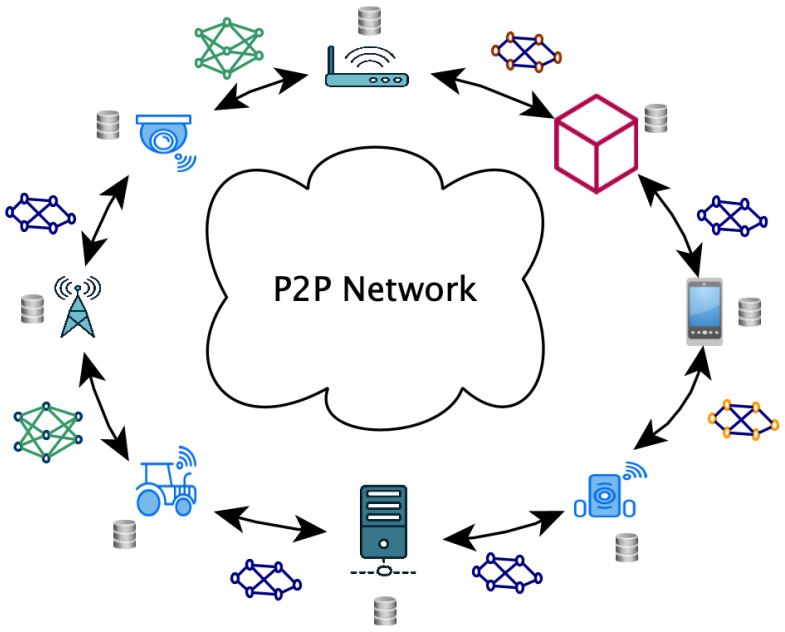

Figure 2.5 – Model training: Decentralized FL

Usually, the global shared model is generated by some aggregation algorithm applied to the locally generated models.

2.3.2 Aggregation Strategies

One possible aggregation algorithm is based on SGD (Stochastic Gradient Descent), which updates the gradient over tiny subsets (mini-batches) of the whole data set or even by selecting the clients that will contribute to the model updates, calculated independently by each client device [128].

One most common aggregation algorithms, proposed by McMahan et al. [65] and called the FedAvg method, consists of iterative model averaging, i.e., the client devices update the model locally with one-step SGD, and then the server averages the resulting models with weights. This aggregation method allows for treating unbalanced and non-IID (non-Independent Identically Distributed) since the distributed data may come from various sources.

Another important issue related to model training in Federated Learning is the communication costs since the client devices will send model updates to the central server and eventually receive whole global models from the server. Some approaches to minimize these communication costs are reported in the literature. For instance, McMahan et al. [65] suggests performing more iterations in the client devices to reduce the training rounds between them and the central server, which is necessary for the training convergence.

Another approach to computing global parameter aggregation, called FedCS (Federated Learning with Client Selection), is described by Nishio et al. [84], which performs a selection of the client devices that will participate in each training round based on information about available computational resources and communication channels collected on randomly selected client devices. From these pre-selected client devices, it is chosen for the training round the ones that allow to aggregate the maximum number of client devices in a given deadline [84]. Figure 2.6 illustrates this protocol.

After initialization (step 1), the central server requests resource information from randomly selected client devices (step 2). Based on the received information, the central server chooses devices that will participate in the training round (step 3) and sends the global model to them (step 4). The central server receives model updates from each client device according to the planned schedule (step 5) and computes model aggregation. Training rounds are iterated until to obtain model convergence.

These approaches to aggregation in federated learning can be applied in an intelligent IDS based on federated learning [36]. Each client device will train a local model based on the network traffic it captures, contributing to model updates for the central server. However, since network traffic patterns are usually specific to given types of devices, it would be important to train global models per device type [45].

In this context, federated learning offers a very efficient approach to dealing with security risks due to the limitations of the resource-constrained devices in IoT systems since

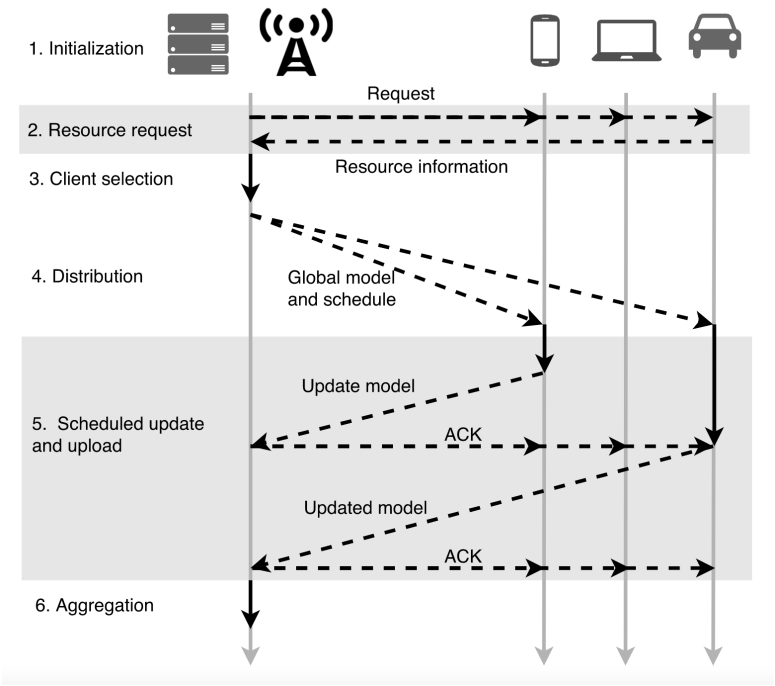

Figure 2.6 – Overview of the FedCS protocol. [128]

these devices take advantage of the knowledge on intrusion detection captured in the globally trained models, even if they don´t have enough resources to take part on the model training of them.

2.3.3 Privacy Issues

Since data privacy-preserving is a big issue due to the recent regulations in Europe and Brazil intended to protect user's privacy and provide data security, federated learning also here offers a significant advantage since model training is not dependent on access to the device data, which is only available locally at each client device [56]. Moreover, since device data is not transmitted through the network, security risks are also minimized.

However, since an initial global model must be shared with the client devices and further model updates should be transmitted to the central server, some malicious client devices may capture the network traffic, threatening data privacy, and even make changes to the models, intending to influence on the final results. Thus, some techniques should be adopted to cope with these threats.

Differential privacy [118] is one of these techniques, which adds noise to the data, or uses generalization methods to obscure model attributes until the third party cannot identify who sent the model updates, thereby making the data impossible to be restored and so to protecting user privacy.

2.3.4 Strategies for Inference Phase in FL

For the inference phase, a client device receives an input (a network traffic sequence in an intrusion detection case). It applies its local model to classify it as normal or attack traffic. Another possibility is for the client device to send the input to an edge server with a trained model to perform the inference and then to receive back the classification [128].

It is also possible to perform part of the inference in a local partial model in the client device that sends the partial result to an edge server, which has another part of the model, that finishes the inference, giving back the classification to the client device. Yet another possibility for the inference is that the edge server uses a trained model at a cloud server to perform part of the inference [5]. One of the above inference modes can be chosen according to the available computation, memory, storage, and communication resources in the client devices and edge servers.

2.4 Distributed Ledgers Technologies

A distributed ledger (DL) maintains transactions or digital interactions in a transparent, immutable, and auditable manner. Blockchain, which was first introduced in 2008 by Nakamoto [76], is one of the technologies to implement DLs [59], by storing a set of transactions in a block and chaining every block using cryptographic techniques [117]. Each transaction can be a transfer of values between different entities that are broadcast to the network and collected into the blocks. The transactions are mined into a block by the so-called miners, and so they add transactions into the blockchain [61].

2.4.1 Mining or Validation Process

Through the mining process, the participant nodes can reach a secure, tamper-resistant consensus, which is a fundamental problem in distributed systems that requires two or more nodes to agree with each other on a given decision, even if some of the nodes are unreliable. Many different consensus algorithms have been used by blockchain, including proof of work (PoW), proof of stake (PoS), and proof of storage.

In a blockchain, there is no central trusted authority to maintain data validity, which the participating nodes in the blockchain-based system guarantee. Each node has a copy of the ledger, and they keep their copy up-to-date by using some consensus algorithm to

decide the validity of any change or update to the ledger. Thus, the ledger replica in a blockchain guarantees high availability for the entire system [1].

Each block stores a cryptographic hash of the block that precedes it and the transactions stored within it. Thus, tampering with blocks requires propagation of the change to all blocks that follow it, creating a chain of blocks from the genesis block (the first one in a blockchain, which has no previous block) to the current block.

2.4.2 Smart Contracts

Smart contracts can execute some action associated with registered transactions on a blockchain. They provide the automation mechanism for the execution of code, according to implemented rules that have been agreed upon by the participant nodes [126].

It is worth noting that there are at least two types of blockchain: public one, where anyone can participate in the network, and private or permissioned one, where only privileged nodes can participate [18].

2.5 Summary

This chapter introduces the main topics related to the present thesis, conceptualizing IoT security, machine learning-based intrusion detection systems (IDSs), and the architecture of IoT systems. Initially, it describes the architecture of IoT systems, particularly the GECA reference architecture. It explains the roles of the IoT, Edge, and Business Solution layers in this architecture. The chapter also introduces the Peer-to-Peer (P2P) networking concept and its implementation in a decentralized architecture. The architecture aims to demonstrate the feasibility of combining distributed ledgers with edge computing concepts in a decentralized manner to implement intrusion detection in the network edge layer.

The chapter outlines the main threats to IoT security, such as cloning of things, firmware replacement, and denial of service (DoS) attacks, emphasizing that these threats can leave distinctive traces in network traffic. The chapter then defines machine learning-based IDSs, highlighting the importance of representative datasets for IoT systems based on deep learning techniques to model training, particularly federated learning, and presenting common datasets used for intrusion detection approaches.

The chapter also discusses strategies for the inference phase in federated learning, explaining that the choice of approach depends on the available resources for computation, memory, storage, and communication in client devices and edge servers. It introduces distributed ledger technologies, including blockchain, and discusses their role in maintaining

transactions or digital interactions in a transparent, immutable, and auditable manner. The mining or validation process in blockchain, which helps reach a secure, tamper-resistant consensus, is also explained.

Thus, this chapter has introduced the main concepts mentioned in the following thesis chapters.

3. DECENTRALIZED INTRUSION DETECTION ORIENTED TO IOT

This chapter presents a DeCentralizeD, Federated Learning-based approach to intrusion detection in IoT, called DCD-FL. First, an overview of DCD-FL and its architecture is given, then the workflow through the developed approach is described. Finally, a prototype, implemented to validate the defined architecture, is presented in detail.

3.1 Overview

DCD-FL is an approach to intrusion detection that combines decentralized computing and federated learning. Its architecture consists of a network of Peer-to-Peer (P2P) nodes, each composed of computation, storage, and communication components, as illustrated in Figure 3.1.

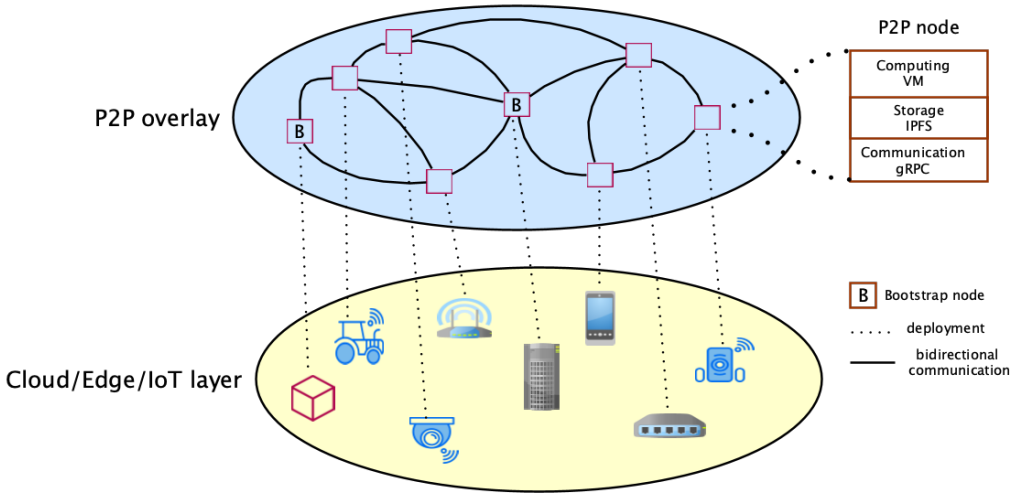

Figure 3.1 – DCD-FL: P2P network as an overlay of components in IoT layers

The P2P nodes can be virtual machines, computers, or devices from the cloud, edge, or IoT layers, selected from available components in the IoT system or some provisioned resource specifically for the intrusion detection process. This P2P structure ensures that DCD-FL has a fault-tolerant architecture, where a set of nodes is oriented to specific tasks in the intrusion detection process.

As indicated in the legend of Figure 3.1, each P2P node may have a specific type and so be responsible for particular tasks; for instance, the two bootstrap nodes are well-known contact peers at start-up to provide integration of joining peers into the P2P network. As shown in Figure 3.2, P2P nodes of specific types can be dynamically grouped to form P2P

clusters, according to the demand from the intrusion detection process, to perform particular tasks in a fault-tolerant and load-balanced way.

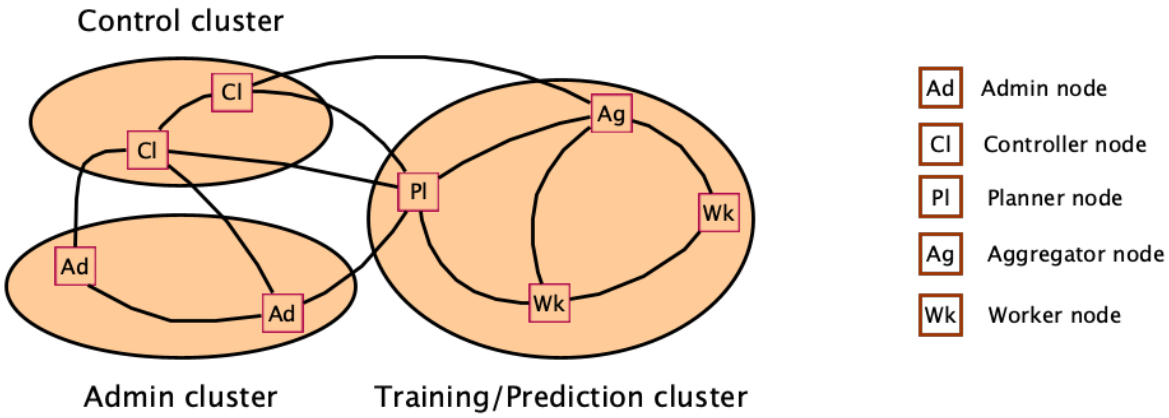

Figure 3.2 – DCD-FL: P2P clusters to perform specific tasks

P2P nodes can be of type: Admin (*Ad*), Controller (*Cl*), Planner (*Pl*), Aggregator (*Ag*), Worker (*Wk*), and Storage (*Str*). Figure 3.2 shows clusters composed of these types of nodes.

Ad nodes in the admin cluster manage information about the registered nodes and tasks to be executed and in execution, in a decentralized way, by using a blockchain to store metadata and a decentralized storage resource to store all data.

Cl nodes in a controller cluster perform access control to the resources and tasks available in DCD-FL by consulting the authentication/authorization data maintained by the admin cluster.

Following the federated learning approach, the training/prediction cluster trains the local model (employing *Wk* nodes) and global models (by using the *Ag* nodes), collecting and sharing them with the nodes involved in each planned task (defined by the *Pl* nodes).

There are also storage clusters composed of *St* nodes (not exemplified in Figure 3.1), which can register information in distributed ledgers and decentralized storage components.

Communication between all the nodes, intra- and cross-clusters, uses TLS, with a digital identity for each node, generated from the node's public key and managed by admin and controller clusters.

Each node uses an API service call to register itself in the admin cluster, providing its characteristic properties (its identity, computational, storage, and energy attributes), including its public key from a key pair (public and private keys) generated by the node itself, using a specific distributed client application (DApp). Only then the new node can participate in the IoT system.

Thus, admin and controller clusters manage the IDS's infrastructure, providing authentication and authorization services for nodes, including the P2P network topology management, by registering/maintaining the participant nodes. Inside each cluster, a lightweight local blockchain is used to register information generated/used by the IDS tasks securely and decentralized.

Thus, DCD-FL incorporates distributed ledgers (global and local blockchains) and off-chain storage to guarantee the integrity and high availability of the necessary information for the IoT security platform in a totally decentralized way. It also supports authentication, authorization, and auditing mechanisms, providing essential security features in a distributed manner.

DCD-FL is designed to handle problems related to IoT security, such as intrusion detection and data privacy. It achieves this by performing model training and predicting intrusion locally at edge nodes, selected based on their available resources. Only models are transmitted to other edge nodes (*Wk* nodes) for global model generation (at *Ag* nodes), ensuring data privacy.

The intrusion detection mechanism of DCD-FL is based on deep learning techniques, which explore network traffic patterns to predict possible attacks. Alerts are generated by *Pl* nodes when anomalous behavior is detected and sent to *Ad* nodes, which can activate *Pl* nodes to create new plans to be executed to handle the alerts.

Thus, one of the main characteristics of DCD-FL is decentralization. Admin, controller, and IDS-specific clusters can be deployed on different available edge components. This decentralized architecture reduces network traffic between edge servers and fog/cloud servers, enhancing privacy since monitored information in the network traffic data is not transmitted to fog and edge servers.

In conclusion, DCD-FL presents a practical approach to intrusion detection in IoT applications. It leverages the power of decentralized computing, federated learning, and distributed ledger technologies to provide a secure and efficient solution to IoT security issues.

3.2 DCD-FL for Intrusion Detection

In this thesis, a decentralized FL approach for intrusion detection in IoT applications generates global models, by means of a set of servers, called aggregators (or *Ag* nodes in the architecture), and not just one central server. Each *Ag* node manages a group of client IoT devices, called workers (*Wk* nodes), and interacts with the other aggregators to generate their models.

3.2.1 DCD-FL Components and Workflow

Figure 3.3 illustrates the intrusion detection workflow through the DCD-FL components, indicating, with little numbered circles, the point where each step occurs in the workflow and also indicating the algorithms used by each group of nodes in the workflow, which will be described in the following.

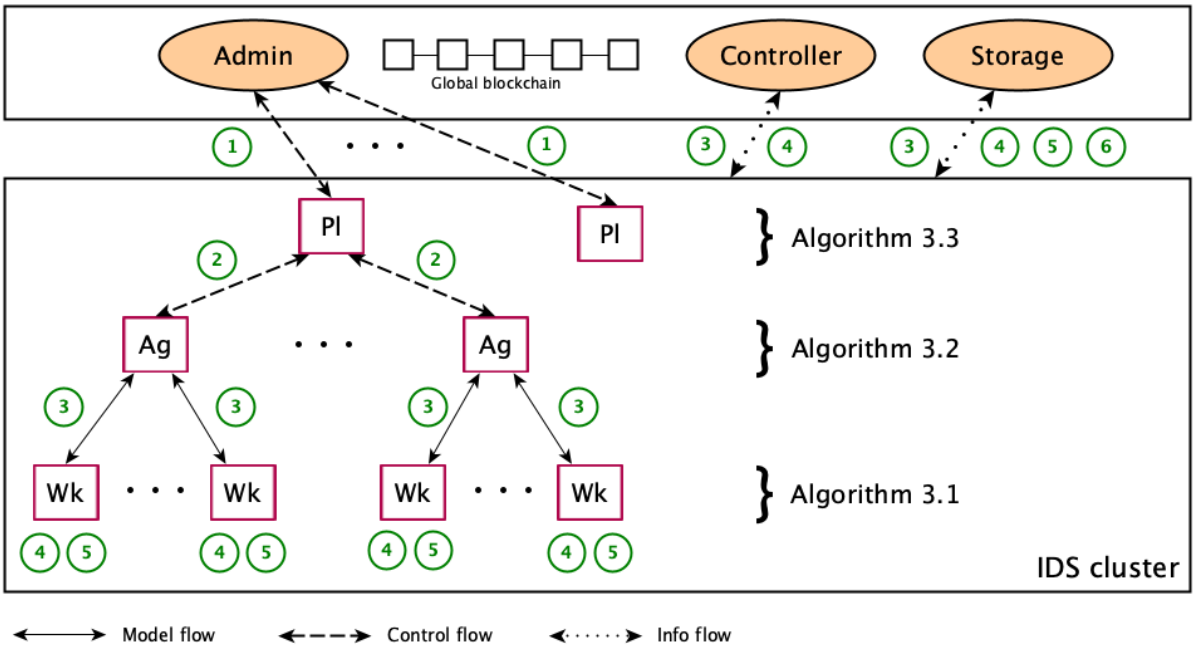

Figure 3.3 – DCD-FL components and workflow.

In **Step 1**, the admin cluster activates *Pl* nodes to perform task scheduling, resource allocation, and operation binding to determine which specific *Ag* and *Wk* nodes perform each task.

In **Step 2**, each planner *Pl*:

- starts the updates of global and local ML models;
- creates training rounds for global and local models, taking into account computational and storage resources, power consumption, and eventual timing constraints in the model training tasks; and,
- starts inferences/predictions in the devices to identify eventual anomalies according to activation by the admin cluster.

In **Step 3**, aggregator nodes *Ag*, which implement the generation of global models per device type and security threat according to the plan sent by the *Pl* node:

- collect local models from worker nodes to be considered in the aggregation, receiving control information from the controller cluster and accessing the storage cluster and its eventual available cache nodes;
- perform model aggregation, selecting and executing an aggregation algorithm indicated in the received plan; and,
- publish generated global models, accessing storage cluster and its eventual available cache nodes and notifying the planner node.

In **Step 4**, worker nodes *Wk*, which perform basic intrusion detection tasks:

- monitor/collect network traffic periodically, using a sniffer and registering it locally or by means of the storage cluster;
- prepare collected data for training, according to scheduling, by:
  - accessing data locally or by the storage cluster, depending on device resources,
  - applying data transformations to adequate them to the deep neural network algorithm to be used in the model training,
  - registering transformed data, locally or in the storage cluster, depending on the device resources;
- execute training of local models at each end device (sensors, actuators, smart devices) or edge devices (routers, switches, small servers), according to schedule, by:
  - accessing data locally or in the storage cluster, depending on device resources,
  - executing deep neural network algorithm for training, and
  - registering trained model updates in the storage cluster.

In **Step 5**, worker nodes also perform inference to identify eventual attacks by:

- using the current ML model at each device, according to scheduling, by:
  - obtaining transformed data, locally available or from storage cluster,
  - executing deep neural network algorithm for inference,
  - notifying inference result to the planner node.

As mentioned in the above DCD-FL workflow, the storage cluster implements decentralized storage for global and local models, offering cache resources to minimize communication between nodes of all clusters in the IDS.

In **Step 6**, the storage cluster can:

- authenticate nodes, which are requesting access, by using information from the controller cluster,
- validate node authorization by using information from the controller cluster,
- execute the operation, as requested by the node, in the case of valid authentication and authorization, and
- register operation logs using the controller cluster.

The following Section describes the algorithms performed by each node in the workflow of the intrusion detection process in DCD-FL in a bottom-up manner.

3.2.2 Algorithms

At each worker Wk_l, it is executed the local model training described in Algorithm 3.1, where θ_l represents the initial local model parameters, η is the learning rate, and t is the aggregation round that the worker Wk_l is participating.

Algorithm 3.1 – Local Model Training

Local Model Training (θ_l, t)
$\mathfrak{D}_l^t \leftarrow$ get_dataset (l, t)
$\theta_l^t \leftarrow \theta_l$
for each local epoch **do**
  for each batch b from $\mathfrak{D}_l^t$ **do**
    $g \leftarrow \nabla \text{Loss}\left(\theta_l^t; b\right)$
    $\theta_l^t \leftarrow \theta_l^t - \eta \times g$
  end for
end for
$\theta_l^{t+1} = \theta_l^t$
return θ_l^{t+1}

The function get_dataset (l, t) obtains a dataset of worker Wk_l to be used in the aggregation round t. The $\nabla \text{Loss}\left(\theta_l^t; b\right)$ expression represents the gradient of the loss function with respect to the local model parameters θ_l^t of worker Wk_l in the local round t, and minibatch data b obtained from the dataset $\mathfrak{D}_l^t$. The algorithm updates the local model parameters θ_l^t, using the SGD update rule $\theta_l^t - \eta \times g$, according to the specified learning rate η, and returns the updated local model parameters θ_l^{t+1} after training.

At each aggregator Ag_k, a global model is generated by the aggregation process, given by the Algorithm 3.2, where $\mathfrak{M}_k$ is the set of global models available in Ag_k and nW is

Algorithm 3.2 – Aggregate Models

Aggregate Models $(\mathfrak{M}_k, nW)$
for each aggregation round r **do**
  $\mathfrak{W}_r \leftarrow$ select_workers (nW, r)
  for each worker $Wk_l \in \mathfrak{W}_r$ **do**
    $\theta_{Wk_l} \leftarrow$ select_model $(\mathfrak{M}_k, Wk_l, r)$
    $\theta_{Wk_l} \leftarrow$ **Local Model Training** $\left(\theta_{Wk_l}, r\right)$
  end for
  $\{n_l, N_l$: number of samples used by worker Wk_l and number total of samples$\}$
  $\theta_k^r \leftarrow \sum_{l=1}^{l=|\mathfrak{W}_r|} n_l/N_l \times \theta_{Wk_l}$
end for
$\theta_k^{r+1} \leftarrow \theta_k^r$
return θ_k^{r+1}

the number of workers to be used, according to the current plan, in the round r, which is the current round of the plan being activated by the planner Pl_j.

For each planned aggregation round r, a set of nW is selected from the available workers. Then for each selected worker Wk_l, one deep learning model is selected from the set of available models $\mathfrak{M}_k$ in the aggregator Ag_k, and this selected model is given to the worker's Local Model Training function to update it. After receiving all the updated models θ_{Wk_l}, they are aggregated by applying the FedAvg technique [65].

At each planner Pl_j, a global model is generated by the aggregation process, given by the Algorithm 3.3, where $\mathfrak{A}_j$ is the set of available aggregators to Pl_j and nA is the number of aggregators to be used, according to the current plan, in the round p, which is the current round of the plan being activated by the planner Pl_j.

Algorithm 3.3 – Aggregate Global Models

Aggregate Global Models $\left(\mathfrak{A}_j, nA\right)$
for each planning round p **do**
  $\mathfrak{A}_p \leftarrow$ select_aggregators $\left(\mathfrak{A}_j, nA, p\right)$
  for each aggregator $Ag_k \in \mathfrak{A}_p$ **do**
    $\theta_{Ag_k} \leftarrow$ **Aggregate Models** $\left(\theta_{Ag_k}^0, nW\right)$
  end for
  $\{m_k, M_k$: number of samples used by aggregator Ag_k and number total of samples$\}$
  $\theta_j^{p+1} \leftarrow \sum_{k=1}^{k=|\mathfrak{A}_j|} m_k/M_k \times \theta_{Ag_k}$
end for
return θ_j^{p+1}

A set of *nA* aggregators are selected from the ones, which are available to the planner Pl_j. Then, a global model θ_{Ag_k} for each aggregator Ag_k is computed by the function Aggregate Models $\left(\theta^0_{Ag_k}, nA\right)$, which receives an initial global model of Ag_k and the number of workers that should be used by the aggregator, according to the current plan. A global model θ_j^{p+1} for Pl_j is generated by applying FedAvg on the models θ_{Ag_k}, which were obtained to each of the selected aggregators.

An Ad_i node from the admin cluster is responsible for starting an intrusion detection process. It must select one or more planners Pl_j, which will participate in the following workflow process and request that each one determine a plan to be executed. Ad_i follows the Algorithm 3.4, where $\mathfrak{P}_i$ is the set of available planners in Ad_i and *nP* is the number of planners to be used by the intrusion detection process to be executed.

Algorithm 3.4 – Intrusion Detection

```
1: Intrusion Detection (𝔓_i, nP)
2: 𝔓_h ← select_planners (𝔓_i, nP)
3: for each planner Pl_j ∈ 𝔓_h do
4:     𝔄_j ← get_aggregators (Pl_j)
5:     Aggregate Global Models (𝔄_j, nA)
6:     Notify_Pl_i ← Intrusion Prediction (𝔄_i, nA)
7: end for
8: return Notify_Pl_j
```

For each selected planner Pl_j, the set of available aggregators $\mathfrak{A}_j$ is obtained, and the function **Aggregate Global Models** $(\mathfrak{A}_j, nA)$ is used to perform the aggregation of the models. Moreover, the function **Intrusion Prediction** $(\mathfrak{A}_i, nA)$ executes model inference at *nA* aggregators in $\mathfrak{A}_i$, which will generate a notification into $Notify_{Pl_i}$ for each detected anomaly.

The select_workers(), select_aggregators(), and select_planners() functions are responsible for making decisions about which P2P node is adequate to perform each of the demanded tasks in the intrusion detection workflow. To select the workers, the select_workers() function must consider their computational and memory resources to decide if it can perform model training with the planned deep learning model and dataset, according to the current plan in execution. The other select functions must also handle this process.

To make these decisions, each deep learning model and dataset has a characterization to indicate the necessary resources they will demand. Also, each P2P node must have a characterization that makes it possible to compare and check which nodes can be allocated. This information is persisted and is available to all DCD-FL nodes from decentralized storage.

The DCD-FL nodes that can be selected for each specific task are obtained from the list of known peers maintained by each node. Thus, when an Ad_i node needs to select planners, it searches for planner nodes in its list of known peers, which is periodically synchronized with the list of known peers of its neighborhood and DCD-FL bootstrap nodes.

Since for each intrusion detection process, all the number of elements (admin, planner, aggregator, and worker nodes, number of planning and aggregation rounds, and number of epochs and batches) are constants, the time complexity of these algorithms is also constant, $O(1)$.

Each DCD-FL node may be in one of the states: NEW, CONNECTED, DISCONNECTED, FREE, or BUSY. So, the select functions will collect only the nodes compatible with the demanded tasks, according to the plan in execution, and are in the state FREE, which will change to BUSY when selected and to FREE again when finished the task. Figure 3.4 shows a State Machine diagram, showing the state transitions for DCD-FL nodes.

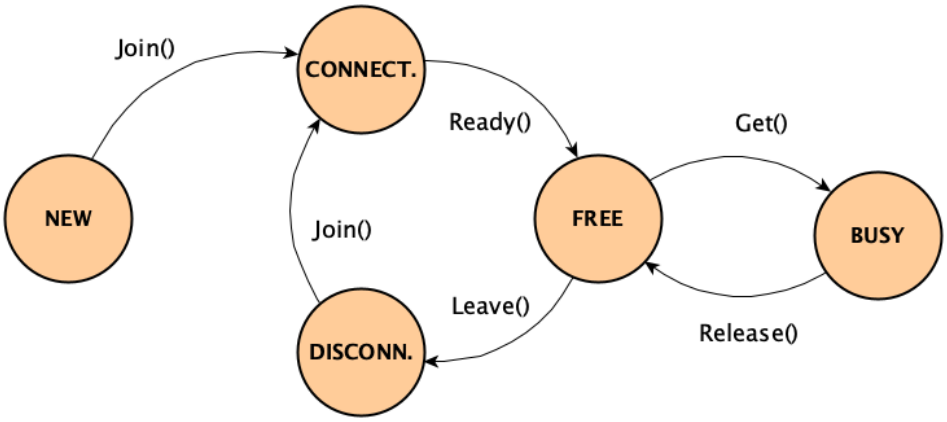

Figure 3.4 – State Machine Diagram for DCD-FL nodes

A DCD-FL node is in state NEW when it registers in the Admin cluster and changes to state CONNECTED when it calls the Join service of a controller cluster to get authenticated and authorized to participate in the DCD-FL network. When the node calls the Leave service of a controller cluster, its state is changed to DISCONNECTED.

When a node is ready to perform tasks, it changes its state to FREE. The state of the nodes is obtained by calling a specific service for this in the node's server or then in a node from an admin cluster, which is also responsible for maintaining this information.

The following Section describes the technologies adopted to satisfy these DCD-FL requirements, providing decentralized storage in an adequate way for IoT-based systems, which involve a P2P network-based approach as DCD-FL.

3.3 Prototype Implementation for DCD-FL

This section presents the implementation of DCD-FL, covering the tools and technologies used in the development, the challenges encountered, and how they were addressed. In the next chapter, the results of the evaluation of DCD-FL are presented, where it is discussed its performance in terms of its ability to detect intrusions, its scalability, and its impact on data privacy through various experiments, which were performed with the DCD-FL prototype here described.

3.3.1 DCD-FL Prototype Architecture

DCD-FL comprises a set of worker and aggregator nodes as P2P clusters and provides a specific API for intrusion detection based on the proposed decentralized, federated learning approach. According to available resources in the corresponding host, each edge server may work as a blockchain node to the controller cluster and a worker or aggregator node to the training/predicting cluster. An edge server can work as different types of nodes simultaneously if its available resources allow it and it is authenticated/authorized with enough permissions.

As usual in P2P networking, each node in DCD-FL provides similar functionalities, working as servers and clients. Thus, each P2P node is implemented by a Python application consisting of the same set of libraries and modules. Table 3.1 lists the implemented modules and their classes for the DCD-FL nodes.

Table 3.1 – DecideApp: set of modules and their classes in each DCD-FL node.

Storage	**Node**	**Services**
DatasetStorage, ModelStorage, PeerStorage, ControllerStorage, IpfsStorage	AdminNode, ControllerNode, PlannerNode, AggregatorNode, WorkerNode, StorageNode	AdminServices, ControllerServices, PlannerServices , AggregatorServices, WorkerServices , ModelServices. PeerServices

Each DCD-FL node implements the `decide_app` Python module, which contains the `DecideApp` class with the node's main function. Figure A.1 in Appendix A contains a class diagram in UML that shows these classes' functionalities and their associations.

When instantiated, the `DecideApp` class instantiates several other classes to implement computation, communication, and storage functionalities. They initiate a gRPC server, connect with a DCD-FL bootstrap node to receive an initial set of known peers, and wait for service calls from other DCD-FL nodes. Figure B.2 in Appendix B contains a UML sequence diagram showing the main workflow related to these classes.

All node classes have a constructor and a `start()` method to initialize a gRPC server, which provides remote access to the functionalities from the modules of services related to the node type. Thus, for instance, the gRPC server of an `AdminNode` will provide the functionalities of the `AdminServices` class.

Storage modules are responsible for persisting information related to datasets, models, and peers, using decentralized storage in IPFS [13] or the Ethereum blockchain [22]. For each type of DCD-FL node, there is a specific module that has a corresponding class. Storage nodes St_m start their respective gRPC server and keep waiting for calls from other nodes to perform their specific services, which include retrieving/storing models from/to IPFS (model parameters) and Ethereum (model metadata). Controller nodes Ct_h behave similarly, but their services consist of performing authentication and authorization for the other nodes.

Each Ad_i is implemented by an `AdminNode` class. After initiating its gRPC server, it waits for remote procedure calls to register a calling node to a session, which requires its node `Id` and the node `Id` of an existing recommender admin node that it will be contacted to confirm it. The calling node must also provide its name and technical specification, including available resources (CPU, RAM, and connectivity characteristics). Ad_i will return a node session token to be used in the authentication/authorization in the access control. `AdminNode` class also provides an undo of the node register from a session.

An Ad_i node also periodically activates intrusion detection processes, which include intrusion prediction and training model tasks. To do this, Ad_i select one or more planner nodes Pl_j from its list of known peers and activates them to perform the tasks, receiving the metadata of the trained models and the results of the predictions with eventual notifications of anomalies.

After starting their gRPC server, Pl_j nodes wait for service calls to be activated. In the case of a training model call, Pl_j generates new plans or selects an existing one, indicated by Ad_i node. To create new plans, a planner uses a configuration file that defines all the factors to consider. Figure 3.5 shows an example of this configuration file.

```
{
  "factors": {
    "nodes_qty": {"admin_node_qty": [1,2,5], "controller_node_qty": [2], "planner_node_qty": [1,2,5,10], "
        aggregator_node_qty": [1,3], "worker_node_qty": [1,5,10], "storage_node_qty": [2]
  },
  "models": {
    "ann" : {"learning_rate":[0.01, 0.001], "layers":[2], "hidden_size":[64, 128, 256], "batch_size": [64, 128,
        256]}, "lstm": {"learning_rate":[0.001],"layers":[2],"hidden_size":[64, 128, 256],"batch_size": [64, 128,
        256]},  "gru": {"learning_rate":[0.001], "layers":[2], "hidden_size":[64, 128, 256], "batch_size": [64, 128,
         256]}
  },
  "datasets": {
    "unswnb15": {"inputs":[57],"outputs":[10]}, "n-baiot": {"inputs":[115],"outputs":[11]}, "iot23": {"inputs":[24],"
        outputs":[4]}, "bot_iot": {"inputs":[10],"outputs":[5]}, "edge-iiotset": {"inputs":[92],"outputs":[15]}
  }
}
```

Figure 3.5 – DCD-FL: Example of configuration file to generate plans

A cartesian product of the specified factors, models, and datasets, in the configuration file, produces all the possible plans. Figure 3.6 presents an example of a generated plan.

```
{
  "plan_name": "ann-edge-iiotset-3-10",
  "num_plan_rounds": 1,
  "num_aggregation_rounds": 4,
  "num_aggregators": 3,
  "num_workers": 10,
  "model_type": "ann",
  "dataset_name": "edge-iiotset",
  "learning_rate": 0.001,
  "num_layers": 2,
  "hidden_size": 128,
  "batch_size": 128,
  "num_epochs": 4,
  "num_inputs": 92,
  "num_outputs": 15,
  "aggregation_algorithm": "FedAvg"
},
```

Figure 3.6 – DCD-FL: Example of generated plan

The execution of this plan, in Figure 3.6, will consist of one planning round, with four aggregation rounds. In each aggregation round, three aggregators will train global models, each using ten workers to train local models. Model training will be on a multilayer perceptron (ANN), using the indicated hyperparameters and the edge-iiotset dataset.

The planner Pl_j activates each selected aggregator Ag_k by calling its gRPC service `AggregateModels()`, which receives the plan as an argument. This service selects worker nodes Wk_l from the list of known peers of Ag_k in the indicated quantity in the plan. It activates them to perform the local model training by calling the worker's gRPC service `TrainModel()`, which receives the plan and an initial model, as indicated in the plan.

After starting its gRPC server, each Wk_l waits for service calls from other nodes. When its gRPC service `TrainModel()` is called, Wk_l collects locally the dataset, indicated in the plan, pre-processes it, which includes splitting it in 75% for training and 25% for testing, and calls its method for local model training and testing. This method obtains the model from IPFS by using the IPFS hash, which is passed to it by Ag_k. After model training and testing, Wk_l persists the model in the IPFS and returns the IPFS hash for it, and also registers model metadata in the blockchain using the Ethereum smart contract, shown in Figure 3.7.

After receiving the local models from its workers, each Ag_k performs its FedAvg method to generate a global model, which is also persisted using IPFS and Ethereum, as done by the workers.

When a planner Pl_j is activated by an Ad_i to perform a prediction, the flow is similar to the one described above. Still, instead of training models, the workers will collect samples of their datasets and execute their local models on them to predict if occurred or not an anomaly.

3.3.2 Technologies Used in the DCD-FL Prototype

The DCD-FL prototype was developed using Python, PyTorch, and several other Python libraries. Python is a high-level, interpreted programming language known for its readability, and PyTorch is an open-source machine learning library based on the Torch library. It is used for applications such as computer vision and natural language processing. Table 3.2 lists the main technologies used in the DCD-FL development, indicating the specific version.

Table 3.2 – Technologies used in DCD-FL development.

Technology	**Version**	**Description**
Python	3.9.13	An interpreted object-oriented programming language; Very popular in the implementation of machine learning-based systems.
Pytorch	1.13.1	Open-source machine learning library developed by Facebook; Oriented to high-performance deep learning models; Used to train all the models in DCD-FL.
Sci-klearn	1.2.1	Also an open-source machine learning library, built on top of two core Python libraries, NumPy and SciPy; More suitable for traditional machine learning models; Used utilities as, for example, `train_test_split()` to partition dataset.
Pandas	1.5.3	Open-source data manipulation and analysis library for Python; Used DataFrame structure (two-dimensional labeled data tables) and its utilities to perform pre-processing of datasets.
`matplotlib`	3.7.1	Data visualization library for Python; Provides various plotting customizable utilities; Used to plot experiment results.
`cryptography`	40.0.2	Open-source Python library, providing cryptographic functionalities for secure communications, encryption, and digital signatures; Used to generate hashes and also server and client certificates.
gRPC	1.51.1	Open source high-performance Remote Procedure Call (RPC) framework developed by Google; Uses Protobuf (Protocol Buffers) as its interface definition language and HTTP/2 as its transfer protocol; Provides type checking, bidirectional streaming, and flow control.
IPFS	0.10.0	Decentralized protocol and network; Uses content addressing and peer-to-peer technology to create a distributed file system.
Ethereum	-	Decentralized, open-source blockchain platform, oriented to the creation and execution of smart contracts and decentralized applications (DApps).
`web3.py`	6.2.0	Python library for interacting with the Ethereum blockchain and building decentralized applications.
`solcx`	0.8.19	Python library for Solidity compilation by providing a programmatic interface to interact with the Solidity compiler.
Pinata IPFS API	-	Service provided by the Pinata company, offering a set of HTTP endpoints and functionalities to store, retrieve, and manage files on the IPFS network.

3.3.3 TLS/SSL and gRPC in DCD-FL

In DCD-FL, it is adopted gRPC [44], combined with TLS/SSL certificates [102], to ensure secure and encrypted communication between all the nodes. TLS/SSL certificates are digital documents that bind cryptographic keys to specific entities, such as organizations, individuals, or servers. They are issued and signed by certificate authorities (CAs) and are essential in establishing trust and authenticity in online communications. Figures C.5 and C.6 in Appendix C shows Python code to generate certificates for server and clients.

Each DCD-FL node has installed its private key and certificate on its gRPC server and gets the public certificates of all other nodes via IPFS.

The gRPC of a node, working as a client, verifies the server's TLS/SSL certificate during the connection handshake. It checks whether the certificate is trusted, valid, and matches the expected server identity, conforming to the certificate of the other node. Once the client validates the server's certificate, a secure TLS/SSL connection is established. This connection encrypts the data exchanged between the client and server, ensuring confidentiality and integrity.

Optionally, gRPC can enable mutual TLS/SSL authentication. In this scenario, the client also presents its own TLS/SSL certificate to the server, verifying its identity. The server can then authenticate the client's certificate and make access control decisions based on it.

3.3.4 Ethereum in DCD-FL

Ethereum [22], a blockchain-based platform, provides a decentralized virtual machine that can execute scripts using a worldwide network of public nodes. This feature is particularly useful in DCD-FL as it allows for the creation of smart contracts, which are self-executing contracts with the terms of the agreement directly written into code.

In DCD-FL, Ethereum is used to register metadata related to the trained models and the properties of the nodes. Figure 3.7 shows the smart contract, coded in the Solidity language [22], implemented to register key/value pairs in the Ethereum blockchain.

By adopting the dictionary format as key/value pairs, the smart contract becomes very flexible and could be used to register all relevant information produced by the DCD-FL nodes. Figures C.2 and C.1 in Appendix C shows the Python code to deploy the above smart contract `KeyValueStore` in the Ethereum blockchain (Mumbai testnet), using the Alchemy API, and to set/get a value by executing the smart contract, respectively.

Using the Remix IDE, the cost to deploy the smart contract `KeyValueStore` in the Polygon network (Mumbai testnet) was circa 0.001178 MATIC, which is the Polygon cryp-

```
1 pragma solidity ^0.8.18
2 // SPDX-License-Identifier: MIT
3 contract KeyValueStore {
4   mapping(string => string) private _data;
5
6   function setData(string memory key, string memory value)
       public {
7     _data[key] = value;
8   }
9
10  function getData(string memory key) public view returns (
       string memory) {
11    return _data[key];
12  }
13 }
```

Figure 3.7 – DCD-FL: Smart contract for key value store

tocurrency. Since cryptocurrency prices are highly volatile, converting this value to USD can vary greatly. Assuming that the price of 1 MATIC is approximate $0.672600, then 0.001178 MATIC would be approximate $0.000792 in USD (0.001178 * 0.672600); And the cost to execute the smart contract was circa 0.000117 MATIC (approximately $0.00008 in USD).

3.3.5 IPFS in DCD-FL

IPFS (InterPlanetary File System) [13] is a protocol and network designed to create a content-addressable, peer-to-peer method of storing and sharing hypermedia in a distributed file system. Figure C.3 in Appendix C shows the Python code to persist a model in the IPFS system using the Pinata HTTP API. In the same Appendix C, Figure C.4 shows a Python code to retrieve a stored model using the IPFS API.

The Pinata HTTP API receives data in JSON format and returns a hash that can be used to retrieve the stored data. It also provides access control to the decentralized store through authentication to a Pinata gateway. It is a low-cost resource since a twenty dollars monthly fee allows the storage of 20,000 files in the system with a total size of 50Gbytes.

Since the DCD-FL trained models have circa 500Kbytes in size, it would be possible to store circa 105,000 new models with the basic fee. Thus, considering the limit of the number of files, the cost per model will be 0.001 dollars per model. However, deploying a local IPFS server in DCD-FL nodes is always possible to become independent of the Pinata gateway.

3.4 Summary

This chapter presents the development of a decentralized approach based on federated learning to intrusion detection in IoT applications, named DCD-FL. This system consists of a network of servers, known as aggregators, to generate global models. These aggregators use groups of IoT devices or workers and collaborate with other aggregators to create their global models. The system's intrusion detection mechanism is built on deep learning techniques, which analyze network traffic patterns to predict potential security breaches. When unusual behavior is detected, planner nodes generate alerts sent to admin nodes, which can then handle these alerts.

The DCD-FL system is characterized by its decentralization, with admin, controller, and IDS-specific clusters deployed across various edge components. This structure reduces network traffic between edge servers and fog/cloud servers and enhances privacy. The prototype of DCD-FL consists of worker and aggregator nodes that form peer-to-peer clusters and provides a specific API for intrusion detection.

The system's storage modules are tasked with persisting information related to datasets, models, and peers, utilizing decentralized storage in IPFS and the Ethereum blockchain. Each type of DCD-FL node has a corresponding class within a specific module. Admin nodes periodically activate intrusion detection processes, which include intrusion prediction and model training tasks. These nodes select planner nodes from their list of known peers and start them to perform the tasks. They subsequently receive the trained models' metadata and the predictions' results, including any anomaly notifications.

4. EXPERIMENTS AND RESULTS

This chapter presents a detailed report of the experimental evaluation conducted to assess the performance and effectiveness of the DeCentralizeD, Federated Learning-based Intrusion Detection System (DCD-FL IDS), developed as part of this research. This research aims to develop a robust, efficient, and effective intrusion detection system for IoT applications that can address the limitations of existing systems. The experiments' results are crucial in determining whether DCD-FL achieves this aim. They offer a quantitative evaluation of its performance, providing the necessary support for the conclusions drawn in this thesis.

4.1 Setup

The experiments were designed to evaluate DCD-FL under various conditions and configurations, using three different machine learning models and three distinct datasets, described in the following. The results obtained from these experiments provide insights into the DCD-FL's capabilities, highlighting its strengths and identifying areas for potential improvement.

All the experiments were executed on a single computer, a 2.3 GHz Intel Core i9 8-core, with 16 Gbytes 2.6 MHz DDR4, running MacOS Ventura 13.4. This setup was chosen to ensure a constrained environment, as with IoT-based systems, for the experiments, eliminating potential variables that could affect the results. Each P2P node of DCD-FL was assigned a unique local IP and a unique port on this computer, effectively simulating a decentralized network environment on a single machine.

The communication between DCD-FL nodes was performed using TLS/SSL encrypted service calls to gRPC servers implemented on each DCD-FL node. This approach ensured secure and reliable communication between the nodes, which is crucial in a decentralized system where data privacy and integrity are the main requirements. Using gRPC servers provided a high-performance, open-source framework for handling remote procedure calls, facilitating efficient and effective node communication.

The process of model training was carried out at the edge nodes. Each edge node was responsible for training a local model using its subset of one of the available datasets, according to the plan being executed at the moment, and making predictions based on this local model. This decentralized approach to model training and prediction leverages the computational resources of the edge nodes and allows for more efficient and scalable machine learning tasks. It also ensures that data privacy is maintained, as the raw data does not need to be transferred between nodes.

Once the worker nodes train the local models, the next step is generating the global model by the aggregator nodes, following the executed plan, using model aggregation of the local models from each edge node. This global model represents the collective learning of all the edge nodes and is used to predict new data. The generation of global models was carried out by the aggregator nodes, which were responsible for collecting the local models from the edge nodes and performing model aggregation.

DCD-FL leverages distributed ledger technologies for the persistence of trained model metadata by executing a smart contract on Ethereum in the Polygon network (Mumbai testnet) using a wallet created in Metamask.

A smart contract allows for the secure and transparent recording of model metadata, ensuring the process is auditable and tamper-proof. The Polygon network provides a scalable and efficient solution for executing the smart contract, with lower transaction costs and faster block mining times compared to the main Ethereum network, which will be necessarily used in a production environment.

The storage of the trained models was implemented off-chain using IPFS [13]. The Pinata HTTPS API was used to save and restore the models by the DCD-FL nodes in IPFS. This approach ensures that the trained models are securely and reliably stored while reducing the load on the blockchain.

4.1.1 Models

In the experiments, it was utilized different machine learning models: Multi-layer Artificial Neural Networks (ANN), Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU). These models were chosen due to their proven effectiveness in various deep learning applications, particularly in handling sequential data, which is common in IoT applications.

The given fully connected neural network (FC NN) or Multi-Layer Perceptron (MLP), also known as a dense network, is an artificial neural network where all the neurons in one layer are connected to all the neurons in the next layer. The specific FC NN used here is a 4-layer network, which includes an input layer, two hidden layers, and an output layer.

The input layer size is variable and depends on the dataset being used, with the number of neurons equal to the number of input features. The hidden layers contain 64 neurons, and the output layer size is also variable, with the number of neurons equal to the number of output classes or targets.

The network is implemented in Python version 3.9.13, using the PyTorch library version 1.13.1 and the torch.nn.Linear class was used to define the fully connected layers, and the ReLU (Rectified Linear Unit) activation function was applied to the outputs of the first

three layers. The output of the final layer is directly returned as the network output, which is suitable for a multi-class classification problem where a softmax function would be applied to the network's output to obtain class probabilities.

The number of parameters in this FC NN depends on the sizes of the input and output layers and is calculated as the sum of the number of weights and biases in each layer. The number of weights in a layer is the product of the number of input and output neurons, and the number of biases is equal to the number of output neurons. The total number of parameters is the sum of the parameters in all layers. Table 4.1 lists the number of parameters for each model in the experiments.

Table 4.1 – Number of parameters of each deep learning model.

Dataset Name	Inputs	Outputs	FC ANN	RNN	LSTM	GRU
UNSW-NB15	57	10	12,682	16,266	63,114	47,498
N-BaIoT	115	11	30,082	37,566	150,114	112,598
IoT23	24	4	10,432	13,066	52,214	39,198
Bot-IoT	10	5	8,322	10,566	42,214	31,698
Edge-IIoTset	92	15	23,682	29,666	118,614	88,998

The Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) are all types of recurrent neural networks that are particularly effective for sequence prediction problems because they can use their internal state (memory) to process sequences of inputs.

The implemented RNN model is a two-layer network with a variable hidden layer size. The torch.nn.RNN class was used to define the RNN layers. The output of the RNN layers is then passed through a fully connected layer to produce the final output. The number of parameters in an RNN layer is calculated as the sum of the parameters for the weights and biases for each gate, which includes the input and hidden states.

The LSTM model is similar to the RNN model but uses torch.nn.LSTM to define the LSTM layers. LSTMs have an advantage over simple RNNs in that they can avoid the long-term dependency problem: they can remember information for extended periods. This characteristic is advantageous when the network needs to learn from significant past events that have taken many time steps. An LSTM layer has more parameters than an RNN layer because it has four gates (input, forget, cell, and output), each with weights and biases.

The GRU model is similar to the RNN and LSTM models but uses torch.nn.GRU to define the GRU layers. GRUs are a variation of LSTMs that combine the forget and input gates into a single "update" gate and merge the cell state and hidden state. As a result, they have fewer parameters than LSTMs and may train faster or need less data to generalize.

The number of parameters in these models depends on the sizes of the input and output layers and is calculated as the sum of the number of weights and biases in each layer. The number of weights in a layer is the product of the number of input and output neurons,

and the number of biases is equal to the number of output neurons. The total number of parameters is the sum of the parameters in all layers.

All these machine learning models were implemented using the PyTorch Python library [91]. Table 4.2 lists the hyperparameters for the models.

Table 4.2 – Hyper-parameters for the models.

Hyperparameter	**Values**
Number of Layers	4 in FC ANN and 2 in others
Number of Neurons per Layer	64
Learning Rate	0.001
Batch Size	128
Number of Training Epochs	4
Optimizer	Adam
Loss Function	Cross-Entropy
Dropout Rate	0.0
Weight Initialization	He, biases initialized to zero

The selection of these models aligns with the orientation of DCD-FL towards deep learning techniques. Deep learning models, such as Multi-layer ANN, LSTM, and GRU, can learn complex patterns from large amounts of data, making them suitable for the diverse and high-dimensional data in IoT applications.

4.1.2 Datasets

The experiments used different datasets: UNSWNB15, Bot-IoT, and N-BaIoT. Each of these datasets corresponds to a different type of IoT application, allowing us to demonstrate the versatility of DCD-FL in various uses of IoT. This diversity demonstrates that DCD-FL can be effectively applied in various IoT contexts, from intrusion detection to malware and botnet detection.

All these datasets have more than one output, so they allow to train of multi-class machine learning models, with a specific output for each possible particular attack class, instead of just a binary output, which can only predict if it was detected or not an attack. All the experiments reported in this section were performed to train multi-class machine learning models.

UNSW-NB15 Dataset

The UNSW-NB15 dataset was created by the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) [73], using the IXIA PerfectStorm tool to generate normal

and anomalous network traffic. The Tcpdump tool was used to capture the network traffic, producing a pcap file of 100 Gigabytes, which Argus, BroIDS, and C# scripts used to extract features. The ground truth table to label the dataset was obtained from an IXIA report generated during the attacks' simulation. Figure 4.1 shows the testbed environment used to generate the dataset.

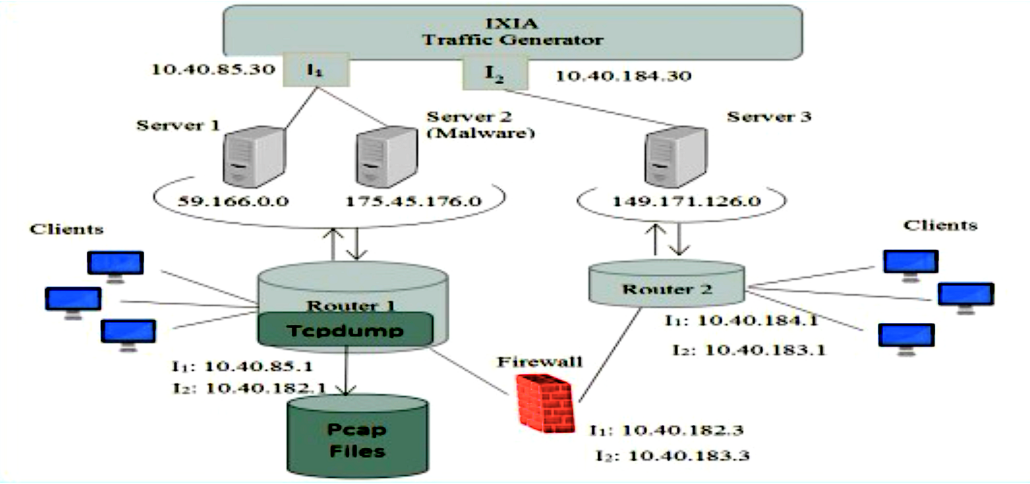

Figure 4.1 – Testbed environment in generating UNSW-NB15 dataset. [73]

As shown in Figure 4.1, there are three servers from where the network traffic generated by the IXIA PerfectStorm tool is propagated. Server 2 transmits anomalous traffic, and the tcpdump tool is running on router 1, capturing the network traffic and storing it in a pcap file, from which the tools mentioned above extracted 49 features. Some of them are listed in Table 4.3.

Table 4.3 – Main features of the UNSW-NB15 dataset. [73]

Feature	**Description**
srcip, dstip,sport, dport	The IP address and port number of the source and of destination
proto	Transaction protocol: TCP, UDP
state	The state and its dependent protocol
dur	Record total duration
sbytes, dbytes	Source to destination and destination to source bytes
sttl, dttl	Source to destination and destination time to live
sloss, dloss	Source and destination packets re-transmitted or dropped
service	http, ftp, ssh, dns ..,
sload, dload	Source and destination bits per second
spkts, dpkts	Source to destination and destination to source packet count

The UNSW-NB15 dataset [74] contains 2,540,044 records, and Table 4.4 lists the nine types of attack into which these records are classified. All these types of attacks were considered in the experiments in the present Thesis.

Table 4.4 – Categories in the UNSW-NB15 dataset. [73]

Class	Count	Description
Benign	1,550,712	Normal traffic
Fuzzers	19,463	Feeding a program/network with randomly generated data to force its suspension
Analysis	1,995	Performs of port scan and use of spam
Backdoor	1,782	Bypass system security to obtain unauthorized access
DoS	5,051	Try to make a server or network resource unavailable for use by causing excessive overhead to a provided service
Exploits	24,736	Use of a known security problem to exploit system vulnerabilities
Generic	5,570	A method that targets cryptography and causes a collision with each block-cipher
Reconnaissance	12,291	A technique for gathering information about a network host and is also known as a probe
Shellcode	1,365	Code as payload in exploiting system vulnerabilities
Worms	153	Attacker self-replicated to other systems

The wide range of attack types in the dataset allows us to test the DCD-FL's ability to detect different kinds of intrusions in a class of applications similar to an IoT-based system with a combined fog and edge setup (servers and clients as devices, respectively, in the Figure 4.1).

Bot-IoT Dataset

BoT-IoT [48] is a realistic dataset consisting of normal and botnet traffic, with 42 features, and containing 477 (0.01%) benign flows and 3,668,045 (99.99%) attack flows, that is 3,668,522 flows in total. Figure 4.2 shows the testbed environment used to generate the dataset.

As shown in Figure 4.2, the testbed environment includes normal and attacking virtual machines (VMs), all with the same IP, and each one is identified by a unique port. In the Ubuntu VMs, it simulated IoT services, using the Node-red tool and the Argus tool to extract features. A PFSense firewall is used to maintain controlled traffic for the experiments. The Kali VMs generates anomalous traffic, performing port scanning, DDoS, and other Botnet-related attacks by targeting the Ubuntu Server, Ubuntu mobile, Windows 7, and Metasploitable VMs. And the Ostinato tool was used to generate realistic benign traffic.

The simulated IoT devices included a weather station, which generates information on air pressure, humidity, and temperature; a smart fridge, which measures the fridge's temperature and, when necessary, adjusts it below a threshold; motion-activated lights, which turn on or off based on a pseudo-random generated signal; a remotely activated garage door, which opens or closes, based on a probabilistic input; and, a smart thermostat, which regulates the house's temperature by starting the Air-conditioning system [48].

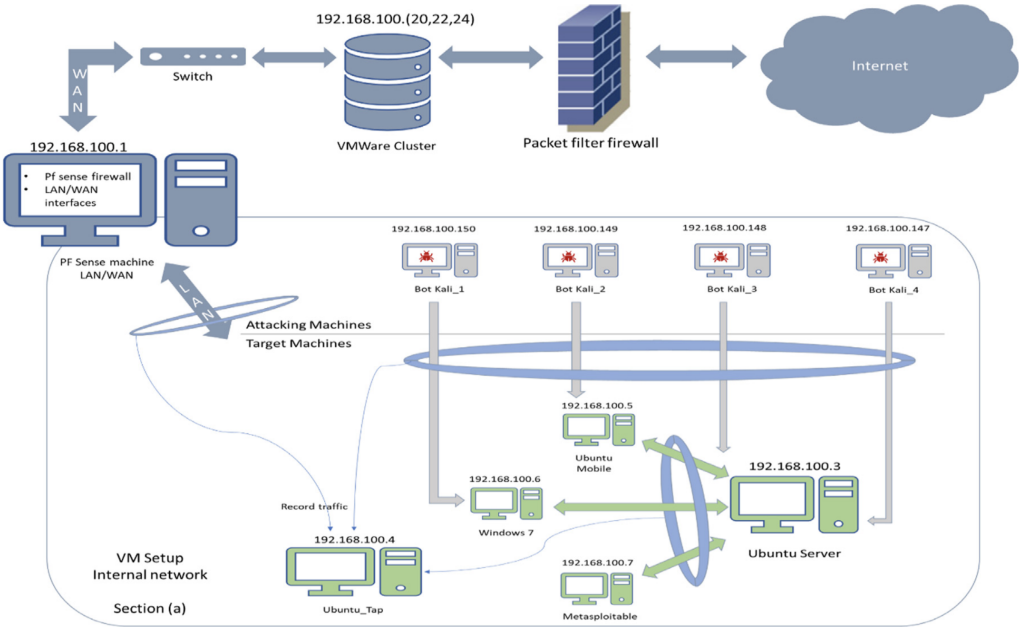

Figure 4.2 – Testbed environment in generating Bot-IoT dataset. [48]

Table 4.5 lists all the extracted features of the Bot-IoT dataset, from which Koroniatis et al. [48] identified the 10 more relevant: 'seq', 'stddev', 'N_IN_Conn_P_SrcIP', 'min', 'state_number', 'mean', 'N_IN_Conn_P_DstIP', 'drate', 'srate', 'max'.

Table 4.5 – Features of the Bot-IoT dataset. [48]

Feature	**Description**
pkseqid	Row Identifier
stime	Record start time
flgs, flgs_number	Flow state flags seen in transactions and its numerical representation
proto, proto_number	Textual and numerical representation of transaction protocols
saddr , sport	Source IP address and port number
daddr, dport	Destination IP address and port number
pkts, bytes	Total count of packets and the total number of bytes in transaction
state, state_number	Transaction state and numerical representation of feature state
ltime, dur	Record last time and record total duration
seq	Argus sequence number
mean, stddev, sum	Average duration, standard deviation, and the sum of aggregated records
min, max	Minimum, and maximum duration of aggregated records
spkts, dpkts	Source-to-destination and destination-to-source packet count
sbytes, dbytes	Source-to-destination and destination-to-source byte count
rate	Total packets per second in transaction
srate, drate	Source-to-destination and destination-to-source packets per second
attack	Class label: 0 for Normal traffic, 1 for Attack Traffic
category, subcategory	Traffic category and subcategory

The additional features, not included in Table 4.5, 'N_IN_Conn_P_SrcIP' (number of inbound connections per source IP) and 'N_IN_Conn_P_DstIP' (number of inbound con-

nections per destination IP) were generated by using sliding windows of length 100 connections. Ten more relevant features were used in all of the experiments performed with this dataset in the present Thesis.

Table 4.6 lists the categories and subcategories for the BoT-IoT dataset's attack types. There are four attack categories plus a normal one, and five categories/subcategories (indicated as Classes 0, 1, 2, 3, and 4 in Table 4.6) were used in all the experiments with this dataset in the present Thesis, i.e., the others were not considered. The used samples were obtained after a pre-processing phase that removed redundancies in the available samples.

Table 4.6 – Categories of attacks in Bot-IoT dataset. [48]

Category/Subcategory	**Tools**	**Samples**	**Samples in experiments**
Denial of Service			
DDoS **(Class 0)**			1,541,315
TCP	hping3	19,547,603	
UDP	hping3	18,965,106	
HTTP	golden-eye	19,771	
DoS **(Class 1)**			1,320,148
TCP	hping3	12,315,997	
UDP	hping3	20,659,491	
HTTP	golden-eye	29,706	
Information gathering (Class 2)			72,919
Service scanning	nmap, hping3	1,463,364	
OS Fingerprinting	nmap, xprobe2	358,275	
Information theft			
Keylogging	Metasploit	1,469	
Normal (Class 3)	Benign	370	370
Data theft (Class 4)	Metasploit	118	65
Total		73,361,270	2,934,817

N-BaIoT Dataset

The N-BaIoT dataset is a collection of network traffic traces from IoT devices infected with the Mirai and BASHLITE botnets [66]. The dataset was created to develop new methods for detecting attacks launched from compromised IoT devices and differentiating between hour and millisecond-long IoT-based attacks. The dataset was generated using the testbed environment shown in Figure 4.3.

The testbed environment contains a total of nine commercial IoT devices (listed in Table 4.7), connected via Wifi and also to a switch, as illustrated in Figure 4.3. To capture the network traffic was used port mirroring and the Wireshark tool.

Five types of BASHLITE attacks and five kinds of Mirai attacks were applied in the network, and network traffic was captured separately for each IoT device. It generated nine separate datasets for each device, including some benign traffic instances (as indicated in

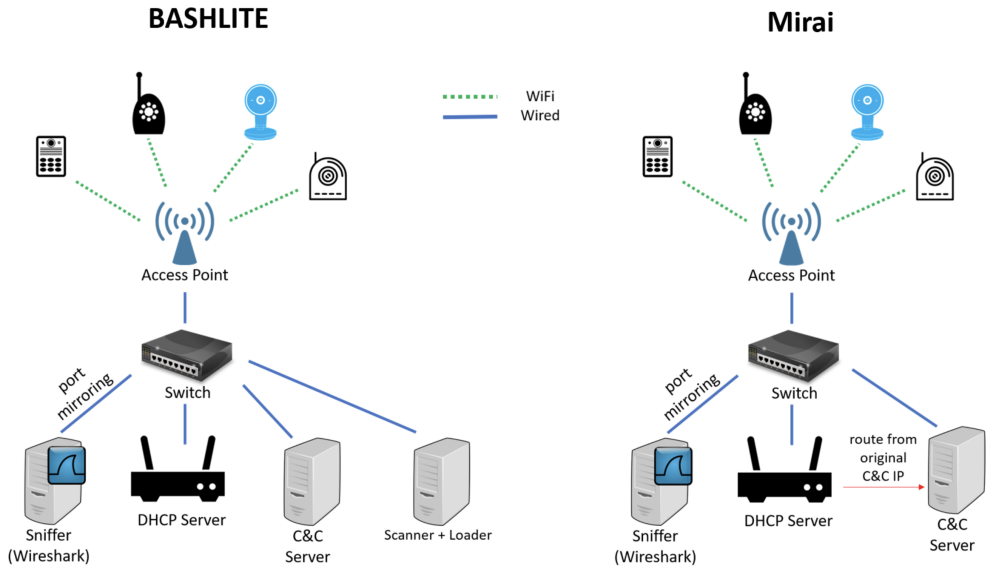

Figure 4.3 – Testbed environment in generating NBaIoT dataset. [66]

Table 4.7 – IoT devices used to generate NBaIoT dataset. [66]

Device ID	Device Maker, Model	Type	Benign Instances
1	Danmini	Doorbell	49,548
2	Ennio	Doorbell	39,100
3	Ecobee	Thermostat	13,113
4	Philips B120N/10	Baby Monitor	175,240
5	Provision PT-737E	Security Camera	62,154
6	Provision PT-838	Security Camera	98,514
7	SimpleHome XCS7-1002-WHT	Security Camera	46,585
8	SimpleHome XCS7-1003-WHT	Security Camera	19,528
9	Samsung SNH 1011 N	Webcam	52,150

Table 4.7). These ten types of attacks constitute the categories or classes which identify the possible anomalies. Table 4.8 lists these classes of attacks and the benign class with the number of samples of each one in the dataset, which were used in the experiments in the present Thesis.

Different from the previous datasets, the N-BaIoT dataset is characterized by 23 features (shown in Table 4.9), which represent statistical data computed on the captured network packets, per IoT device, and not by properties of the network traffic. These 23 features were computed for five different time windows (100 ms, 500 ms, 1.5 s, 10 s, and 1 min), thus resulting in a total of 115 features. All these features are used in the experiments of the present Thesis. However, it is essential to note that devices three (thermostat) and seven (security camera) do not have data on Mirai attacks, so they were not included in the performed experiments.

Table 4.8 – Categories in the N-BaIoT dataset. [66]

Class Id	Class	Count	Description
0	Benign	15,538	Normal traffic
1	BASHLITE COMBO	15,345	Sending spam data and opening a connection to a specified IP
2	BASHLITE Junk	15,449	Sending spam data
3	BASHLITE Scan	14,648	Scanning the network for vulnerable devices
4	BASHLITE TCP	15,676	TCP flooding
5	BASHLITE UDP	15,602	UDP flooding
6	Mirai Ack	15,138	Ack flooding
7	Mirai Scan	14,157	Automatic scanning for vulnerable devices
8	Mirai Syn	16,436	Syn flooding
9	Mirai UDP	15,625	UDP flooding
10	Mirai UDP Plain	15,304	UDP flooding with fewer options

Table 4.9 – Extracted features for NBaIoT dataset. [66]

Value	Statistic	Aggregated by	Nr. Features
Packet size (of outbound packets only)	Mean, Variance	Source IP, Source MAC-IP, Channel, Socket	8
Packet count	Number	Source IP, Source MAC-IP, Channel, Socket	4
Packet jitter (the amount of time between packet arrivals)	Mean, Variance, Number	Channel	3
Packet size (of both inbound and outbound together)	Magnitude, Radius, Covariance, Correlation coefficient	Channel, Socket	8

The N-BaIoT dataset was selected for the evaluation of DCD-FL due to its focus on botnet attacks, a major threat in IoT environments. The dataset allows us to test the system's ability to detect these types of attacks, demonstrating its effectiveness in protecting IoT devices from botnet infections, as shown in the next section on the experiment results.

4.1.3 Metrics

In evaluating the DCD-FL system, several metrics and measures were collected to assess the trained models' quality, time efficiency, and size. These metrics and measurements are crucial in understanding the performance and effectiveness of the DCD-FL system in a decentralized intrusion detection context.

Quality metrics are essential in evaluating the performance of the trained models. These include accuracy, precision, recall, and F1-score. Accuracy is the proportion of true results (both true positives and true negatives) in the population. It provides a general measure of the model's performance. Precision is the proportion of true positive results out of all

positive results. It measures the model's ability to identify only relevant instances correctly. Recall (also known as sensitivity) is the proportion of true positive results out of the actual positive results. It measures the model's ability to find all the relevant cases within a dataset. F1-score is the harmonic mean of precision and recall. It balances the two metrics and is particularly useful when the class distribution is imbalanced.

These metrics were collected for each local model generated by each worker node execution and for each aggregated global model generated by each aggregator node from the aggregation of the local models. Collecting these metrics allows for a comprehensive evaluation of DCD-FL's performance in detecting intrusions.

Time metrics were collected to assess the efficiency of the model training tasks. These include training execution time per epoch, per local model generation, and model aggregation execution. These metrics were further separated by dataset loading time, model training execution time, and model testing time.

These time metrics provide insights into the efficiency and scalability of DCD-FL. They allow for an understanding of how the system performs under different configurations and workloads, which is crucial in a decentralized setting where resources may be limited or variable.

Size metrics were collected to understand the resource requirements of the trained models. These include model size (in terms of the number of parameters), model memory use, and model storage space.

The model size is an essential factor in understanding the complexity and computational requirements of the model. Model memory use and storage space are crucial in a decentralized setting with limited resources. These metrics provide insights into the feasibility and scalability of deploying the DCD-FL system in a real-world IoT environment with constrained resources.

4.2 Experiments

The experiments explored several factors, including the number of aggregation rounds and varying topologies of the P2P networking, according to the number of available aggregator and worker nodes. These factors were chosen based on their potential impact on the quality of the trained models, both locally at each worker and globally by aggregation in the aggregator nodes.

The aggregation rounds were varied to understand their effect on the learning process and the final model performance. The topologies of P2P networking were varied to examine the impact of different network structures on the learning process. For instance, it was considered different configurations, such as one aggregator with one worker, one ag-

gregator with three workers, one aggregator with ten workers, three aggregators with one worker, and so on.

4.2.1 Model Quality versus Participant Nodes

The first experiment was designed to investigate the correlation between the number of participant nodes (aggregators and workers) and the quality of the local and global models. The quality of the models was assessed in terms of accuracy, precision, recall, and F1 score. The experiment involved varying the number of aggregators and workers and measuring the resulting model quality. The relevance of this experiment lies in its potential to reveal how the system's performance scales with the number of nodes.

In this experiment, model training was executed using the Multi-layer Artificial Neural Network (ANN) and dataset Bot-IoT with one aggregator node and 1, 5, and 10 worker nodes. Figures 4.4, 4.5, and 4.6 show model quality metrics (accuracy, precision, recall, and f1 score) for these three cases, plotting the metrics per aggregator and worker nodes.

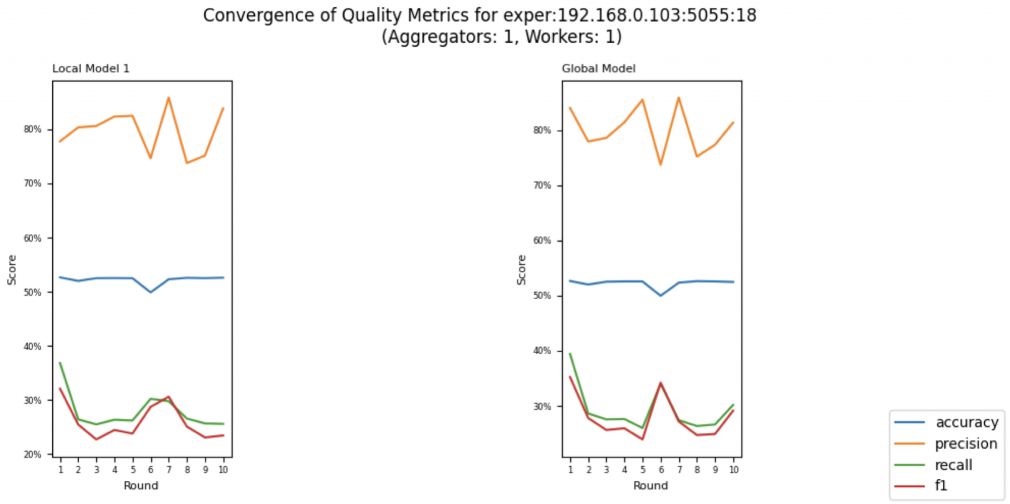

Figure 4.4 – Model quality x Nodes: ANN, Bot-IoT, 10 rounds, 1 aggr., 1 worker.

As noted in Figure 4.4, with just one worker, the global model will reflect the obtained local model generated by it without taking advantage of information from other available workers. Even after ten rounds, the quality metrics are unsatisfactory.

In Figure 4.5, it can be noted that even the workers presenting a diverse quality, the global model obtained by aggregation converge in less than six aggregation rounds, with precision, recall, and F1 score of around 80%, which is satisfactory, since there was no fine-tuning of the ANN hyperparameters.

Also, the global model convergence occurs independently of the number of workers, as seen with the results of Figure 4.6, where the global model converges by aggregation

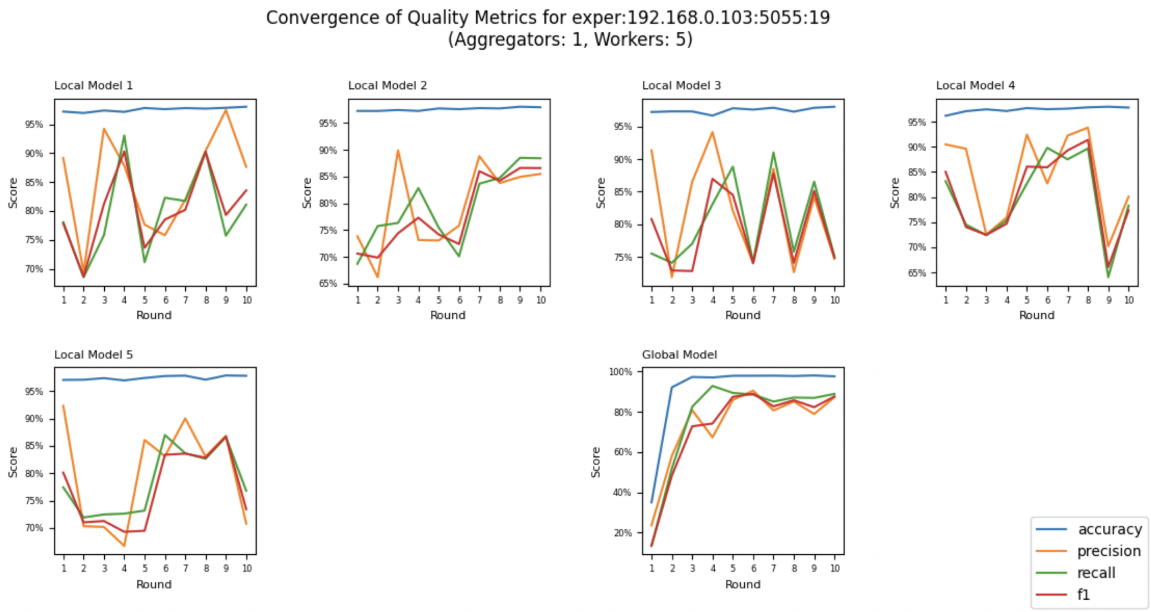

Figure 4.5 – Model quality x Nodes: ANN, Bot-IoT, 10 rounds, 1 aggr., 5 workers.

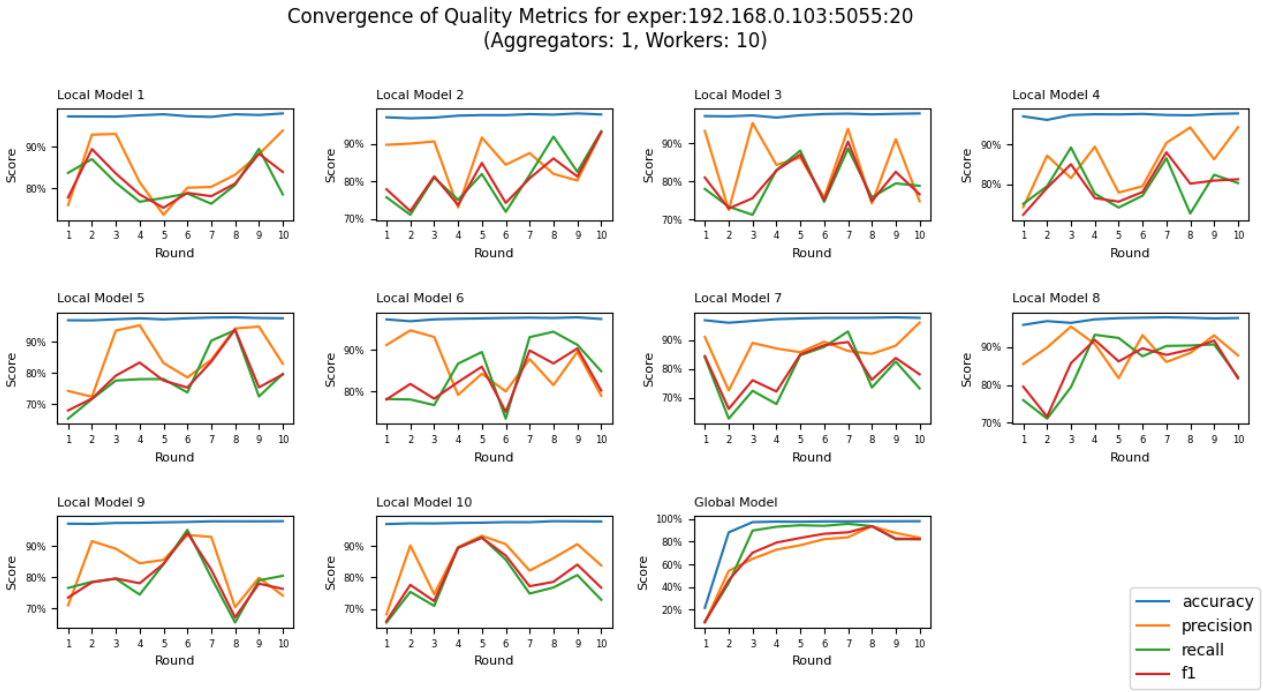

Figure 4.6 – Model quality x Nodes: ANN, Bot-IoT, 10 rounds, 1 aggr., 10 workers.

round six, with metrics also around 80%. This same pattern in the results could be perceived using the other datasets. It can also be noted that the higher the number of workers, the smoother the convergence of the global model, but it does not guarantee a better quality, which depends strongly on the dataset and hyperparameters, as stated in the literature.

Since the objective here is not fine-tuning the model hyperparameters, there is no change in the model hyperparameters during the rounds, so the model size doesn't change

within the same dataset. The Bot-IoT dataset has 8,022 parameters, so it has a memory use of 32K bytes. And this is the same for the local and global models.

During each round, initially, each worker may receive an updated model from the aggregator node and, at the end of the aggregation round, will send its local model to the aggregator node. Thus, the communication cost C will consist of the sum of storage use of the models exchanged during each aggregating round:

$$C = \sum_{r=1}^{R} (K * (L_{rk} + G_r)) \tag{4.1}$$

where R is the number of aggregation rounds, K is the number of workers, L_{rk} is the size of local model for worker k in round r, and G_r is the size of the global model in round r.

Thus, for one aggregator and five workers, ten messages are exchanged (five from aggregator to workers and five from workers to aggregator), so the total communication cost is $C = 10 * (5 * 2 * 32) = 10 * 320 = 3,200$ Kbytes $\approx$ 3.2Mbytes. Similarly, for one aggregator and ten workers, the total communication cost is $C = 10 * (10 * 2 * 32) = 10 * 640 = 6,400$ Kbytes $\approx$ 6.4Mbytes.

So, the communication cost is linearly proportional to the number of participant nodes in the aggregation rounds. Also, considering the number of aggregation rounds, for the case with one aggregator and five workers, for example, from the first round to the fifth round, there is an increase of 320Kbytes to 1,600Kbytes, i.e., an increase of 5 times in communication cost, and an increase of 6.3 times in the recall (detection rate), from 14% to 89%, which compensates the communication cost, given yet the advantages of the decentralized federated learning.

Model training experiments were performed to evaluate the effect of the number of aggregators, using the same ANN model with the same Bot-IoT dataset, taking one, three, and six aggregators and always ten workers. Figures 4.7, 4.8, and 4.9 show metrics for each of these cases.

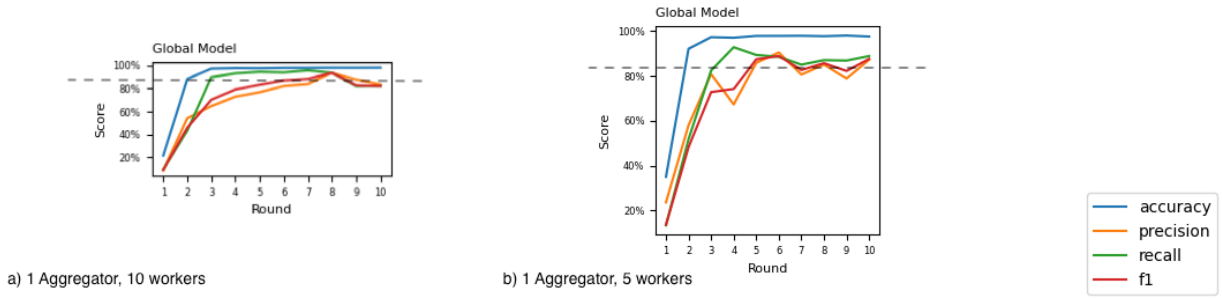

Figure 4.7 – Model quality x Nodes: ANN, Bot-IoT, 10 rounds, 1 aggr., 5 and 10 workers.

In Figure 4.7a) and Figure 4.7b), there are one aggregator and ten and five workers, respectively. In the case of using more workers and just one aggregator, the more relevant

metrics are below 80% at round ten, clearly showing that the aggregation will need more rounds to converge to the metric values obtained by using only five workers.

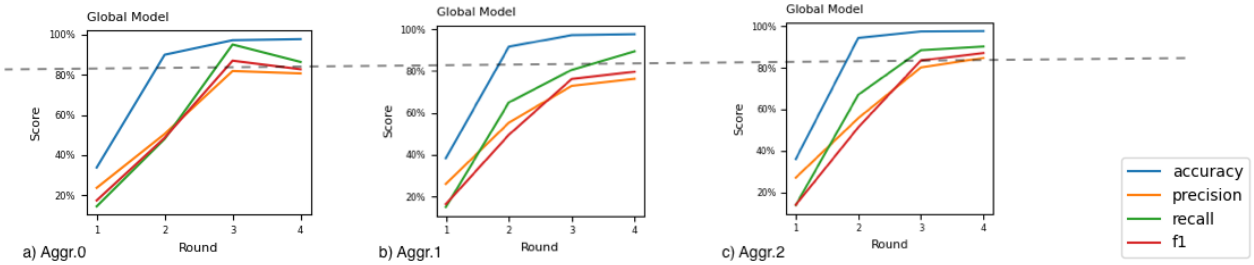

Figure 4.8 – Model quality x Nodes: ANN, Bot-IoT, 4 rounds, 3 aggrs., 5 workers.

Figures 4.8a), Figure 4.8b), and Figure 4.8c) show results for the three aggregators (0, 1, and 2), each one using five workers. It can be seen that, at round 4, aggregator 2 has all metrics over 80%. This result can be explained by the fact that an aggregator can take advantage of the use of model training results from the other aggregators. Thus, by increasing the number of aggregators without increasing the number of workers, some aggregators can eventually converge faster than others.

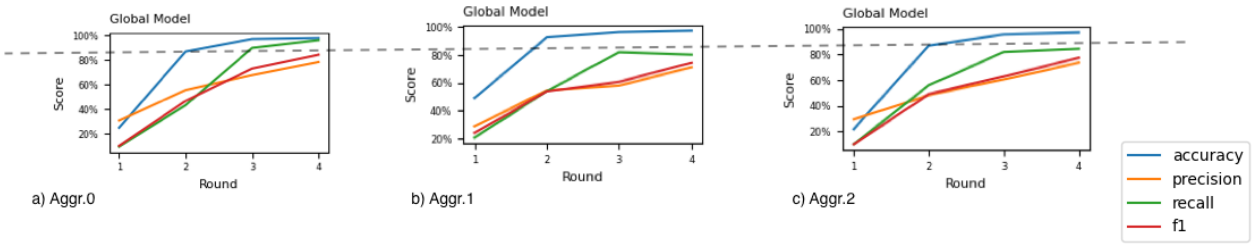

Figure 4.9 – Model quality x Nodes: ANN, Bot-IoT, 4 rounds, 3 aggrs., 10 workers.

Figures 4.9a), Figure 4.9b), and Figure 4.9c) show results for the three aggregators (0, 1, and 2), each one using ten workers. It can be seen that, at round 4, all aggregators have the more relevant metrics under 80%. Thus, increasing the number of workers demands more rounds to the convergence of the model, which has shown metrics over 80% in the previous above-reported experiments.

4.2.2 Convergence of Global Model Quality

The second experiment was designed to demonstrate if the convergence of the quality of the global model is correlated with the number of aggregation rounds. At each round, a fixed number of aggregators and workers were used to train local and global models. This experiment is relevant as it can provide insights into the system's learning efficiency and the impact of the aggregation process on the global model's quality.

This experiment also executed model training for ten aggregation rounds, using the Multi-layer Artificial Neural Network (ANN) and dataset UNSW-NB15 with one aggregator node and 1, 5, and 10 worker nodes. Figure 4.10 shows model quality metrics (accuracy, precision, recall, and f1 score) for these three cases (one worker in 4.10a, five workers in 4.10b, and ten workers in 4.10c), plotting the global model metrics per aggregator.

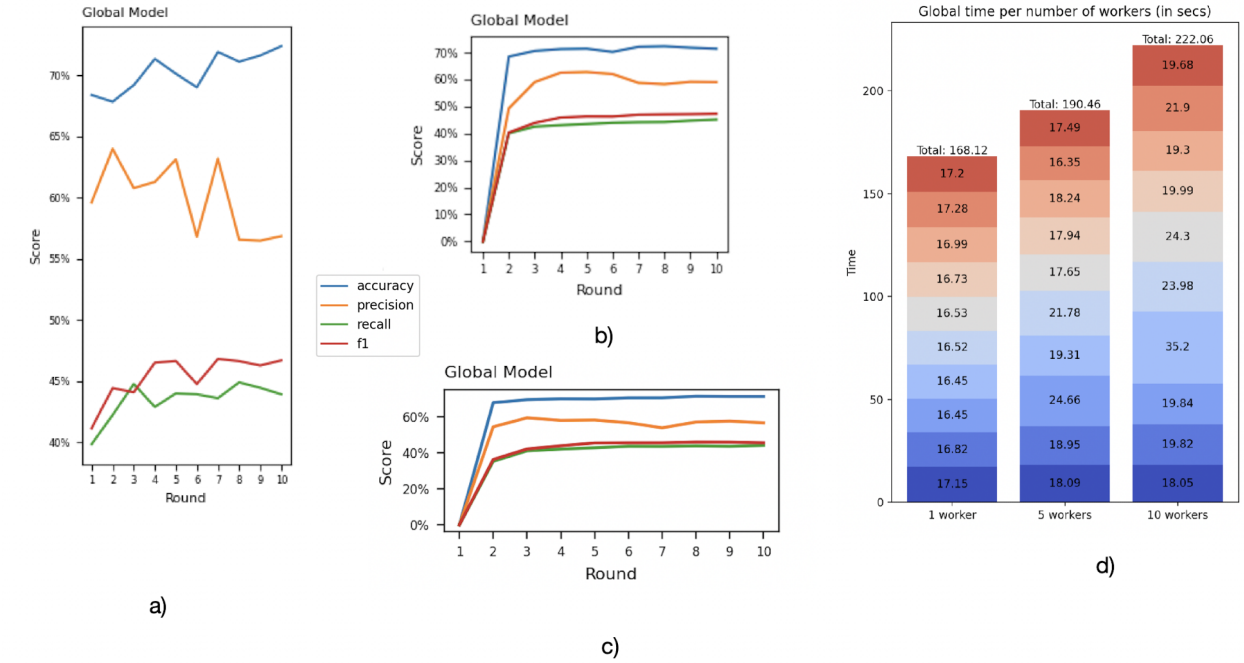

Figure 4.10 – Convergence: ANN, UNSW-NB15, 1 aggreg., and 1, 5, 10 workers.

Figure 4.10d) shows the time model training takes in the three cases. It can be noted that the time overhead is not high when using a higher number of aggregation rounds (practically the same time for the different numbers of workers).

Using the UNSW-NB15 dataset and basic configuration for the ANN model (without tuning its hyper-parameters), the trained global model has not had a satisfactory quality (metrics should be around 90%, as suggested in the literature). Still, it can be noted that aggregation round 3 already occurred with a convergence of the aggregated global model metrics.

In the previous Figures 4.4, 4.5, and 4.6, for the experiment using the Bot-IoT dataset, it could be noted that this convergence occurred around aggregation round 5. So, the convergence of the global model quality depends on the quality of the dataset and, of course, on using better hyperparameters for the model, which can eventually produce better metrics and in less time too. It is worth noting that this same pattern was also obtained for the N-BaIoT dataset.

4.2.3 Model Quality per Class

The third experiment was designed to analyze the performance of the trained models for each class of attack in a dataset. In this experiment, model training was executed for ten aggregation rounds, using the Long Short Term Memory (LSTM) algorithm and dataset N-BaIoT, with one aggregator node and five worker nodes.

Figures 4.11 and 4.12 show the ROC-AUC and Precision-Recall curves at the aggregation rounds 2 and 4, respectively.

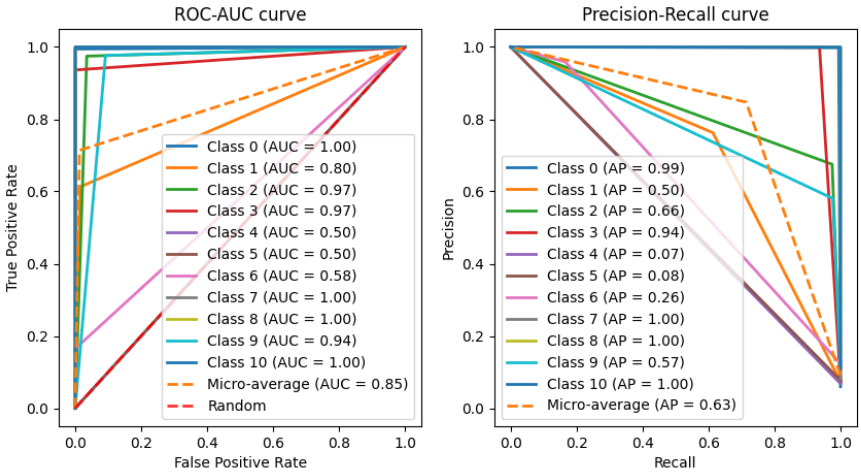

Figure 4.11 – Model quality per class: LSTM, N-BaIoT, aggreg. round 2.

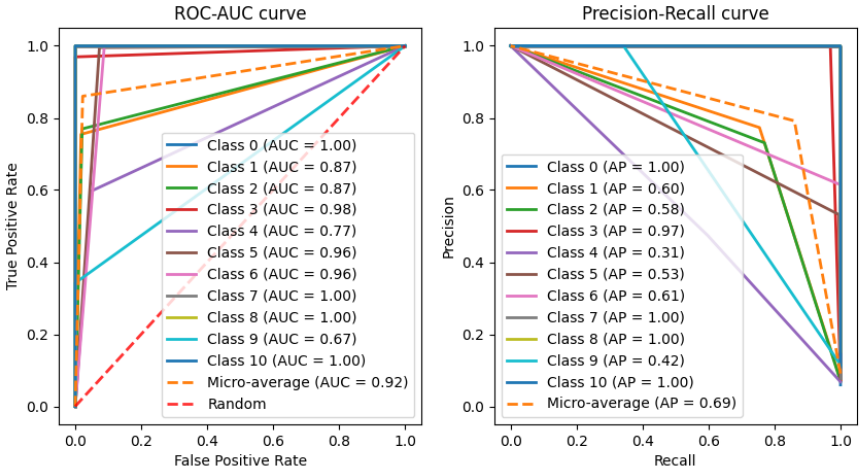

Figure 4.12 – Model quality per class: LSTM, N-BaIoT, aggreg. round 4.

In Figures 4.11 and 4.12, each class of attack is indicated by the class ID, listed previously in Table 4.8, thus "Class 0" represents the benign class, with normal traffic, and classes 4, 5, and 6, correspond to BASHLITE TCP and UDP, and Mirai Ack, which has unsatisfactory metrics at all aggregation rounds. Even so, with the basic configuration for the

LSTM model (no tunning of hyperparameters), the obtained metrics, already at the aggregation round 4, for almost all classes, are above 90%.

4.2.4 Effect of Deep Learning Model

The fourth experiment was designed to analyze the effects of the deep learning model on the quality of the trained models. This experiment used the same dataset BoT_IoT to model training with a Multi-layer Artificial Neural Network, a Long Short Term Memory (LSTM), and a Gated Recurrent Network (GRU) machine learning algorithm. Figures 4.13, 4.14, and 4.15 show the ROC-AUC and Precision-Recall curves for each of these cases.

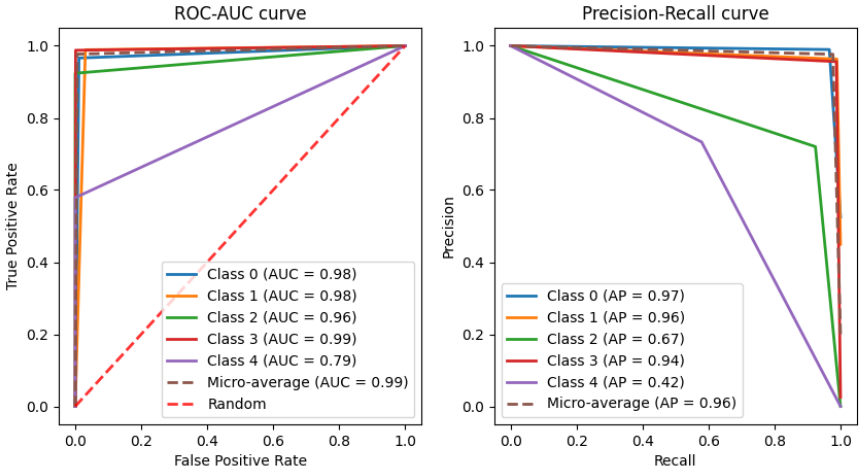

Figure 4.13 – Effect of the DL model: ANN, Bot-IoT, 1 aggreg., 5 workers.

All three DL models present satisfactory metrics (above 90%), except for "Class 4" which corresponds to the "Data theft" attack, which has only 65 samples (as indicated in Table 4.6), in this very imbalanced dataset.

Figure 4.16 shows the model training and testing time the three models took using the Bot-IoT dataset, with one aggregator and five worker nodes during four aggregation rounds. It can be noted that the time metrics for the FC ANN model are circa ten times shorter than the recurrent NN-based models (LSTM and GRU) and presented results similar to them in terms of trained model quality.

Moreover, looking at the size metrics for the three models (shown in Figure 4.17), it can be noted that the ANN model is circa five times smaller than the other DL models for the three datasets used in the experiments. Thus, based on these results, ANN represents the better alternative to IoT-based systems, where limited time and resource constraints must be satisfied.

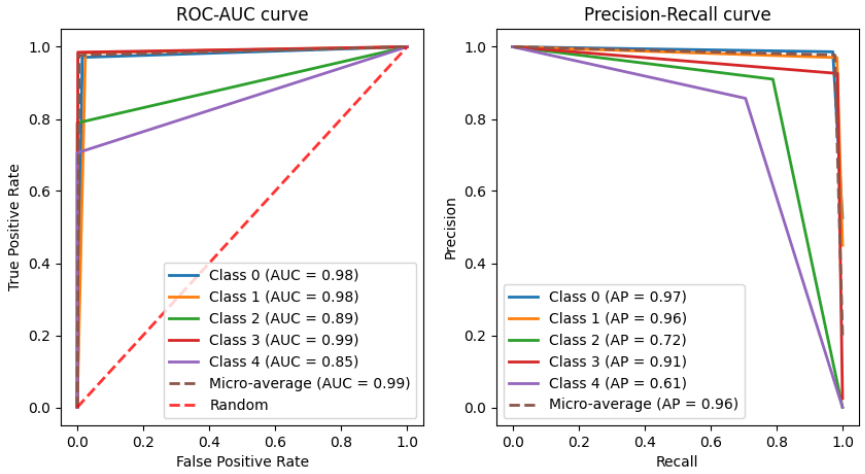

Figure 4.14 – Effect of the DL model: LSTM, Bot-IoT, 1 aggreg., 5 workers.

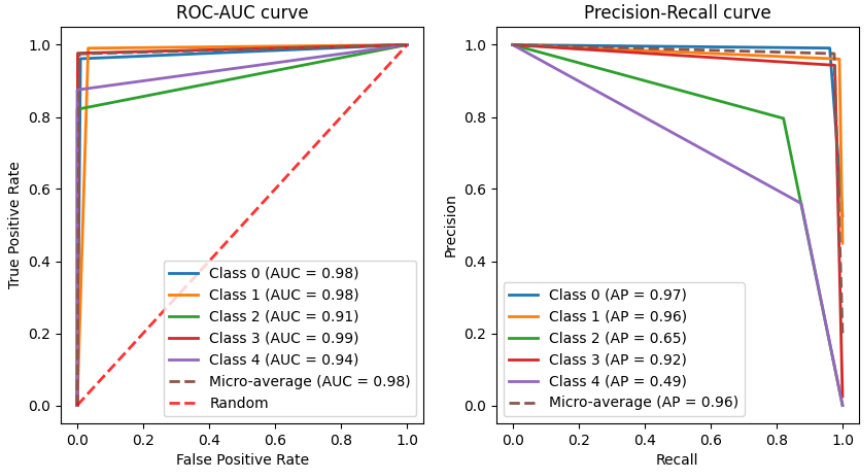

Figure 4.15 – Effect of the DL model: GRU, Bot-IoT, 1 aggreg., 5 workers.

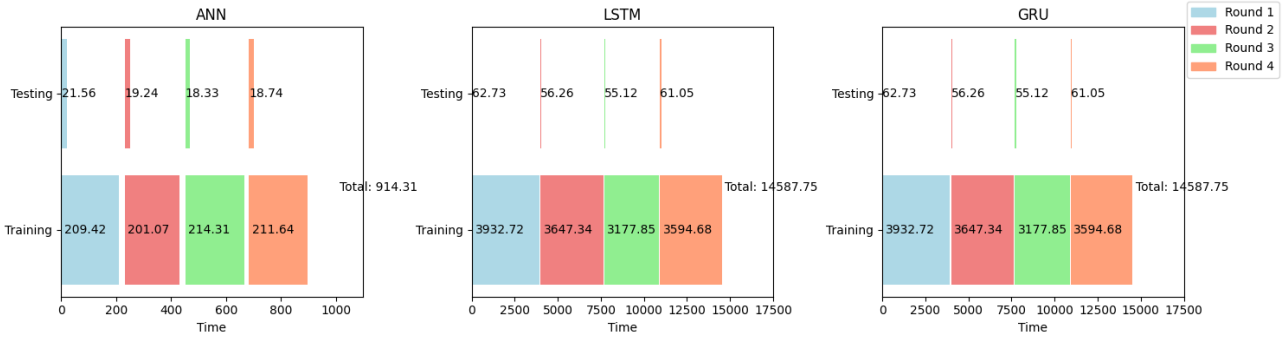

Figure 4.16 – Time metrics for models: ANN, LSTM, GRU.

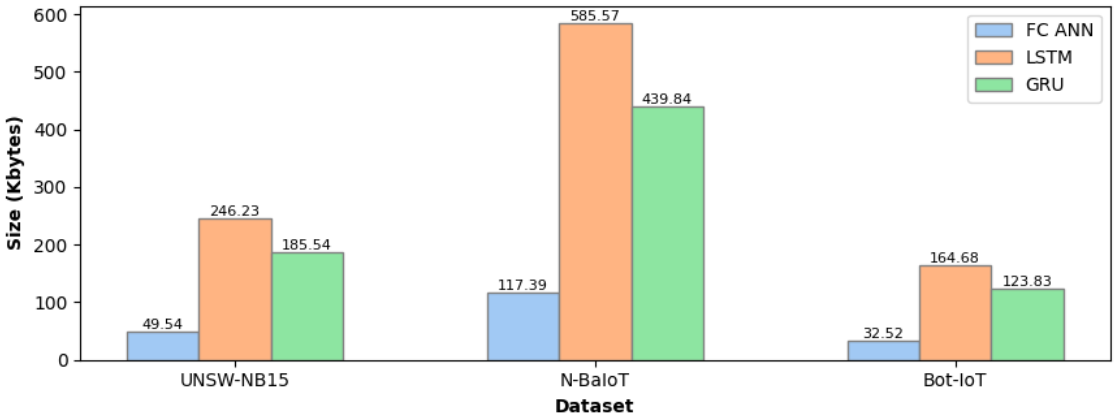

Figure 4.17 – Size metrics for models: ANN, LSTM, GRU.

4.2.5 Comparative Analysis

This experiment was designed to demonstrate that DCD-FL produces results comparable to those reported in the literature of related works but aggregates all advantages of a decentralized approach to intrusion detection.

Table 4.10 compares DCD-FL results with the results from Kim et al. [46] for N-BaIoT and deep learning models RNN and LSTM. The comparison could show only four of the IoT devices, the only ones reported in [46].

Table 4.10 – Average F1-score of RNN and LSTM for N-BaIoT dataset.

	Doorbell		**Baby monitor**		**Security Camera**		**WebCam**	
	[46]	DCD-FL	[46]	DCD-FL	[46]	DCD-FL	[46]	DCD-FL
RNN	0.41	**0.84**	0.44	**0.87**	0.37	**0.88**	0.55	**0.88**
LSTM	0.62	**0.84**	0.25	**0.86**	0.25	**0.83**	0.43	**0.89**

DCD-FL results were obtained with nine workers acting as the nine IoT devices from the N-BaIoT dataset, i.e., each worker received only subsets of the dataset that came from the corresponding IoT device (note that in N-BaIoT, there are nine separated CSV files, one for each IoT device). As seen, DCD-FL has a better F1 score for all devices and classes (benign and attacks) reported by Kim et al. [46].

Ge et al. [35] report results obtained using a Multi-Layer Perceptron and the Bot-IoT dataset, in the form of a confusion matrix, from which it was computed the accuracy, precision, recall, and F1 score metrics, shown in Table 4.11, which compares their metrics and the produced by DCD-FL, using an aggregator and ten workers nodes after ten aggregation rounds.

The results in the paper [35] were obtained by manually fine-tuning the hyperparameters of the MLP. DCD-FL produced similar results, even not performing any tuning of

Table 4.11 – Accuracy, Precision, Recall, and F1 Score for Bot-IoT using a MLP

	Accuracy		**Precision**		**Recall**		**F1 Score**	
	[35]	DCD-FL	[35]	DCD-FL	[35]	DCD-FL	[35]	DCD-FL
Normal	99.67%	**99.84**%	99.84%	95.06%	98.31%	**98.65**%	99.07%	96.83%
DDoS/DoS	99.42%	98.22%	99.37%	98.15%	99.70%	98.20%	99.53%	98.17%
Reconnaissance	99.12%	**99.99**%	98.40%	88.00%	97.91%	70.21%	98.15%	78.21%
Information theft	99.82%	**100.0**%	76.80%	50.00%	93.81%	30.77%	84.44%	38.10%

hyperparameters. However, the results for the Information theft class are not satisfactory (precision, recall, and F1 score under 50%), which is justified by the fact that this class has only 65 samples from circa 2,934,817 samples in the Bot-IoT dataset, and so making it difficult to obtain better results without some specific adjusts in the model.

4.2.6 Support to Different IoT Applications

It is described here how the testbed environments used to generate the datasets adopted in the experiments can be easily mapped to the DCD-FL architecture. Thus it can be deployed into any of them, and so it demonstrated that DCD-FL can cope with specific characteristics of different types of IoT applications,

Figure 4.1 illustrates the testbed environment for the UNSW-NB15 dataset, which has elements that characterize a smart city IoT system. In a direct map from this testbed environment to the DCD-FL architecture, each DCD-FL worker node corresponds to a given client in the testbed, receiving batches from the UNSW-NB15 dataset, to be used in model training, as it was its monitored network traffic. Servers 1 and 3 are mapped to DCD-FL aggregator nodes to collect local models and generate and share global models, all working in a decentralized manner.

As shown in Figure 4.2, described previously, the testbed environment used to generate the Bot-IoT dataset has all the characteristics of a smart home IoT application. This testbed can be easily mapped to a DCD-FL setup, where each VM and simulated IoT device corresponds to P2P nodes in DCD-FL, either collecting traffic and performing local model training or collecting local models, performing aggregations and sharing the generated global models with the other P2P nodes in DCD-FL.

Similarly, the testbed environment shown in Figure 4.3, which was used to produce the N-BaIoT dataset, has typical characteristics of a smart home IoT application and can also easily be mapped to the DCD-FL decentralized architecture. Each IoT device can assume the role of a DCD-FL worker node, or even an aggregator node, if it has enough resources, to perform model training tasks activated by the DCD-FL planner nodes. In this testbed, the

sniffer server can assume the role of an aggregator node to produce the global models and share them with the other nodes.

4.3 Summary

This chapter presents experiments designed to analyze the effects of various parameters on the quality of trained models. One experiment evaluated the impact of the number of aggregators on model training, revealing that increasing the number of aggregators without increasing the number of workers can lead to faster convergence. However, with more workers and just one aggregator, the metrics are below 80%, indicating the need for more rounds to achieve the metric values obtained with fewer workers.

The chapter also explores the impact of different deep learning models, including a Multi-layer Artificial Neural Network (ANN), a Long Short Term Memory (LSTM), and a Gated Recurrent Network (GRU), applied in the same dataset, Bot_IoT. All models showed satisfactory metrics (above 90%), except for "Class 4" representing the "Data theft" attack. Notably, the ANN model was about five times smaller than the other models and had a shorter execution time, making it a preferable choice for IoT-based systems, which usually have constrained resources.

Finally, a comparative analysis demonstrated that DCD-FL produces results comparable to those in the literature but with the benefits of a decentralized approach to intrusion detection. The communication cost was linearly proportional to the number of participant nodes in the aggregation rounds, with an increase in communication cost offset by a significant increase in the relevant model metrics, such as recall and f-1 score, highlighting the advantages of decentralized, federated learning.

5. RELATED WORK

This chapter presents the work with similar characteristics to the present one. First, the conducted systematic literature review is described and discussed, and then each selected work is described and compared with the work of the present thesis.

A systematic literature review was conducted to identify the IoT state-of-the-art, including IoT security-related issues, which are the main focus of this work. This review was important to unify the main concepts and bring together the technologies related to IoT and IoT security aspects.

The initial step of the review consisted of searching the most relevant scholarly academic databases: IEEE Explorer (http://ieeexplore. ieee.org), Google Scholar (http://scholar. google.com), ACM Digital Library (http://dl.acm.org), Wiley (http://onlinelibrary.wiley.com), Springer (http://link.springer.com), ArXiv (https://arxiv.org/), and Science Direct (http://www. sciencedirect.com).

The key terms used in the electronic searching, chosen according to the research questions of this thesis proposal, including combinations of *Internet of Things*, *IoT*, *security*, *intrusion detection*, *federated machine learning*, *deep learning*, *distributed ledgers*, and *survey*.

The obtained search results were grouped into four categories: surveys (to unify the concepts), approaches (to determine state-of-the-art), security issues (to identify open problems), applications. The results were filtered according to document relevance, the number of citations, as given by search engines, and the kind of content based on analysis of the abstract section.

In the next step of the literature review, after applying the exclusion criteria (published not from 2015 to 2022, content not related to the focus of the present thesis, and low relevance as indicated by search engines), the Author of this work read the more relevant papers. And so new bibliographical references of interest were included in the review, allowing one to consider papers that had not matched to key terms used in the electronic search. Table~5.1 shows the number of publications in each category.

Table 5.1 – Literature Review: number of publications (2015-2023)

Topic	Surveys	Approaches	Security Issues	Applications	Total
Intrusion detection for IoT	36	407	28	8	479
Federated learning for IoT	19	373	28	10	430
Distributed ledgers for IoT	48	229	112	46	435
Total	103	1,009	168	64	1,344

According to the search engines and further analysis of the present Author, a subset of the more relevant collected papers are listed in the following tables and discussed.

Table 5.2 lists the selected surveys and reviews about intrusion detection, federated learning, and distributed ledgers focused on IoT.

Table 5.2 – Selected surveys and reviews

Refs	Year	Title	Description
		IoT Concepts	
[15]	2012	Fog Computing and Its Role in the Internet of Things	Definition of Fog layer characteristics, showing it as the appropriate platform for critical Internet of Things (IoT) services and applications. Included here, even in the exclusion criterium, since it is where Fog computing is first characterized
[17]	2014	The Internet of Things vision: Key features, applications, and open issues	Included here, even in the exclusion criterium, since it presents an interesting vision of IoT operations in terms of four phases: collection, transmission, and processing/management/utilization phases
[99]	2018	A survey on Internet of Things architectures	Survey on IoT-oriented architectures, discussing related tools, technologies, and methodologies to solve real-life problems by building and deployment of IoT applications
[127]	2021	Internet of Things 2.0: Concepts, Applications, and Future Directions	Survey on the evolution of IoT, presenting the vision for IoT 2.0, focusing on the fields of machine learning intelligence, mission-critical communication, scalability, energy harvesting-based sustainability, interoperability, user-friendly IoT, and security
		IoT Security	
[10]	2018	Internet of Things: A survey on the security of IoT frameworks	Survey on the security of eight IoT frameworks, discussing the proposed architecture, compatible hardware, and security features
[28]	2021	Centralized, Distributed, and Everything in between: Reviewing Access Control Solutions for the IoT	Review centralized, hierarchical, federated, and distributed access control approaches to IoT systems and examine the suitability of each one for specific requirements of access control in IoT
[43]	2020	Machine Learning in IoT Security: Current Solutions and Future Challenges	Review security requirements, attack vectors, and security solutions based on ML and DL, which can be applicable to IoT networks
[49]	2018	Internet of Things security: A top-down survey	Survey of security and privacy solutions in IoT, using blockchain and Software Defined Networking, with comparisons based on various parameters
[52]	2016	Intrusion detection in mobile ad hoc networks	Survey of intrusion detection techniques in MANETs considering technologies and detection algorithms, classified into nine categories, comparing different techniques, according to strengths and limitations

Continued on next page

Table 5.2 – Selected surveys and reviews (continued)

Refs	Year	Title	Description
[67]	2019	Internet of Things Applications, Security Challenges, Attacks, Intrusion Detection, and Future Visions	Review of different security issues in IoT layers (perception, network, support, and application), focusing on Distributed Denial of Service (DDoS) attacks, making a comparison between IDSs oriented to this type of attack
[85]	2019	Current research on Internet of Things (IoT) security: A survey	Review of research work in IoT security from 2016 to 2018, discussing relevant tools, IoT modelers, and simulators
[96]	2023	Access Control for IoT: A Survey of Existing Research, Dynamic Policies, and Future Directions	Survey on access control in the IoT, including access control requirements, technologies, models, policies, research challenges, and future directions
[101]	2019	A survey of network-based intrusion detection data sets	Literature survey on data sets for network-based intrusion detection, identifying 15 different properties to assess the suitability of each data set for specific scenarios and highlighting the peculiarities of each data set
[115]	2020	Machine Learning Based Intrusion Detection Systems for IoT Applications	Review on ML classifiers to use as IDSs, trained and validated with CIDDS-001, UNSW-NB15, and NSL-KDD datasets, including deployment in a Raspberry Pi to evaluate timing on an IoT specific hardware
[125]	2018	Data Security and Privacy-Preserving in Edge Computing Paradigm	Review of the data security and privacy threats, protection technologies, and countermeasures inherent in edge computing
		Intrusion Detection Systems	
[4]	2020	Intrusion detection: Issues, problems, and solutions	Study of intrusion detection, discussing popular attacks, examining problems associated with their detection, and exploring possible solutions
[14]	2018	A Critical Review of Practices and Challenges in Intrusion Detection Systems for IoT	Review of intrusion detection systems for IoT technologies, focusing on architecture types, pointing out future directions
[19]	2019	Network Intrusion Detection for IoT Security Based on Learning Techniques	Review of existing IoT NIDS implementation tools and datasets and survey of the state-of-the-art in terms of architecture, detection methodologies, validation strategies, treated threats, and algorithm deployments
		Federated Learning for IoT	
[56]	2022	From distributed machine learning to federated learning	Survey of federated learning works, proposing a taxonomy of related techniques and classifying FL systems according to types of parallelism, aggregation algorithms, data communication, and the security that they use or provide

Continued on next page

Table 5.2 – Selected surveys and reviews (continued)

Refs	Year	Title	Description
[57]	2021	A Systematic Literature Review on Federated Machine Learning	Systematic literature review on federated learning, from a software engineering perspective, considering development phases of background understanding, requirement analysis, architecture design, implementation, and evaluation
[98]	2019	Intrusion Detection at the Network Edge	Review the possible IDS solutions to different Fog tiers, with related deployment and design issues; and indicate future research directions
Distributed Ledgers in IoT			
[1]	2018	The Blockchain of Things, Beyond Bitcoin	Overview on the use of blockchain-based platforms for IoT
[18]	2019	A systematic literature review of blockchain-based applications	Investigate the current state of blockchain technology and its applications, based on a systematic review of the literature, pointing out research gaps and suggesting future directions
[39]	2019	Privacy preservation in blockchain-based IoT systems	Review implementation of five privacy preservation strategies in blockchain-based IoT systems (anonymization, encryption, private contract, mixing, and differential privacy)
[59]	2019	The blockchain: State-of-the-art and research challenges	Review of research work on the blockchain, exploring its main components, blockchain-based IoT, blockchain-based security, blockchain-based data management, and applications based on the blockchain
[61]	2019	Blockchain's adoption in IoT: The challenges, and a way forward	Review of the peculiarities of the IoT environment and how they can be efficiently supported by blockchain technologies, including some practical issues related to the integration of IoT devices with the blockchain
[64]	2019	Blockchain in healthcare applications: Research challenges and opportunities	Review of blockchain focusing on its application in the healthcare sector, which requires more stringent authentication, interoperability, and record/sharing constraints. It is discussed the advantages and open issues related to this use of blockchain technologies
[71]	2019	Securing IoT in distributed blockchain: Analysis, requirements and open issues	Literature review on the application of blockchain to IoT-based systems, providing a taxonomy with strengths, weaknesses, opportunities, and threats of blockchain in IoT and also indicating its implementation requirements
[117]	2019	Survey on blockchain for the Internet of Things	Survey on Blockchain technologies used in IoT applications, identifying critical issues and possible adaptations and enhancements to the Blockchain consensus protocols and data structures

The selected papers are discussed in the following sections according to the mentioned categories and the topics related to our research questions.

5.1 Intrusion Detection Systems for IoT

This section discusses the existing research on intrusion detection systems in IoT, introducing the different types of systems that have been proposed, their key features, and their limitations, highlighting the novelty and significance of the present thesis. Table 5.3 lists the selected papers about approaches, security issues, and applications related to intrusion detection for IoT systems.

Table 5.3 – Intrusion Detection for IoT Systems

Refs	Year	Title	Description	Category
[3]	2019	Mobile Encrypted Traffic Classification Using Deep Learning	Deep learning approach to mobile traffic classifiers, automatically extracting features, and able to cope with encrypted traffic	Deep learning-based IDS
[6]	2019	Deep recurrent neural network for IoT intrusion detection system	Use of the multi-layered recurrent neural network to implement IDS at the Fog layer	RNN-based IDS
[29]	2020	Passban IDS: An Intelligent Anomaly-Based Intrusion Detection System for IoT Edge Devices	Lightweight IDS to protect IoT devices that are directly connected to it. It can be deployed directly on very cheap IoT gateways	IDS for IoT
[31]	2020	Deep learning for cyber security intrusion detection	Review of intrusion detection systems based on deep learning approaches, describing 35 cyber datasets distributed into seven categories, analyzing seven deep learning models (binary and multi-class setups), and evaluating the performance of these methods	Deep learning-based IDSs
[35]	2019	Deep Learning-based Intrusion Detection for IoT Networks	Intrusion detection scheme for IoT, classifying traffic using a feed-forward neural network model for binary and multi-class, trained with the Bot-IoT dataset	IDS for IoT
[46]	2020	Intelligent Detection of IoT Botnets Using Machine Learning and Deep Learning	Botnet detection intrusion for each device of the N-BaIoT dataset, using different machine learning models to identify the more effective in terms of F1-score in binary and multi-class classification	Deep learning-based IDS
[51]	2020	Machine Learning-Based Early Detection of IoT Botnets Using Network-Edge Traffic	Lightweight IoT botnet detection solution, which can be deployed in Raspberry Pi, based on machine learning algorithms.	IDS for IoT

Continued on next page

Table 5.3 – Intrusion Detection for IoT Systems (continued)

Refs	Year	Title	Description	Main topics
[53]	2021	UIDS: a unified intrusion detection system for IoT environment	Unified intrusion detection system for IoT environment (UIDS), based on a combination of four different decision tree models, which were trained using the UNSW-NB15 dataset, in order to detect four specific types of attacks (exploit, DoS, probe, and generic)	Decision Tree-based IDS
[54]	2017	A Semi-Supervised Intrusion Detection System Using Active Learning SVM and Fuzzy C-Means Clustering	Hybrid semi-supervised machine learning technique, combining Support Vector Machine (SVM) and Fuzzy C-Means (FCM) clustering, using the NSL-KDD dataset for model training	IDS for IoT
[58]	2021	E-GraphSAGE: A Graph Neural Network-based Intrusion Detection System	Network intrusion detection system (NIDS) based on Graph Neural Networks (GNNs), using an adapted GraphSAGE model, which was trained using UNSW-NB15, Bot-IoT, and TON_IoT datasets	GNN-based IDS
[119]	2020	Large-Scale and Robust Intrusion Detection Model Combining Improved Deep Belief Network With Feature-Weighted SVM	IDS combining Deep Belief Network (DBN) with feature-weighted support vector machines (WSVM), using the NSL-KDD dataset to validation	ML-based IDS

As summarized in Table 5.3, many recent works are oriented to intrusion detection in IoT systems. Most are trying to apply machine learning techniques to implement intelligent IDSs. All these approaches are centralized, have the single point of failure problem, do not adopt the federating learning technique, and cannot provide data privacy.

Table 5.4 compares DCD-FL, and other IDS approaches for IoT systems.

Table 5.4 – Comparison between DCD-FL and other approaches

IDS	Technique	Datasets	Better results	Comments
Aceto et al. [3]	Auto-encoder, LSTM, CNN, Encrypted traffic	Private wireless traffic, and Facebook and Messenger traffic in a lab	93.45% Accuracy in Android setup and 93.32% in iOS	CNN with best results, but no detail on datasets makes any comparison difficult
Almiani et al. [6]	Recurrent Neural Network at Fog layer	Balanced NSL-KDD	Simulation in Matlab, FPR 4.51%, Recall 48.99%, Precision 49.57%, and F1 score 49.28%	NSL-KDD is deprecated and results by simulation only

Continued on next page

Table 5.4 – Comparison between DCD-FL and other approaches (continued)

IDS	Technique	Datasets	Better results	Comments
Eskandari et al. [29]	Local Outlier Factor (LOF) and Isolation Forest (iForest)	User-generated benign traffic, captured using tcpdump, and also four types of attacks	On Raspberry Pi 3 Model B, port scan attack detected with precision 98% and recall 100%	Tiny dataset and very specific few attacks, so not very representative of IoT systems
Ge et al. [35]	Feed-Forward neural network	Bot-IoT dataset	On a Google's Colab setup, obtained metrics above 90% for binary and multi-class cases	Straightforward ML model, a FF with two layers, which was fine-tuned for the Bot-IoT dataset
Kim et al. [46]	CNN, RNN, LSTM, NB, KNN, DT, RF	N-BaIoT dataset	Identified as the most effective ML models in detecting Bashlite and Mirai botnets are decision trees and random forests in both binary and multiclass cases	Both the identified models are not suitable for IoT applications since these models are static models that do not adapt well to changes in the underlying data distribution, which is the usual case for IoT-based systems
DCD-FL [77]	Decentralized FL using MLP, RNN, LSTM, GRU	N-BaIoT, UNSW-NB15, Bot-IoT	As reported previously, different types of experiments and main metrics around 90% in all datasets and models, except by particular under-sampled attack classes	Three different recent datasets of IoT applications and results with actual execution of ML models

5.2 Federated Learning in IoT

Here, the existing research on the use of federated learning in IoT is discussed, presented the benefits of federated learning in this context, such as improved privacy and scalability, and how different researchers have implemented it.

Machine learning techniques, particularly deep learning techniques, have been widely applied to intrusion detection [31]. However, federated learning application in intrusion detection is recent (last four years), and the combination with distributed ledgers for some aspects of this area is also very recent. Table 5.5 summarizes the FL-based approaches to implement IDSs, found in the literature of the last six years.

Table 5.5 – Federated Learning-based IDSs.

Refs	Description	C/D[a]	BC[b]	IoT[c]
Nguyen et al.[82]	Representing network packets as symbols in a language to use a Recurrent Neural Network to train local models and a central server to produce a global model per device.	C	No	Yes
Diro et al.[26]	Global model produced by a master fog node by collecting model updates from IoT devices, which use an SGD-based algorithm in the model training.	C	No	Yes
Prabavathy et al.[92]	Intrusion detection is oriented to the fog layer and acts not close to the IoT devices at the edge layer.	C	No	Yes
Sahan et al.[106]	IDS for conventional networks across various organizations. Each organization performs model training locally using LSTM, sending model updates back to the central, which performs aggregation to update the global model.	C	No	No
A.-Basset et al.[2]	IDS for vehicular networks, using Transformers to learn vehicular traffic patterns, which allows for detecting attacks. A blockchain is used for registering valid model updates.	C	Yes	Yes
Mothukuri et al.[72]	Using LTSM and GRU to implement local model training at IoT devices and a Random Forest decision tree to perform global intrusion prediction.	C	No	Yes
Liu et al. [55]	Specifically for vehicles, each vehicle works as an edge server training local models based on its data. RSUs perform aggregation to produce global models.	D	Yes	Yes
Friha et al. [33]	Using just one machine learning model (MLP) and Edge IIoTset dataset, specifically oriented to IIoT, each factory performs centralized FL by training local models at edge devices, which are also responsible for aggregating and sharing a global model with other factories. In the aggregation process, inside each factory, all edge devices share local models with every other device, so there is a high communication cost to consider.	D	No	Yes
Nascimento & Hessel [77]	Using different ML models and datasets, use decentralized federated learning-based aggregation to train global models combining models from aggregators and from local models, which are trained at the edge of IoT networks, without having access to data of devices; and trained global models are shared back with all devices.	D	Yes	Yes

[a]Centralized or Decentralized, [b]Blockchain use, [c]For IoT

One of these recent works is from Liu et al. [55], reporting an intrusion detection system built using a federated learning approach specifically for vehicles, where each vehicle works as an edge server training local models based on its data. The aggregation is performed by RSUs (Road Side Units) to produce global models. All the models are stored in a public blockchain to improve the models' security. Unlike our approach, the adopted aggregation algorithms are mainly oriented to vehicles, not any IoT device. Moreover, using a

public blockchain and a PoW consensus algorithm makes the approach resource and time-consuming to be adopted for any kind of IoT system. As discussed previously, the reported results are on the KDDCup99 dataset, which is already obsolete.

DÏoT [82] is a federated learning intrusion detection approach based on representing network packets as symbols in a language. This strategy allows the implementation of a language analysis technique to detect anomalies using GRU (Gated Recurrent Neural Network), a kind of Recurrent Neural Network. [83]. According to the IoT device type, it adopts a federated learning approach for aggregating anomaly-detection profiles for intrusion detection. Unlike our work, where an entirely decentralized federated learning approach is defined, DÏoT depends on a centralized IoT security service to aggregate all local models generated by security gateways to produce new models later sent back to the security gateways. This situation constitutes a single point of failure, which is not desirable in robust approaches to security.

Diro et al. [26] developed an IDS based on a distributed deep learning approach at the fog layer. A master fog node collects model updates from other fog nodes that perform SGD-based training on their local data to generate local models. So, the master node aggregates the model updates to produce a global model, which is shared back to all fog nodes. The fog nodes directly connected to IoT devices collect network traffic and perform predictions to detect eventual intrusion. This approach assumes a single global model that can demand growing computational resources to be generated proportionally to the increasing number of connected IoT devices. Also, it can not consider the many different network traffic characteristics according to the IoT device types. Moreover, it is assumed a single fog node is a master, representing a single point of failure in the system.

Prabavathy et al. [92] developed an intrusion detection technique in the fog layer, adopting machine learning to interpret the attacks from network traffic and taking advantage of the distributed architecture of fog computing to obtain an intrusion detection mechanism with scalability, flexibility, and interoperability. In this approach, ML-based intrusion detection is oriented to the fog layer. So it will not act close to the IoT devices at the edge layer, which can cause a communication overhead between the nodes.

Sarhan et al. [106] proposed a federated learning-based IDS for conventional networks across various organizations, i.e., not for IoT. Each participating organization receives a global model from a central organization and performs model training locally using LSTM, sending model updates back to the central, which performs aggregation to update the global model. The approach evaluation used the UNSW-NB15, and BoT-IoT datasets, converted to NetFlow format [106].

FED-IDS, proposed by Abdel-Basset et al. [2], is an FL-based IDS for vehicular networks, using deep learning, specifically Transformers, to learn spatial and temporal representations of vehicular traffic, allowing for detecting attacks. In addition, a blockchain

implements a decentralized and secure platform for the vehicle's interaction, registering valid model updates. The dataset TON_IoT is used in the evaluation of the IDS.

Mothukuri et al. [72] proposed an FL-based IDS using Recurrent Neural Networks (LTSM and GRU - Gated Recurrent Networks) to implement local model training at IoT devices and a Random Forest decision tree to execute a kind of model aggregation to perform global intrusion prediction for man-in-the-middle and DDoS attacks.

5.3 Distributed Ledger Technologies Applied to IoT

This section discusses the existing research on the use of distributed ledger technologies in IoT systems. In general, as a distributed ledger, blockchain has decentralized control (no central authority dictates the rules); allows data transparency and audibility (a full copy of every transaction ever executed in the system is stored in the blockchain and is public to all the participant nodes); distributes information (every node keeps a copy of the blockchain to avoid having a centralized authority privately keep all that information); implements a decentralized consensus (transactions are validated by all the participant nodes instead of a central entity, even in the presence of unreliable nodes); guarantees data immutability (once registered, data can not be modified anymore); and, is secure (the blockchain is tamper-proof and cannot be manipulated by malicious nodes).

Given the characteristics mentioned above of blockchain, many approaches to access control are adopting it in implementing authentication, authorization, and auditing, which are the main mechanisms to implement IoT security. Table 5.6 presents some relevant approaches based on blockchain technologies for IoT security.

Table 5.6 – Distributed Ledger Approaches for IoT Security

Refs	Goal	Solution	Review	Limitations
[7]	Secure authorized access to IoT resources	IoTChain, a scheme that provides End-to-End authorization	Ethereum blockchain is used to implement a trustless way to handle authorization, with the possibility of multicast groups for authorized clients	Not yet actually applied and yet using a permissioned and PoW version of Ethereum
[11]	Decentralized authorization in IoT systems	Using smart contracts on a public blockchain to implement decentralized authorization	Blockchain-based authorization with a mechanism to guarantee privacy to stored resources	Nodes may not stay up to date on the chain, depending on some conditions

Continued on next page

Table 5.6 – Distributed Ledger Approaches for IoT Security (continued)

Refs	Goal	Solution	Review	Limitations
[12]	Ensure the integrity of shared datasets in IoT-based systems	Two conceptual blockchain-based approaches to share datasets for IoT	Survey on IoT datasets and discussion on how to use blockchain to store them	No concrete results are presented; it is a position paper
[20]	Reduce blockchain storage overhead in the initial synchronization process and the following maintenance	BC-Store, a combination of local in-cache blockchain storage and external storage in IPFS	Solution is detailed, and interesting empirical results are reported	Reported results only for Bitcoin blockchain
[25]	Decentralized access control in IoT systems	Attribute-based access control scheme for IoT, based on records in the blockchain.	The access control is optimized to perform lightweight calculations, and security and performance analysis show resiliency to multiple attacks	Oriented to permissioned blockchain only
[24]	End-to-end trust model for IoT without relying on any common root of trust	Distributed trust model for the IoT, using a credit-based Blockchain with a built-in reputation mechanism	Provide a security analysis for both the used blockchain and the overall architecture and also experimental results	In the case of operators refusing to endorse their clients, a denial of service (DoS) attack may occur
[27]	A lightweight blockchain adequate to be used in IoT by eliminating the Proof of Work (POW) and the concept of coins	A private blockchain, used for controlling and auditing communications, which can be deployed in a smart home device	Security analysis argues that the blockchain-based smart home framework is secure, and simulation results show irrelevant overheads	"Miner", responsible for local blockchain, and its local storage is a single point of failure
[60]	Blockchain technology to define access control systems, oriented to auditability of access control policies evaluation	A smart contract-based access control system and its application to a reference scenario, where the resources to be protected are themselves smart contracts	Presents also an implementation exploiting XACML policies and Solidity written smart contracts deployed on the Ethereum blockchain	Due to auditability requirements, there is no implemented data privacy in the access control policies stored in the blockchain

Continued on next page

Table 5.6 – Distributed Ledger Approaches for IoT Security (continued)

Refs	Goal	Solution	Review	Limitations
[88]	Framework for access control in IoT based on the blockchain technology	FairAccess, a decentralized and privacy-preserving authorization management framework where users own and control their data	Blockchain-based framework providing transactions to grant, get, delegate, and revoke access, with implementation and deployment in a Raspberry PI device and local blockchain	Implementation using Bitcoin blockchain not satisfies usual restrictive time constraints of IoT-based applications
[94] [95]	Access control system for protection from unauthorized access and automated detection of compromised nodes in IoT	Blockchain-based Trust and Reputation System (TRS) for IoT access control, which evaluates the participant node trust and reputation score, providing a self-adaptive access control system	Implementation in private Ethereum blockchain with Docker containers and reported performance metrics compatible with IoT applications	Not suitable to resource-constrained IoT devices since they need to perform asymmetric cryptography
[126]	Distributed and trustworthy access control for IoT systems	A smart contract-based framework, which consists of multiple access control contracts (ACCs), each one with a specific role in the access control process	Provide a case study in an IoT system, where the ACCs, JC, and RC are implemented based on the Ethereum smart contract platform to achieve the access control	Case study is too simple and not suitable to demonstrate the feasibility of the approach, mainly in terms of smart contract execution costs

Alphand et al. [7] present IoTChain, a scheme that provides End-to-End authorization for secure authorized access to IoT resources, where a permissioned Ethereum blockchain is used to implement a trustless way to handle authorization, with the possibility of multicast groups for authorized clients. However, it still needs to be used in an IoT application.

Andersen et al. [11] present WAVE, a decentralized authorization system that provides fine-grained permissions, noninteractive delegation, and proofs of permission and supports revocation. WAVE uses smart contracts on a public Ethereum blockchain to implement decentralized authorization, with a mechanism to guarantee privacy to registered information in the blockchain. However, nodes may not always stay up to date on the chain, depending on some conditions related to where they are deployed.

Banerjee et al. [12] proposes using blockchain to ensure the integrity of shared datasets in IoT-based systems. They describe two conceptual blockchain-based approaches to share datasets for IoT. But, no concrete results are presented; it is just a position paper.

Ding et al. [25] also present blockchain-based access control oriented to IoT. Authentication and authorization attributes are registered in a blockchain, which validates any

posterior request for access to resources. Since any request must go through a global blockchain, even if it is permissioned, an unacceptably high latency may occur for access request response.

Di Petro et al. [24] present a blockchain-based protocol to establish trust between service clients and service providers by combining Bitcoin blockchain with a subchain (called *obligation chain*) where the client/providers agreements are registered.

An architecture for access management in IoT based on the use of a blockchain is proposed by Novo et al. [87]. The IoT devices are isolated from the blockchain network and must register in a manager node, which interacts with the blockchain network. The manager defines access control rules for the IoT devices' resources and can modify and delete policies for the IoT devices. These management operations are performed employing smart contract transactions in the blockchain, which is one of the main limitations of the proposed architecture since the waiting time for transactions to complete in the blockchain is too much longer and not appropriate for some management operations, for example, the revocation of some privilege for a given IoT device.

The FairAccess framework [88] also uses smart contracts in a blockchain for controlling access to resources in an IoT system based on registered policies in the blockchain. Putra et al. [95] make use of smart contracts in a blockchain to develop an authorization model based on trust, where the evaluation of positive and negative interactions of nodes in the IoT system determines the level of trust of each one. These smart contract-based approaches depend on computational, communication, and storage resources beyond the edge devices' capacities without providing some mechanism to consider these constraints, as our approach proposes.

5.4 Summary

This chapter explores the existing research on intrusion detection systems (IDS) for the Internet of Things (IoT) based on machine learning techniques, mainly based on federated learning techniques and the application of distributed ledger technologies for IoT security. The chapter introduces various IDS types, key features, and limitations. It highlights that most current systems are centralized and do not adopt federated learning techniques, leading to potential single points of failure and lack of data privacy.

The chapter also presents a comparative analysis between DCD-FL, the developed approach in the present thesis, and other IDS approaches for IoT systems. This analysis includes a summary of IDS techniques, datasets they utilize, their results, and comments on their effectiveness and limitations.

Lastly, the chapter reviews the application of distributed ledger technologies for IoT security. It discusses solutions to reduce blockchain storage overhead, decentralize access control in IoT systems, and create end-to-end trust models for IoT without relying on any common root of trust. Each solution is evaluated based on its goals, the details of the resolution, its review, and its limitations. The chapter underscores the need for more decentralized and privacy-preserving solutions in the IoT security landscape.

The next chapter details the present thesis's advances and gives some indications for future work related to the topics discussed in this chapter.

6. FINAL CONSIDERATIONS

This work contributes to solutions for critical IoT security issues. It is oriented to developing decentralized security architecture for intrusion detection in IoT-based systems, allowing secure IoT applications. Decentralized, federated machine learning in a data privacy-preserving manner, using deep learning techniques to learn and adapt itself to new kinds of intrusions dynamically, is combined with distributed ledgers to minimize the security risks associated with not guaranteeing integrity, confidentiality, and availability of IoT-based systems. The present thesis has as main contributions:

- a decentralized, federated learning architecture for IoT-based systems, which can predict attacks based on previous incidents, as well on the behaviors of IoT devices, in a privacy-preserving and fault-tolerant way;
- use of distributed ledgers providing security mechanisms for authentication, authorization, integrity, confidentiality, and high availability in a way that IoT-based systems can effectively and efficiently use; and
- prototype implementation for the developed architecture, which demonstrated its feasibility.

6.1 Revisiting the Thesis Research Proposal

In this Section, the thesis research proposal is reviewed to analyze the research problems and objectives, hypotheses, and research questions, which were proposed in the context of what was accomplished with the developed work in the present thesis.

6.1.1 Research Problems and Objectives

DCD-FL, as a decentralized, federated machine learning-based approach, allows us to handle problems P.1, P.3, and P.4 (see Section 1.1) since model training and prediction for intrusion are performed locally at edge nodes, selected from the ones that can execute the tasks, according to their available resources, and only models are transmitted to other edge nodes for global models generation.

DCD-FL intrusion detection mechanism is based on deep learning techniques, which explore the network traffic patterns to predict possible attacks. Model training and prediction for intrusion are periodically scheduled and performed at each node, generating alerts when some anomalous behavior is detected.

Moreover, distributed ledgers (global and local blockchains) and off-chain storage, included in the DCD-FL approach, guarantee the integrity and high availability of the necessary information for the IoT security platform in a totally decentralized way. They also support authentication, authorization, and auditing mechanisms, essential to treat problem P.2 (see Section 1.1) by providing necessary security features in a distributed manner.

According to the available computation, memory, and energy resources in the corresponding host, instances of the IDS clusters can also be deployed on different edge servers and provide services (e.g., for monitoring network traffic, intrusion detection, etc.) and requests services (e.g., for obtaining trained models, for performing authentication and authorization, etc.).

Since computation tasks in DCD-FL will be primarily executed at the edge layer, DCD-FL decentralized architecture will reduce network traffic between edge servers and fog/cloud servers. Moreover, privacy will be enhanced since the monitored information in the network traffic will not be transmitted to fog and edge servers. Adopting this architecture will also be possible to cope with problem P.1 (see Section 1.1). In summary, problems considered in the present thesis include:

- *P.1 - vulnerable and constrained devices should be protected and supported by any IoT security platform, intended to be effective and efficient*: DCD-FL makes use of distributed security mechanisms in a decentralized way, allocating processing loads to the devices according to their available resources and also allowing the use of computation and storage resources from the fog level, when necessary, thus not depending only on the edge devices; this is possible due to the totally decentralized architecture of our developed security system;

- *P.2 - a considerable amount of generated data by devices should not be transmitted to fog and cloud to be processed by the intrusion detection system; instead, the IDS should be near to where the data is produced*: in DCD-FL, data is maintained close to or even only locally in the edge devices, which generated it, depending on the available resources of each device

- *P.3 - traditional machine learning techniques for intrusion detection, demanding some kind of manual feature engineering, are not enough anymore, given the very dynamic aspects of IoT, in terms of growing new types of devices and applications*: DCD-FL makes use of deep learning algorithms in the machine learning tasks, which not depend on manual activities, as manual feature engineering;

- *P.4 - data privacy preserving is becoming even more critical, since penalties for privacy violation become very severe, according to recently approved regulations in many countries*: in DCD-FL, model training is performed using local data, minimizing data traffic between the devices, ideally stored only locally at the edges devices; and, when

necessary, only the models are transmitted through the network to be used by the aggregation algorithms, which generate global models that are sent back to the edge devices;

- *P.5 - typical fog and cloud cyber threats will also affect the edge since they are all interconnected, and so all of them must be carefully considered*: DCD-FL uses cryptography to protect the data that need to be exchanged between all the system components and digital identity to be used by the system components to get authentication and authorization to access the data.

6.1.2 Hypotheses and Research Questions

In the present thesis, hypotheses that were validated include:

- *H.1 - Federated machine learning techniques for intrusion detection allow obtaining an autonomous and decentralized, effective mechanism to minimize security risks in IoT systems*: It was developed a decentralized software architecture based on concepts from P2P networking and a detection intrusion mechanism, based on deep learning algorithms from the machine learning area of Artificial Intelligence, which effectively minimizes security risks by alerting possible security threats in IoT based systems; and the mechanism is performed mainly on the edges devices; also, a prototype was implemented based on the defined architecture, which results can demonstrate the feasibility and efficiency of the developed solution;
- *H.2 - Distributed ledgers guarantee the adequate availability and integrity of digital identities and all necessary access control information to the authentication and authorization for resource uses in IoT systems*: DCD-FL adopted distributed ledgers to persist information about the trained models and also to register the digital identities of the edge devices, which are authorized to participate in the security system tasks; moreover, the contents of the trained models are persisted in decentralized stores, to alleviate the demands for access to the distributed ledgers and to maintain the decentralized nature of the developed security system;
- *H.3 resulting security platform can be adopted in many kinds of IoT applications*: an implementation of the developed decentralized architecture for DCD-FL, based on abstractions from the P2P networking, shows the possibility of deployment of the security system in any IoT based system, where there are devices at the edge, fog, or cloud level, which can perform some basic computation and store data (to be used in the model trained task in the edge devices) and models (generated by the devices and used by them to predict eventual intrusions in the system)

Research questions (RQ) explored in the present thesis, and specific objectives (SG) include:

- *RQ.1 - What are the state-of-the-art of intrusion detection and prevention techniques, and how can they be applied to IoT systems?*
  - *SG.1 - To identify and report the state-of-the-art of intrusion detection and prevention approaches that can be applied to IoT systems*: a systematic literature review was conducted, which identified the current approaches to intrusion detection techniques applied to IoT-based systems, presented in the Related Work section of the thesis (see Chapter 5), allowing the development of architecture and its prototype to handle some open problems, reported in the literature, and to demonstrate its feasibility by performing various practical experiments, respectively
  - *SG.5 To report the research results by publishing them in scientific publications and presenting at academic events, as well as exploring the possibility of technology transfer to the industry*: It was published two papers, one review on decentralized, federated learning (WF-IoT'2022 [78]) and one on DCD-FL, developed decentralized, federated learning architecture (AINA'2022 [77]).
- *RQ.2 - Can Federated machine learning techniques effectively apply to intrusion detection and prevention in IoT systems?*
  - *SG.2 - To develop an efficient, effective, and flexible IoT security platform*: performed experiments, which correlate model quality and topology of the P2P network (number of aggregators and workers), correlate model size and topology/-model type/dataset, and correlate number of local models used to aggregate to generate the global model, produce comparable results to the existing centralized approaches (see Section 4.2.5), with the additional advantages of the decentralized approach of DCD-FL;
  - *SG.3 - To explore the use of federated learning to implement distributed machine learning algorithms for intrusion detection and prevention in IoT systems*: the DCD-FL developed decentralized architecture, allowed the implementation of federated learning techniques, and the results of the performed experiments, by using the implemented prototype of our security system, allows to answer this question positively (see Chapter 4).
- *RQ.3 - How can distributed ledgers be efficiently deployed on the edge in IoT systems to guarantee information integrity in IoT systems?*
  - *SG.2 - To develop an efficient, effective, and flexible IoT security platform*: by adopting federated learning techniques, only trained models are exchanged between the edge, fog, and cloud devices, since data are maintained locally, thus,

the use of distributed ledgers can be restricted to store only the trained models; the developed prototype of our decentralized architecture demonstrated that just one distributed ledger could be used by all the devices, including the edge, fog and cloud ones; avoiding the necessity of deployment of many distributed ledgers at the edge level; moreover, the trained models are encrypted to be exchanged between the nodes, avoiding non-authorized access to them (see Chapter 3).

- *RQ.4 - Is it possible to develop an IoT security platform based on these techniques mentioned above, which can be adopted in different kinds of IoT applications?*
  - *SG.4 - To evaluate the developed IoT security platform and compare its performance with results reported by other similar platforms*: the implemented prototype of DCD-FL, the developed decentralized architecture, allows one to answer this question positively since it can be deployed on any IoT based systems, where there are interconnected nodes, which have some degree of computational and storage power, as argued in Section 4.2.6. Experiments using the N-BaIoT dataset, which provides separated data for each type of IoT device, allowed to show how DCD-FL can handle IoT systems with different types of devices, each one with specific possibilities of attacks (see Section 4.2.5)

6.2 Limitations and Future Work

This Section discusses the limitations of the present research. It suggests areas for future work, pointing out aspects of DCD-FL that could be improved and additional features that could be added.

6.2.1 Problem P1: Constrained resources

Since vulnerable and constrained devices should be protected and supported by any IoT security platform intended to be effective and efficient, deciding when and where each intrusion detection task will be executed is crucial. It is not acceptable nor effective the overhead of resource-constraint devices, but at the same time, they need to be monitored and protected in the security platform.

Thus, techniques and algorithms must be developed to estimate the necessary resources for training and inference tasks. Another important aspect related to this issue is the selection of participant nodes in the model training rounds. It will be necessary to define algorithms to perform load balance between cloud/fog/edge nodes quickly and efficiently

and determine the best routing between the participant nodes, minimizing communication costs.

To cope with these above issues, results from real-time scheduling theory (various types of schedulers, feasibility analysis techniques) may be helpful in how to estimate if available computational and memory resources are enough to execute a given task and how to estimate execution time for computation and memory access (including cache effects) for allocated tasks [120].

Minimizing communication costs is also related to the massive amount of generated data by IoT devices that should not be transmitted to the cloud; instead, intrusion prediction should be performed near where the data is produced. Considering the significant heterogeneity of IoT devices, scheduling/allocation algorithms must consider these aspects when deciding when and where to execute model training and intrusion prediction.

Moreover, given that in many IoT-based systems, due to their mobility and scalability peculiarities, nodes may become unavailable and new nodes appear in the system, a dynamic re-scheduling based on currently available resources at each time point must be considered, as well as mechanisms to dynamically reconfigure clusters of nodes to adjust them for the newly defined scheduling.

In this aspect, results from Distributed Systems research area [113] may help; for instance, the middleware concept applied to developing security platforms as a service, accessed by secure API, can be a good starting point to deal with these problems.

6.2.2 Problem P2: Privacy Risks

Since data privacy-preserving is a big issue, mainly due to the recent regulations (in Europe, GDPR [116], in Brazil, LGPD), intended to protect user's privacy and provide data security, FL offers a significant advantage since model training is not dependent on access to the device data, which is only available locally at each client device [56] Moreover, many security risks are minimized since device data is not transmitted through the network.

However, since an initial global model must be shared with the client devices and further model updates should be transmitted to aggregator servers, some malicious client devices may capture the network traffic (man-in-the-middle attack), threatening data privacy and even make changes to the models, intending to influence on the final results (poison attack). Thus, some techniques should be adopted to cope with these threats.

Differential privacy [118] adds noise to the data or uses generalization methods so that a third party cannot identify who sent the model updates, making the data impossible to be restored. However, differential privacy usually demands a high computational cost, which is incompatible with many IoT devices.

Other approaches to guarantee data privacy-preserving in FL and avoid poison attacks on the transmitted models between the nodes in an FL-based system use distributed ledger technologies.

6.2.3 Problem P3: Deep Learning Overhead

Since deep learning algorithms have a high demand for computational and storage resources and usually IoT devices have limited available energy, techniques and algorithms must be defined to improve power consumption efficiency by all kinds of cloud/fog/edge nodes. And this definition involves estimating available energy at each node at each time point to identify if tasks will be finished in the node or if a re-scheduling will be necessary, for example.

In the Design Automation for embedded systems research area, the present Author has some published work [81] [80] [79], there are many techniques and algorithms already developed to implement CAD tools for automatized synthesis and verification of integrated circuits that can be explored to the solutions for the above issues, as future work.

6.2.4 Problem P4: Blockchain-based Access Control Overhead

Energy efficiency in the blockchain is mainly related to consensus algorithm execution. PoW and PoS have a high energy consumption [18], being utterly incompatible with the energy levels which are available in the devices of IoT-based systems. However, several consensus mechanisms and procedures could be adapted to decrease energy waste.

There are many other alternatives to implement consensus as, for example, Cardano's protocol [32], which provides a very effective consensus algorithm and also allows fast transaction processing.

6.3 Concluding Remarks

This work contributes to the research for solutions to significant IoT security issues. The thesis introduced some main concepts related to intrusion detection in IoT-based systems and promising technologies that have been shown to cope with many severe problems in developing Intrusion Detection Systems: Federated Learning and Distributed Ledgers.

Decentralized, federated machine learning in a data privacy-preserving manner, using deep learning techniques to dynamically learn and adapt itself to new kinds of intru-

sions, combined with distributed ledgers to minimize the security risks associated with not guaranteeing integrity, confidentiality, and availability of IoT-based systems, is a promising solution to the IDS for IoT implementation. However, there are yet many open problems. The thesis indicated some of them and suggested possible research paths that one interested in IoT-based systems can consider.

REFERENCES

[1] Abadi, F. A.; Ellul, J.; Azzopardi, G. “The Blockchain of Things, Beyond Bitcoin: A Systematic Review”. In: International Conference on Internet of Things (iThings) and Green Computing and Communications (GreenCom) and Cyber, Physical and Social Computing (CPSCom) and Smart Data (SmartData), 2018, pp. 1666–1672.

[2] Abdel-Basset, M.; Moustafa, N.; Hawash, H.; Razzak, I.; Sallam, K. M.; Elkomy, O. M. “Federated Intrusion Detection in Blockchain-Based Smart Transportation Systems”, *IEEE Transactions on Intelligent Transportation Systems*, vol. PP–99, Jun 2021, pp. 1–15.

[3] Aceto, G.; Ciuonzo, D.; Montieri, A.; Pescapé, A. “Mobile Encrypted Traffic Classification Using Deep Learning: Experimental Evaluation, Lessons Learned, and Challenges”, *IEEE Transactions on Network and Service Management*, vol. 16–2, Jun 2019, pp. 445–458.

[4] Adeleke, O. “Intrusion Detection: Issues, Problems and Solutions”. In: 3rd International Conference on Information and Computer Technologies, ICICT, 2020, pp. 397–402.

[5] Aledhari, M.; Razzak, R.; Parizi, R. M.; Saeed, F. “Federated Learning: A Survey on Enabling Technologies, Protocols, and Applications”, *IEEE Access*, vol. 8, Aug 2020, pp. 140699–140725.

[6] Almiani, M.; AbuGhazleh, A.; Al-Rahayfeh, A.; Atiewi, S.; Razaque, A. “Deep Recurrent Neural Network for IoT Intrusion Detection system”, *Simulation Modelling Practice and Theory*, vol. 101, Dec 2020, pp. 102031.

[7] Alphand, O.; Amoretti, M.; Claeys, T.; Dall’Asta, S.; Duda, A.; Ferrari, G.; Rousseau, F.; Tourancheau, B.; Veltri, L.; Zanichelli, F. “IoTChain: A Blockchain Security Architecture for the Internet of Things”. In: IEEE Wireless Communications and Networking Conference (WCNC), 2018, pp. 1–6.

[8] Alsaedi, A.; Moustafa, N.; Tari, Z.; Mahmood, A.; Anwar, A. “TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems”, *IEEE Access*, vol. 8–3, Sep 2020, pp. 165130–165150.

[9] Ammar, A. “A Decision Tree Classifier for Intrusion Detection Priority Tagging”, *Journal of Computer and Communications*, vol. 03–04, Mar 2015, pp. 52–58.

[10] Ammar, M.; Russello, G.; Crispo, B. “Internet of Things: A Survey on the Security of IoT Frameworks”, *Journal of Information Security and Applications*, vol. 38–2, Feb 2018, pp. 8–27.

[11] Andersen, M. P.; Kumar, S.; AbdelBaky, M.; Fierro, G.; Kolb, J.; Kim, H.-S.; Culler, D. E.; Popa, R. A. “WAVE: A Decentralized Authorization System for IoT via Blockchain Smart Contracts”, Technical Report UCB/EECS-2017-234, EECS Department, University of California, Berkeley, University of Berkeley, CA, USA, 2017, 16p.

[12] Banerjee, M.; Lee, J.; Choo, K. K. R. “A Blockchain Future for Internet of Things Security: A Position Paper”, *Digital Communications and Networks*, vol. 4–3, Jan 2018, pp. 149–160.

[13] Benet, J. “IPFS - Content Addressed, Versioned, P2P File System”, *arXiv*, vol. 1–1, Jul 2014, pp. 1–11, 1407.3561.

[14] Benkhelifa, E.; Welsh, T.; Hamouda, W. “A Critical Review of Practices and Challenges in Intrusion Detection Systems for IoT: Toward Universal and Resilient Systems”, *IEEE Communications Surveys & Tutorials*, vol. 20–4, Nov 2018, pp. 3496–3509.

[15] Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. “Fog Computing and Its Role in the Internet of Things”. In: First Edition of the MCC Workshop on Mobile Cloud Computing, 2012, pp. 13–16.

[16] Booij, T. M.; Chiscop, I.; Meeuwissen, E.; Moustafa, N.; Hartog, F. T. H. d. “ToN_IoT: The Role of Heterogeneity and the Need for Standardization of Features and Attack Types in IoT Network Intrusion Data Sets”, *IEEE Internet of Things Journal*, vol. 9–1, Jun 2022, pp. 485–496.

[17] Borgia, E. “The Internet of Things vision: Key features, applications and open issues”, *Computer Communications*, vol. 54–09, Oct 2014, pp. 1–31.

[18] Casino, F.; Dasaklis, T. K.; Patsakis, C. “A Systematic Literature Review of Blockchain-based Applications: Current Status, Classification and Open issues”, *Telematics and Informatics*, vol. 36–5, Jan 2019, pp. 55–81.

[19] Chaabouni, N.; Mosbah, M.; Zemmari, A.; Sauvignac, C.; Faruki, P. “Network Intrusion Detection for IoT Security Based on Learning Techniques”, *IEEE Communications Surveys & Tutorials*, vol. 21–3, Dec 2019, pp. 2671–2701.

[20] Chou, I.-T.; Su, H.-H.; Hsueh, Y.-L.; Hsueh, C.-W. “BC-Store: A Scalable Design for Blockchain Storage”. In: 2nd International Electronics Communication Conference, 2020, pp. 33–38.

[21] Choudhary, S.; Kesswani, N. “Analysis of KDD-Cup’99, NSL-KDD and UNSW-NB15 Datasets using Deep Learning in IoT”, *Procedia Computer Science*, vol. 167–3, Mar 2020, pp. 1561–1573.

[22] Dannen, C. “Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners”. New York, NY: Apress, 2017, 1st ed., 185p.

[23] Denniss, W.; Bradley, J.; Jones, M. B.; Tschofenig, H. “OAuth 2.0 Device Authorization Grant”. Source: https://www.rfc-editor.org/info/rfc8628, 2023-02-10.

[24] Di Pietro, R.; Salleras, X.; Signorini, M.; Waisbard, E. “A Blockchain-based Trust System for the Internet of Things”. In: 23nd ACM on Symposium on Access Control Models and Technologies, 2018, pp. 77–83.

[25] Ding, S.; Cao, J.; Li, C.; Fan, K.; Li, H. “A Novel Attribute-Based Access Control Scheme Using Blockchain for IoT”, *IEEE Access*, vol. 7–10, Apr 2019, pp. 38431–38441.

[26] Diro, A. A.; Chilamkurti, N. “Distributed Attack Detection Scheme using Deep Learning Approach for Internet of Things”, *Future Generation Computer Systems*, vol. 82, May 2018, pp. 761–768.

[27] Dorri, A.; Kanhere, S. S.; Jurdak, R.; Gauravaram, P. “Blockchain for IoT Security and Privacy: The Case Study of a Smart Home”. In: International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), 2017, pp. 618–623.

[28] Dramé-Maigné, S.; Laurent, M.; Castillo, L.; Ganem, H. “Centralized, Distributed, and Everything in between: Reviewing Access Control Solutions for the IoT”, *ACM Computing Surveys*, vol. 54–7, May 2021, pp. 1–34.

[29] Eskandari, M.; Janjua, Z. H.; Vecchio, M.; Antonelli, F. “Passban IDS: An Intelligent Anomaly-Based Intrusion Detection System for IoT Edge Devices”, *IEEE Internet of Things Journal*, vol. 7–8, Aug 2020, pp. 6882–6897.

[30] Ferrag, M. A.; Friha, O.; Hamouda, D.; Maglaras, L.; Janicke, H. “Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralized and Federated Learning”, *IEEE Access*, vol. PP–99, Apr 2022, pp. 1–1.

[31] Ferrag, M. A.; Maglaras, L.; Moschoyiannis, S.; Janicke, H. “Deep Learning for Cyber Security Intrusion Detection: Approaches, datasets, and comparative study”, *Journal of Information Security and Applications*, vol. 50–10, Mar 2020, pp. 102419.

[32] Foundation, C.; EMURGO; IOHK. “Cardano’s Homepage”. Source: https://cardano.org/, 2023-05-12.

[33] Friha, O.; Ferrag, M. A.; Benbouzid, M.; Berghout, T.; Kantarci, B.; Choo, K.-K. R. “2DF-IDS: Decentralized and Differentially Private Federated Learning-based

Intrusion Detection System for Industrial IoT", *Computers & Security*, vol. 127–3, Jan 2023, pp. 103097.

[34] Garcia, S.; Parmisano, A.; Erquiaga, M. J. "IoT-23: A Labeled Dataset With Malicious and Benign IoT Network Traffic", Technical Report, Zenodo, 2020, 20p, last accessed in 05/20/2023.

[35] Ge, M.; Fu, X.; Syed, N.; Baig, Z.; Teo, G.; Robles-Kelly, A. "Deep Learning-based Intrusion Detection for IoT Networks". In: IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC), 2019, pp. 256–265.

[36] Ghimire, B.; Rawat, D. B. "Recent Advances on Federated Learning for Cybersecurity and Cybersecurity for Federated Learning for Internet of Things", *IEEE Internet of Things Journal*, vol. 9–11, Dec 2022, pp. 8229–8249.

[37] Grammatikis, P. I. R.; Sarigiannidis, P. G.; Moscholios, I. D. "Securing the Internet of Things: Challenges, Threats and Solutions", *IEEE Internet of Things*, vol. 5–11, Jan 2019, pp. 41–70.

[38] Han, W.; Gu, Y.; Wang, W.; Zhang, Y.; Yin, Y.; Wang, J.; Zheng, L.-R. "The Design of an Electronic Pedigree System for Food Safety", *Information Systems Frontiers*, vol. 17–2, Aug 2015, pp. 275–287.

[39] Hassan, M. U.; Rehmani, M. H.; Chen, J. "Privacy Preservation in Blockchain based IoT Ssystems: Integration Issues, Prospects, Challenges, and Future Research Directions", *Future Generation Computer Systems*, vol. 97–1, Jan 2019, pp. 512–529.

[40] Heartfield, R.; Loukas, G.; Budimir, S.; Bezemskij, A.; Fontaine, J. R.; Filippoupolitis, A.; Roesch, E. "A Taxonomy of Cyber-Physical Threats and Impact in the Smart Home", *Computers and Security*, vol. 78–1, Jan 2018, pp. 398–428.

[41] Hodo, E.; Bellekens, X.; Hamilton, A.; Dubouilh, P.-L.; Iorkyase, E.; Tachtatzis, C.; Atkinson, R. "Threat Analysis of IoT Networks Using Artificial Neural Network Intrusion Detection System". In: International Symposium on Networks, Computers and Communications (ISNCC), 2016, pp. 1–6, 1704.02286.

[42] Hou, J.; Qu, L.; Shi, W. "A Survey on Internet of Things Security from Data Perspectives", *Computer Networks*, vol. 148–1, Jan 2019, pp. 295–306.

[43] Hussain, F.; Hussain, R.; Hassan, S. A.; Hossain, E. "Machine Learning in IoT Security: Current Solutions and Future Challenges", *IEEE Communications Surveys & Tutorials*, vol. 22–3, Dec 2020, pp. 1686–1721.

[44] Indrasiri, K.; Kuruppu, D. "gRPC: Up and Running: Building Cloud Native Applications with Go and Java for Docker and Kubernetes". New York, NY: O'Reilly Media, 2020, 204p.

[45] Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; D'Oliveira, R. G. L.; Eichner, H.; Rouayheb, S. E.; Evans, D.; Gardner, J.; Garrett, Z.; Gascón, A.; Ghazi, B.; Gibbons, P. B.; Gruteser, M.; Harchaoui, Z.; He, C.; He, L.; Huo, Z.; Hutchinson, B.; Hsu, J.; Jaggi, M.; Javidi, T.; Joshi, G.; Khodak, M.; Konecný, J.; Korolova, A.; Koushanfar, F.; Koyejo, S.; Lepoint, T.; Liu, Y.; Mittal, P.; Mohri, M.; Nock, R.; Özgür, A.; Pagh, R.; Qi, H.; Ramage, D.; Raskar, R.; Raykova, M.; Song, D.; Song, W.; Stich, S. U.; Sun, Z.; Suresh, A. T.; Tramèr, F.; Vepakomma, P.; Wang, J.; Xiong, L.; Xu, Z.; Yang, Q.; Yu, F. X.; Yu, H.; Zhao, S. "Advances and Open Problems in Federated Learning", *Foundations and Trends® in Machine Learning*, vol. 14–1–2, Dec 2021, pp. 1–210.

[46] Kim, J.; Shim, M.; Hong, S.; Shin, Y.; Choi, E. "Intelligent Detection of IoT Botnets Using Machine Learning and Deep Learning", *Applied Sciences*, vol. 10–19, Mar 2020, pp. 7009, results were reported to N-BaIoT for each IoT device. Very interesting to compare with our results.

[47] Kolias, C.; Kambourakis, G.; Stavrou, A.; Voas, J. "DDoS in the IoT: Mirai and Other Botnets", *Computer*, vol. 50–7, Jul 2017, pp. 80–84.

[48] Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. "Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network forensic Analytics: Bot-IoT Dataset", *Future Generation Computer Systems*, vol. 100–5, May 2019, pp. 779–796.

[49] Kouicem, D. E.; Bouabdallah, A.; Lakhlef, H. "Internet of Things Security: A Top-Down Survey", *Computer Networks*, vol. 141–1, Jan 2018, pp. 199–221.

[50] Krebs, B. "Hacked Cameras, DVRs Powered Today's Massive Internet Outage". Source: https://krebsonsecurity.com/2016/10/hacked-cameras-dvrs-powered-todays-massive-internet-outage/, 2023-05-20.

[51] Kumar, A.; Shridhar, M.; Swaminathan, S.; Lim, T. J. "Machine Learning-Based Early Detection of IoT Botnets Using Network-Edge Traffic", *arXiv*, vol. abs/2010.11453–1, Dec 2020, pp. 1–15, 2010.11453.

[52] Kumar, S.; Dutta, K. "Intrusion Detection in Mobile ad hoc Networks: Techniques, Systems, and Future Challenges", *Security and Communication Networks*, vol. 9–14, Sep 2016, pp. 2484–2556.

[53] Kumar, V.; Das, A. K.; Sinha, D. "UIDS: A Unified Intrusion Detection System for IoT Environment", *Evolutionary Intelligence*, vol. 14–1, Mar 2021, pp. 47–59.

[54] Kumari, V. V.; Varma, P. R. K. “A Semi-Supervised Intrusion Detection System Using Active Learning SVM and Fuzzy C-Means Clustering”. In: International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2017, pp. 481–485.

[55] Liu, H.; Zhang, S.; Zhang, P.; Zhou, X.; Shao, X.; Pu, G.; Zhang, Y. “Blockchain and Federated Learning for Collaborative Intrusion Detection in Vehicular Edge Computing”, *IEEE Transactions on Vehicular Technology*, vol. PP–99, Jan 2021, pp. 1–1.

[56] Liu, J.; Huang, J.; Zhou, Y.; Li, X.; Ji, S.; Xiong, H.; Dou, D. “From Distributed Machine Learning to Federated Learning: a Survey”, *Knowledge and Information Systems*, vol. 64–4, Mar 2022, pp. 885–917, 2104.14362.

[57] Lo, S. K.; Lu, Q.; Wang, C.; Paik, H.-Y.; Zhu, L. “A Systematic Literature Review on Federated Machine Learning: From a Software Engineering Perspective”, *ACM Computing Surveys*, vol. 54–5, Feb 2021, pp. 1–39, 2007.11354.

[58] Lo, W. W.; Layeghy, S.; Sarhan, M.; Gallagher, M.; Portmann, M. “E-GraphSAGE: A Graph Neural Network based Intrusion Detection System”, *arXiv*, vol. abs/2103.16329, Jan 2021, pp. 1–18, 2103.16329.

[59] Lu, Y. “The Blockchain: State-Of-The-Art and Research Challenges”, *Journal of Industrial Information Integration*, vol. 15–April, Jan 2019, pp. 80–90.

[60] Maesa, D. D. F.; Mori, P.; Ricci, L. “A Blockchain based Approach for the Definition of Auditable Access Control systems”, *Computers & Security*, vol. 84, Mar 2019, pp. 93–119.

[61] Makhdoom, I.; Abolhasan, M.; Abbas, H.; Ni, W. “Blockchain’s Adoption in IoT: The Challenges, and a Way Forward”, *Journal of Network and Computer Applications*, vol. 125, Jan 2019, pp. 251–279.

[62] Marikyan, D.; Papagiannidis, S.; Alamanos, E. “A Systematic Review of the Smart Home Literature: A User Perspective”, *Technological Forecasting and Social Change*, vol. 138–11, Jan 2019, pp. 139–154.

[63] Marwedel, P. “Embedded System Design”. Cham, Switzerland: Springer International Publishing, 2018, 3rd ed., 430p.

[64] McGhin, T.; Choo, K. K. R.; Liu, C. Z.; He, D. “Blockchain in Healthcare Applications: Research Challenges and Opportunities”, *Journal of Network and Computer Applications*, vol. 135–September 2018, Jan 2019, pp. 62–75.

[65] McMahan, H. B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B. A. y. “Communication-Efficient Learning of Deep Networks from Decentralized Data”, *arXiv*, vol. abs/1602.05629–1, May 2016, pp. 1–14, 1602.05629.

[66] Meidan, Y.; Bohadana, M.; Mathov, Y.; Mirsky, Y.; Shabtai, A.; Breitenbacher, D.; Elovici, Y. “N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders”, *IEEE Pervasive Computing*, vol. 17–3, Jul 2018, pp. 12–22, 1805.03409.

[67] Mishra, N.; Pandya, S. “Internet of Things Applications, Security Challenges, Attacks, Intrusion Detection, and Future Visions: A Systematic Review”, *IEEE Access*, vol. 9–3, Apr 2021, pp. 59353–59377.

[68] Mishra, S.; Singh, N. K.; Roussea, V. u. “System on Chip Interfaces for Low Power Design”. New York, NY: Morgan Kaufmann, 2015, 520p.

[69] Mitchell, R.; Chen, I.-r. “A Survey of Intrusion Detection Techniques for Cyber-Physical Systems”, *ACM Computing Surveys*, vol. 46–4, Mar 2014, pp. 1–29.

[70] Mohammadi, M.; Rashid, T. A.; Karim, S. H.; Aldalwie, A. H. M.; Tho, Q. T.; Bidaki, M.; Rahmani, A. M.; Hoseinzadeh, M. “A Comprehensive Survey and Taxonomy of the SVM-based Intrusion Detection Systems”, *Journal of Network and Computer Applications*, vol. 178–03, Jan 2021, pp. 102983.

[71] Moin, S.; Karim, A.; Safdar, Z.; Safdar, K.; Ahmed, E.; Imran, M. “Securing IoTs in Distributed Blockchain: Analysis, Requirements and Open issues”, *Future Generation Computer Systems*, vol. 100, Jan 2019, pp. 325–343.

[72] Mothukuri, V.; Khare, P.; Parizi, R. M.; Pouriyeh, S.; Dehghantanha, A.; Srivastava, G. “Federated Learning-based Anomaly Detection for IoT Security Attacks”, *IEEE Internet of Things Journal*, vol. PP–99, Sep 2021, pp. 1–1.

[73] Moustafa, N.; Slay, J. “UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set)”. In: Military Communications and Information Systems Conference (MilCIS), 2015, pp. 1–6.

[74] Moustafa, N.; Slay, J. “The evaluation of Network Anomaly Detection Systems: Statistical Analysis of the UNSW-NB15 Data Set and the Comparison with the KDD99 Data Set”, *Information Security Journal: A Global Perspective*, vol. 25–1-3, Mar 2016, pp. 18–31.

[75] Mukherjee, M.; Matam, R.; Shu, L.; Maglaras, L.; Ferrag, M. A.; Choudhury, N.; Kumar, V. “Security and Privacy in Fog Computing: Challenges”, *IEEE Access*, vol. 5–3, Oct 2017, pp. 19293–19304.

[76] Nakamoto, S. “Bitcoin: A Peer-to-Peer Electronic Cash System”, Technical Report, bitcoin.org, 2008, 9p, available at https://bitcoin.org/bitcoin.pdf.

[77] Nascimento, F. A. M.; Hessel, F. “A Decentralized Federated Learning Architecture for Intrusion Detection in IoT Systems”. In: 36th International Conference on Advanced Information Networking and Applications (AINA), Volume 2, Barolli, L.; Hussain, F.; Enokido, T. (Editors), 2022, pp. 256–268.

[78] Nascimento, F. A. M.; Hessel, F. “Decentralized Federated Learning for Intrusion Detection in IoT-based Systems: A Review.” In: 8th IEEE World Forum on the Internet of Things (WFIoT), 2022, pp. 200–208.

[79] Nascimento, F. A. M.; Oliveira, M. F. S.; Wagner, F. R. “A Model-Driven Engineering Framework for Embedded Systems Design”, *Innovations in Systems and Software Engineering*, vol. 8–1, Nov 2012, pp. 19–33.

[80] Nascimento, F. A. M. d.; Oliveira, M. F. S.; Wagner, F. R.; IEEE; Nascimento, F. A. M.; Oliveira, M. F. S.; Wagner, F. R. “MDE Approach to the Co-Synthesis of Embedded Systems using a MOF-based Internal Design Representation”. In: ICSE Workshop on Model-Based Methodologies for Pervasive and Embedded Software, 2009, pp. 53–60.

[81] Nascimento, F. A. M. d.; Rosenstiel, W. “A Repartitioning and HW/SW Partitioning Algorithm to the Automatic Design Space Exploration in the Co-Synthesis of Embedded Systems”. In: 14th Symposium on Integrated Circuits and Systems Design, 2001, pp. 85.

[82] Nguyen, T. D.; Marchal, S.; Miettinen, M.; Fereidooni, H.; Asokan, N.; Sadeghi, A.-R. “DÏoT: A Federated Self-Learning Anomaly Detection System for IoT”. In: 39th IEEE International Conference on Distributed Computing Systems (ICDCS), 2019, pp. 756–767.

[83] Nguyen, T. D.; Rieger, P.; Yalame, H.; Möllering, H.; Fereidooni, H.; Marchal, S.; Miettinen, M.; Mirhoseini, A.; Sadeghi, A.-R.; Schneider, T.; Zeitouni, S. “FLGUARD: Secure and Private Federated Learning”, *arXiv*, vol. abs/2101.02281–1, Feb 2021, pp. 1–12, 2101.02281.

[84] Nishio, T.; Yonetani, R. “Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge”. In: IEEE International Conference on Communications (ICC), 2019, pp. 1–7, 1804.08333.

[85] Noor, M. b. M.; Hassan, W. H. “Current Research on Internet of Things (IoT) Security: A Survey”, *Computer Networks*, vol. 148–1, Jan 2019, pp. 283–294.

[86] Nord, J. H.; Koohang, A.; Paliszkiewicz, J. “The Internet of Things: Review and Theoretical Framework”, *Expert Systems with Applications*, vol. 133–11, Nov 2019, pp. 97–108.

[87] Novo, O. “Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT”, *IEEE Internet of Things Journal*, vol. 5–2, Jan 2018, pp. 1184–1195.

[88] Ouaddah, A.; Elkalam, A. A.; Ouahman, A. A. “FairAccess: A New Blockchain-based Access Control Framework for the Internet of Things”, *Security and Communication Networks*, vol. 9–18, Feb 2016, pp. 5943–5964.

[89] Ouaddah, A.; Mousannif, H.; Elkalam, A. A.; Ouahman, A. A. “Access Control in the Internet of Things: Big Challenges and New Opportunities”, *Computer Networks*, vol. 112–11, Nov 2017, pp. 237–262.

[90] Pace, P.; Aloi, G.; Gravina, R.; Caliciuri, G.; Fortino, G.; Liotta, A. “An Edge-Based Architecture to Support Efficient Applications for Healthcare Industry 4.0”, *IEEE Transactions on Industrial Informatics*, vol. 15–1, Jan 2018, pp. 481–489.

[91] Paszke, A.; et al.. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”, *arXiv*, vol. 1–1, Jul 2019, pp. 1–18, 1912.01703.

[92] Prabavathy, S.; Sundarakantham, K.; Shalinie, S. M. “Design of Cognitive Fog Computing for Intrusion Detection in Internet of Things”, *Journal of Communications and Networks*, vol. 20–3, Jun 2018, pp. 291–298.

[93] Prajapati, P.; Bhatt, B.; Zalavadiya, G.; Ajwalia, M.; Shah, P. “A Review on Recent Intrusion Detection Systems and Intrusion Prevention Systems in IoT”. In: 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2021, pp. 588–593.

[94] Putra, G. D.; Dedeoglu, V.; Kanhere, S. S.; Jurdak, R. “Trust Management in Decentralized IoT Access Control System”. In: International Conference on Blockchain and Cryptocurrency (ICBC), 2020, pp. 1–9.

[95] Putra, G. D.; Dedeoglu, V.; Kanhere, S. S.; Jurdak, R.; Ignjatovic, A. “Trust-Based Blockchain Authorization for IoT”, *IEEE Transactions on Network and Service Management*, vol. 18–2, Jun 2021, pp. 1646–1658.

[96] Ragothaman, K.; Wang, Y.; Rimal, B.; Lawrence, M. “Access Control for IoT: A Survey of Existing Research, Dynamic Policies and Future Directions”, *Sensors*, vol. 23–4, Feb 2023, pp. 1805.

[97] Rahman, S. A.; Tout, H.; Talhi, C.; Mourad, A. “Internet of Things Intrusion Detection: Centralized, On-Device, or Federated Learning?”, *IEEE Network*, vol. 34–6, Mar 2020, pp. 310–317.

[98] Raponi, S.; Caprolu, M.; Pietro, R. D. “Intrusion Detection at the Network Edge: Solutions, Limitations, and Future Directions”. In: Third International Conference on Edge Computing – EDGE, 2019, pp. 59–75.

[99] Ray, P. P. “A Survey on Internet of Things Architectures”, *Journal of King Saud University - Computer and Information Sciences*, vol. 30–3, Jan 2018, pp. 291–319.

[100] Rescorla, E. “The Transport Layer Security (TLS) Protocol Version 1.3”. Source: https://datatracker.ietf.org/doc/html/rfc8446, 2023-03-12.

[101] Ring, M.; Wunderlich, S.; Scheuring, D.; Landes, D.; Hotho, A. “A Survey of Network-based Intrusion Detection Data Sets”, *Computers & Security*, vol. 86–6, Jun 2019, pp. 147–167, 1903.02460.

[102] Ristic, I. “Bulletproof SSL and TLS: Understanding and Deploying SSL/TLS and PKI to Secure Servers and Web Applications”. New York, NY: Feisty Duck, 2013, 2nd ed., 512p.

[103] Saadat, H.; Aboumadi, A.; Mohamed, A.; Erbad, A.; Guizani, M. “Hierarchical Federated Learning for Collaborative IDS in IoT Applications”. In: 10th Mediterranean Conference on Embedded Computing (MECO), 2021, pp. 1–6.

[104] Sadique, K. M.; Rahmani, R.; Johannesson, P. “Towards Security on Internet of Things: Applications and Challenges in Technology”, *Procedia Computer Science*, vol. 141–01, Jan 2018, pp. 199–206.

[105] Sarhan, M.; Layeghy, S.; Moustafa, N.; Portmann, M. “NetFlow Datasets for Machine Learning-based Network Intrusion Detection Systems”. In: 10th EAI International Conference, BDTA, and 13th EAI International Conference on Wireless Internet, WiCON, 2021, pp. 117–135, 2011.09144.

[106] Sarhan, M.; Lo, W. W.; Layeghy, S.; Portmann, M. “HBFL: A Hierarchical Blockchain-based Federated Learning Framework for Collaborative IoT Intrusion Detection”, *Computers and Electrical Engineering*, vol. 103–10, Sep 2022, pp. 108379.

[107] Sfar, A. R.; Natalizio, E.; Challal, Y.; Chtourou, Z. “A Roadmap for Security Challenges in the Internet of Things”, *Digital Communications and Networks*, vol. 4–2, Jan 2018, pp. 118–137.

[108] Shen, X.; Yu, H.; Buford, J.; Akon, M. “Handbook of Peer-to-Peer Networking”. New York, NY: Springer Publishing Company, Incorporated, 2009, 1st ed., 1400p.

[109] Shiravi, A.; Shiravi, H.; Tavallaee, M.; Ghorbani, A. A. “Toward Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection”, *Computers & Security*, vol. 31–3, Dec 2011, pp. 357–374.

[110] Simonovich, V. “Imperva Blocks Our Largest DDoS L7/Brute Force Attack Ever (Peaking at 292,000 RPS)”. Source: https://www.imperva.com/blog/

imperva-blocks-our-largest-ddos-l7-brute-force-attack-ever-peaking-at-292000-rps/, 2021-06-21.

[111] Sindhu, S. S. S.; Geetha, S.; Kannan, A. “Decision Tree based Light Weight Intrusion Detection using a Wrapper Approach”, *Expert Systems with Applications*, vol. 39–1, Jan 2012, pp. 129–141.

[112] Sittón-Candanedo, I.; Alonso, R. S.; Corchado, J. M.; Rodríguez-González, S.; Casado-Vara, R. “A Review of Edge Computing Reference Architectures and a New Global Edge Proposal”, *Future Generation Computer Systems*, vol. 99–01, Jan 2019, pp. 278–294.

[113] Tanenbaum, A. S.; van Steen, M. “Distributed Systems: Principles and Paradigms”. Upper Saddle River, NJ: Pearson Prentice Hall, 2007, 800p.

[114] Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A. A. “A Detailed Analysis of the KDD CUP 99 Data Set”. In: Symposium on Computational Intelligence for Security and Defense Applications, 2009, pp. 1–6.

[115] Verma, A.; Ranga, V. “Machine Learning Based Intrusion Detection Systems for IoT Applications”, *Wireless Personal Communications*, vol. 111–4, Jan 2020, pp. 2287–2310.

[116] Voigt, P.; Bussche, A. v. d. “The EU General Data Protection Regulation (GDPR), A Practical Guide”. New York, NY: Springer, 2017, 383p.

[117] Wang, X.; Zha, X.; Ni, W.; Liu, R. P.; Guo, Y. J.; Niu, X.; Zheng, K. “Survey on Blockchain for Internet of Things”, *Computer Communications*, vol. 136–1, Jan 2019, pp. 10–29.

[118] Weippl, E.; Katzenbeisser, S.; Kruegel, C.; Myers, A.; Halevi, S.; Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H. B.; Mironov, I.; Talwar, K.; Zhang, L. “Deep Learning with Differential Privacy”. In: ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 308–318, 1607.00133.

[119] Wu, Y.; Lee, W. W.; Xu, Z.; Ni, M. “Large-Scale and Robust Intrusion Detection Model Combining Improved Deep Belief Network With Feature-Weighted SVM”, *IEEE Access*, vol. 8, Jun 2020, pp. 98600–98611.

[120] Xia, Q.; Ye, W.; Tao, Z.; Wu, J.; Li, Q. “A Survey of Federated Learning for Edge Computing: Research Problems and Solutions”, *High-Confidence Computing*, vol. 1–1, Mar 2021, pp. 100008.

[121] Yang, Y.; Chen, X.; Tan, R.; Xiao, Y. “Intelligent IoT for the Digital World”. Hoboken, NJ: John Wiley & Sons, Inc., 2021, 200p.

[122] Yin, C.; Xiong, Z.; Chen, H.; Wang, J.; Cooper, D.; David, B. “A Literature Survey on Smart Cities”, *Science China Information Sciences*, vol. 58–10, Oct 2015, pp. 1–18.

[123] Yu, J. Y.; Kim, Y. G. “Analysis of IoT Platform Security: A Survey”. In: International Conference on Platform Technology and Service, PlatCon, 2019, pp. 1–5.

[124] Zarpelão, B. B.; Miani, R. S.; Kawakani, C. T.; Alvarenga, S. C. d. “A Survey of Intrusion Detection in Internet of Things”, *Journal of Network and Computer Applications*, vol. 84–02, Jan 2017, pp. 25–37.

[125] Zhang, J.; Chen, B.; Zhao, Y.; Cheng, X.; Hu, F. “Data Security and Privacy-Preserving in Edge Computing Paradigm: Survey and Open Issues”, *IEEE Access*, vol. 6–Idc, Jan 2018, pp. 18209–18237.

[126] Zhang, Y.; Kasahara, S.; Shen, Y.; Jiang, X.; Wan, J. “Smart Contract-Based Access Control for the Internet of Things”, *IEEE Internet of Things Journal*, vol. 6–2, Apr 2019, pp. 1594–1605.

[127] Zhou, I.; Makhdoom, I.; Shariati, N.; Raza, M. A.; Keshavarz, R.; Lipman, J.; Abolhasan, M.; Jamalipour, A. “Internet of Things 2.0: Concepts, Applications, and Future Directions”, *IEEE Access*, vol. 9–10, May 2021, pp. 70961–71012.

[128] Zhou, Z.; Chen, X.; Li, E.; Zeng, L.; Luo, K.; Zhang, J. “Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing”, *Proceedings of the IEEE*, vol. 107–8, Jan 2019, pp. 1738–1762.

APPENDIX A – CLASS DIAGRAM

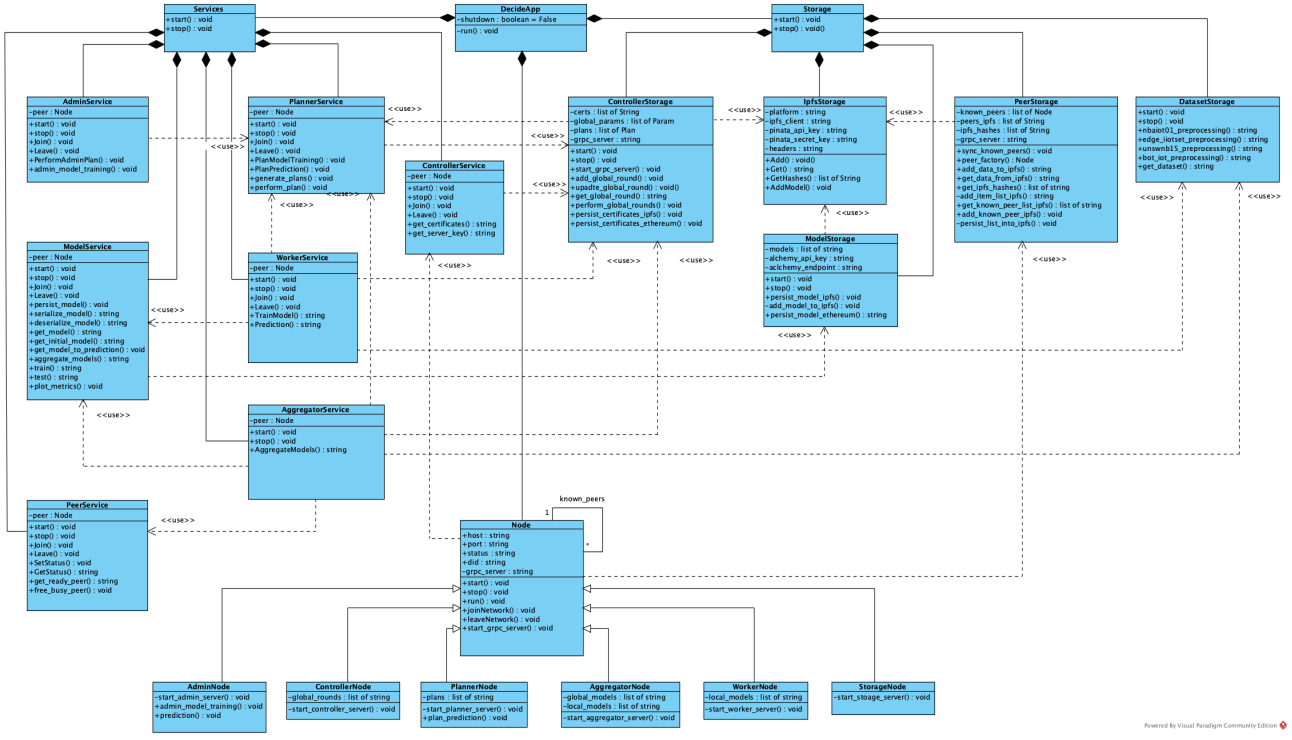

Figure A.1 – Class Diagram: DecideApp and main classes

APPENDIX B – SEQUENCE DIAGRAMS

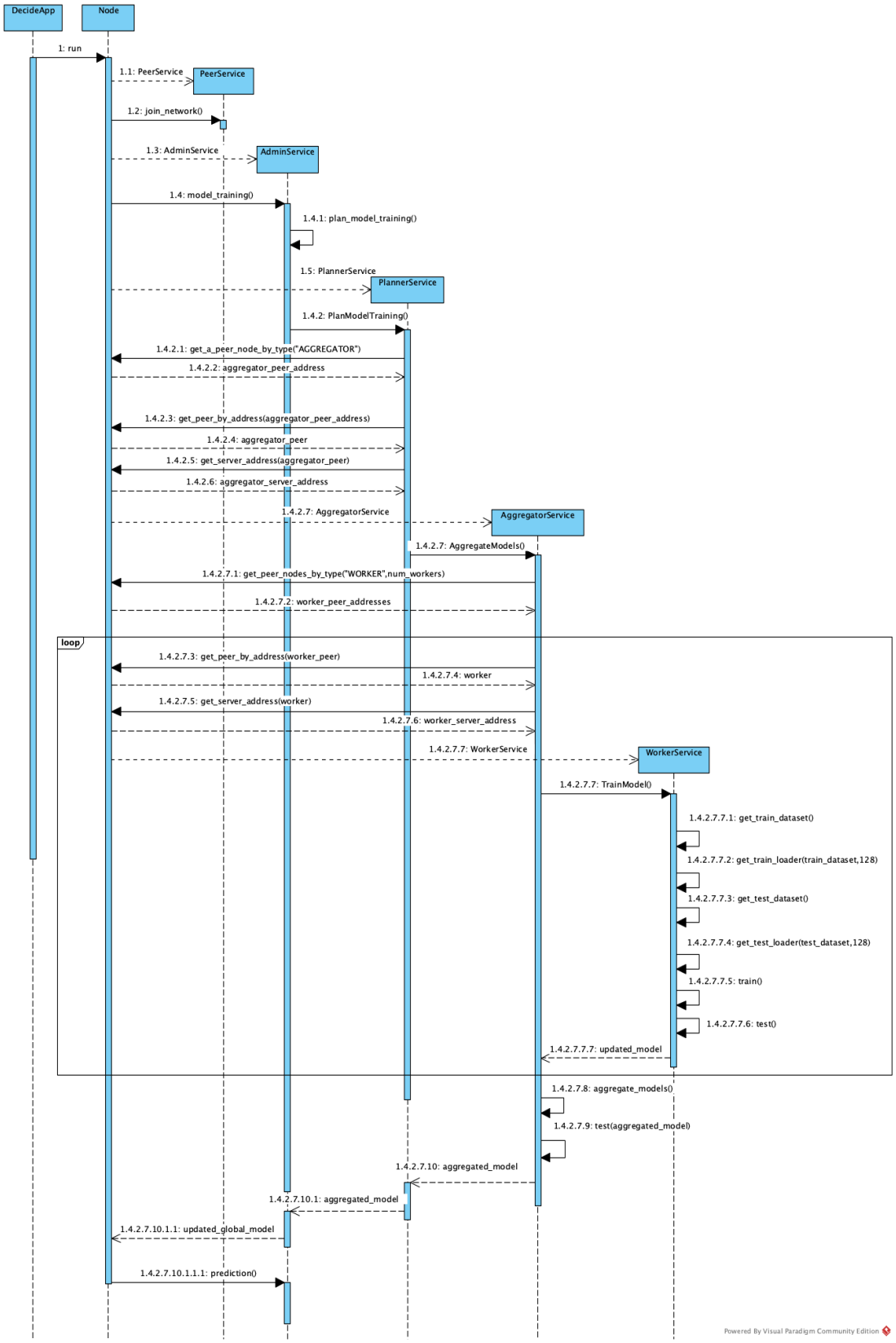

Figure B.1 – Sequence Diagram: Decentralized aggregation in DCD-FL

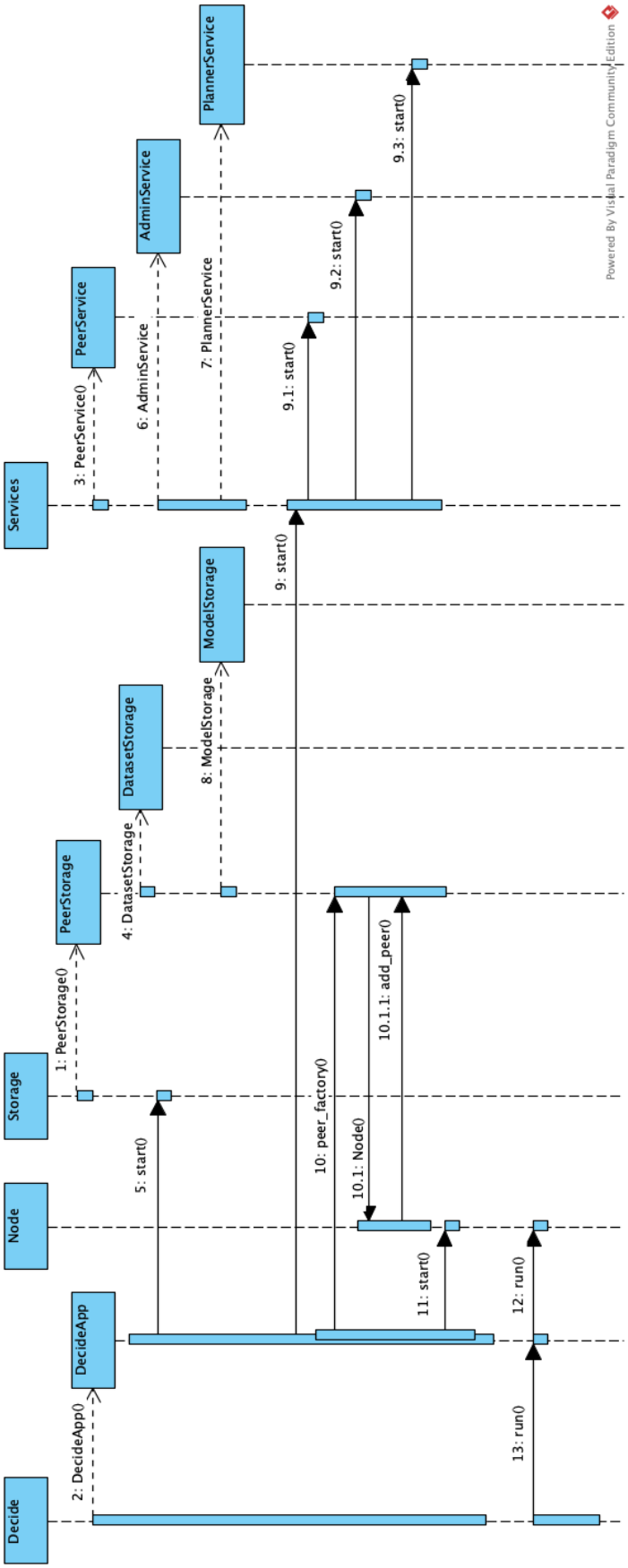

Figure B.2 – Sequence Diagram: Main execution flow in DCD-FL

APPENDIX C – PYTHON SOURCE CODE

```
...
alchemy_endpoint = 'https://polygon-mumbai.g.alchemy.com/v2/' + self.alchemy_api_key
w3 = Web3(Web3.HTTPProvider(alchemy_endpoint))
w3.middleware_onion.inject(geth_poa_middleware, layer=0)

contract_address = self.contract_address
contract = w3.eth.contract(address=contract_address, abi=self.abi)

model_id = model_metadata["model_id"]
model_ipfs_hash = model_metadata["model_ipfs_hash"]

# Set data with pair model_id/model_ipfs_hash
tx_hash = contract.functions.setData(model_id, model_ipfs_hash).transact()
tx_receipt = w3.eth.wait_for_transaction_receipt(tx_hash)

# Get data with key = model_id
data = contract.functions.getData(model_id).call()
...
```

Figure C.1 – Python code to persist and retrieve model metadata in Ethereum

```
...
alchemy_endpoint = 'https://polygon-mumbai.g.alchemy.com/v2/' + self.alchemy_api_key

w3 = Web3(Web3.HTTPProvider(alchemy_endpoint))
w3.middleware_onion.inject(geth_poa_middleware, layer=0)

contract_source_code = self.contract
compiled_sol = compile_standard({ 'language': 'Solidity','sources': {'decidefl.sol': {'content': contract_source_code,},}, 'settings': {'outputSelection': {'*': {'*': ['abi', 'evm.bytecode'],},},},},
})
abi = compiled_sol['contracts']['decidefl.sol']['KeyValueStore']['abi']
bytecode = compiled_sol['contracts']['decidefl.sol']['KeyValueStore']['evm']['bytecode']['object']

from web3.middleware import construct_sign_and_send_raw_middleware
acct = w3.eth.account.from_key(self.wallet_private_key)
w3.middleware_onion.add(construct_sign_and_send_raw_middleware(acct))
w3.eth.default_account = acct.address

gas_estimate = w3.eth.estimate_gas({"data": bytecode})
w3.eth.set_gas_price_strategy(medium_gas_price_strategy)
w3.middleware_onion.add(middleware.time_based_cache_middleware)
w3.middleware_onion.add(middleware.latest_block_based_cache_middleware)
w3.middleware_onion.add(middleware.simple_cache_middleware)
gas_price = w3.eth.generate_gas_price()

transaction = {"from": acct.address,"data": bytecode, "gasPrice": gas_price, "gas": gas_estimate,"chainId": 80001,  "value": w3.to_wei("0", "ether"), "nonce":  w3.eth.get_transaction_count(acct.address)}
signed_transaction = acct.sign_transaction(transaction)
transaction_hash = w3.eth.send_raw_transaction(signed_transaction.rawTransaction)
transaction_receipt = w3.eth.wait_for_transaction_receipt(transaction_hash)
contract_address = transaction_receipt.contractAddress
contract = w3.eth.contract(address=contract_address, abi=abi)
...
```

Figure C.2 – Python code to deploy smart contract in Ethereum

```
...
data_json = json.loads(data)
url = 'https://api.pinata.cloud/pinning/pinFileToIPFS'
headers = {
  'pinata_api_key': self.pinata_api_key,
  'pinata_secret_api_key': self.pinata_secret_api_key
}
metadata = {"name": data_json["key"],
  "description": "Models",
  "model_filename": data_json["model_filename"],
  "aggregated_model": data_json["aggregated_model"],
  "model_type": data_json["model_type"],
  "model_dataset": data_json["model_dataset"],
  "creator": data_json["creator"]
}
data = {'file': (data_json["model_filename"], data_json["model"]),
  'pinataMetadata': metadata
}
payload={'pinataOptions': '{"cidVersion": 1}',
  'pinataMetadata': '{"name": "'+data_json["key"]+'","keyvalues": {"app": "dcd-fl","name": "'+data_json["key"]+'","description": "Models","model_filename": "'+data_json["model_filename"]+'","aggregated_model": "'+data_json["aggregated_model"]+'","model_type":"'+data_json["model_type"]+'","model_dataset": "'+data_json["model_dataset"]+'","creator": "'+data_json["creator"]+'"}}'
}
files=[ ('file',((data_json["model_filename"], data_json["model"])) ]
response = requests.request("POST", url, headers=self.headers, data=payload, files=files)
ipfs_hash = response.json()['IpfsHash']
...
```

Figure C.3 – Python code to persist model in IPFS

```
...
url = ('https://ipfs.io/ipfs/' + str(cid)).replace("'","")
file_content = requests.get(url).content
file_content = file_content.decode("utf-8")
response = ipfs_storage_pb2.GetResponse(data=file_content)
...
```

Figure C.4 – Python code to retrieve model from IPFS

```
1 ...
2 # Generate the CA key
3 ca_key = rsa.generate_private_key(public_exponent=65537,
      key_size=2048,backend=default_backend())
4
5 # Generate the CA certificate
6 ca_name = x509.Name([x509.NameAttribute(NameOID.COMMON_NAME
      , u'sec4iot.com.br'), x509.NameAttribute(NameOID.
      ORGANIZATION_NAME, u'sec4iot'), x509.NameAttribute(
      NameOID.COUNTRY_NAME, u'BR'),])
7 ca_cert = x509.CertificateBuilder().subject_name(ca_name).
      issuer_name(ca_name).public_key(ca_key.public_key()).
      serial_number(x509.random_serial_number().
      not_valid_before(datetime.utcnow().not_valid_after(
      datetime.utcnow() + timedelta(days=365)).add_extension(
      x509.BasicConstraints(ca=True, path_length=None),
      critical=True).sign(ca_key, hashes.SHA256(),
      default_backend())
8
9 # Generate the server key
10 server_key = rsa.generate_private_key(public_exponent
      =65537,key_size=2048,backend=default_backend())
11
12 # Generate the server certificate signing request (CSR)
13 server_name = x509.Name([x509.NameAttribute(NameOID.
      COMMON_NAME, u'sec4iot.com.br'), x509.NameAttribute(
      NameOID.ORGANIZATION_NAME, u'sec4iot'), x509.
      NameAttribute(NameOID.COUNTRY_NAME, u'BR'),])
14 server_csr = x509.CertificateSigningRequestBuilder().
      subject_name(server_name).add_extension(x509.
      BasicConstraints(ca=False, path_length=None), critical=
      True,).add_extension(x509.SubjectAlternativeName([x509.
      DNSName(u"ca.sec4iot.com.br"), x509.DNSName(u"localhost"
      ),x509.DNSName(node_DNSname),x509.IPAddress(ipaddress.
      IPv4Address(u"127.0.0.1")),
15 x509.IPAddress(ipaddress.IPv4Address(node_IpAddress)),x509.
      IPAddress(ipaddress.IPv6Address(u"::1")),]),
16 critical=False,).sign(server_key, hashes.SHA256(),
      default_backend())
17 ...
```

Figure C.5 – Python code to generate TLS/SSL certificates (part 1)

```
...
# Sign the server CSR with the CA
server_cert = x509.CertificateBuilder().subject_name(
    server_csr.subject).issuer_name(ca_cert.subject
).public_key(server_csr.public_key()).serial_number(x509.
    random_serial_number()).not_valid_before(datetime.utcnow
    ()).not_valid_after(datetime.utcnow() + timedelta(days
    =365)).add_extension(x509.BasicConstraints(ca=False,
    path_length=None), critical=True,).add_extension(x509.
    SubjectAlternativeName([x509.DNSName(u"server.sec4iot.
    com.br"),x509.DNSName(node_DNSname), x509.DNSName(u"
    localhost"),x509.IPAddress(ipaddress.IPv4Address(u"
    127.0.0.1")),x509.IPAddress(ipaddress.IPv4Address(
    node_IpAddress)),x509.IPAddress(ipaddress.IPv6Address(u"
    ::1")),]),critical=False,).sign(ca_key, hashes.SHA256(),
     default_backend())
# Save the certificates and keys to files
with open(filename_path+sub_filename+"ca.crt", 'wb') as f:
f.write(ca_cert.public_bytes(serialization.Encoding.PEM))
with open(filename_path+sub_filename+"ca.key", 'wb') as f:
f.write(ca_key.private_bytes(serialization.Encoding.PEM,
    serialization.PrivateFormat.PKCS8,serialization.
    NoEncryption()))
with open(filename_path+sub_filename+"server.crt", 'wb') as
     f:
f.write(server_cert.public_bytes(serialization.Encoding.PEM
    ))
with open(filename_path+sub_filename+"server.key", 'wb') as
     f:
f.write(server_key.private_bytes(serialization.Encoding.PEM
    , serialization.PrivateFormat.PKCS8, serialization.
    NoEncryption()))
...
```

Figure C.6 – Python code to generate TLS/SSL certificates (part 2)

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Pesquisa e Pós-Graduação
Av. Ipiranga, 6681 – Prédio 1 – Térreo
Porto Alegre – RS – Brasil
Fone: (51) 3320-3513
E-mail: propesq@pucrs.br
Site: www.pucrs.br