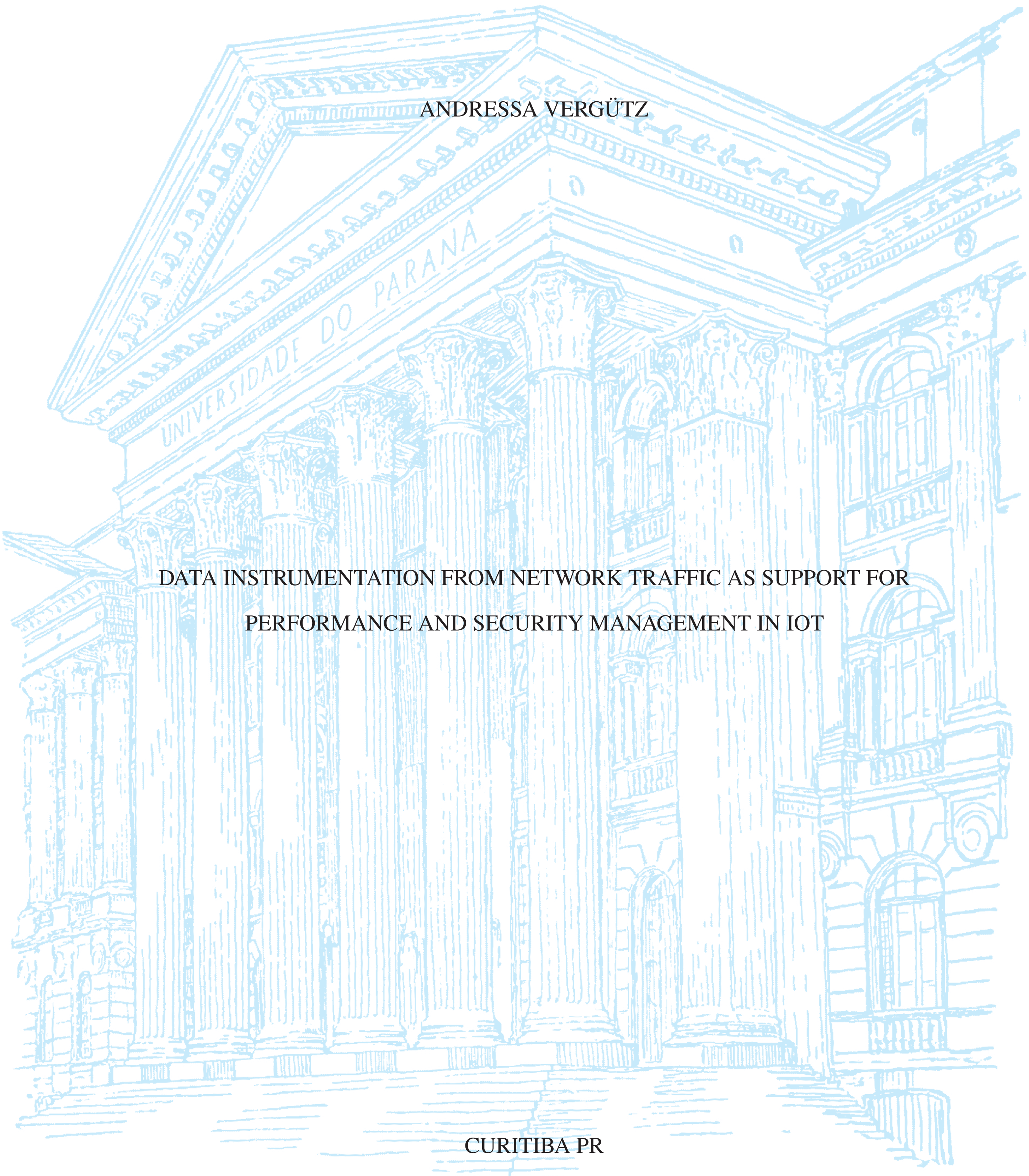
UNIVERSIDADE FEDERAL DO PARANÁ



ANDRESSA VERGÜTZ

DATA INSTRUMENTATION FROM NETWORK TRAFFIC AS SUPPORT FOR PERFORMANCE AND SECURITY MANAGEMENT IN IOT

CURITIBA PR

2018

ANDRESSA VERGÜTZ

DATA INSTRUMENTATION FROM NETWORK TRAFFIC AS SUPPORT FOR PERFORMANCE AND SECURITY MANAGEMENT IN IOT

Tese apresentada como requisito parcial à obtenção do grau de Doutor em Ciência da Computação no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Prof. Dr. Michele Nogueira Lima.

Coorientador: Prof. Dr. Burak Kantarci.

CURITIBA PR

2018

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)
UNIVERSIDADE FEDERAL DO PARANÁ
SISTEMA DE BIBLIOTECAS – BIBLIOTECA CIÊNCIA E TECNOLOGIA

Vergütz, Andressa
Data instrumentation from network traffic as support for performance and security management in IOT. / Andressa Vergütz. – Curitiba, 2022.
1 recurso on-line : PDF.

Tese (Doutorado) – Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática.
Orientadora: Profa. Dra. Michele Nogueira Lima.
Coorientador: Prof. Dr. Burak Kantarci.

1. Informática. 2. Ciência da computação. 3. Internet das coisas. 4. Redes de computação – Medidas de segurança. I. Lima, Michele Nogueira. II. Kantarci, Burak. III. Universidade Federal do Paraná. Programa de Pós-Graduação em Informática. IV. Título.

Bibliotecário: Nilson Carlos Vieira Junior CRB-9/1797

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da tese de Doutorado de **ANDRESSA VERGÜTZ** intitulada: **DATA INSTRUMENTATION FROM NETWORK TRAFFIC AS SUPPORT FOR PERFORMANCE AND SECURITY MANAGEMENT IN IOT**, sob orientação do Prof. Dr. MICHELE NOGUEIRA LIMA, que após terem inquirido a aluna e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.
A outorga do título de doutora está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 21 de Março de 2022.

Assinatura Eletrônica
23/03/2022 09:50:09.0
MICHELE NOGUEIRA LIMA
Presidente da Banca Examinadora

Assinatura Eletrônica
27/04/2022 21:36:18.0
ANELISE MUNARETTO FONSECA
Avaliador Externo (UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ)

Assinatura Eletrônica
24/03/2022 16:55:27.0
WAGNER MACHADO NUNAN ZOLA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
24/03/2022 16:26:22.0
LEOBINO NASCIMENTO SAMPAIO
Avaliador Externo (UNIVERSIDADE FEDERAL DA BAHIA)

This thesis is dedicated to my family.

ACKNOWLEDGEMENTS

First, I would like to thank God for supporting me spiritually throughout my life.

This thesis would not have been possible without the help of so many people who supported me in different ways. I would like to thank my advisor, Prof. Michele Nogueira, for all the advice, dedication, and patience during the orientation. Thanks for believing in my abilities and for providing opportunities for me to grow professionally. Also, I thank my co-advisor, Prof. Burak Kantarci, for the discussion and contribution to the work. I am grateful to the members of my Thesis' committee. Thank you for accepting the invitation. I thank Prof. Guevara Noubir for my visit to Northeastern University and his comments on improving my research. I thank Prof. Aldri Santos for his practical discussions and encouraging my progress. I thank all professors and colleagues from the HealthSense Project for the great work and extraordinary experiences. Also, I thank all my colleagues from the Arquimedes Project for the teamwork and so much learning.

I would like to thank my colleagues at CCSC and NR2 labs. Thank you for the friendship, discussions, coffee, and beers. All discussions, issues, problems, and negative or positive critics greatly impacted my professional and personal life. Particularly, I would like to thank people who work closer to me, especially Nelson and Alemão (Bruno), for the partnership, friendship, and hard work so late at night. We learn a lot together. Thank also to my co-authors, especially Uelinton, Bruna, Ricardo, and Iago, for their support and teamwork. It was great to work and learn with you. Thank Matheus for the friendship and calls. Also, I would like to thank people from Home Sweet Home 220 Vanessa, Natalia, Sayuri, Thais, Lilian, Judi, and Romeu; I hope our friendship can stay for a long time. Thank my friends Carol, Erica, Camila, Vanessa, Schai, and Lari for all the support words, for believing in me and encouraging words.

I am grateful for all the financial support from the Brazilian Government by CAPES and RNP. I would also thank all professors and administrators in the Dinf-UFPR.

Finally, I would like to deepest thank my loved ones. Thanks to my parents, brothers, and Nona, who have always been by my side; For their love, support, encouragement, and dedication. Mom, as you always said to me: "I did this Ph.D. with you", and you really did by supporting me and staying by my side all the time. Thank Dad for always helping me find ways to continue strong and never give up. I would also like to express my deepest gratitude to my love Paulo Roberto for all his support, love, patience, and for believing in me more than I sometimes. Thank you for staying by my side so many times, waiting for me to finish my work. Thank you for encouraging me and showing me that "I can do it". I really appreciate your love and support. I love you all! Thank you so much!

RESUMO

A evolução da inteligência computacional e o avanço da próxima geração de tecnologias sem fio têm impulsionado o avanço da Internet das Coisas (IoT). A IoT compreende um ambiente heterogêneo com diferentes dispositivos gerando uma massiva quantidade de dados desestruturados e desorganizados, por exemplo, desde dispositivos vestíveis à sensores de movimento e lâmpadas inteligentes geram tráfego. Entretanto, a natureza de larga escala da IoT e o aumento significativo do tráfego de rede desestruturado introduziu novos problemas de desempenho, segurança e privacidade. De um lado, aplicações inteligentes (ex., aplicações de saúde inteligente) exigem requisitos como alta confiabilidade e baixa latência. Alcançar tais requisitos não é uma tarefa fácil, dado, nos últimos anos, o significativo aumento no tráfego da rede. Por outro lado, dispositivos IoT contêm dados sensíveis dos usuários e dispositivos; abrindo brechas para adversários explorarem o meio sem fio inativo para capturar informações sobre os usuários e suas atividades. Um efetivo gerenciamento de desempenho e segurança de redes depende de uma eficiente instrumentação dos dados para suportar decisões inteligentes baseadas em dados. No entanto, este processo não é simples sem uma organização e instrumentação do tráfego da rede. Desse modo, neste trabalho advocamos na seguinte hipótese: é possível criar uma metodologia de instrumentação dos dados da rede para suportar diferentes propósitos, como atividade de desempenho e segurança da rede. Portanto, esse trabalho investiga os dados do tráfego da rede, os estrutura e organiza para obter informações valiosas, suportar decisões inteligentes e suportar operações de gerenciamento de rede (ex., desempenho e segurança). Esta tese agrega à literatura com as seguintes contribuições: *(i)* a introdução de uma metodologia eficiente para análise do tráfego da rede IoT como suporte as operações de gerenciamento de rede, chamado FLIPER; *(ii)* um novo método para identificar aplicações e-Health usando características side-channel (chamado MOTIF); *(iii)* a introdução de uma arquitetura adaptável aos requisitos das aplicações de saúde inteligente baseado no fatiamento de recursos da rede (chamado PRIMUS) a fim de auxiliar o gerenciamento da rede; *(iv)* a introdução de um método IoT que revela e mascara características do tráfego da rede (intitulado IoTReGuard), o qual analisa os impactos e destaca os danos de uma instrumentação dos dados da rede da perspectiva de um adversário. A avaliação de desempenho por meio de análises do tráfego da rede e baseado em simulação orientada à traços mostra a viabilidade da instrumentação dos dados e exploração das características como suporte para diferentes atividades de gerenciamento de rede. Uma análise eficiente do tráfego da rede auxilia diversas situações, como tratamentos especiais de acordo com as demandas das aplicações inteligentes, alocação adaptativa dos recursos e o desenvolvimento de mecanismos de defesa. Finalmente, existem oportunidades de pesquisa relacionadas à avaliação das análises de tráfego em cenários experimentais 5G, redução do tempo de classificação, latência, entre outros.

Palavras-chave: IoT, casa inteligente, análise de tráfego da rede, instrumentação dos dados da rede, gerenciamento de rede, desempenho, segurança, confiabilidade, latência, privacidade.

ABSTRACT

The evolution of computational intelligence and the advances of next-generation wireless technologies have boosted the advance of the Internet of Things (IoT). IoT comprises a heterogeneous environment with different devices generating a massive amount of unstructured and unorganized data, *e.g.*, everything from wearables to motion sensors and light bulbs generates traffic. However, the large-scale nature of IoT and the significant increase in unstructured network traffic have introduced new performance, security, and privacy concerns. On the one hand, smart applications (*e.g.*, smart healthcare) demand restrict requirements, such as high-reliability and low-latency. The achievement of such requirements is not a trivial task since, in the last years, given the significant increase in network traffic. On the other hand, IoT devices contain much sensitive information about users and devices; opening lacks to adversaries to exploit the innate wireless medium to capture sensitive information about the user and his/her activities. Effective network performance and security depend on efficient IoT network data instrumentation to support smart data-driven decisions. However, this process is not straightforward without an efficient organization and network traffic instrumentation. Thus, in this work, we advocate on the following thesis: it is possible to create a broader methodology of network data instrumentation to support different purposes, such as network performance and security activities. Hence, this work investigates the network traffic data, structure it, and organize it to obtain valuable information to support data-driven decisions and network management operations (*e.g.*, performance and security). This thesis contributes to the literature as follows: *(i)* the introduction of an Efficient Methodology for IoT Network Traffic Analysis as Support to Network Management Operations, called FLIPER; *(ii)* a novel Method to identify e-Health Applications using Side-channel Features (called MOTIF); *(iii)* the introduction of an Architecture for Adaptable Smart Healthcare Applications Requirements based on Network Resource Slicing (called PRIMUS) to assist network management through network slicing; *(iv)* the introduction of an IoT Method to Reveal and Guard Network Traffic Features (called IoTReGuard) that analyzes the impacts and highlight the damages of a network data instrumentation from the adversary perspective. Performance evaluation by network traffic analyses and trace base simulations on network traffic features show the viability of data instrumentation and feature exploration as support for different network management activities. An efficient network traffic analysis based on data instrumentation assist several situations, such as special treatments according to smart applications demands, adaptive resource allocation, and tailored development of defense mechanisms. Finally, there are research opportunities on evaluating the analysis in an experimental 5G network scenario, reducing the classification time, reducing latency, among others.

Keywords: IoT, smart home, network traffic analysis, network data instrumentation, network management, performance, security, reliability, latency, privacy.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

3GPP	3rd Generation Partnership Project
5G	5th Generation
5GPP	5th Generation Partnership Project
AIFS	Arbitration Inter-frame Spacing
AP	Access Point
Bagg	Bagging
CART	Classification and Regression Trees
CFA	Confirmatory Factor Analysis
CSV	Comma-separated Values
CW	Contention Window
DHCP	Dynamic Host Configuration Protocol
DLP	Dependent Link Padding
DNS	Domain Name System
DPI	Deep Packet Inspection
DSCP	Differentiated Services Code Point
DT	Decision Tree
ECG	Electrocardiogram
EDCA	Enhanced Distributed Channel Access
EFA	Exploratory Factor Analysis
eMMB	Enhanced Mobile Broadband
FA	Factor Analysis
FCAPS	Fault, Configuration, Accounting, Performance, Security
FDA	U.S. Food and Drug Administration Staff
FS	Flow Scenario
FTP	File Transfer Protocol
GPS	Global Positioning System
HIPAA	Health Insurance Portability and Accountability Act
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
HRA	Human Reliability Analysis
IANA	Internet Assigned Numbers Authority
IAT	Inter Arrival Packet Time
IETF	Internet Engineering Task Force
ILP	Independent Link Padding
IoT	Internet of Things

IP	Internet Protocol
IPv4	Internet Protocol Version 4
ISO	International Organization for Standarization
ITU-T	International Telecommunication Union - Telecommunication Standardization Sector
JSON	JavaScript Object Notation
KMO	Kaiser-Meyer-Olkin
k-NN	k-Nearest Neighbors
MAC	Media Access Control
mIoT	Massive Internet of Things
ML	Machine Learning
MSS	Multi-State System
MTBF	Mean Time Between Failures
MTU	Maximum Transmission Unit
NB	Naive Bayes
NF	Network Function
NFV	Network Function Virtualization
NGMN	Next Generation Mobile Network Alliance
NHS	National Health Service
NIST	National Institute of Standards and Technology
NS	Network Slice
NSI	Network Slice Instance
NSMF	Network Slicing Management Function
P2P	Peer to Peer
PC	Principal Component
PCA	Principal Component Analysis
PCAP	Network Packet Capture
PS	Packet Scenario
QoS	Quality of Service
RF	Random Forest
RSVP	Resource Reservation Protocol
SD	Slice Differentiator
SLA	Service Level Agreements
SDN	Software Defined Networks
SST	Service/Slice Type
SVM	Support Vector Machine
TCP	Transport Control Protocol
ToS	Type of Service
Tstat	TCP Statistic and Analysis Tool

XGBoost	Extreme Gradient Boosting
UDP	User Datagram Protocol
URLLC	Ultra-Reliability and Low-Latency Communications
VIM	Virtual Infrastructure Manager
VLAN	Virtual Local Area Network
VM	Virtual Machine
VNF	Virtual Network Functions
VPN	Virtual Private Networks
XML	Extensible Markup Language

LIST OF SYMBOLS

R	Reliability rate
P_d	Packets delivery successfully
t_c	time constraint
P_s	Total number of sent packets
$d_{end-to-end}$	End-to-end latency
d_{proc}	Processing delay
d_{que}	Queuing delay
d_{trans}	Transmission delay
d_{prop}	Propagation delay
D	Set of devices
d_i	Device class
F	Set of features
f_i	Network feature
S	Set of statistical measures
s_i	Statistical measure
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
k	Cross-validation parameter
N	Set of tenants
n_i	Given tenant
App	Set of applications
app_i	Given application
s	Given slice
t	Instant time
v	Network traffic
Q	Set of requirements
q_i	Application requirement
R	Set of available resources
r_i	Available resource
C	Set of controllers
c_i	Controller
u_{num}	User identification
u	User

num	Number of slices
t_d	Threshold duration time
SST	Slice/Service Type
SD	Slice Differentiator
P	Set of performance variables
p_i	Performane variable

CONTENTS

1 INTRODUCTION

The evolution of computational intelligence along with the advances of next-generation wireless technologies, such as the Fifth Generation (5G) cellular network, has boosted the advance of the Internet of Things (IoT) (Abdellatif et al., 2019). IoT collects, quantifies and provides insights about the surrounding environment (Al-Garadi et al., 2020). It integrates billions of smart devices that communicate with one another with minimal human intervention. Such devices are mostly equipped with embedded sensors that collect data from the environment and help users to control them. For instance, smart home users control their home's electronic systems (*e.g.*, smart bulbs, smart locks, sensors) using appropriate smartphone applications and from remote locations. Moreover, sensors have been attached to clothes or wearable accessories to sense the position, motion, and changes in the vital signs of patients (He et al., 2018). Thus, IoT devices offer personal digital assistants, home security, climate control, remote patient monitoring, telemedicine, and physical activity trackers (Khan et al., 2021).

IoT revolutionizes human life by inaugurating the concept of smart scenarios. Typical examples of smart scenarios include smart home, smart hospital, smart city, smart vehicles, among others (Stoyanova et al., 2020). Smart homes involve appliances like smart locks, sensors for temperature and gas. Smart hospitals include wearables and implantable sensors like smartwatches, glasses, and insulin pump sensors. The smart city contains monitoring cameras and air pollution sensors, while smart vehicles comprise drones and applications for industrial and vehicles automation. Each smart scenario has proper reasons and monitoring necessities; hence, each one involve devices and sensors specific to collect and monitor data according to the scenario reasons and demands. In the real-world, there are situations that mix devices from different smart scenarios due to user needs. For instance, a smart home can contain typical IoT devices for home monitoring and also health-related devices to monitor the residents health. Thus, this thesis considers this situation, a smart home with diverse IoT devices.

IoT gains significant attention from the industry and research community since it transforms modern life and business model by improving efficiency, service level, and customer satisfaction (Vidya and Vijaya Kumar, 2016; Neshenko et al., 2019). Forecasts suggest that, by 2025, there will be more than 75 billion IoT connected devices in use, projecting to reach a massive total of 79.4 zettabytes on data volume (Statista, 2016). Forecasts also show a growing market for smart healthcare of 24.1% until 2022 (More, 2019). Nowadays, the most popular IoT solutions and applications comprise human activity automation. In smart homes, there are automatic shades to open and close with the sun and the Google latitude doorbell that rings when a family member's Global Positioning System (GPS) shows they are heading home. Moreover, startups are providing remote patient monitoring. OpenTeleHealth (OpenTeleHealth, 2019) provides a monitoring kit according to the patient's situations, *e.g.*, for pregnant women, the kit has electrodes that she sets on her belly for 20 minutes daily, and then data is automatically sent to the obstetrician. In academia, medical researchers have shown that if detecting fatal diseases (*e.g.*, cancer, Parkinson) in their early stages, it is possible to prevent them (Movassaghi et al., 2014; Health Organization, 2013), which can be provided by continuous remote monitoring (Vergütz et al., 2017). Therefore, IoT is revolutionizing daily scenarios, improving the quality of life.

However, the large-scale nature of IoT and the significant increase in network traffic, given to the connectivity among devices, have introduced new three-fold concerns: performance, security, and privacy (Al-Garadi et al., 2020; Khan et al., 2021). From the network performance perspective, devices and applications have experienced high delays and packet loss rates. Critical applications (*e.g.*, smart healthcare – s-health – applications) are highly sensitive to poor performance and network failures, demanding to restrict requirements (De la Torre Díez et al., 2018). For instance, a severe loss of connectivity can prevent medical professionals to access the necessary information to treat patients. Nonetheless, given the high level of reliability required by s-health, conventional network mechanisms (*e.g.*, priority queues) become unsuitable due to traffic diversity and density (Afolabi et al., 2018; He et al., 2018; Abdellatif et al., 2019). The variety of s-health applications (*e.g.*, telemedicine and critical care monitoring) makes it even more difficult since each application has its own requirements. It is hard to differentiate these requirements and make efficient network decisions since s-health traffic is mixed with a massive amount of general network traffic (*e.g.*, social networks, video streaming, cameras). Then, it is indispensable to effectively analyze the vast amount of network traffic to adapt network resources.

IoT devices contain much sensitive information about users and devices from the security and privacy perspective. Network traffic analysis poses major concerns regarding data privacy and security (Statista, 2016). An adversary inside or near a smart home environment can potentially exploit the innate wireless medium used by these devices to capture sensitive information about the user and his/her activities (Acar et al., 2020). This malicious activity is known as traffic-based attacks or traffic-based side-channel attacks (Neshenko et al., 2019). For example, a motion sensor allows a user to detect any movement in a physical space, but the sensor has only two states (motion and no-motion). If an adversary reveals the current state of the sensor, the adversary also reveals the user's presence at home. Therefore, IoT technologies create new challenges and problems involving security, privacy, and network performance. Regarding the potential IoT environment and its network traffic data, this thesis concentrates the effort on such three-fold concerns: security (network traffic-based attacks), privacy (IoT devices transmitting sensitive information), and performance (achievement of smart applications requirements). To assist in improving these three issues, a crucial previous step is to deal with the unstructured and unorganized network traffic data collected from the IoT environment. The IoT data may efficiently support smart data-driven decisions to improve performance, security, and privacy.

1.1 MOTIVATION AND PROBLEM DESCRIPTION

There is a data explosion, partially, due to the rise of IoT devices from smart scenarios that generate daily a wide variety, volume, and diverse amount of data (Saha, 2020). IoT connects people, places, and things in the physical world. It involves different IoT devices and applications generating data, *e.g.*, everything from wearables to motion sensors and light bulb generate traffic. Thus, it results in a heterogeneous environment with different devices generating a massive amount of unstructured and unorganized data. Unstructured data has no pre-defined format or organization, such as sensor data, differently from structured data that follows a standard order (*e.g.*, names and dates) (Sarker, 2021). In order to be meaningful, unstructured, uncorrelated, and unorganized IoT data requires special handling and analysis to gain insights and knowledge, support smart data-driven decisions, and discover new possibilities.

Extracting knowledge or valuable insights from IoT data boosts smart decision-making (Saha, 2020). Efficient IoT network data analysis provides actionable insights or more profound knowledge about data for network performance and security (Sarker, 2021). However, this process is not straightforward without an efficient organization and network traffic instrumentation. In order to gain actionable insights and useful for network performance and security, two actions are essential: network data instrumentation and feature exploration. Nonetheless, IoT is complex, and it poses challenges in performing data instrumentation efficiently (CDW, 2017). IoT data is often noisy, has misses values or inconsistencies, which increases the challenges. Another challenge is understanding the real-world scenario and their problems with meaningful and complex data. For instance, data-driven decisions on business models, medical treatments, or network security demand different data and features, influencing data instrumentation. It is essential to extract and select features useful for each scenario. Once IoT network traffic data is structured and organized, it assists in intelligent and efficient use, giving valuable and helpful information for network administrators, users, and organizations.

Effective network performance and security depends on efficient IoT network data instrumentation to collect, clean, extract, handle, and explore relevant features and support smart data-driven decisions. For instance, the data instrumentation offers valuable insights on achieving the performance requirements of IoT applications. IoT contains a wide variety of applications such as several health-related applications (*e.g.*, extreme critical care monitoring and remote surgery) (Abdellatif et al., 2019). However, each application has specific requirements (Alliance, 2016). Extreme critical care applications monitor patient health and react immediately, thus requiring very low latency and high-reliable communication. Achieving the specific requirements of each application is not a trivial task. IoT data volume, variety, and veracity pose an enormous burden on conventional networking infrastructures since it increases the competition for medium access, packet loss, and delays. Notably, health application does not support losses and delays (Vergütz et al., 2017). Understanding the network traffic from critical and sensitive applications may assist in resource allocation and network adaptation to transmit exclusive network packet data. An efficient network data instrumentation allow IoT devices characterization and traffic identification. Traffic identification associates the network traffic with the generating application or device, enabling any performance management operations (Valenti et al., 2013).

Similarly, data instrumentation and feature exploration assist data-driven decisions on IoT network security. For instance, they may assist the development of efficient defense mechanisms against traffic-based attacks. Such attacks take benefit from valuable information obtained by IoT network traffic analysis (Huang et al., 2020). Generally, these attacks employ specific network traffic features extracted from data instrumentation and feature exploration (*e.g.*, network packet size) to identify IoT devices and understand users' behaviors. An efficient network data instrumentation and feature exploration give insights into the most meaningful network feature. Meaningful network feature is valuable information since it reveals the traffic feature that most represent an IoT device behavior. Thus, once discover the most relevant feature representing the IoT device behavior, it allows the development of an efficient defense mechanism masking only the key network traffic feature to avoid traffic-based attacks. Therefore, IoT unstructured data demands efficient data instrumentation and feature exploration to give insights and support data-driven network performance and security decisions. An efficient IoT data instrumentation may assist the achievement of specific applications requirements and protect users against traffic-based attacks, boosting the users' adoption of smart scenarios.

1.2 OBJECTIVES

Considering the new issues involving performance, security (network traffic-based attacks), and privacy (IoT devices transmitting sensitive information), and the difficulties related to IoT network data instrumentation, this thesis investigates the following research questions: 1) Is it possible to create a broader methodology for network data instrumentation to support different purposes, such as network performance and security activities? 1.A) If we instrument network data efficiently and apply some intelligence, is it possible to know what application/device generates traffic? 1.B) This knowledge can assist network management and the achievement of smart application requirements? 2) Once an efficient network data instrumentation assists network management, can adversaries also benefit to violate users' privacy and security? 2.*A*) Network data instrumentation can also assist the tailored development of network security mechanisms?

Under the perspective of the aforementioned research questions, the main goal of this thesis consists in instrumenting IoT network data through network data analysis and feature exploration to support other network actions. Such actions evolve the adaptive network performance and security management. Hence, this work introduces the FLIPER methodology, *an e**F**ficient methodo**L**ogy for **I**oT network traffic analysis as su**P**port to n**E**two**R**k management operations*. The FLIPER methodology aims at analyzing the IoT network traffic data, structuring it, and organizing it to obtain valuable information to support smart data-driven decisions on network management operations (*e.g.*, performance and security management).

The FLIPER methodology supports achieving performance requirements of different applications, traffic identification, development of security mechanisms, network management, and others. Thus, the ***M**eth**O**d for identifying eheal**T**h applications using s**I**de-channel **F**eatures* (MOTIF) instantiates the FLIPER methodology to identify applications, such as smart healthcare (s-health) applications in the IoT context. Based on network traffic analysis, MOTIF identifies applications traffic through a statistics-based approach with envisioned Machine Learning (ML) techniques to not violate user privacy since it analyzes network traffic measurements (*e.g.*, packet size). In order to analyze FLIPER supports on network performance management, this thesis introduces *an architecture for ada**P**table sma**R**t healthcare appl**I**cations require**M**ents based on network **R**eso**U**rce **S**licing* (PRIMUS) to assist network management through network slicing. The network slicing approach emerged to slice the same physical network into virtually isolated sub-networks, offering flexibility and fast adaptation. The benefits of network slicing for IoT include adapting network resources to specific requirements and situations, such as s-health reliability and latency (Afolabi et al., 2018). An important prior step consist of data analysis to support the adaptive resource allocation; hence, the FLIPER methodology supports PRIMUS architecture by structuring and obtaining valuable information for network management.

Finally, this work presents an ***IoT** method to **Re**veal and **Guard** Network Traffic Features* (IoTReGuard) that analyzes the impacts and highlight the damages of a network data instrumentation from the adversary perspective. The adversaries attempt to take advantage of ML's strengths during network data instrumentation and feature exploration, using them to make data-driven decisions about what information (in network features) is most relevant to employ in the attacks. Hence, they assist adversaries in performing traffic-based attacks. The IoTReGuard method obfuscates network traffic by well-known methods to improve users' privacy. Hence, the steps of the FLIPER methodology offered support in different ways for the MOTIF method, the PRIMUS architecture, and the IoTReGuard method by offering data-driven decision in traffic identification, network management, and security.

1.3 CONTRIBUTIONS

The main contributions of this thesis are two-fold: *1)* Novel solutions to support network performance management and *2)* Securing the smart home communications against traffic-based side-channel attack. The sub-contributions for contribution 1 are *1.a)* an efficient network data instrumentation methodology for IoT network traffic data, *1.b* an IoT traffic instrumentation-based approach to support smart health applications traffic identification, and *1.c)* network data instrumentation to support the adaptive network resource allocation, and consequently, the smart health applications. The sub-contributions for contribution 2 are *2.a)* a network data instrumentation and exploratory analysis of feature exploration under the adversary perspective and *2.b)* a method to mask IoT network traffic. Next, the summary of these contributions:

1. Novel methodology for network data instrumentation.
    (a) **FLIPER: An Efficient Methodology for IoT Network Traffic Analysis as Support to Network Management Operations.** The FLIPER methodology comprises a set of steps to perform an efficient network traffic analysis and data instrumentation. The motivation to create a new methodology is to offer insights and measures crucial to providing effective and specialized services. We realized that efficiently analyzing data can provide valuable information for obtaining patterns and behaviors, assisting network management, improving network security, allowing specialized services, and others. Hence, an efficient methodology for analyzing network traffic is essential for decision making and knowledge acquisition. Thus, we introduce the FLIPER methodology to perform an efficient network traffic analysis and support several needs (*e.g.*, network management).
    (b) Novel solutions to network support network operations and management.
        i. **MOTIF: A Method For Identifying E-Health Applications Using Side-Channel Features.** Identifying legitimate traffic of a health-related application is a non-trivial task. Researches must deal with an increasing amount of traffic, diversity of applications as well as traffic complexity. MOTIF method takes into account IoT devices network traffic as input to identify different s-health applications. MOTIF method extracts side-channel features from the network traffic, submits them to a distinct type of ML-algorithms, and identifies the applications.
        ii. **PRIMUS: An Architecture for Adaptable Smart Healthcare Applications Requirements based on Network Resource Slicing.** PRIMUS aims to meet the requirements of low-latency and high-reliability in an adaptive way for s-health applications. As network slicing is central to adapt network resources, PRIMUS architecture manages traffic through network slicing. Unlike existing proposals, it supports device and service heterogeneity based on the autonomous knowledge of s-health applications. Emulation results in Mininet-WiFi show the feasibility of meeting the s-health application requirements in virtualized environments.
    (c) Securing smart-home against traffic-based attacks:
        i. **IoTReGuard: An IoT Method to Reveal and Guard Network Traffic Features.** IoTReGuard method explores network traffic features to reveal the key ones and mask smart home devices. It quantifies the potential impact of feature exploration when exploited by adversaries on IoT network traffic. Based on network traffic capture, IoTReGuard method reveals the key network traffic features through a feature exploration employing statistical techniques. Then, it masks network traffic data or specific features by obfuscation techniques to prevent traffic-based attacks.

1.4 STRUCTURE OF THE THESIS

This thesis is structured into six chapters. Chapter 2 presents the background about IoT, smart applications diversity (*e.g.*, smart healthcare), user privacy, network traffic management, network traffic analysis and data instrumentation, network slicing, and traffic-based attacks. Chapter 3 introduces the FLIPER methodology and the MOTIF method. It describes the state-of-the-art related to identifying applications and presents the MOTIF method performance evaluation. Chapter 4 details the existing techniques used to achieve application requirements and introduces the PRIMUS architecture. Chapter 5 discusses the related works on network traffic analysis and defenses against traffic-based attacks. It also presents the IoTReGuard method. Finally, Chapter 6 concludes the thesis as well as presents the future works, publications, and project researches.

2 BACKGROUND

This chapter provides a brief introduction of the Internet of Things (IoT) user privacy, applications diversity on IoT, and the requirements of smart healthcare applications. It also describes performance management of network traffic and security, network traffic analysis and monitoring, feature exploration, and traffic identification. Moreover, to achieve application requirements and protect user privacy, this work advocate for an efficient network traffic analysis and data instrumentation as support for performance and security management. Thus, this chapter also details the network slicing paradigm and the traffic-based attacks on IoT that violate users' privacy. Hence, this chapter follows five sections. Section 2.1 presents a brief introduction of IoT, applications diversity and s-health requirements, as wel as, user's privacy. Section 2.2 describes network traffic management, network traffic analysis, feature exploration, and traffic identification approaches. Section 4.1.3 details the main concepts of network slicing and its key enabling technologies. Section 2.4 presents the definition of network traffic-based side-channel attacks and the main countermeasures against such attacks. Finally, Section 2.5 summarizes the chapter.

2.1 INTERNET OF THINGS

The Internet of Things (IoT) is one of the important technologies of this era that can add automation and smartness in both working and domestic fields (Khan et al., 2021). In fact, the basic idea of IoT is "the presence of a variety of *things* or *objects* – such as tags, sensors, and actuators – which are able to interact with each other and cooperate with their neighbors to reach common goals" (Atzori et al., 2010). Hence, IoT defines a dynamic environment of interrelated devices for seamless connectivity and data transfer (Stoyanova et al., 2020; Khan et al., 2021). The IoT transforms physical objects or "things" to smart by exploring technologies such as embedded devices, communication technologies, sensor networks, and applications. It integrates billions of smart devices that can communicate with one another with minimal human intervention. Such devices are mostly equipped with embedded sensors that collect data from the environment and help users control them. Hence, IoT can collect, quantify and understand the surrounding environment through IoT devices and the collected data (Al-Garadi et al., 2020).

Some typical examples of IoT devices include: i) smart home appliances like smart locks, sensors for temperature, gas or ambient light, ii) wearable devices like smartwatches, glasses or health monitoring system (Stoyanova et al., 2020). Smart home contributes to enhancing the personal life-style by making it easier and more convenient to monitor and operate home appliances and systems remotely (*e.g.*, air conditioner, heating systems, energy consumption meters) (Khan et al., 2021). For example, a smart home can automatically close the windows and lower the blinds of upstairs windows based on the weather forecast. Moreover, a smart home can have health-related devices to collect and monitor residents health. Health-related sensors have been attached to clothes or wearable accessories to sense the position, motion, and changes in vital signs of patients (He et al., 2018). Different smart healthcare applications (s-health) use the IoT data collected from health-related devices to monitor patients remotely.

In the IoT, any object communicates by wireless networks. These objects are equipped with sensors and actuators to collect, transmit, and interact with the environment. The objects act like sensors, actuators, and gateways. Sensor collects information, actuators interacts with the environment, and the gateway acts like a central communication entity that allows the interaction between devices. This organization involves the cooperation between different communication technologies, protocols and services. Communication technologies connect heterogeneous devices together to deliver smart services. Examples of communication protocols used for the IoT are Wi-Fi, Bluetooth, and IEEE 802.15.4 (Al-Fuqaha et al., 2015). Once sensors and actuators have low computational power and transmission range, they employ short-range personal area networks communication technology, such as Bluetooth. Hence, sensors communicate between them and with the gateway by these technologies. In contrast, the gateway has more computational power supporting long-range communication, such as Wi-Fi. Hence, the gateway processes the data and transmits them to an access point (AP), and then to the cloud or to a server for some company (*e.g.*, hospital server) by Internet (Movassaghi et al., 2014).

The smart home can contain health-related devices from the smart healthcare context. Hence, a smart home considering smart healthcare involve the following devices: wearable sensors, implantable sensors, actuators, and gateway. Wearable sensors are embedded in devices (*e.g.*, smartwatch, smart textiles, and smart tattoos) and positioned on the body, while implantable sensors are positioned inside the body (*e.g.*, smart contact lenses, insulin pump, microbubble sensor for intracranial and bladder pressure). Wearable and implantable sensors monitor, collect, and send to the gateway physiological data, such as body temperature and blood pressure. These sensors allow wireless monitoring of a person anywhere, anytime, and with anybody (Movassaghi et al., 2014). The actuator interacts with the user upon receiving data from the sensors. Its role is to provide feedback in the network by acting on sensor data, *e.g.*, pumping the correct dose of medicine into the body in ubiquitous healthcare applications. Finally, the gateway, in s-health also called node coordinator, receive data collected by sensors and transmit to hospital servers, patients devices, healthcare professional devices, or database of the healthcare practitioners.

2.1.1 IoT Application Diversity

Nowadays, there are a wide variety and amount of different applications (Boulos et al., 2014). IoT applications range from from temperature air monitoring, smart home and city security by monitoring cameras, industry automation, quality and reliable soil moisture monitoring, to remote patient monitoring and telemedicine. Each application has specific requirements according to the data collect and context. Smart healthcare applications comprise the most critical and sensitive applications since they monitor and collect daily health-related data (Khan et al., 2021; He et al., 2018). Thus, this thesis concentrates the performance efforts on s-health applications. Smart healthcare (s-Health) uses mobile and electronic technologies to improve the diagnosis of diseases, patients treatment, and enhanced quality of life. The 'smart' in healthcare refers to the use of technologies and data analytics to monitor patients' health remotely and make decisions. Once sensors positioned in or on the human body, collect physiological data, these data must be efficiently captured, and securely transmitted to the healthcare system. Hence, healthcare professionals and machine learning models make decisions based on the data collected.

S-health assists remote patient monitoring, medical diagnosis, physical performance, and many other contexts (Movassaghi et al., 2014). For instance, wearable sensors under control by s-health applications monitor non-critical patients at home rather than in the hospital, reducing strain on hospital resources, such as physicians and beds. Telecare and telemedicine open new opportunities for providing medical care at home, providing better access to healthcare for those living in rural areas and enabling older people to live independently at home (Baker et al., 2017). Thus, there are also a diversity of applications in s-health. S-health encompasses any e-health (electronic/digital health) application. E-Health consists of any Internet application, used in conjunction with other information technologies, focused on providing better conditions to the clinical processes, treatment of patients, and better conditions of cost to the health system. Hence, e-health applications involve from critical care monitoring, telehealth, and telemedicine, to m-health (mobile health), physical activities, and fitness applications. Furthermore, the progress in personalized care impulsion by 5G enhance the quality of experience of surgeons, allowing the remote use of robots from everywhere, called as remote surgery applications. Therefore, the diversity of applications is still increasing, and s-health involves all health-related applications.

Each s-health application has its characteristics, behavior, importance, and type of data analyzed. For instance, telemedicine allows clinical services to leverage information technologies, video imaging, and telecommunication linkages to enable doctors to provide healthcare services at a distance. In contrast to telemedicine, telehealth is an umbrella term that covers telemedicine and a variety of non-physician services, including telenursing, teledermatology, telepathology, and telepharmacy (WHO, 2016). M-health describes applications supported by mobile communication devices, such as wireless patient monitoring devices, smartphones, personal digital assistants, and tablet computers. Hence, m-health is an application used as an accessory to regulated medical device (*e.g.*, an application that turns a smartphone into an ECG machine) (FDA, 2015).

5G Public-Private Partnership (5GPP) and Next Generation Mobile Networks (NGMN) provided situations with specific requirements for s-health (Alliance, 2016; 3GPP, 2018). For instance, people suffering from fatal diseases (*e.g.*, chronic and cardiovascular diseases) are prone to sudden critical situations (*e.g.* heart attack) that need urgent care and actions. Generally, critical care monitoring applications (also called as urgent care or extreme critical application) monitor these type of patients. To effectively assist the quick identification and early diagnosis, urgent care in s-health requires timely, accurate, and reliable communication. Hence, critical care applications require ultra low-latency (1ms) and high-reliability (99.999%). Remote surgery also requires low-latency and high reliability, because surgeries do not support errors, even more errors caused by packet losses or delays. Besides critical applications, there are an enormous variety of s-health applications with more flexible requirements. Table 2.1 present a set of applications with their main performance requirements. Body temperature, and diabetes monitoring consist of daily continuous monitoring applications. When some abnormal situations occurs (*e.g.*, high values of body temperature), they need to alert the physician or patient, supporting 125 ms of latency (Movassaghi et al., 2014; Islam et al., 2015).

Table 2.1: Example of Selection of S-Health Applications and their Requirements (Alliance, 2016; 3GPP, 2018; Pekar et al., 2020)

Application Type	**Requirements**
Body Temperature Monitoring	Low-latency (125 ms) and low packet loss
Remote Monitoring (rural areas)	Ubiquitous Broadband
Critical Care Monitoring	Ultra low-latency (1 ms) and high-reliability
Diabetes Monitoring	Low-latency and data accuracy
Emergency Ambulance	Mobility, latency, and reliability
Smart Medication Management	Reliability and resilience
Remote Surgery	Ultra low-latency and high-reliability
Telemedicine	Low-latency and high data rate

In general, almost all s-health applications require reliable communications, since health patient consist of a critical use case (Alliance, 2016). Applications more flexible, such as fitness and calories counting applications, do not have restrict requirements. In contrast, telemedicine needs high data rate throughput, and low-latency. Emergency ambulance applications is a crucial scenario (*e.g.*, rescue people from accidents) requiring mobility, latency, and reliability. Diabetes monitoring and smart medication encompass treatment reactions based on monitored data. Hence, they require resilience and reliability to inform the actuator about the moment to pump medication in the body. Due to the variety of applications and requirements, it is essential data analysis and instrumentation, traffic identification, and network adaptation to achieve quality of service for s-health applications. Moreover, users can use multiple types of devices and sensors simultaneously. For instance, many ultra-light, low power, waterproof sensors integrated into people's clothing, monitoring different data, such as heart rate, blood pressure, breathing rate, and others. Hence, a key challenge is the overall management of several devices, as well as the applications associated with these devices (3GPP, 2018).

2.1.1.0 Reliability and Latency Requirements

Based on the literature, most frequent requirements of s-health applications encompass reliability and latency (Alliance, 2016; Baker et al., 2017). They are crucial for applications to work as expected. Further, critical care applications, remote surgery, emergency ambulance need both of them since these applications deal with urgent care scenarios, rescuing people. Hence, we focus on reliability and latency requirements.

The definition of reliability is related to dependable systems since the dependability covers several useful requirements, such as reliability, and availability. In this context, the reliability of a component or system is its ability to function correctly over a time interval (Avizienis et al., 2004). In contrast to availability, reliability is defined in terms of a time interval instead of instant time, since availability refers to the probability that the system is operating correctly at any given moment (Birman, 2005). Hence, a highly reliable system works without interruption during a relatively long time If a system goes down for one millisecond every hour, it has an availability of over 99.9999% but is still highly unreliable. Similarly, a system that never crashes but is shut down for two weeks every August has high reliability but only 96% availability (Tanenbaum and Van Steen, 2002). The reliability of a system is measured by Mean Time Between Failures (MTBF) (Jain, 1990). In the context of s-health applications, the reliability rate (R) characterize the reliability of a communication. Thus, the reliability rate is the amount of sent packets successfully delivered (P_d) to the destination within the time constraint (t_c) required by the targeted service, divided by the total number of sent packets (P_s) (Alliance, 2016). The reliability rate is evaluated only when the network is available.

Stringent latency requirements are of utmost importance for providing s-health applications. In network communications, the definition of latency is the time taken to a packet data traverse from source to destination (Kurose and Ross, 2014). A variety of sources contribute to increasing the latency, including network congestion, routers, firewalls, the distance between source and destination, and others. To measure the application's latency is necessary to consider the end-to-end latency. It measures the duration between the transmission of a data packet from the application layer at the source node and the successful reception at the application layer at the destination node (Alliance, 2016). Hence, the end-to-end latency ($d_{end-to-end}$) includes the processing delay, queuing delay, transmission delay, and propagation delay. Processing delay (d_{proc}) consists of the time needed to examine the packet header and determine its direction. Queuing delay (d_{que}) is the time the packet stays in the queue (*e.g.*, router queue). Transmission delay (d_{trans}) is the time needed to transmit the data packet to the link. Finally, propagation delay (d_{prop}) consists of the time to propagate the data packet in the link. Thus, end-to-end latency is the sum of these delays, considering the number of routers in the path. Based on the literature, a key issue involves the significant number of applications that require reliability and ultra low-latency, because the current network technologies have lack support for reliability (Birman, 2005). Hence, it is essential to perform network traffic analysis to assist the management and adaptation of the network to achieve requirements (Khan et al., 2021).

2.1.2 User Privacy in IoT

Security and privacy are essential components of IoT. The claim of individuals' privacy is not new (Westin, 1968). Unlike last year's perception, the claim for privacy consists of a complex and broader theme of data protection (Pinheiro et al., 2020). The Federal Constitution of Brazil in art. 5, item X, sought to protect privacy, thus ensuring: the intimacy, private life, honor, and image of people are inviolable, ensuring the right to compensation for material or moral damage resulting from its violation. The consecration of the right to privacy encompasses all the manifestations of the intimate, private, and personal spheres of people (Brazil, 1988). Thus, privacy is the individual's right to exclude the knowledge from third parties that is only relevant to him and his private life (Westin, 1968; Prates et al., 2020). Recently, governments around the world defined effective data protection rules and regulations, such as General Data Protection Regulation (GDPR) and General Personal Data Protection Law (free translation from "Lei Geral de Proteção de Dados Pessoais" (LGPD)). The LGPD, which is not restricted to cyberspace, provides for the processing of personal data, by a natural person or by a legal entity governed by public or private law, in order to protect the fundamental rights of freedom and privacy.

In the technological context, the guarantee of privacy is not only provided by laws and rules of use but also requires efficient technical solutions to offer options to the user in the way of handling the data generated by him (Prates et al., 2020). The main privacy protection techniques rely on cryptography primitives. For example, encrypt data based on a secret key, turning it into a string of uninterpretable characters to anyone who reads it firsthand. Although the concepts are secure in theory, it is possible to break the privacy of users, even with encrypted data, or without having access to the data in its entirety, such as in network traffic-based side-channel attacks (Spreitzer et al., 2018). In general, these attacks employ sniffers to snoop wireless network traffic, probe unintended side-channel leakages (*e.g.*, packet length and headers), and infer personal information that violates the users' privacy (Dyer et al., 2012). In the literature, this attack is also called traffic-analysis attacks and traffic-based attacks (Neshenko et al., 2019; Papadogiannaki and Ioannidis, 2021). We use them as similar throughout the text but mainly using the name: traffic-based side-channel attacks. Network traffic analysis and feature exploration may assist the development of efficient defense mechanisms. Thus, new security management is necessary to protect users' privacy, especially in smart homes with health-related data.

2.2 NETWORK MANAGEMENT

Network management includes implementing, integrating, and coordinating hardware, software to monitor, test, configure, evaluate, and control network resources (Kurose and Ross, 2014). International Organization for Standardization (ISO) defined five key areas for network management, called FCAPS model: Fault, Configuration, Accounting, Performance, and Security Management, as shown in Figure 2.1 (Leinwand and Fang, 1993). Fault management detects and reacts immediately to fault networks (*e.g.*, service interruption on links and hosts). Configuration management allows the administrator to know the network's devices, components, and hardware and software configuration. Accounting management registers and verifies the network resources accessed by users and devices to control privileged access and user quotas. Performance management analyzes and controls performance (*e.g.*, link use, and flow rate) of different network components (*e.g.*, links, routers, and hosts). Finally, security management controls the access to network resources according to well-defined politics and detects attacks by intrusion detection and prevention systems. In general, the FCAPS model is based on network monitoring since it is necessary to monitor network behavior to perform network control tasks.

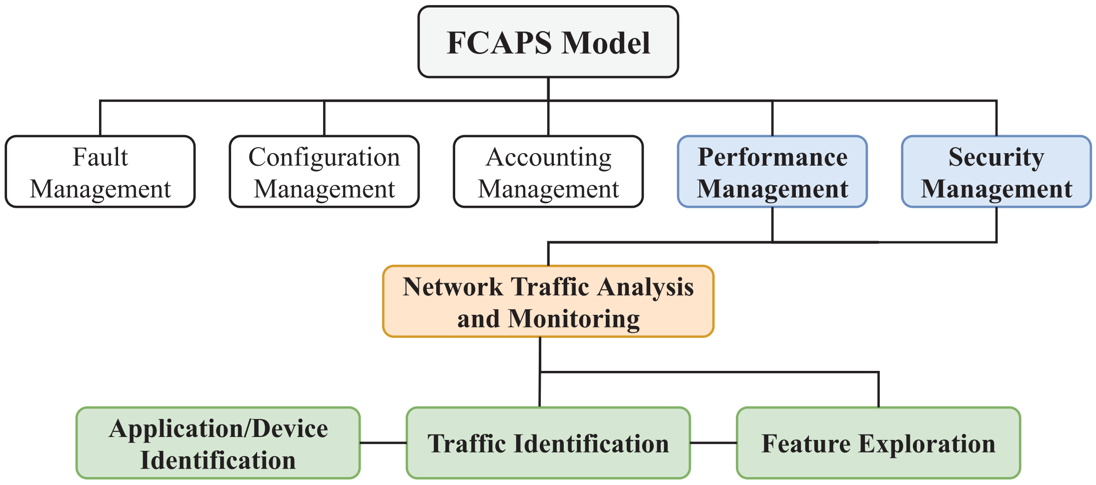


Figure 2.1: FCAPS Model with Focus on Performance and Security Management

Network management can improve applications and service performance in different scenarios (*e.g.*, smart healthcare and home), each one requiring a certain quality level and network resources. Improper use of these resources can degrade Quality of Service (QoS) parameters and, consequently, lead to unsatisfactory behavior, both from the user's point of view (response time) and application (available resources). Through performance management, it is possible to adapt the resources used by applications to their real needs, assisting network operators to take actions regarding the maintenance of performance levels of services offered, such as response time. Performance management uses some key concepts: service and its behavior, resource allocation, and QoS parameters. The services consist of computational and network procedures to perform some tasks, such as file storage and video transmission. Computational and network resources (*e.g.*, bandwidth, energy power, others) are needed for a time to perform services' tasks. Hence, the resources and the time used by the service define its behavior. In general, applications services require resource allocation to achieve QoS requirements, *e.g.*, low-latency. Thus, performance management analyzes parameters such as response time, throughput, and error rate, to evaluate QoS. If the application requirements are not achieved, the service will be degraded, leading even to its non-execution. Therefore, it is essential to monitor the network to learn its behavior, idle resources, and link usage to achieve requirements.

In the same vein, network management assists in protecting the security of applications, data, and services (Avizienis et al., 2004). Security management is a technical and administrative security process that involves security policies, controls, and security monitoring (Kizza et al., 2013). It improves the security of the communication channel through defense mechanisms like cryptography and services such as authentication and access control. Generally, cryptography protects the communications channels from sniffers. Sniffers are programs written for and installed on the communication channels to eavesdrop on network traffic, examining all traffic on selected network segments. Sniffers are easy to write and install but difficult to detect. There are attacks on wireless networks based on network data collected from sniffers (*e.g.*, traffic-based attacks). Hence, security administrators manage the network by analyzing data from all devices to secure network resources and users' data information. Network traffic monitoring allows understanding devices' characteristics and behavior, assisting in developing defense mechanisms.

Therefore, network monitoring is crucial to management operations of performance and security. A significant function of network monitoring is identifying trends and patterns in network traffic and devices. According to these measurements, network operators understand the current state of a network (Lee et al., 2014). It is crucial to investigate and classify the types of network applications that generate user traffic to provide significant traffic insights (*e.g.*, filtering unwanted traffic) (Callado et al., 2009). However, there are issues related to network traffic identification, such as traffic patterns changes, unstructured and unorganized data, encryption, diversity, and amount of traffic and applications. Hence, efficient network traffic analysis and monitoring can assist in traffic identification and network management.

2.2.1 Network Traffic Analysis and Monitoring

Analyzing and classifying network traffic assist various network management activities, such as capacity planning and provisioning, traffic engineering, application performance, anomaly detection, and others (Callado et al., 2009). This analysis takes the network traces of a group of devices as input and gives information about those devices, their users, their applications, or the traffic itself as output. In general, the methods associate traffic packets and/or flows with the application types that generate them. When the focus is on detecting specific applications or devices, literature also uses the term traffic identification (Figure 2.1) (Dainotti et al., 2012). Hence, we use classification and identification terms as the same. The information provided by traffic identification is precious for quite a few networking operations such as prioritizing traffic according to criteria, Service Level Agreements (SLA) verification and addressing network security problems like intrusion detection (Valenti et al., 2013; Mehta and Shah, 2017).

Essential properties of traffic identification consist of granularity, timeliness, and computational cost (Valenti et al., 2013). Granularity distinguish identification methods into two categories: *(i)* coarse-grained algorithms that classify only a large family of protocols (*e.g.*, P2P versus non-P2P), *(ii)* fine-grained algorithms that identify the specific applications (*e.g.*, Skype versus Facebook). The second property, timeliness, refers to the classification time. Early classification quickly identifies applications only by analyzing a few packets. In contrast, late classification takes time to collect traffic features and sometimes needs to wait for traffic flow to finish. Finally, computational cost consists of the power required to inspect traffic and make a classification decision. It is an essential factor when choosing an identification algorithm since some algorithms can be expensive. For instance, in packet processing, the most expensive operation is usually packet memory access, followed by regular expression matching.

Moreover, there are two categories for network traffic monitoring: active or passive (Lee et al., 2014). Active monitoring involves the injection of test traffic, and it usually runs on an end-system. It can directly measure what operators want to observe without waiting for a particular event to happen. In this case, several probe packets are sent continuously across the network to infer its properties (Callado et al., 2009). However, the test traffic needs to resemble real traffic as much as possible. Otherwise, the observed network behavior may not fully match with the real one. In contrast, passive monitoring observes the actual behavior of the network, analyzing network packets and flows. A flow is a set of packets that share the origin and destination addresses, origin and destination ports, and transport protocol. Hence, network monitoring by flow measurements only considers traffic summaries of the flows. When observing packets, most techniques analyze packet headers. Moreover, passive monitoring runs on dedicated devices or is performed by network devices (*e.g.*, routers and switches), suitable for traffic identification and capacity planning since it depicts traffic distribution.

2.2.2 Network Data Instrumentation

In order to analyze and monitor efficiently the network traffic, there is a role prior step: data instrumentation. Data instrumentation is a key term from data analysis, machine learning, and big data to support data-driven systems and decisions (Goodell and Kolodner, 2022; Sarker, 2021). Data instrumentation deals with structured, unstructured, and semi-unstructured (Goodell and Kolodner, 2022; Verma et al., 2020). Structured data follows a standard order, such as names, dates, addresses, geolocation, and others. Unstructured data has no pre-defined format or organization, such as sensor data, emails, blog entries, audio files, videos, web pages, and others. Semi-structured data has elements from both the structured and unstructured data containing certain organizational properties, such as JavaScript Object Notation (JSON) and Extensible Markup Language (XML) documents, databases, and others. Besides these types of data, there is the metadata that represents the data, such as author, file type, file size, creation date and time, last modification date and time, and others (Sarker, 2021). Hence, network traffic fits into unstructured data because it has an expressive amount of data without format or organization. Moreover, it contains network traffic features (*i.e.*, metadata) such as packet size, inter packet time, IP address, and MAC address.

Data instrumentation structures and organizes the network traffic (Goodell and Kolodner, 2022). It is essential on network traffic analysis due to the heterogeneous environment, *e.g.*, IoT is a heterogeneous environment generating heterogeneous and unstructured data. Besides the network traffic may follow a Network Packet Capture (PCAP), it still necessary to perform an organization and characterization of network data since the network traffic contains null values, incomplete data, and also missing values. Hence, the network data instrumentation consists of cleaning by removing unnecessary data, pre-processing and extracting data to achieve well-structured data. The structured data is useful for analysis, data visualization, learn patterns, and others. Network data instrumentation and feature exploration offers valuable information for data-driven decisions. Next subsection details the network feature exploration.

2.2.3 Feature Exploration on Network Traffic

Necessary actions on network traffic analysis consist of feature exploration and characterization since they organize and structure network data (Papadogiannaki and Ioannidis, 2021). They are also an essential component of Machine Learning (ML) because the performance of ML algorithms heavily depends on pre-processing and data transformation. Network traffic feature exploration aims to infer its proprieties, patterns, behaviors, applications, users, or the traffic itself. Even with encrypted network traffic communication, there are ways to analyze the network traffic patterns, *i.e.*, through statistical traffic characteristics. These characteristics involve features transmitted with the network packets, such as the packet size, timestamp, and packet direction. In the security context, such features are also called unintentional side-channel data leaks.

A thorough network traffic feature analysis and exploration collects, cleans, pre-process, extracts, and selects meaningful traffic features (Li et al., 2017a; Conti et al., 2018; Shen et al., 2020). As a first step, sniffers collect the network traffic from a capturing point (*e.g.*, the in-home gateway or access point). Then, the cleaning and pre-processing involve removing unnecessary network traffic, according to the network traffic analysis context. For instance, in IoT device identification is interesting to remove broadcast packets. The feature extraction comprises grouping the network packets per packet or per flow to extract and provide a list of network traffic features. The packet-level extracts features from each network packet, such as packet size and timestamp, the interval between packets, and packet direction. In contrast, the flow-level extraction involves features belonging to the flow 5-tuple (same source and destination IP address, same source, and destination port numbers, and same transport-layer protocol). Flow-level features include flow volume, timestamp, the interval between packets, flow duration, and packet direction. It is also possible to extract statistical primitives (*e.g.*, mean, standard deviation, and variance) from the network traffic features. For instance, extract several statistical primitives from a sequence of packet sizes.

Based on the network traffic extracted, it is often necessary to select a subset of meaningful features (Conti et al., 2018). However, the main challenge is to identify the relevant features from network traffic suitable for specific contexts. Generally, each network context contains specific behavior and patterns. Thus, feature extraction and selection algorithms map the entire feature set onto a lower-dimensional and most miniature necessary set of features to achieve high accuracy. The quality of the feature set is crucial to the performance of an ML algorithm. Using irrelevant or redundant features often leads to negative impacts on the accuracy of most ML algorithms. It can also make the system more computationally expensive, as the amount of information stored and processed (Shen et al., 2020). Hence, selecting a small subset of features is essential to retain essential and useful information about the classes of interest.

In ML, the definition of feature selection consists of selecting only those input dimensions that contain the relevant information solving particular problems (Khalid et al., 2014). Generally, feature selection algorithms follow two methods: filter and wrapper. Filter methods make independent assessments based on general characteristics of data, relying on a particular metric (*e.g.*, error rate) to rate and select the best subset before the learning commences. Wrapper methods evaluate the performance of different subsets using the ML algorithm that is ultimately employed for learning. Wrapper consists of forwarding selection and backward elimination, *i.e.*, it selects features, classifies using these features, and based on the results, eliminate features or select new ones. However, less explored, feature extraction and selection algorithms can identify the most relevant features, such as Principal Component Analysis (PCA) and Factor Analysis (FA). Through a deep feature exploration, these algorithms assist in understanding the feature most represent the entire dataset.

2.2.3.0 Principal Component Analysis

PCA is used in all forms of analysis - from neuroscience to computer graphics and intrusion detection systems - because it is a simple, non-parametric method of extracting relevant information from datasets (Shlens, 2014; Ullah et al., 2021). It allows to summarize and visualize information in a dataset containing observations described by multiple features. Each feature can be considered as a different dimension. Thus, PCA transforms a set of original correlated features into a set of few uncorrelated features, called Principal Components (PC) (Hoang and Nguyen, 2018). These PCs are linear combinations of the original features. The number of PCs is less than or equal to the number of original features, allowing lower complexity. Therefore, PCA identifies patterns in data to highlight their similarities and differences. Since patterns can be hard to find in high dimension data, PCA is a powerful tool for analyzing data by reducing the number of dimensions without much loss of information.

Figure 2.2 shows a example of two-dimensional data with features variation. PCA aims to identify directions (or PCs) along with the maximal variation in the data. For instance, in the first graphic, the data are represented in the X-Y coordinate. The dimension reduction and relevant features are achieved by identifying the principal directions, or PCs, in which the data varies. PCA assumes that the directions with the largest variances are the most important. Therefore, the PC1 axis is the first principal direction along which the samples show the largest variation. The PC2 axis is the second most important direction, and it is orthogonal to the PC1 axis. The second graphic shows the dimensionality reduction by projecting PCs. Precisely, for PCA to work properly, extracting the mean from each data dimension is necessary. The mean subtracted is the average across each dimension. Hence, considering a two-dimensional space, all the x values have $\bar{x}$ (the mean of the x values of all data points) subtracted, and all the y values have $\bar{y}$ subtracted from them (Smith, 2002). Then, since the data is two-dimensional, the covariance matrix will be $2x2$. Afterward, the eigenvectors and eigenvalues for this matrix are calculated, showing useful information about data patterns (*e.g.*, amount of variance retained by each PC). In other words, PCA reduces the data dimensionality to two or three principal components that can be visualized graphically, with minimal loss of information.

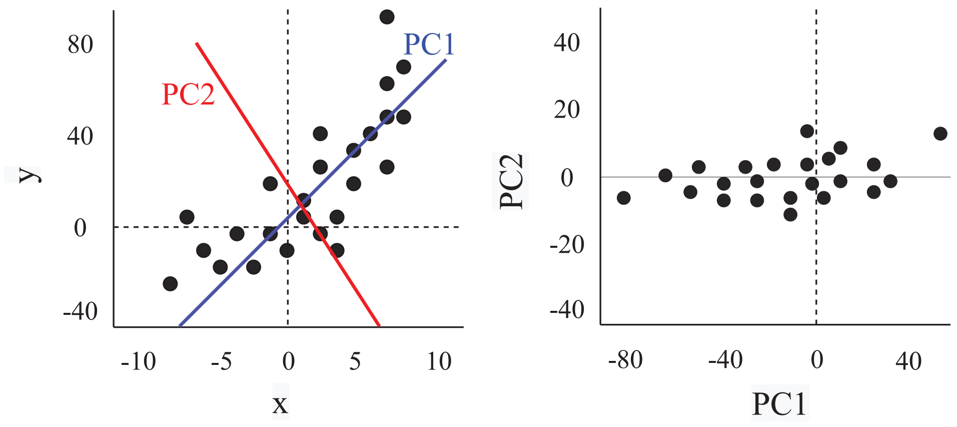


Figure 2.2: Example of PCA

Note that the PCA is beneficial when the features within the dataset are highly correlated. Correlation indicates that there is redundancy in the data. Due to this redundancy, PCA reduces the original variables into a smaller number of new variables (= PCs), explaining most of the variance in the original features. It is possible to ignore the components of lesser significance. By leaving out some components, the final dataset will have fewer dimensions than the original. Precisely, if the original dataset has n dimensions with n PCs, and it is decided to choose only the first p PCs, then the final dataset has only p dimensions.

2.2.3.0 Factor Analysis

There are two basic forms of factor analysis, exploratory and confirmatory. In the confirmatory factor analysis (CFA), the goal is to prove or disprove some hypothesis about the data. FA will confirm, or not, where the latent features are and how much variance they have. In contrast, the exploratory factor analysis (EFA) does not follow any hypothesis. It is an investigatory process to assist in the understanding of data by discovering the association between features. This work follows the most common method used in the literature, the exploratory factor analysis based on PCA (Brown, 2009). This method extracts the maximum variance and puts them into the first factor. After, it removes the variance explained by the first factor and then starts extracting maximum variance for the second factor. This process goes to the last factor.

Factor analysis extracts the maximum common variance from all variables and puts them into a common score (factor). There are key concepts in FA: variance, eigenvalue, and factor score. Variance expresses how much the variable values differ from the average. FA analyzes how the different factors influence the variance among features. Every factor has an influence, but some will explain more variance than others. The amount of variance a factor explains is expressed in an eigenvalue. If a factor has an eigenvalue of 1 or above, it explains more variance than a single observed variable. Factors with eigenvalues less than one account for less variability than a single variable. The factor score describes how strongly a feature from the original data is related to a given factor. Thus, FA performs by examining the pattern of correlations between the observed variables. Features that are highly correlated (either positively or negatively) are likely influenced by the same factors, while relatively uncorrelated ones are likely influenced by different factors (DeCoster, 1998).

Therefore, feature exploration, including feature selection, improve data details by pointing out the most relevant and contribution features. Assist network performance and security management by giving detailed information on the monitored network. It is possible to learn network traffic patterns and device behavior and discover the most significant feature in representing such patterns and behavior based on network traffic analysis. In addition, network traffic analysis assists ML algorithms in the traffic identification task. For instance, traffic identification can employ only the most relevant feature and its statistical primitives.

2.2.4 Traffic Identification Approaches

Traffic identification brings several benefits to network management (Conti et al., 2018). For instance, the knowledge of the applications used by the network's clients can help the administrators tune the network equipment and parameters to deliver QoS services. It can benefit critical care applications monitoring patients' health that need to transmit data more quickly as possible. However, identifying applications or devices also has privacy implications. It is possible to target a high-profile user and discover whether she/he uses privacy-sensitive applications (*e.g.*, health). Moreover, knowing the set of applications installed on a mobile device can reveal sensitive information about the user, such as relationship status and country. Therefore, besides performance methods to achieve application requirements, it is also crucial for new security methods. The identification of devices and applications follow usual approaches.

In general, the traffic identification comprises four main approaches (as shown in Figure 2.3), *i.e.*, those based on *(i) transport-layer port number*, *(ii) inspection of packet payload*, *(iii) flow or host behavior*, and *(iv) statistics* that explores flow and packet statistical measurements. In each of these approaches there are several challenges and limitations to be addressed (Aceto et al., 2017). In the identification of IoT devices and s-health application context, these approaches have vantages and disadvantages. Hence, the following subsections detail each approach.

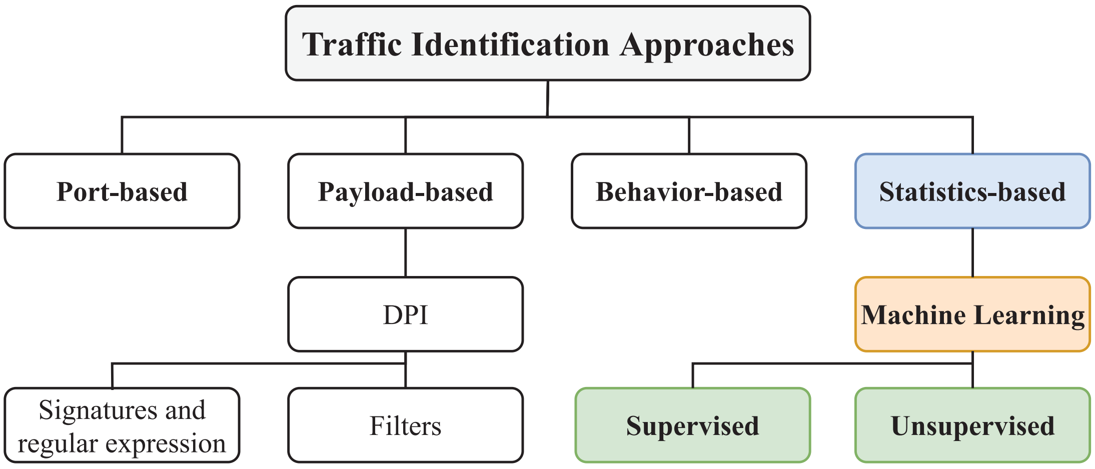


Figure 2.3: Identification/Identification Approaches

2.2.4 Port-Based Approach

The most common traffic identification approach matches port numbers to applications, *i.e.*, an application is associated with a known port number (Callado et al., 2009). This approach extracts the port number value from the packet header and then associates it with the port number of the port-application table. It often associates a well-known port number to given traffic, *e.g.*, web traffic is associated with TCP port 80. Historically, many applications use a well-known port on their local host as a rendezvous point to which other hosts may initiate communication. A classifier or identification algorithm position in the middle of a network need only look for TCP SYN packets (the first step in TCP three-way handshake during session establishment) to know the server-side of a new client-server TCP connection. The application is then inferred by looking up the TCP SYN packet target port number in the Internet Assigned Numbers Authority (IANA) list of registered ports. UDP uses ports similarly (though without packet guaranteed delivery) (Nguyen and Armitage, 2008).

The main advantage of the port-based approach is fast, as it does not apply complex calculations. Its implementation is quite simple, and the knowledge of the classification system is easily extended by adding new application-port pairs to its database. For general services, *e.g.*, DNS, FTP, and e-mail, it works well. However, this approach has some limitations. Firstly, some applications may not have their ports registered with IANA (*e.g.*, peer-to-peer applications such as Napster and Kazaa). Many applications choose to hide their traffic behind other port numbers, *e.g.*, HTTP traffic carried over TCP port 80, in order to get through firewalls or to avoid the traffic identification (Callado et al., 2009). Moreover, nowadays, networks exceedingly carry more traffic that uses dynamically allocated port numbers as needed. For instance, RealVideo streamer allows the dynamic negotiation of the server port used for the data transfer. This server port is negotiated on an initial TCP connection (Nguyen and Armitage, 2008). It is also common for a specific application to use a non-default port number, which fails the port-based approach. Consequently, this approach becomes insufficient in many cases since no specific application can be associated with a dynamically allocated or hidden port number.

2.2.4 Payload-Based Approach

The payload-based approach or Deep Packet Inspection (DPI) inspects the packet payload to identify the traffic of applications and devices (Nguyen and Armitage, 2008). DPI searches for known patterns, keywords or regular expressions. For instance, l7-filter is a packet classifier for Linux that, besides look port numbers, analyzes regular expression matching on the application layer data to determine what protocols are being used (Team, 2006). Moreover, to identify specific applications for particular purposes, the researchers create tables containing all possible applications' signatures (Paglialonga et al., 2017). Generally, DPI has been used in conventional applications such as File Transfer Protocol (FTP), HyperText Transfer Protocol (HTTP), HyperText Transfer Protocol Secure (HTTPS), Domain Name System (DNS), and others (Kim et al., 2008). Additionally, intrusion detection systems use DPI as a preliminary step to identifying network anomalies since it achieves high accuracy. For being extremely accurate, many studies use payload-based algorithms to establish ground-truth for the traces (Shen et al., 2017; Aouini et al., 2018). From ground-truth data, the algorithms can filter the network traffic.

Despite its numerous advantages, DPI has some drawbacks. First, the complexity and computation cost are generally high, as several accesses to packet memory are needed. It must keep up-to-date with extensive knowledge of application protocol semantics and must be powerful enough to perform concurrent analyzes of a potentially large number of flows (Nguyen and Armitage, 2008). Some protocols are designed to be secure (*e.g.*, HTTPS); thus, the data stream is encrypted (Callado et al., 2009). Therefore, DPI can be difficult or impossible when dealing with proprietary protocols or encrypted traffic. Moreover, payload-based algorithms that compare known signatures offline analyze packets with limited length since packets are often recorded with limited length. Packet fragmentation also can add computational complexity. Finally, looking at the packet-payload violates user privacy. HIPAA, FDA, and Brazil government impose rules about data privacy, not being allowed to access private data (FDA, 2015; da República, 2018). Hence, DPI is not suitable for IoT since it contains critical and private information.

2.2.4 Behavior-Based Approach

The behavioral-based approach looks at the whole traffic received by a host or an endpoint in the network (Valenti et al., 2013). It observes the patterns on the traffic generated, *e.g.*, how many hosts are connected, with which transport layer protocol, and how many different ports. Hence, the behavior-based approach identifies the application running on the target host. In general, different applications generate diverse traffic patterns. For instance, a P2P host contacts many different peers, usually using a single host port, whereas a Web server contacts different clients with multiple parallel connections. The characterization of traffic patterns can be at different levels of detail, such as social and functional. Some researchers proposed a dispersion graph to fingerprint applications using the network interactions of hosts (Kim et al., 2008). This approach avoids access to packet payload. Moreover, it provides essential information for the administrator network about patterns. However, it generally focuses on endpoints, does not work well in the network core, and requires large traffic.

2.2.4 Statistics-Based Approach

The statistics-based approach relies on statistical traffic characteristics to identify devices and applications (Nguyen and Armitage, 2008). The network layer traffic contains packet size, flow duration, timestamp, and packet inter-arrival time (Conti et al., 2018). In general, these data are transmitted without encryption techniques enabling the information extraction or computing statistics measurements. Hence, the statistical feature extraction applies statistical primitives (*e.g.*, mean, minimum, standard deviation). For instance, from network traffic features is possible to calculate statistical properties such as distribution of flow duration, mean packet inter-arrival time, and minimum packet size. These properties are unique for some applications and devices. For instance, s-health applications that monitor physiological data yield small packet sizes (*e.g.*, around 50 bytes) and thus generate a small flow volume. Moreover, studies point out that network layer traffic has unique features, such as flow volume and packet size, for certain applications classes, able to differentiate one application from another (Pacheco et al., 2018).

In general, the network traffic features follow two categories: packet-level and flow-level (Conti et al., 2018). Packet-level consists of data related to only one packet, such as packet size and timestamp. In contrast, flow-level encompasses data obtained from several packets that belong to the same flow. Hence, to extract flow-level features, it is necessary to wait for the end of the flow. Applications with a small amount of data, such as critical care monitoring, achieve better identification accuracy with packet-level measurements since it contains more packets samples than flow-level. These applications do not have many packets. Streaming applications with more traffic data available fit well with flow-level measurements. Hence, these measurements extracted from network traffic serve as input for ML techniques (Valenti et al., 2013).

ML techniques are popular approaches to identify and classify patterns in different domains (Pacheco et al., 2018). Its main objective is to give the computers learning capabilities. ML tries to extract knowledge from a set of features or attributes, representing the measurable properties of a process (*e.g.*, measurable properties of network traffic). These techniques have a wide range of applications, including search engines, medical diagnosis, text and handwriting recognition, image screening, marketing, and sales diagnosis, among others (Nguyen and Armitage, 2008). In 1994, ML techniques were first utilized for Internet flow classification in the context of intrusion detection (Frank, 1994), the starting point for much of the work using

ML techniques on the Internet traffic classification. In general, ML takes input in a dataset of instances (also known as examples). An instance refers to an individual, an independent example of the dataset. The values of features characterize each instance. In the traffic identification context, the instances consist of network traffic features from the packet-level or flow-level, including statistical measurements.

In ML, there are two classical types of learning: supervised and unsupervised (or clustering) (Pacheco et al., 2018). Supervised learning algorithms create knowledge structures that support classifying new instances into pre-defined classes, *i.e.*, historical data has to be labeled. The output of the learning process is a classification model constructed by examining and generalizing from the provided instances. There are several supervised algorithms such as Naive Bayes Tree, Bayesian Network, and C4.5 decision tree. In contrast, unsupervised algorithms try to find relationships between the inputs without previous knowledge. These relationships can be similarities, proximity, and statistical relationships, among others. An example of unsupervised algorithms is K-Means. As a derived consequence of the learning process, supervised algorithms are commonly used to perform classification tasks. In contrast, the unsupervised ones are used to cluster inputs to find anomalous or similar behaviors between themselves. Hence, the ML model and the type of learning are associated with the problem to solve (Nguyen and Armitage, 2008).

Machine learning can provide more efficient network management since it assists in network traffic classification, application identification, traffic pattern behavior identification, and others (3GPP, 2018). It assists the security and performance management from FCAPS model. However, there is a need for integrated and automatic services based on machine learning and network management. Hence, the network operator can use ML techniques output to adapt the network according to service requirements and security concerns. The statistics-based approach associated with ML techniques can identify applications and devices and assist in achieving applications requirements and the development of security mechanisms. An emerged paradigm, called network slicing, comprises the autonomous analysis of network traffic and the adaptation of network resources to specific requirements and situations. It offers service customization and isolation on physical network infrastructure, enabling the logical as well as physical separation of network resources (Afolabi et al., 2018).

2.3 NETWORK SLICING PARADIGM

Network slicing is gaining momentum among an ever-growing community of researchers from both academia and industry. It is a fundamental concept of the 5G, and the architecture and specification is under standardization in different organizations, such as 3rd Generation Partnership Project (3GPP) (3GPP, 2018), Internet Engineering Task Force (IETF) (Homma et al., 2019), International Telecommunication Union - Telecommunication Standardization Sector (ITU-T) (ITU-T, 2012), and Next Generation Mobile Network Alliance (NGMN) (Alliance, 2016). Network slicing receives attention due to several factors, such as diversity of services and devices, and it is essential to various types of requirements because it provides separate virtual network based on service requirements (Rost et al., 2017). The definitions and scopes of network slicing vary to some degree from one organization to another. We harmonize the definition of network slicing based on 3GPP, IETF Internet-Draft, ITU-T, and NGMN reports. Hence, network slicing allows to create separate virtual networks in support of services, depending on several requirements, on a common physical infrastructure and provides optimized functions to different market scenarios (*e.g.*, health, vehicle, factory). These logical networks are Network Slices (NS), enabling custom network operations and true service differentiation (Afolabi et al., 2018).

The type of application associated with an NS would determine the resource and service treatment the generated traffic would receive, *i.e.*, NS gives the appropriate resources and traffic treatment to fulfill ultra low-latency requirements of a real-time communication (Afolabi et al., 2018). In general, a resource is a set of attributes or capabilities used to deliver a service (Ordonez-Lucena et al., 2017). A network slice is composed of a collection of resource combined appropriately to meet the service requirements. There are three main classes of resources: network, computing, and functionality (Homma et al., 2019). Network resources encompass bandwidth, forwarding route, communication priority, encryption, and others. The attributes of network resource are latency, jitter, packet loss rate, reliability, and others. Computing resources refer to storage, memory, virtual deployment unit (*e.g.*, virtual machine, container), others. These two are virtualized and managed to provide a set of software functionalities. Finally, functionality resources include network functions on data plane (*e.g.*, security with firewall, DPI), and control plane (*e.g.*, number of registered devices, mobility management).

2.3.1 Network Slice Enabling Technologies

Virtualization technologies are essential for network slicing since have shown tremendous disruptive advantages and opportunities in terms of programmability and flexibility to networking initiatives (Ordonez-Lucena et al., 2017). These technologies consist of Software Defined Networks (SDN) and Network Function Virtualization (NFV) that allow to slice a network into logically isolated network slices, each tailored for a given use case (Afolabi et al., 2018). The objective is to virtualize as many functions as possible to be programmable and configurable. Even more, in the case of slices defined per type of application or device, with knowledge about the services of each slice, the network can be simplified by removing functions that are not necessary. For instance, if a slice is giving access to a static sensor, mobility management can be reduced to minimum. Therefore, virtualization technologies simplifies and facilitates the autonomic management for each specific slice (Richart et al., 2016).

SDN decouples control and data plane, removing control functionality from network devices (Kreutz et al., 2015). These network devices become simple packet forwarding devices (the data plane) that can be programmed. Thus, the network control becomes directly programmable via an open interface (*e.g.*, OpenFlow). This programmability automates network configurations in such a way that network administrators can run SDN applications that help to optimize particular services. An external entity, the SDN controller, accommodates the control logic and provides the essential resources and abstractions to facilitate the programming of forwarding devices. The network is programmable through software applications running on top of the SDN controller. This is a fundamental characteristic of SDN. A set of network resource abstraction, that support control, form a network slice. Hence, the SDN controller manages network slices by applying rules when necessary and following the corresponding network policy. Moreover, a SDN controller can maintain a distinct slice context originating from different sources. This allows to dynamically manage network slices through grouping of slices belonging to the same context and maintaining a global map between the corresponding client context.

SDN boosted the emergence of NFV because the flexibility of separate control and data planes assist on virtualizing network devices. NFV decouples of physical network equipment from the functions that run on them (Mijumbi et al., 2015). It dispatches a network function (NF), such as a firewall, to a provider (cloud data center) as an instance software. It allows the consolidation of many network equipment types onto high volume servers, switches, and storage that can be located in data centers and distributed network nodes. The detachment of software from hardware helps reassign and share the infrastructure resources, thus together, hardware and software, can perform different functions at various times. This helps network operators deploy new network services faster over the same physical platform. Therefore, any NFV-enabled device in the network can have components instantiated with flexibly set up connections. Figure 2.4 presents a NFV architecture with virtualized and physical resources. Hence, a set of Virtual Network Functions (VNFs) implemented in software running on several physical servers composes a given service. The VNFs are relocated and instantiated at different network locations without necessarily requiring the purchase and installation of new hardware.

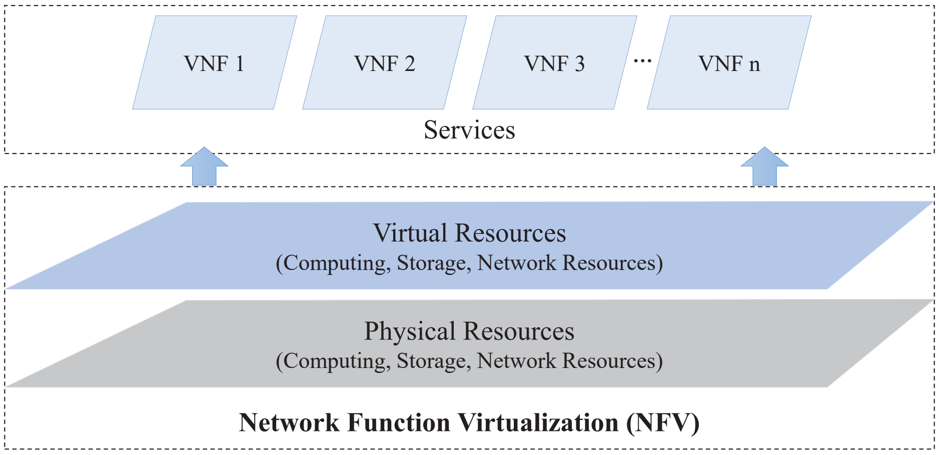

VNF 1
VNF 2
VNF 3
...
VNF n
Services
Virtual Resources
(Computing, Storage, Network Resources)
Physical Resources
(Computing, Storage, Network Resources)
Network Function Virtualization (NFV)

Figure 2.4: Network Function Virtualization

A VNF is an implementation of a network function (NF) that is deployed on virtual resources, such as Virtual Machine (VM). Hence, as NFV decouples of physical network equipment from the functions that run on them to a virtual environment, thus, the VNFs are the NFs of NFVs. NF is a functional block within a network infrastructure that has well defined external interfaces and well-defined functional behavior (Mijumbi et al., 2015). Examples of NFs are elements in a home network, such as a residential gateway, and conventional network functions (*e.g.*, DHCP servers, firewalls). Thus, VMs support VNF. In the NFV, the virtual resources support the virtualization and deployment of NFs that make up the service. However, from the user perspective, the services have the same performance. The services functional and behavioral specifications determine the number, type, and ordering of VNFs. Therefore, the behavior of the service is dependent on the VNFs. Hence, VNFs allow networks to be agile and capable of responding automatically to the needs of the traffic and services running over it. Key enabling technology for VNF, besides NFV, is SDN. However, SDN and NFV are different concepts. They are complementary but increasingly co-dependent.

SDN and NFV address different aspects of a software-driven networking solution (Mijumbi et al., 2015). NFV decouples NFs from hardware elements to VNFs, while SDN focuses on separating the handling of packets and connections from overall network control. In the SDN architecture, virtualization is the allocation of abstract resources to particular clients or applications; in NFV, the goal is to abstract NFs away from dedicated hardware to host them on cloud servers. Moreover, while NFV works on existing networks because it resides on servers and interacts with specific traffic sent to them, SDN requires a new network construct where the data and control planes are separated. There are many studies involving the combination of SDN and NFV to enhanced either of them, *e.g.*, network slicing. Combining SDN and NFV benefit each other, for instance, as an SDN controller can run in a VM, it can be implemented as a part of service. Thus, VNFs realize control and management applications (*e.g.*, load balancing, traffic analysis) used in SDN, benefiting from NFV elasticity. In the same way, SDN assists NFV by offering automation of operations, security, and control policy. Hence, SDN and NFV provide flexibility, programmability, elasticity, and other characteristics required by network slicing.

Network slicing involves different networks and multiple heterogeneous physical resources. However, heterogeneity causes difficulty in accessing and managing the networks. Network operators have to manage network resources with multiple types of equipment and different types of access interfaces. The virtualization technologies resolve such issue. As network virtualization allows the abstraction of physical resources, other systems, applications, or users can access the capabilities of resources by using abstracted interfaces. These interfaces guarantee compatibility for accessing the virtual resources and provide efficient control of the physical resources (Alliance, 2016). Moreover, there are concerns about wireless virtualization since it needs to consider characteristics such as limited resource usage or signal interference. One challenge consists of wireless link virtualization, since wireless link requires the configuration of wireless channel parameters, such as a channel of operation, and transmit power setting. However, there are already researchers with this direction, 5GPPP is studying virtualization (3GPP, 2018), and there is an emulated platform available to wireless virtualization on Wi-Fi (*e.g.*, Mininet-WiFi) (dos Reis Fontes and Rothenberg, 2019). Therefore, network slicing needs virtualization technologies to employ on different networks and physical resources.

2.3.2 Main Concepts

There are important terms related to network slicing concept. There are five main vertical customers envisioned for network slicing: automotive, healthcare, factories, media and entertainment, and energy (3GPP, 2018). In many cases, a vertical customer become the final NS tenant (an end-user that rents and occupies NS). Moreover, according to NGMN, the network slicing process has three main layers: service instance layer, network slice instance layer, and resource layer, as shown in Figure 2.5 (Alliance, 2016; Afolabi et al., 2018). Service instances layer represents the end-user. Network slice instance layer encompasses one or several sub-network instances. The sub-network instance comprises a set of network functions and resources to run these functions in order to support the service requirements. Finally, the resource layer contains the physical and logical resources. A didactic and straightforward example from vertical health sector is: a telemedicine application need to run streaming service. Thus, the application creates a new service instance for streaming, requiring high-bandwidth. The network operator establish an end-to-end NS instance with network characteristics that were required by the service instance. This NS contains one or more sub-networks that can be a network function, or network resources realizing a part of a NS. These sub-networks run on the resources from the resource layer. A sub-network use the resources in an isolated or shared manner following the NS configurations.

Moreover, there are key aspects needed to realize the network slicing concept, such as automation, isolation, customization, elasticity, and end-to-end (3GPP, 2018). These aspects support the network slices creation and management. They allow to (re-) configure the slices as appropriate to provide an end-to-end service. The main aspects are detailed:

- Automation - enables an on-demand configuration of network slicing without the need for fixed contractual agreements. It relies on signaling-based mechanisms, which allow third parties to place a slice creation request indicating besides conventional SLA. It would reflect the desired capacity, latency, jitter, timing information considering the starting and ending time, and duration of a network slice.

- Isolation - allows to operate parallel slices on a common physical network. It assures performance, security and privacy, and management. Each slice is defined to meet

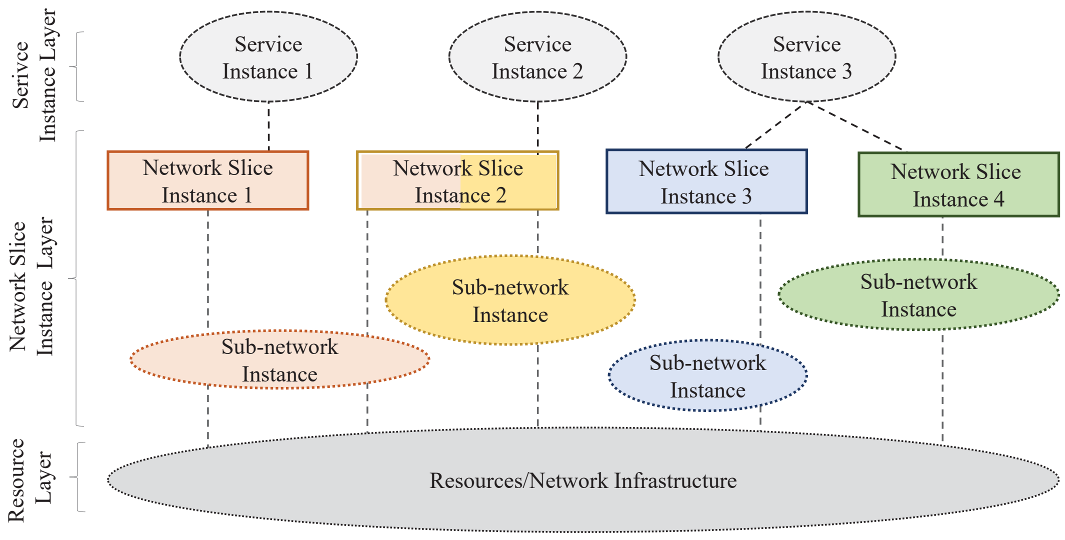


Figure 2.5: NGMN Network Slicing Concept (Alliance, 2016)

particular service requirements. Hence, performance isolation ensures the achievement of service-specific performance requirements on each slice. Moreover, each slice has independent security functions that prevent unauthorized entities from having read or write access to slice-specific configuration/management information. Finally, the management of each slice is independent, as a separate network (Ordonez-Lucena et al., 2017). The notion of isolation involves data and control plane, while its implementation defines the degree of resource separation. Thus, isolation can be deployed (i) by using a different physical resource, (ii) when separating via virtualization means a shared resource, and (iii) through sharing a resource with the guidance of a respective policy that defines the access rights for each tenant.

- Customization - guarantees the efficient use of resource allocated by NS tenant, application, or service. The customization is employed by considering data and control plane separated, data plane with service-tailored network functions, or control plane with programmable policies.

- Elasticity - assures the achievement of requirements to a network slice under varying network conditions, amount of serving users, or user mobility. It reshapes the use of allocated resources. Therefore, elasticity operation reallocates virtual network functions (VNFs) or alters the amount of initially allocated resources modifying physical and virtual network functions (*e.g.*, adding a new VNF, enhancing the network capacity). However, this operation requires inter-slice communications since it may influence the performance of other slices that share the same resource.

- End-to-End - facilitates the service delivered from source to the destination. Thus, it combines resources that belong to different infrastructure providers, and it unifies heterogeneous technologies (*e.g.*, considering core network transport, and cloud). Virtualization concepts, such as Software Defined Networks (SDN), assist in achieving an end-to-end service. Subsection 2.3.1 presents these concepts.

Although the concept of separated virtual networks deployed over a single network is not new (*e.g.*, Virtual Local Area Networks (VLAN)), some differences make NS a novel concept (Afolabi et al., 2018). Network slices are end-to-end logical networks running on a common physical, mutually isolated, with independent management, created on-demand. The dynamism is the key difference from existing proposals, such as VLANs. The idea is to create slices to offer different services and assure some QoS within the slice. Thus, besides, be a set of network resources allocated to a service/application, a slice also is a set of flows belonging to different end-users. For instance, a slice can be all the flows whose source or destination is a given type of device such as sensors. Depending on the specification of a slice, an end-user can participate on different slices, but slices are always independent between each other. In the context of wireless networks, a slice can be created to give service to a specific group of devices with the same requirements or by type of application (Richart et al., 2016).

2.3.3 Network Slice Types

In general, applications requirements determine the type of network slice (Service/Slice Type - SST). However, due to the rapid advance of IoT and 5G, there are several applications with different requirements originated from the five main vertical sectors (automotive, healthcare, factories, media and entertainment, and energy) (3GPP, 2018). 3GPP presented market segments specifying more than 70 use cases and applications (3GPP, 2016), and NGMN described several applications, such as high-speed trains, healthcare services, and autonomous vehicles (Alliance, 2016). Thus, due to the number of use cases and applications, the standard bodies grouped all use cases in three categories to organize the requirements and vertical sectors (3GPP, 2018; Foukas et al., 2017): *(1)* Enhanced Mobile Broadband (eMBB), *(2)* Ultra-Reliable and Low-Latency Communications (URLLC), and *(3)* Massive Internet of Things (mIoT) (shown in Figure 2.6). The eMBB has very high data traffic and bit rate. URLLC characteristics include low latency, ultra-high reliability, high-density distribution, and high precision accuracy. Finally, mIoT involves massive numbers of connections in environments with high user density.

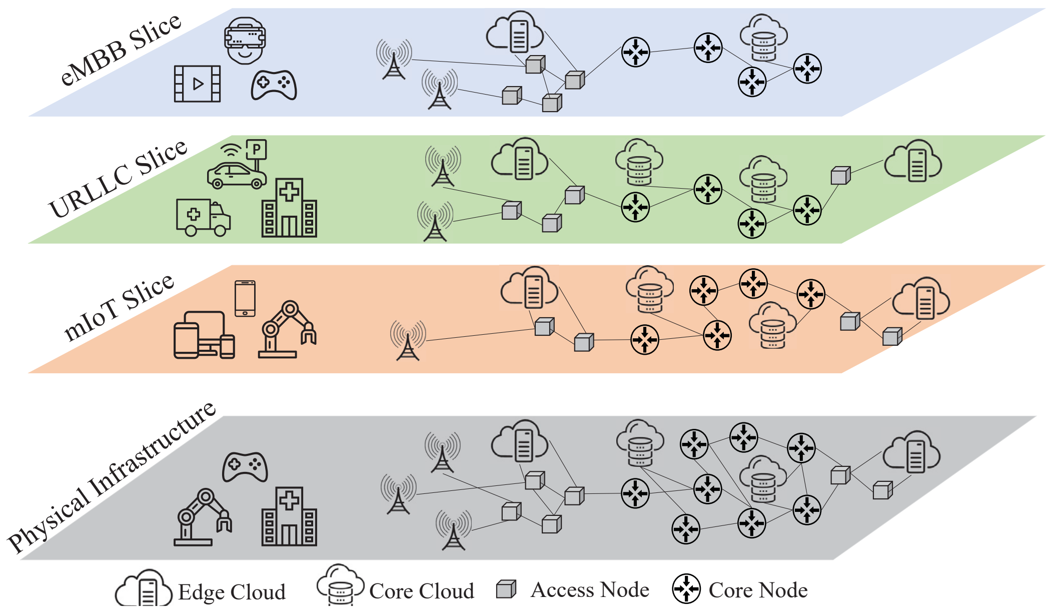


Figure 2.6: Network Slice Type (adapted from (Ordonez-Lucena et al., 2017))

As shown in Figure 2.6, each slice type has specific devices, applications, as well as requirements. The slice types share the same physical infrastructure. Hence, the physical infrastructure has different devices, such as sensors, mobile devices, autonomous cars, virtual reality devices, robot arms, and several network components, such as base stations, access nodes, and core nodes. Moreover, there are cloud servers both in edge and core network to realize virtualization concepts since network slicing utilize such concepts. Each slice encompasses different applications and services, *e.g.*, eMBB has virtual reality applications and streaming; URLLC has autonomous cars, emergency, and healthcare services; mIoT has a massive amount of devices and sensors communicating with each other, such as smart homes, wearable sensors, and stadiums. Thus, to achieve the specific requirements, each slice allocates network resources according to application requirements, using only a part of the physical infrastructure. Depending on the application requirement, it can use more than one slice type. At the moment, these three slices encompass several envisioned use cases. However, additional use cases are likely to emerge. Thus, flexibility is necessary to adapt to new use cases with a wide range of requirements.

In particular, the URLLC slice type characterizes by stringent latency and reliability requirements targeted at supporting critical communications use cases and services. It emphasizes the delivery of critical packets with high resiliency to variations in channel and load conditions rather than high throughput. Thus, it fits with smart healthcare applications and services, especially remote surgery, critical, and urgent care monitoring (Alliance, 2016; 5GPPP, 2015). For instance, urgent care applications that monitor ill patients remotely, face to emergency scenarios (*e.g.*, heart attack) needs to send messages as soon as possible to the physicians, not supporting delays and packet losses. Remote surgery enables remotely assisted surgery. Specialists are not available in many hospitals and can join a local surgeon remotely to perform specific procedures that require expert skills. Hence, network slicing through virtualization technologies instantiates network elements at an appropriate location for the communication to proceed as close to a direct path as possible, reducing latency. The collection of virtualized elements forming network slices allows multiple instances of applications and viable to share network, computation, and other resources (Rost et al., 2017). Moreover, network slicing assists reliability through the reservation of resources when they are not necessary. Therefore, network slicing brings enormous benefits and assist in realizing the envisioned smart healthcare scenario.

2.3.3.0 Benefits of Network Slicing for S-Health

Mainly, s-health requires efficient networking capabilities to provide ultra-low latency, low loss rate, and high reliability. To achieve such requirements, network slicing offers service customization and isolation on physical network infrastructure, enabling the logical as well as physical separation of network resources (Afolabi et al., 2018). The main idea lies in a single physical piece of network infrastructure able to cost-effectively deliver multiple logical networks (slices) over the same network infrastructure. Figure 2.7 shows a network slicing perspective for s-health. Slice 1 (NS 1) is isolated for telemedicine applications monitoring patients, receiving then the slice type of eMBB with broadband connectivity. NS 2 contains network resources and services to achieve low latency and ultra-reliability since it is established for critical care monitoring applications. Hece, it receives the slice type URLLC. NS 3 is established for emergency response in natural disasters, *e.g.*, a hurricane, where many people are using wearables. In this case, massive devices from the survivors send messages to the hospital. This scenario requires a slice type of mIoT to manage the number of devices. Hence, these three NSs perspective can represent a smart hospital with different use cases that contains a vast and different type of traffic, each one with its specific requirements.

Another network slicing benefit for s-health is the flexibility to provide virtual analytic data and automation. Virtual analytic data consists of analyzing the network traffic and its behavior, and, then, automation provides the network services and resources necessary according to the network behavior and performance. Hence, data analytics complement automation. In general, to observe network traffic, techniques of data analytics can be applied like data

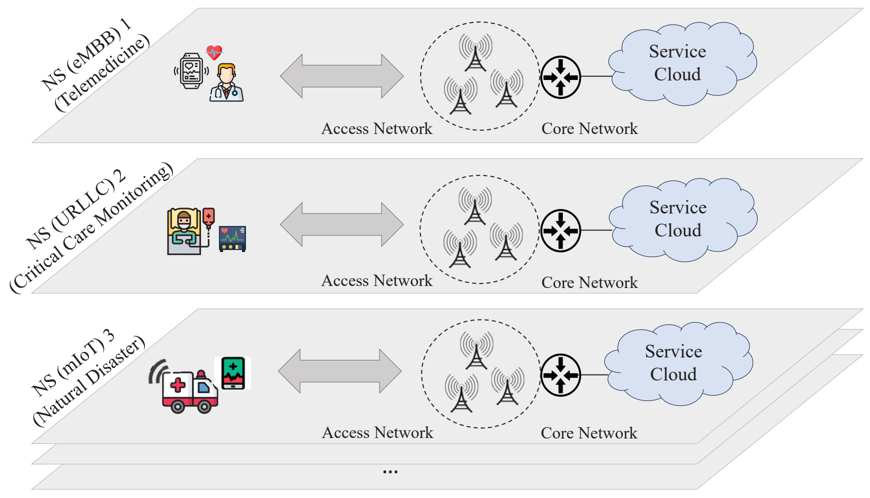


Figure 2.7: Example of Network Slices for S-health

instrumentation, feature exploration, and fingerprinting. However, once network slicing is a new paradigm, there is no standard indicated to do such traffic analysis. Machine learning algorithms can assist network slicing management since such algorithms quickly learn traffic features and behaviors. Thus, data instrumentation and fingerprinting s-health traffic can assist the automation of decisions about network resources adaptation. However, traffic-based attacks also benefit from network traffic analysis and ML techniques to infer sensitive information, posing issues related to network traffic analysis.

2.4 NETWORK TRAFFIC-BASED SIDE-CHANNEL ATTACKS

Traffic-based side-channel attacks take advantage of any data transmitted. The leakages consist of data generated on network transmissions, such as the size and timing of packets. Adversaries through sniffers capture network traffic passively and probe for side-channel leaks. The leaks, when statistical crossed with human behavior patterns, violates users' privacy. For instance, Srinivasan et al. (2008) inferred information about users through traffic analyses in a smart home scenario. The authors obtained the information through the similarity of timestamps between devices transmissions with the home residents behavior, enabling the identification of rooms, number of residents, and even possible visitors. The side-channel leakages occur in different layers (Prates et al., 2020). Captures directly on the devices analyze leaks related to the energy consumed to reveal the cryptography key. On the device embedded sensors, the captures analyze activation/deactivation to discover passwords, behavior patterns, and position. Analysis of network traffic captures reveals information related to the operational system, running applications, and even in-home rooms. It is also possible to observe population information through social data to discover social profiles and mass public patterns. However, captures from the devices and sensors require the target devices' possession. At the same time, the network data can be captured remotely, being vulnerable to attacks and feasibility the break of privacy.

2.4.1 Attack Definition

The traffic-based attack analyzes unintended data leaks to reveal personal information and harm users' privacy. Such attacks can have different targets such as devices, protocols, applications, or services. Thus, the literature usually sorts them among two orthogonal axes: A) invasive or non-invasive; B) active or passive (Standaert, 2010; Prates et al., 2020). In axes A, invasive attacks require the physical and direct access of the devices to observe leaks like the energy consumption and activation or deactivation of sensors. A non-invasive attack only exploits externally available information. It performs observations on leaks from the network traffic capture, such as the packet size. In axes B, active attacks control the input and output of the device to tamper with their proper functioning. For instance, an attacker can manipulate the device input to observe leaking information via abnormal behavior of the target subsequently. As opposed, passive attacks observe the device outputs without disturbing it (Spreitzer et al., 2018).

The side-channel attacks we consider in this thesis are passive and non-invasive attacks. Since they exploit leakages from network traffic captures such as packet size to infer personal information, they pose a severe threat to the security of most devices. Such devices range from laptops, gateways to IoT devices. The passive and non-invasive side-channel attack does not require physical access to the device since it collects information remotely. This attack is one of the most worrying regarding privacy since the user is unaware of capturing their information. It follows three phases: collecting, analyzing, and attack phases (Spreitzer et al., 2018; Prates et al., 2020). In the collecting phase, the adversary sniffs the network traffic and extracts network features. In the analyzing phase, the adversary finds patterns on the network or device behavior to reveal information. In the attack phase, the adversary builds ML models based on the network traffic features to identify devices, applications, and users activities to violate users' privacy.

Identifying the behavioral patterns of devices is the essence of traffic side-channel attacks. Then, machine learning technologies are applied to network captures to find patterns that identify information about the targets. Thus, traffic-based side-channel attacks use classification algorithms (supervised and unsupervised) as their main tool. Given the side-channel leakages (*e.g.*, packet size and timestamp), the adversary can identify the devices, users' actions/activities, and applications (Barman et al., 2021). It is possible to identify the device brand/manufacturer or type, the user actions (*i.e.*, interactions with the device), and the running application on the device. For instance, identify a specific device (*e.g.*, a monitoring camera), the user movement in-home, and a smartwatch application, respectively. Moreover, one identification can complement the other, *e.g.*, based on the device identification and its primary functions, the adversary can monitor the users' actions. For instance, the adversary can identify a sensor motion that works on/off; thus, network traffic means the user is in-home (Spreitzer et al., 2018).

The device, users' activities, and application identification use side-channel data, which involves data from the network traffic packet, such as packet size, timestamp, flow volume, flow duration, response time, and others. Because the device's behavior is valuable information, attackers seek to find patterns to infer information about each device present on the network. Once each device or application follows some specific pattern, either in packet size or timestamp, ML algorithms can identify them. Some IoT devices usually generate a tiny amount of network traffic with small packets sizes (*e.g.*, blood pressure sensor and smart plug), while others transmit a high number of packets with big packets sizes (*e.g.*, monitoring cameras and smart speaker assistant). Moreover, some IoT devices generate network traffic only with the user interaction (*e.g.*motion sensor and smart speaker assistant). In contrast, others transmit network traffic continuously (*e.g.*, monitoring cameras and baby monitor). Therefore, packet size and timestamp differentiate from one device to another, even in small values. Based on previous knowledge on devices functioning, the adversary can monitor the in-home users' activities.

2.4.2 Countermeasures Against Traffic-Based Attacks

With the ever increasing number of smart devices connected to people's everyday lives, the threat of traffic-based side-channel attack increases (Spreitzer et al., 2018). Malicious people can statistically analyze side-channel leakages to infer user information and violate security concerns.

Thus, countermeasures to prevent adversaries from performing traffic-based side-channel attacks have been considered in mobile and IoT devices. According to the existing defenses types, the most popular approaches include injecting fake network traffic and padding packet sizes to mask side-channel data to improve user privacy (Papadogiannaki and Ioannidis, 2021).

The fake traffic approach confuses the adversary by adding fake (or dummy) traffic (Barman et al., 2021). The fake traffic patterns conceal genuine use network traffic patterns (Yu et al., 2021). In contrast, the packet padding approach makes packet traces harder to distinguish by removing their differences, *e.g.*, by enforcing constant bit-rates and packet sizes or by altering the traces into the common closest sequence of packets (Pinheiro et al., 2020; Chaddad et al., 2021). The decision on the obfuscation approach regard in the network traffic features. For instance, some approaches transmit fake packets at fixed timing intervals to mask time-related information. Moreover, some works combine different countermeasures, such as randomly generating fake traffic with fixed or dynamic packet sizes (Dyer et al., 2012). Other works learn the statistical distributions from the original network traffic to generate the fake traffic as similar as possible (Yu et al., 2021). There are still some approaches that hide the device or application traffic into another device traffic. However, it is necessary to combine pairs of devices traffic.

The packet padding approaches are often costly since they add bytes to the network bandwidth. On the contrary, injection of fake traffic approaches consists of more practical and lightweight defenses since it allows to remove fake traffic and transmit it without increasing the packet size (Barman et al., 2021). Both approaches contain complex decisions about the size to increase the padding and the number of fake packets to generate. Once in the IoT environment, several devices have inactive periods, it is possible to send fake traffic during the idle time to represent the device activity. For instance, an adaptive packet padding mixes to inject fake network packets into the gaps in the packet flow to avoid identifying based on timing information. Hence, first, it identifies the gaps in the network volume flow to add fake traffic in the gaps to prevent long gaps from being a distinguishing feature to traffic-based side-channel attacks. Although this work aims to prevent traffic-based attacks, it is essential to mention that these countermeasures can add overhead in terms of bandwidth and data consumption which might not be acceptable in some contexts, *e.g.*, IoT devices (Spreitzer et al., 2018). Thus, most lightweight approaches seem more feasible for IoT environments.

2.5 SUMMARY

This chapter provided an overview of IoT, application diversity and requirements, types of devices, and communication technologies, such as wearable devices and wireless networks. It defined the main concepts required to achieve smart healthcare applications' requirements, such as latency and reliability, and concerns regarding the users' privacy. Thus, the main applications, scenarios, and requirements were defined (*e.g.*, urgent care monitoring applications). The main applications and their requirements were discussed, focusing on latency and reliability since several s-health applications require them. It was also discussed users' privacy because IoT devices and s-health applications contain sensitive information. Hence, it was emphasized the need for network performance and security management to achieve such requirements and protect privacy. Then, based on the FCAPS model, details were presented about performance management to discuss the QoS requirements of applications and services and security management to develop defense mechanisms. Therefore, it was provided insights about the crucial necessity of analyzing network traffic, exploring features, and identifying devices and applications to obtain valuable insights on devices and applications. Then, it was presented the essential properties of traffic identification and the main approaches. It was detailed the emerging concept of network slicing that allows to adapt the network resources according to services and application requirements. Finally, this chapter detailed the traffic-based attacks that allow adversaries to infer sensitive data from smart homes and smart scenarios. The main countermeasures against such attacks were presented, showing the necessity of defense mechanisms tailored for smart homes.

3 NETWORK DATA INSTRUMENTATION AS SUPPORT FOR E-HEALTH APPLICATION IDENTIFICATION

This chapter introduces a network data instrumentation methodology as support for network operations and, presents a novel method for e-Health application identification. In the same chapter, the motivation to present both proposals, the methodology and the method, is because the methodology supports the method; hence, it is a simple way to introduce the methodology and show its use in practice by a novel method. Moreover, the motivation to create a new methodology is to offer insights and measures crucial to providing effective and specialized services; and the motivation to propose a novel method is to identify e-health applications since these applications demand requirements. Hence, this chapter presents the FLIPER, an e**F**ficient methodo**L**ogy for **I**oT network traffic analysis as su**P**port to n**E**two**R**k management operations. The FLIPER methodology comprises steps to perform an efficient network data instrumentation. Such steps assist and support the e-health application identification performed by the MOTIF Method, a novel **M**eth**O**d for identifying eheal**T**h Applications through s**I**de-channel **F**eatures. Note that side-channel features consist of network traffic features (*e.g.*, packet size) since they are not from the packet payload; thus, both terms are used as similar throughout the text. The MOTIF method aims to identify the traffic of applications, especially health applications. It is important to identify health applications traffic to support network traffic management since they transmit sensitive data. MOTIF innovates employing side-channel features that can be openly observed without the necessity of previous knowledge about the application. The following section presents the FLIPER methodology and discusses the data instrumentation demands. Section 3.2 presents the main related works on identifying applications. Section 3.3 introduces the MOTIF method, setting overview. Section 3.4 presents the performance evaluation on MOTIF. Finally, Section 3.5 concludes the chapter.

3.1 THE FLIPER METHODOLOGY

The FLIPER Methodology (an e**F**ficient methodo**L**ogy for **I**oT network traffic analysis as su**P**port to n**E**two**R**k management operations) comprises a set of steps to perform an efficient network data instrumentation. The FLIPER methodology innovates by creating and organizing a useful sequential set of steps to instrument network data. However, before explaining FLIPER, it is crucial to explain the need to create it. Thus, data instrumentation supports network traffic analysis by structuring unorganized data. Moreover, data instrumentation and network traffic analysis are vital to providing efficient, effective, and safe services and enabling specialized services. Network traffic analysis on structured data offers insights and measures crucial to prioritize time-critical applications, provide expected experience and service level agreements (SLA), accommodate new network management requirements, prevent congestion, and develop specialized services. Due to providing support for several operations, such insights obtained by network data analysis comprise valuable information. An example of insights supporting network operations involves the 5G networks since they demand network data analysis to assist the network management to deliver specialized services for the different smart applications (5GPPP, 2015).

The definition of data instrumentation emerged from the broader concept of data analysis and learning engineering to explore, structure, and organize data and get valuable information (Goodell and Kolodner, 2022; Pekar et al., 2020). In the literature, the broader concept of data analysis and exploration is generic for different contexts, such as traffic classification and disease detection (Nguyen and Armitage, 2008; Papadogiannaki and Ioannidis, 2021; Panda and Panda, 2020; Sarker, 2021). However, they taper for the scenario specificity and data nature at some point. Data analysis and exploration usually follow generic steps such as data cleaning, feature extraction, treatment of missing values and outliers, and transformation or creation of new statistics features, according to the context (*e.g.*, health data, images, network traffic). Nonetheless, in some points (*e.g.*, feature extraction step), the data analysis and exploration funnel operate tailored to the context. The term commonly used in a computer network context is network traffic analysis. For instance, in the health-related application identification, data analysis considers network features such as packet size since this application generates a small packet size—similar to IoT device identification (Vergütz et al., 2019). Though, in the web application identification is interesting to consider features such as DNS domain (Ducange et al., 2017). In another context, physiological data analysis involves heartbeats data, and so on. In summary, even though the data analysis is generic for the contexts following similar steps, at some point, it is tailored for the context. Such tailored data analysis may offer relevant and valuable information to assist knowledge acquisition and decision-making.

Although several works in the literature proposed methods and architectures to address specific problems, such as traffic identification (Sivanathan et al., 2018; Aceto et al., 2017; Hafeez et al., 2020), big data analysis (Azad et al., 2019; Verma et al., 2020; Hou et al., 2020), and machine learning application on network traffic (Nguyen and Armitage, 2008; Callado et al., 2009; Conti et al., 2018), none network traffic analysis methodology considering data instrumentation was proposed in order to assist adaptive network solutions. For instance, the FLIPER methodology supports the National Institute of Standards and Technology (NIST) cybersecurity framework in identifying and detecting functions (NIST, 2018). Considering the fast evolution of network attacks, the steps of the FLIPER methodology also assist the development of adaptive defense solutions because static solutions may not be effective against zero-day attacks (Burns et al., 2001; Nogueira, 2009). Reactive defenses work efficiently against well-known intrusions, being vulnerable to unknown intrusions. Therefore, the data instrumentation and network traffic analysis of FLIPER methodology are preliminary steps for several purposes. It is the starting point for creating adaptive network solutions for management and security. Although the data instrumentation term is not used in the literature on network data, this work employs it as a fundamental step to organize and structure data, innovating for network traffic analysis.

Based in the literature review, this work argue that an efficient methodology tailored for analyzing network traffic is beneficial for decision-making, obtaining patterns and behaviors, assisting network management, improving network security, allowing specialized services, and others (Sarker, 2021). Thus, we introduce the FLIPER methodology to perform an efficient network traffic analysis considering data instrumentation and support several needs (*e.g.*, network management). Figure 3.1 presents the set of steps of the FLIPER methodology. The input consists of unstructured and unorganized data from a network capture (*e.g.*, a PCAP file). It is essential to understand the context and the reason for network traffic analysis. For instance, the network traffic analysis addresses traffic identification, device behavior learning, and application traffic management. The context and its needs assist in identifying the type of data (numeric or categorical) and data removing. For some situations, some data is unnecessary, *e.g.*, broadcast traffic. Thus, defining the cleaning and removing data is based on the analysis context and goals.

Afterward, it is essential to extract relevant network traffic features for analysis, like packet size, timestamp, IP address, port number, and others. In the extraction, it is possible to follow a flow-level or packet-level analysis. For applications or devices that generate a small amount of traffic, a packet-level gives more samples than a flow-level. In cases it is unknown, the application behavior may be helpful to extract from both levels to compare and learn the best

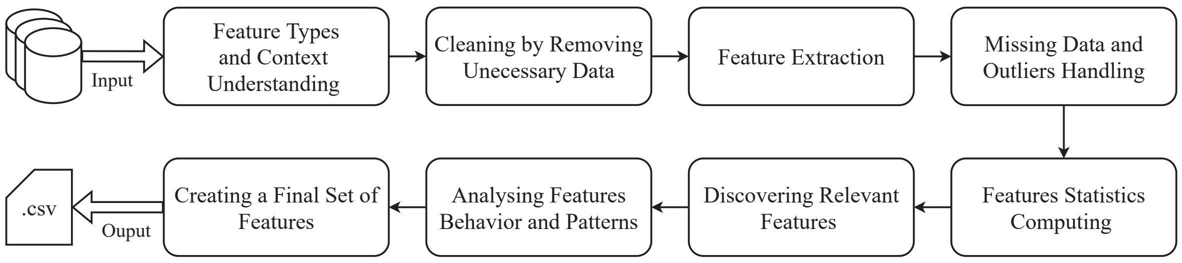

Input
Feature Types and Context Understanding
Cleaning by Removing Unecessary Data
Feature Extraction
Missing Data and Outliers Handling
Features Statistics Computing
Discovering Relevant Features
Analysing Features Behavior and Patterns
Creating a Final Set of Features
Ouput
.csv

Figure 3.1: The Sequential Set of Steps of FLIPER Methodology

level for the analysis context. Thus, the analysis goals also assist in defining the extraction level of the network traffic features. Then, there are missing values and outliers (*e.g.*, extreme values significantly different from the others) that demand some handling. Otherwise, they impact inefficiently in the network traffic analysis. Hence, replacing the missing value with the average, deviation pattern, or another statistic measurement computed from the original features. For instance, the replacement uses the average packet size if the missing value is related to packet size. Moreover, it is possible to remove or use outliers. The outliers may be necessary for some contexts because they represent an essential behavior. Therefore, it also depends on the context.

From the network features, it is common to compute statistics measurements to increase data sampling. These measurements involve average, deviation pattern, variance, among others. They increase the granularity of data sampling, improving the details. In sequence, a step, no less important but limited adopted, consists in discovering the most representative feature, *i.e.*, the features with more variance that most represent the entire data sampling. Sometimes, it is unnecessary to use the entire set of features since some represent most of the data. Hence, it reduces the features sampling, and it assists considerably in understanding patterns. In order to discover the most relevant features, the FLIPER workflow considers feature extraction and selection algorithms, such as PCA and FA. These algorithms analyze the variance and correlation between different features to reveal the most important. Thus, a crucial step is analyzing these representative features by data visualization (graphically) to understand the behaviors. Moreover, these key features allow learning more efficiently since the administrator is not learning from random and not representative features.

Finally, with knowledge from essential features, statistics measurements, and analysis context, it is possible to create a set of features to use as support for several purposes such as network management, specialized services, security mechanisms, traffic identification, and others. For instance, the FLIPER output may be a file containing relevant information for analysis, as shown in Table 3.1. This output file considers an analysis in an IoT environment; hence it contains the device name labeled from some features (*e.g.*, MAC address), the number of packets, mean of the number of packets, and the day capture. It contains only the number of packets as network traffic features since we assume in this example that it was the most relevant feature found in the analysis. Moreover, the capture day may be a key feature for a network administrator or an adversary. This set of features allows understanding behaviors, device patterns, and users activities per day. In other words, it assists achieving valuable information.

3.2 RELATED WORKS ON IDENTIFYING APPLICATIONS

Identifying legitimate traffic of an s-health application is a non-trivial task. Researches must deal with an increasing amount of traffic, diversity of applications as well as traffic complexity (Dainotti et al., 2012). Further, researchers look for lightweight algorithms with little computational requirements as possible. Hence, we have investigated methods existing in the literature to

Table 3.1: Example of the FLIPER Output File

Device Name	No. of Packets	Mean No. of Packets	Day Capture
Amazon Echo Assistant	10	8	16-29-23
Monitoring Camera	80	67	16-29-23
Smart Plug Device	32	29	16-29-23

identify an application. In general, the state-of-the-art on the identification of network application comprises four main classes of approaches, *i.e.*, those based on *(i) transport-layer port number*, *(ii) inspection of packet payload*, *(iii) flow behavior*, and *(iv) statistics* that explores flow-level measurements. In each of these approaches, there are several challenges and limitations to be addressed (Aceto et al., 2017). As MOTIF method is focused on identifying healthcare traffic from s-health applications or devices, a discussion of the four approaches relates to the feasibility of application in the s-health context.

3.2.1 Port-Based Approach

The most basic traffic identification method uses the port number information included in the packet header to associate with a specific application. More specifically, this approach extracts the transport-layer port number from the packet header and then looks it up in the table containing the port-application associations (Valenti et al., 2013). Each application is connected to a specific port number since well-known Internet protocols have specific port numbers according to Internet Assigned Numbers Authority (IANA) (Iana, 2019). For instance, the Transport Control Protocol (TCP) activity is connected with port number 80, Domain Name System (DNS) with port number 53, and others. Thus, associating transport-layer ports with specific applications is still the fastest and most straightforward technique for continuous monitoring and reporting (Dainotti et al., 2012). However, this can be very inaccurate and not reliable, because current applications can hide traffic behind well-known ports (*e.g.*, using TCP port 80) or use dynamically allocated ports to speak with each other (*e.g.*, Peer-to-Peer (P2P) applications) (Grajzer et al., 2012; Lopez-Martin et al., 2017). Therefore, due to these disadvantages, the port-based approach does not fit with the fingerprinting of s-health application.

Moreover, regular administration ports might be utilized by malicious (*e.g.*, malware), and port numbers can be covered up by encryption (Aceto et al., 2017). Network administrators employ techniques to hide packet header information to difficult the identification of traffic applications by malicious. Among these techniques, one consists of hiding the traffic by well-know ports numbers from the Internet protocols. Another technique uses a dynamic port number for the applications. Thus, for all new traffic, a different port number is used (Grajzer and Szczechowiak, 2012). Finally, administrators also benefit from encryption techniques to prevent application identification. Because of these challenges imposed by network administrators, a port-based approach is not appropriate for identifying health-related applications since such applications carry extremely critical data and do not allow unsatisfactory accuracy.

3.2.2 Payload-Based Approach

Numerous researches have begun to investigate payload-based methods, generally known as Deep Packet Inspection (DPI), to achieve better accuracy in traffic application fingerprinting (Valenti et al., 2013). DPI techniques are commonly used in the intrusion detection context, inspecting the network traffic (Prithi. et al., 2017). Intrusion detection is a network process to identify when illegal or unauthorized traffic/access misuses network resources. Hence, DPI allows a deep and detailed analysis of network traffic, both the header and the packet content. Regarding applications identification, studies using DPI focus on others contexts, such as IoT (Lopez-Martin et al., 2017), and Internet or mobile application identification (Alcock and Nelson, 2013; Yao et al., 2015; Shen et al., 2017). There are few studies in the context of wearable devices and health-related applications (Wood et al., 2017; Paglialonga et al., 2017).

The authors in (Lopez-Martin et al., 2017) explored the use of DPI at IoT applications and highlighted the difficulty introduced by the heterogeneity of the distinct connected devices. Moreover, the authors confirmed that payload inspection methods are becoming more difficult due to data cryptography, user's privacy policies, and the need for a continuously updated database. Hence, DPI techniques are combined with machine learning algorithms to use the information other than data packet payloads, such as packet size and packet inter-arrival times. In particular, the authors used the combination of convolutional neural networks and recurrent neural network to classify the packet by deep learning model. Thus, the machine learning approach assisted DPI on IoT devices. With the same problem as encrypted data, the authors in (Yao et al., 2015) presented a framework for classifying network traffic generated by mobile applications at a per-flow granularity. The framework uses a small sample of mobile applications to extract the application name and the lexical context found in the traffic. However, the framework depends on the content of the Hypertext Transfer Protocol (HTTP) protocol header, so it does not classify applications with encrypted Hypertext Transfer Protocol Secure (HTTPS) protocol data.

In (Shen et al., 2017), the authors used DPI methods to extract the ground-truth information to assure the correct identification. Thus, they extracted the application name from the packet payload and proposed a method based on the second-order Markov Chain. Results achieved 90% of accuracy. Only considering DPI techniques, the authors in (Alcock and Nelson, 2013) compared the performance of four DPI techniques on general Internet identification. The techniques encompass the L7 filter, nDPI library, libprotoident library, and Tstat (TCP Statistic and Analysis Tool). The L7 is a classifier for Linux that identifies packets based on application layer data. The nDPI is an open-source library for deep packet inspection. L7 and nDPI associate the packet payload with pre-known signatures. Libprotoident is a library that performs application-layer protocol identification for flows. Finally, Tstat is a passive sniffer that identifies traffic patterns at both the network and the transport levels. The authors considered 28 applications in the analysis, such as Facebook, YouTube, and others. Results showed that nDPI and libprotoident provide the highest accuracy.

In the health context, the authors in (Wood et al., 2017) searched for users' medical or personal information in the packet payload. However, rather than identifying application traffic, the authors identified critical data or sensitive medical information that is transmitted unsafe. Hence, the authors follow a three-step approach involving traffic collection, cleartext detection (*i.e.*, unencrypted data), and metadata analysis. First, IoT devices collect the data traffic, including smart blood pressure and smart scale monitor. Then, in the captured traffic, DPI technique searches for cleartext data to reveal patient information. Finally, the metadata analysis examines the device activity to infer user behavior. Results showed that numerous devices send health information without cryptography. The authors in (Hahn et al., 2018) included machine learning methods to distinguish compressed from encrypted traffic, since IoT devices transmit packet payloads compressed to reduce the number of packets transmitted. Both studies did not identify applications. However, the security content is promising. In order to identify different health applications, the authors in (Paglialonga et al., 2017), based on text analytics, extracted information from the application webpages on the app stores, and identified relevant features such as the medical specialty. Results showed around 18 applications specialties available, *e.g.*, pharmacology, and nutrition. Although the studies considered health-related applications, they used pre-defined signatures and DPI methods that violate user privacy.

Based on the literature, several studies are related to mobile application identification, and few studies are related to health applications due to high computational complexity and inability of DPI methods to work with encrypted traffic (Wood et al., 2017; Paglialonga et al., 2017). Hence, it is clear that the approach based on the packet payload still presents difficulties in the classification of encrypted traffic. Moreover, health-related traffic generally employs safety measures (*e.g.*, cryptography techniques) since it consists of sensitive information regarding the status of patient's health. Although there are works that perform DPI under encrypted traffic (Sherry et al., 2015), inspecting packet payload on such health information violates the user's privacy and, therefore, is not suitable for s-health applications. Finally, the computational cost is also high due to packet memory access and association with regular expressions (Valenti et al., 2013). Hence, DPI techniques are not suitable for s-health applications.

3.2.3 Behavior-Based Approach

Traffic behavior-based approach looks at all traffic received by an end host (*e.g.*, smartphone) on the network and infers information about pattern traffic behavior. Therefore, this approach acts close to the user in the access network (Kim et al., 2008). For instance, this approach captures all host traffic, in terms of the destination port number and transport-layer protocol, and then compares the host application signatures with the captured traffic profile to classify traffic according to the application that generated it. In general, a behavior-based approach observes the traffic patterns generated, such as how many hosts are connected to the network, transport-layer protocol, and how many different ports are employed to identify the application generating traffic. The authors in (Valenti et al., 2013) created a graph with all hosts connected to the same network and analyzed their behaviors. However, the authors abstain from health-related applications. Further, due to the use of previously known signatures of applications, few works use this approach. Also, standard host behavior analysis requires a considerable amount of data in order to extract such a level of information. Hence, the application identification time will be longer, which can damage s-health applications that require immediate transmission. Hence, some numerous challenges and requirements need to be addressed for s-health applications, since by fingerprinting the traffic of such applications, it makes it possible to offer reliability, availability, and security requirements.

3.2.4 Statistics-Based Approach

Much attention has been invested in data mining and machine learning algorithms (ML-techniques). Generally, the literature divides ML-techniques into supervised learning and unsupervised learning. Supervised learning requires training data to be labeled in advance and produces a model that fits the training data. In contrast, unsupervised learning creates clusters with similar characteristics together, not requiring training data (Nguyen and Armitage, 2008). In general, ML-techniques are based on a *statistics-based approach*. Instead, analyze the payload, this approach considers network traffic features from the flow-level and packet-level (also called side-channel data since the feature used is not from the payload) such as packet size, inter-packet time, flow volume, and statistics measurements (*e.g.*, mean packet size) (Conti et al., 2018). However, most studies in the literature use this approach to identify common Internet applications (Peng et al., 2016; Jenefa and Moses, 2017), mobile applications (Uddin and Nadeem, 2016; Aceto et al., 2017; Shen et al., 2017; Taylor et al., 2018; Chaddad et al., 2018), Wi-Fi and IoT devices (Vo-Huu et al., 2016; Sivanathan et al., 2018), and we found only one study that address health-related applications (Grajzer et al., 2012). Hence, this subsection presents these studies and discusses their applicability in health context.

There are several studies identifying Internet traditional applications (*e.g.*, HTTP and FTP) to assist network management (Erman et al., 2006). The authors in (Erman et al., 2006; Bernaille et al., 2006) classified Internet applications using unsupervised algorithms (also called clustering algorithms), such as K-Means and AutoClass. Packet and flow-level features were used in the analysis, reaching 90% of accuracy. The authors in (Peng et al., 2016; Ducange et al., 2017; Ding et al., 2017; Jenefa and Moses, 2017) identified Internet traditional applications (*e.g.*, HTTP, P2P, and others) considering supervised and unsupervised ML-techniques (*e.g.*, Naive Bayes, K-means and Random Forest). The authors in (Peng et al., 2016) considered packet-level features, such as packet size, and statistical measures like mean packet size. The authors in (Jenefa and Moses, 2017) analyzed flow-level features from the applications, labeling each application flow. The authors in (Ducange et al., 2017) extracted both packet-level and flow-level features, such as flow volume and packet size, also considered statistical measures, *e.g.*, minimum packet size. The authors in (Ding et al., 2017) explored relationships between flow and its neighbors flows, such as flows sharing the same source/destination IP addresses. Hence, they analyzed the dominant application of the flows. All studies achieved good results, with an accuracy of around 0.993 in (Peng et al., 2016). Based on them, it is possible to confirm that the features employed are significant for a successful identification, because the features are the primary input for the classifiers. However, these studies did not address health application.

The authors in (Cao et al., 2019) fingerprinted SDN applications, such as Traffic Monitor, Load Balancer, and Scan Detection. The SDN controller controls these applications. Hence, many attackers infer information about the applications running on the controller before launching their attacks. As SDN separates data and control data, the attackers can infer information only analyzing the control packets. Thus, the authors adopted deep learning methods to extract the features from massive data and identify the applications. They used different types of deep neural networks: convolutional neural networks, long short-term memory, and stacked denoising autoencoder. The algorithms input consisted of capture time of packets, packet size, and packet directions, considering all control packet transmitted between switch and controller. Convolutional neural networks achieved the best results, with 94% of accuracy. However, fingerprint applications only using packet control can not identify applications that generate low packet control since the difference between the amount of traffic will be massive. Further, some applications have similar control packets, which can result in a false positive.

In the context of mobile applications, such as Facebook and WhatsApp, many studies (Alan and Kaur, 2016; Aceto et al., 2017; Aouini et al., 2018; Taylor et al., 2018) used ML-techniques to identify the applications. The authors in (Alan and Kaur, 2016) proposed a method to identify Android mobile applications even when traffic is encrypted. Hence, the method collects TCP/IP header information (*e.g.*, packet size) and applies supervised machine learning algorithms to identify the applications. Results achieved 88% of accuracy. The authors in (Aceto et al., 2017) presented a multi-classification approach to improve the classification performance of mobile applications. Hence, the authors compared five types of combiners (*e.g.*, Majority Voting, Weighted Majority Voting, Behavior Knowledge, and WERnecke's method), each with different complexity and training set requirements. Additionally, packet-level features and statistical measurements were considered, such as inter-packet time. The evaluation of combining techniques showed improvements in the classification performance, reaching the best results with Majority Voting and Naive Bayes. However, although the multi-classification approach improves performance, it also increases the complexity, cost computational, and classification time. In s-health context, wearable devices do not handle high complexity. Further, the s-health application for critical care monitoring requires low-latency. Hence, classification time needs to be small.

The authors in (Uddin and Nadeem, 2016) proposed the TrafficVision framework that classifies the network traffic flow of mobile applications using ML-techniques and performs some network management tasks. In a Software Defined Network (SDN) context, the framework identified both the application name (*i.e.*, Youtube, Vimeo, Skype, others) and the flow types (*i.e.*, video stream, audio stream, video chat, others). The flow types identification was based on flow-level statistical measurements, such as total byte count and total packet count, and was performed through the k-NN classifier. While the application name was identified through c5.0 decision tree classifier using labeled training data. Two datasets were used in the evaluation since the first dataset contains 89 different applications with the corresponding application name (training data), and the second dataset contains 45 applications (testing data). Considering a home scenario with tablets and smartphones, after identifying the applications, TrafficVision performs a policy control service to allocate higher bandwidth to the video flows streamed by the tablet to guarantee its quality. This service allocates higher bandwidth to the tablet by throttling the traffic of the other devices. Hence, tablet users experienced better performance, achieving 1200 kbps of average bitrate, from 600 kbps without the policy. However, the network management employed is static since it can not automatically adapt to different applications or other devices. Moreover, in the s-health context, the time taken to throttle the other traffic device can be high and cause consequences during patient monitoring.

The authors in (Taylor et al., 2016, 2018) proposed the AppScanner framework that identifies mobile applications by network traffic features (packet and flow-level measurements). Through the Random Forest classifier, the framework contains a reinforcement phase to detect ambiguity and validate the classification. Hence, the framework identifies labels and removes ambiguous traffic, and further validates the probability that the classification is correct. The validation employs a prediction probability threshold function, which evaluates the threshold value reached during classification. Thus, it was considered only classification values higher than the threshold. AppScanner results reached an accuracy of 94%. However, accuracy falls when the framework considers different versions of applications. Additionally, if an application generates only ambiguous traffic, it will not be identified, since ambiguity detection deletes all ambiguous traffic. In order to identify anomalous traffic, the authors in (Ajaeiya et al., 2018) proposed a framework that differentiates mobile applications and detects anomalous flow traffic based on flow-level measurements. The framework used Bagging decision tree classifier. Thus, if traffic is not classified usually into any of the classes of applications, it is considered an outlier and receive a prediction score. The score represents the posterior probability of a sample belonging to a class. Hence, through a score threshold, the flow is classified as anomalous. Although reached accuracy around 92%, the authors did not consider a realistic scenario with anomalous traffic.

The authors in (Aouini et al., 2018) considered a residential scenario with users using different applications, such as Facebook and Google Services, since the large amount of traffic present in this environment. Thus, the authors collected traffic for two days and classified it using C5.0 tree-based classifier with flow and packet-level measurements. Results achieved 98% of accuracy overall. However, it was obtained the ground-truth by DPI methods, *e.g.*, nDPI, which violates user privacy. Moreover, the decision tree algorithm increases the complexity as the tree size increase; hence, the authors needed to control the size in order to reduce the training time. The best result was 23.8 seconds of training time. The authors in (Liu et al., 2018) also used Random Forest decision tree to classify mobile application traffic. Hence, it was collected real mobile traffic for 16 days from several applications. Further, the authors proposed a Cascade Forest that contains multiple Random Forest to increase the performance. However, each Random Forest has 500 decision trees, which can increase the complexity and training time in a significant proportion. Nonetheless, there is no result of training time.

The authors in (Vo-Huu et al., 2016; Sivanathan et al., 2018; Prates et al., 2019a) proposed frameworks to fingerprint devices, Wi-Fi and IoT devices, respectively. The authors in (Vo-Huu et al., 2016) combined features from the MAC, network, and physical layer to increase the classification performance and accuracy. Further, the authors presented a new classification algorithm. The authors in (Prates et al., 2019a) fingerprinted IoT devices, such as temperature and light sensors, by side-channel features, such as timestamp. In contrast, the authors in (Sivanathan et al., 2018) used packet and flow-level features to fingerprint IoT devices by classifiers. Thus, in a smart home scenario with several devices sharing data, the authors collected the network traffic for 26 weeks. Devices include monitoring cameras, baby monitors, wearable sensors (*e.g.*, Sleep Sensor and Blipcare blood pressure), laptops, smartphones, hubs/controllers, environment light sensors (*e.g.*, LIFX light bulb), movement sensors (*e.g.*, Belkin Motion) and portable speaker assistance (*e.g.*, Triby Speaker and Amazon Echo). Among these devices, two devices monitor patients' health: Sleep Sensor and BlipcareBP. Network traffic pattern behavior were analyzed considering flow volume, flow duration, sleep time, server port number, and others. To distinguish common devices (*e.g.*, laptops) from IoT devices (*e.g.*, light sensors), the authors also observed the number of servers contacted and unique DNS requests since common devices communicate with a large number of servers. Results reached an overall accuracy of 97%. However, there are no metrics results for each device nor wearable devices.

The authors in (Hameed and Leivadeas, 2020; Kumar et al., 2022) classified IoT devices by machine learning algorithms. They employed both statistics and network traffic features for the classification. Both works compared the classification performance of different classifiers; however, they did not consider health-related devices. The authors in (Panda and Panda, 2020) classified Parkinson's disease using IoT based data. They employed an enterprise dataset and a voice dataset for binary classification problems. By analyzing the data of voice assistance from the labeled voice dataset, the authors classified the users with Parkinson's disease. However, the data used in the classification refer to the packet payload, abstaining from the network traffic features. Moreover, the authors did not classify any health-related application or device.

To identify different health-related applications and differ from general network traffic, the authors in (Grajzer et al., 2012) presented a statistic-based approach that combines the results from multiples classifiers. Hence, the authors selected nine well-known classifiers in the literature, encompassing supervised and unsupervised techniques. These classifiers include Bayes Networks, Naive Bayes, J48 decision tree, Random Tree, Random Forest, Multilayer Perceptron, Jrip (rule-based), k-NN (K Nearest Neighbors), and Simple K-Means (unsupervised). Moreover, the authors extracted 24 features in the packet and flow-level, such as payload size and inter-packet time, of the first ten packets of each flow. To evaluate the proposed approach, the authors created a mobile s-health application prototype that generates physiological data, telemedicine videos, and medical images. In addition to the health data, it was generated regular network traffic (also called background traffic). Thus, the performance evaluation considered the overall classifiers performance of all types of traffic. In the initial results, none classifier presented a high accuracy; for instance, K-Means reached an accuracy lower than 50%.

To achieve better results, in (Grajzer et al., 2012) the authors added combination techniques, such as Majority Vote, and feature selection technique, such as Wrapper Subset Evaluation method. Thus, it was possible to achieve better accuracy, such as 92% in the classification of all traffic. However, due to the poor performance of the K-Means classifier, it was no longer considered in the analysis, since unsupervised techniques are not able to associate clusters with application classes. Also, classifier combination techniques generally tend to benefit the most frequent types of traffic in the database. Thus, because generally, health-related traffic is smaller, it is often classified as some other similar and more frequent traffic. For instance, telemedicine traffic is usually associated with Skype traffic. Moreover, the authors did not present results related to computational cost and classification time, which can be increased by the combination and feature selection techniques. Since s-health applications have many requirements, such as critical applications that carry information about the vital signs of the patient support only 125 ms of latency (De la Torre Díez et al., 2018), they need special attention. Hence, it is essential to consider the time taken to identify the applications.

The study in (Lloret et al., 2017) proposed architecture for smart health continuous monitoring of chronic patients considering the classification time. The architecture contains wearable devices that collect measures from the body and transmit through Bluetooth to a smartphone at the patient side. The smartphone processes the data received and transmit to a hospital database through 5G networks to achieve a better bandwidth and low delay. The database contains an intelligent system with ML-techniques able to send an alarm to the doctor when it detects an anomalous data in the patient records. To detect anomalous data, the system compares records with data from other hospital databases. Experiment results using 5G networks and Multilayer Perceptron achieved 10ms of round trip time with 800 users. However, the authors did not consider other types of traffic, only medical traffic, since the goal was to identify medical alerts and improve continuous monitoring. Nonetheless, this is not a realistic scenario.

Based on the state-of-the-art presented about application identification statistics-based, in general, the studies use different classifiers following supervised, unsupervised, semi-supervised, and combination techniques. However, in most cases, there is no accurate traffic information regarding the application, special health applications that collect vital signs since they generate low network traffic. Therefore, the unsupervised approach fits best in identifying applications. In contrast, some studies pointed out that unsupervised algorithms in identifying specific applications not perform well since application with low network traffic are classified as other common and more frequent application (Taylor et al., 2018). Hence, despite the great effort of the literature, challenges still exist related to the identification of specific applications, especially s-health applications, due to their requirements and level of critically.

3.2.4.1 Discussion

Table 3.2 summarize the studies presented about application identification. Since literature little uses port-based and behavior-based approaches, we summarize only payload-based and statistics-based approaches. The statistics-based approach is considered as ML-techniques once the studies that use statistical measurements also use ML-techniques. Several studies adopt ML-techniques to identify applications due to the facility, big data analysis, and adaptability to different situations provided by such techniques. Further, some studies use DPI to obtain the ground-truth information and ML-technique to classify the network traffic; thus, classification algorithms based on ground-truth (Lopez-Martin et al., 2017). Although DPI achieves high accuracy to obtain the ground-truth since it analyzes the payload and signatures known, for some scenarios, this approach violates user privacy (*e.g.*, smart home, health data). Moreover, most of the studies focus on Internet traditional and mobile applications. Only three researches considered health-related applications (Wood et al., 2017; Paglialonga et al., 2017; Grajzer et al., 2012), since it is not a trivial task to identify these applications. Another study considered wearable devices (Sivanathan et al., 2018). Health applications and wearable devices that monitor vital signs and generate low network traffic are difficult to identify and differentiate from other network traffic since network traffic is always increasing. Identify health-related applications brings enormous benefits, such as enables to provide management network tasks that assist in reaching requirements (*e.g.*, latency, and reliability), and enables to offer and improve security.

Table 3.2: A summary table of cited papers and used identification approaches

Reference	**Target**				**Approach**		**Features**			
	Internet Apps	Mobile Apps	Health Apps	Others	DPI	ML-technique	Signatures	Packet-level	Flow-level	Statistics
Shen et al. (2017)	✓				✓	✓	✓	✓		
Bernaille et al. (2006)	✓					✓		✓		
Erman et al. (2006); Ducange et al. (2017)	✓					✓		✓	✓	✓
Peng et al. (2016)	✓					✓		✓		✓
Jenefa and Moses (2017); Ding et al. (2017)	✓					✓			✓	
Yao et al. (2015)		✓			✓		✓			
Uddin and Nadeem (2016)		✓				✓			✓	
Alan and Kaur (2016)		✓				✓		✓		
Aceto et al. (2017)		✓				✓		✓		✓
Ajaeiya et al. (2018)		✓				✓			✓	✓
Taylor et al. (2018); Liu et al. (2018); Aouini et al. (2018)		✓				✓		✓	✓	✓
Lopez-Martin et al. (2017)				✓	✓	✓	✓	✓		
Vo-Huu et al. (2016); Sivanathan et al. (2018)				✓		✓		✓	✓	✓
Cao et al. (2019)				✓		✓		✓	✓	
Panda and Panda (2020)				✓	✓	✓	✓			
Hameed and Leivadeas (2020); Kumar et al. (2022)				✓		✓		✓		✓
Wood et al. (2017); Paglialonga et al. (2017)			✓		✓		✓			
Grajzer et al. (2012)			✓			✓		✓		✓
Vergütz et al. (2019)			✓			✓		✓	✓	✓

3.3 THE MOTIF METHOD

This section describes the MOTIF method, which takes into account IoT devices network traffic as input to identify different s-health applications. The MOTIF method instantiates some steps from the FLIPER methodology To organize and structure network traffic data and identify the traffic of s-health applications. Hence, MOTIF extracts network traffic features (a.k.a, side-channel data) and submits them to ML-algorithms, such as Random Forest. The next subsections give an overview of the addressed scenario and detail MOTIF.

3.3.1 Setting Overview

This work consider a healthcare setting following an IoT environment, such as a smart home or a smart hospital, comprising of heterogeneous devices, such as thermostats, cameras, laptops, baby monitors, wearable devices and others (Figure 3.2). These devices collect data by various types of sensors, transmit them to a central station (the access point – AP – in the figure), and then the station sends the data to the cloud via the existing network infrastructure. In this IoT scenario, there is a number of wearable devices continuously monitoring user vital signs, *e.g.*, blood pressure, heart rate, and others. Also, there are devices running non-Health applications, *e.g.*, social network applications, browser, and others, generating network traffic.

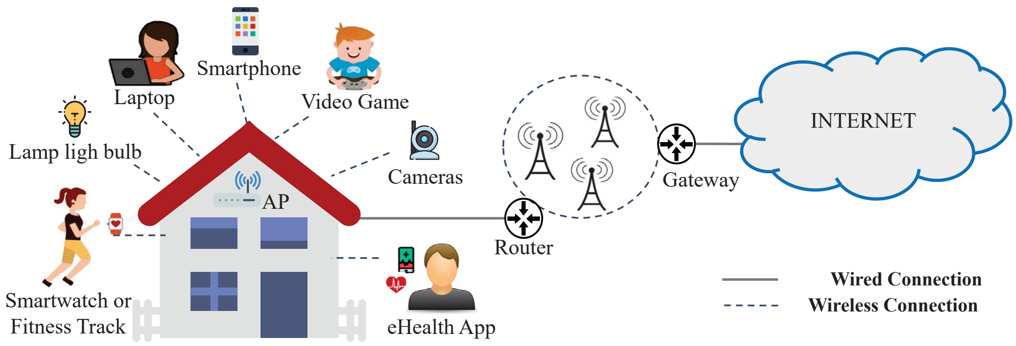


Figure 3.2: Overview of the Considered Scenario

MOTIF works based on side-channel data from the transport, network, and MAC layer. Hence, it can be positioned in a router that serves as gateway for a given network. In the router, MOTIF analyzes the network traffic. Taking as basis the results from MOTIF, router can be adapted to apply some network mechanism, such as adjust priorities in data transmission or re-allocating resources, aiming to achieve s-health application requirements (the definition of these actions based on the results from MOTIF is out of scope of this work). MOTIF employs side-channel data encompassing non-encrypted data from the network traffic (features), *e.g.*, flow volume, flow duration, packet size, inter-packet time, and others. Although network traffic features are not directly linked to the applications, they provide insights about the network traffic based on its behavior. For instance, wearable sensors that collect vital signs send periodic data, unlike another network application, such as the Facebook application, that generates traffic with a constant rate. Hence, through these features, it is possible to differentiate application traffic.

3.3.2 The MOTIF Description

The MOTIF method identifies the traffic of different classes of device applications employing side-channel data and following three main phases: *(1) Pre-Processing, (2) Feature Extraction*, and *(3) Fingerprinting*. The first phase pre-processes the network traffic filtering traffic according to certain feature, such as MAC address, to associate the traffic with the devices. The second phase selects and extracts the side-channel data and the statistical properties from the traffic of each device. The third phase gets the labeled data and creates datasets for training, validation, and tests (if for holdout analysis) and then applies ML-algorithms to fingerprint devices applications.

3.3.3 Pre-Processing Phase

This phase handles the network traffic into two components: *(i)* the collection of ground truth-information and *(ii)* data processing for each device, as shown in Figure 3.3. Together, they create subsets of network traffic belonging to each device (D). In the literature, ground-truth comprises the correct labels of the classes. It correctly indicates which class the data examples belong to (Pacheco et al., 2018). In the MOTIF method, the ground-truth correctly indicate what device each traffic flow or packet data belongs to. MOTIF uses the MAC address as ground-truth to correctly indicate what device each traffic belongs to. Thus, based on the ground-truth information, MOTIF method splits the network traffic and labels according to each device. As a

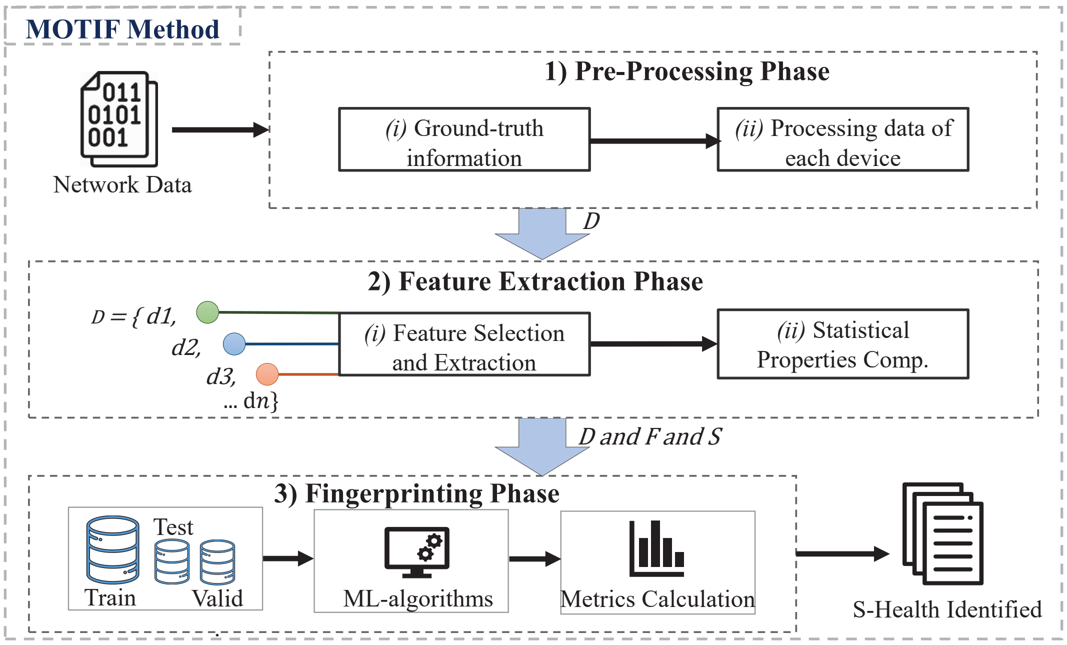

MOTIF Method
Network Data
1) Pre-Processing Phase
(i) Ground-truth information
(ii) Processing data of each device
D
2) Feature Extraction Phase
D = { d1, d2, d3, ... dn}
(i) Feature Selection and Extraction
(ii) Statistical Properties Comp.
D and F and S
3) Fingerprinting Phase
Test
Train
Valid
ML-algorithms
Metrics Calculation
S-Health Identified

Figure 3.3: MOTIF Method

result, MOTIF method creates a subset of network data for each class of devices, denoted by a set $D = \{d_1, d_2, ..., d_n\}$, where each d means a specific device class with its network traffic.

3.3.4 Feature Extraction Phase

This phase receives as input the set of device classes D. Afterwards, for each class $d_i \in D$, MOTIF method selects and extracts the features from the network traffic, defined as a set F of m network features, $F = \{f_1, f_2, ..., f_m\}$ that belongs to each class. MOTIF method employs packet size and flow volume as side-channel data since these two features play a relevant effect in the traffic of s-health applications. For instance, s-health applications related to monitoring physiological data yield small packet size (*e.g.*, around 50 bytes (Movassaghi et al., 2014)), and thus generate a small volume compared to other applications. Moreover, studies point out that network layer traffic has unique features, as flow volume and packet size, for certain applications classes, being able to differentiate one application from another (Pacheco et al., 2018). In order to further differentiate the classes of devices, after extracting the network features, the MOTIF method computes their statistical properties. These properties consist of measures such as average, variance, and others. Thus, for each set of features F, MOTIF method calculates a set of x statistical measures, $S = \{s_1, s_2, ..., s_x\}$, where $D \ni F \ni S$. For instance, for the packet size feature, the MOTIF method computes the maximum ($s1$), minimum ($s2$), average ($s3$), and variance ($s4$). The MOTIF method follows this procedure for all employed statistical measures. Then, this phase offers for the third one, the set of network features F and the set of statistical properties S for each class of device D.

3.3.5 Fingerprinting Phase

This phase uses distinct types of ML-algorithms to fingerprint different s-health applications. Once there are several classes of applications to identify, *i.e.*, a multi-classification problem, the MOTIF method employs algorithms suitable for solving this issue. The well-known algorithms involve the Naive Bayes (NB), Random Forest (RF) based on decision tree rules, non-parametric algorithm K-Nearest Neighbors (k-NN), non-probabilistic Support Vector Machine (SVM), and ensemble-based Bagging (Bagg). Hence, this module receives as input the network traffic features F and statistical properties S of each class of device. As each class is labeled, the MOTIF method merges them in a final file, called training data. The MOTIF method supports unbalanced data (*i.e.*, a dataset containing applications with different amounts of data). For instance, a health-related application monitoring physiological data send few packets with small packet size, while a social network application generates constant packet data, and a streaming application sends big packet size regularly.

To identify different applications with unbalanced data by ML-algorithms, the MOTIF method follows the holdout and cross-validation evaluation methods. The holdout is the traditional model evaluation method. It splits the data in training (60%), testing (20%), and validation (20%). Once health-related applications generally cause unbalanced data, MOTIF mothod also follows the cross-validation method. Cross-validation receives only one file as input and has a parameter called k, which refers to the number of groups the input data will be split (*i.e.*, create k subsets). One subset is the test set and the rest for training the model. Hence, cross-validation performs the same procedure for the k folds and gives a global performance by the combination (average) of the evaluation score of each test set (Pacheco et al., 2018). After creating the final file for each method, MOTIF method applies the ML-algorithms and computes performance metrics (*e.g.*, accuracy, recall) to fingerprint the classes of applications. The final output lies in a table with the percentage values for the metrics to each identified class, *e.g.*, s-health applications.

3.4 PERFORMANCE EVALUATION

This section presents the evaluation methodology and results obtained on the MOTIF method. The MOTIF method employs a statistics-based approach with envisioned machine learning algorithms since it does not violate users' privacy. It analyzes network traffic features such as packet size and flow volume. Such features allow monitoring different network traffic patterns, suitable for smart scenarios (*e.g.*, smart home, smart healthcare) since they generate a high diversity of traffic. The MOTIF evaluation methodology follows two scenarios: Packet Scenario (PS) and Flow Scenario (FS), where we extract the statistical properties through Numpy library (developers, 2021) from Python v2.7. Hence, the next subsections describe the dataset, scenario, feature extraction, ML algorithms, and performance metrics. Section 3.4.4 presents and discusses the results. Finally, Section 3.5 concludes the chapter.

3.4.1 Dataset Characteristics

Once MOTIF works in an IoT environment (*e.g.*, smart healthcare) with several devices sharing data, the representative *IoT Traffic Analysis* dataset (Sivanathan et al., 2018, 2021) serves as input for MOTIF performance evaluation. The dataset presents network traffic from 20 days (Sep. 22, 2016 to Oct. 12, 2016) and it is relevant because it contains records of real network traffic from different types of devices, including monitoring cameras, baby monitors, wearable sensors (*e.g.*, Sleep Sensor and Blipcare blood pressure), laptops, smartphones, hubs/controllers, environment light sensors (*e.g.*, LIFX light bulb), movement sensors (*e.g.*, Belkin Motion) and portable speaker assistance (*e.g.*, Triby Speaker and Amazon Echo). Among these devices, two devices monitor patients health: Sleep Sensor and BlipcareBP. The goal lies in identifying the traffic from these two devices. This variety of devices makes possible to evaluate MOTIF in a more realistic scenario as well as a better comprehension of data analysis and conclusions. Moreover, for being an online available dataset, it allows the easy experiment reproducibility.

The dataset is the base for the extraction of side-channel data and statistical properties. As mentioned, MOTIF employs as reference for analysis the packet size and the flow volume, extracted by Tshark (Team, 2021) and Cisco-Joy tools (Cisco, 2019). Thus, the input file to identify applications is composed of both features and the associated capture time. The capture time encompass the trace capture day from the dataset. Based on this, we normalize the capture time to an interval from 1 to 20 days, and we extract packet sizes from the header of the network layer. A flow is the data-plane of packets between sender and receiver that shares key IP header information. Thus, a flow encompasses the 5-tuple information: same source address and destination address, same source port and destination port, and same protocol. Flow volume is the sum of all bytes sent from a source to a destination. Here, we ignore packets destined to DNS servers and packets or flow volume with size equal to 0. Moreover, the MAC address of each device is used for labelling the training data since this information is available in the dataset.

3.4.2 Details of Feature Selection and Extraction

The *IoT Traffic Analysis* dataset comprises of 20 pcaps files, one for each day of capture. For the pre-processing phase of MOTIF method, we filter these files by the MAC address feature, resulting in different files, one for each MAC address (device) identified in the network traffic. For each device, we compose its associate file with the packet size and flow volume per day, *i.e.*, each device file contains the network traffic features over the 20 days. To calculate the values for the statistical properties, we create small samples for each traffic feature (samples size = 3 values of each feature). Next, we calculate the *min*, *max*, *average*, and *variance* of the network traffic features. At the end, each device has its own values for the network traffic features, statistical properties and a label according to its MAC address. Then, we merge the files of all devices into one unique final file, called training data. As shown in Table 3.3, we follow two evaluation scenarios: *(i)* the packet scenario, where we employ the features related to the network packet size; and *(ii)* the flow scenario, that we use the features related to the flow, such as flow volume.

Table 3.3: Features on the Evaluation Scenarios

SCENARIOS	**FEATURES**
Packet Scenario (PS)	packet size, min packet size, max packet size, average packet size, and variance of packet size
Flow Scenario (FS)	flow volume, flow packet size, min flow packet size, max flow packet size, average flow packet size, and variance flow packet size

3.4.3 Machine Learning Algorithms

This analysis deals with five ML-algorithms to identify the classes of applications. As our problem lies in the multi-classification, *i.e.*, there are several classes of applications or devices to identify, we employ algorithms suitable for solving this issue. Mainly, the algorithms encompass the probabilistic Naive Bayes (NB) based on Bayes' Theorem. The non-parametric algorithm K-Nearest Neighbors (k-NN), which gather together similar samples through measuring their distance. The linear and non-probabilistic classifier Support Vector Machine (SVM) that classifies a new sample by analyzing example data from two classes. Random Forest (RF) based on decision tree, and the meta-algorithm Bagging (Bagg, also called Bootstrap aggregation) based on ensembles. In general, the ensemble method uses multiple learning algorithms to obtain better performance results than using only one algorithm. Hence, in the ensembles, each algorithm is trained individually, and the results are combined. Thus, ensembles can use decision tree algorithms, such as RF and others. Moreover, although SVM is a linear classifier, we apply in our evaluation since it presents high-performance results in the literature (Pacheco et al., 2018).

For the identification by ML-algorithms, we follow the holdout and cross-validation methods. The holdout method splits the data in training (60%), testing (20%), and validation (20%). Once the dataset is unbalanced, *i.e.*, some devices have heavier traffic than others (*e.g.*, the Laptop generates more traffic than BlipcareBP); we also apply the cross-validation method. The cross-validation receives only one file as input and has a parameter called k (Pacheco et al., 2018), which refers to the number of groups the input data will be split. As our dataset contains traffic from nine devices, we define $k = 9$. We run the 9-fold cross-validation and holdout 50 times with different seeds for the random function. After training the ML-algorithms, we calculate the performance metrics: accuracy (Equation 3.1), recall (Equation 3.2), precision (Equation 3.3) and F1 Score (Equation 3.4). These metrics are based on the values about true positive (TP), false positive (FP), true negative (TN), and false negative (FN) achieved during classification.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{3.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

$$F1\ Score = 2 * \frac{Recall * Precision}{Recall + Precision} \tag{3.4}$$

Accuracy represents the overall performance classification result. Precision and recall consider the FP and FN, respectively. There is a trade-off between them: precision represents situations where the FP is more harmful than FN, while recall describes situations where FN is more harmful than FP. Hence, recall fits on medical/healthcare situations since the ML-algorithm must identify all ill patient, even if classify a healthy patient as sick (FP situation). Hence, it is better to identify an ill patient than do not identify. In our case, recall represent an essential metric, since it is crucial to identify a health-related application, especially critical care application, than do not identify. The F1-score metric is the harmonic mean of precision and recall, providing a good measure of overall classification performance between them. Finally, the last metric is the classification time, given in seconds.

3.4.4 MOTIF Results

This subsection presents the MOTIF results about network traffic (side-channel data) behavior and fingerprinted applications, including the health-related applications. The results are based on the *IoT Traffic Analysis* dataset that contains network traffic of several devices. Therefore, this subsection presents the side-channel data behavior of 9 devices considered in the analysis the fingerprinting results. Note that we use the words fingerprinting and identification as similar.

3.4.4.0 Analysis of Side-Channel Data Behavior

The analysis results about the side-channel data behavior considered 9 types of devices from the *IoT Traffic Analysis* dataset. Table 3.4 shows these devices and a summary of the data byte rate and average packet size of each device for the 20 days of capture. The devices range from monitoring cameras and speakers to wearable sensors. We highlight the wearable devices BlipcareBP and Sleep Sensor since this work focuses on smart healthcare context. BLipcareBP reached the smaller data byte rate (0 bytes/sec), while Sleep Sensor presented 27 bytes/s. In contrast, the Laptop and Netatmo (monitoring camera) reached the higher values for data byte rate and average packet size, because both devices generate traffic continuously. The other devices present similar values. These behaviors confirm the unbalanced data presented in the dataset, which is typical for a real IoT scenario. However, it is harder to identify the smaller traffic.

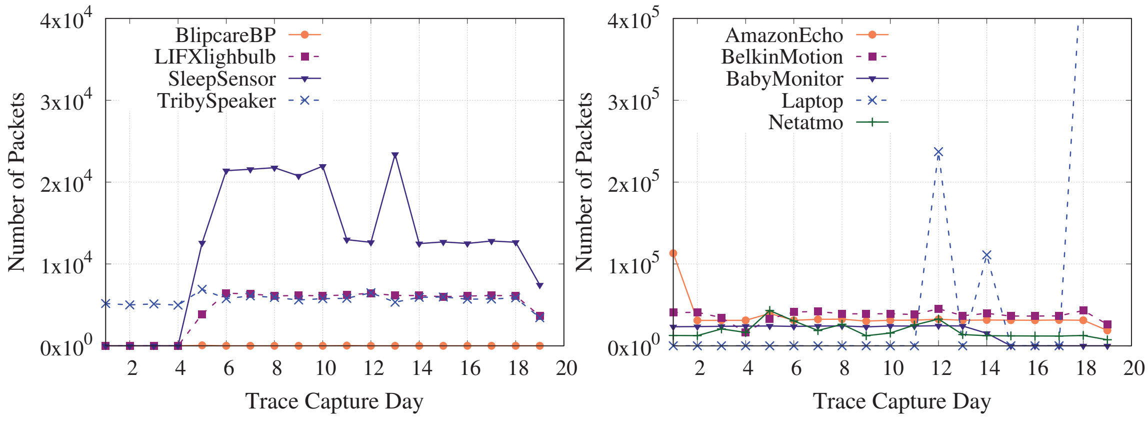
Table 3.4: Devices identified in the evaluation dataset

Device Description	Data Byte Rate	Average Packet Size
Amazon Echo	46 bytes/s	115.79 bytes
Baby Monitor	22 bytes/s	79.48 bytes
Belkin Motion	50 bytes/s	112.81 bytes
Blipcare BP	**0 bytes/s**	**105.69 bytes**
Laptop	147 bytes/s	122.54 bytes
LIFX ligh bulb	6 bytes/s	96.43 bytes
Netatmo (camera)	103 bytes/s	471.34 bytes
Sleep Sensor	**27 bytes/s**	**138.46 bytes**
Triby Speaker	6 bytes/s	104.43 bytes

Figures 3.4(a) and 3.4(b) present the number of packets per device identified in the dataset. Due to the difference in the number of packets values, we divided the devices into two plots. Figure 3.4(a) shows the devices that generated the least amount of packets, where BlipcareBP presented a tiny amount of traffic since it only generated traffic for 3 days. In contrast, Figure 3.4(b) shows the devices that generate more number of packets. The Laptop reached more than 300K packets, at the 17th day. While other devices, *i.e.*, Belkin Motion, BabyMonitor, AmazonEcho, and Netatmo, presented a similar behavior since they encompass similar devices, such as movement monitoring, cameras, and assistance speakers. Figure 3.4(c) endorses the behavior of all devices by showing the average flow volume over the 20 days. We ignore the values for Laptop and Netatmo, to better show the smaller flow volume. Regarding to flow volume, AmazonEcho and BabyMonitor exhibit higher average flow volume. Healthcare devices present small values, which confirms the difficult to identify the traffic from them.

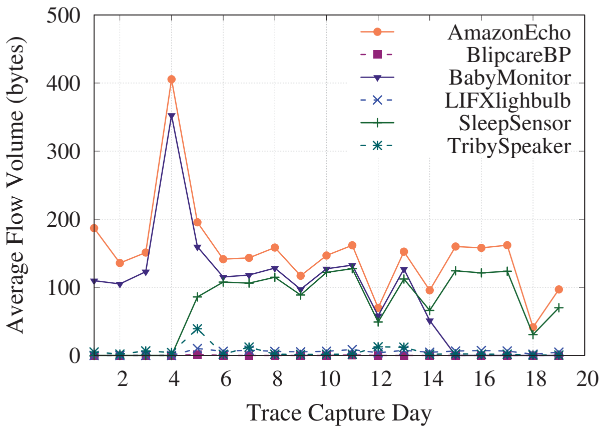
3.4.4.0 Results from Applications Fingerprinting

Unless we tell otherwise, results encompass a macro average overall 9 classes, where each class corresponds to one device. For each evaluation scenario (Packet Scenario – PS and Flow Scenario – FS), we compare the performance metrics between the five ML-algorithms (NB, Bagg, k-NN, RF, and SVM). We also compare the results between holdout and cross-validation methods. First, we discuss the holdout results for PS and FS. Figure 3.5(a) and Figure 3.5(b) show that all ML-algorithms have achieved an accuracy spanning from 90% to 100% in PS and FS, except NB. The lower performance of NB is because it assumes independent predictors, *i.e.*, if only one feature of whole set matches, NB classifies the data as belonging to the class that has the feature. However, it is almost impossible to have a set of completely independent features. Figure 3.6(a) and Figure 3.6(b) show results for F1 score, in which we observe a higher discrepancy between the ML-algorithms. For FS, the results show a decrease, since, in this setting, the data traffic is



(a) Devices with Low Volume of Traffic

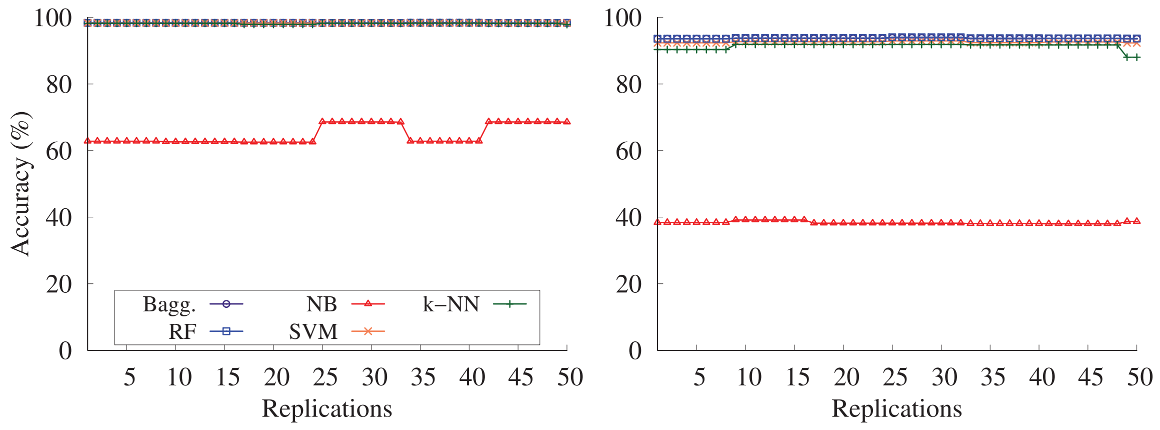
(b) Devices with High Volume of Traffic



(c) Devices Average Flow Volume

Figure 3.4: Traffic Applications Behavior vs Capture Day

divided by flow, outcoming training data with fewer samples than in PS. Generally, a training data with few samples reduce the classifier performance, because there is no enough data to compare.

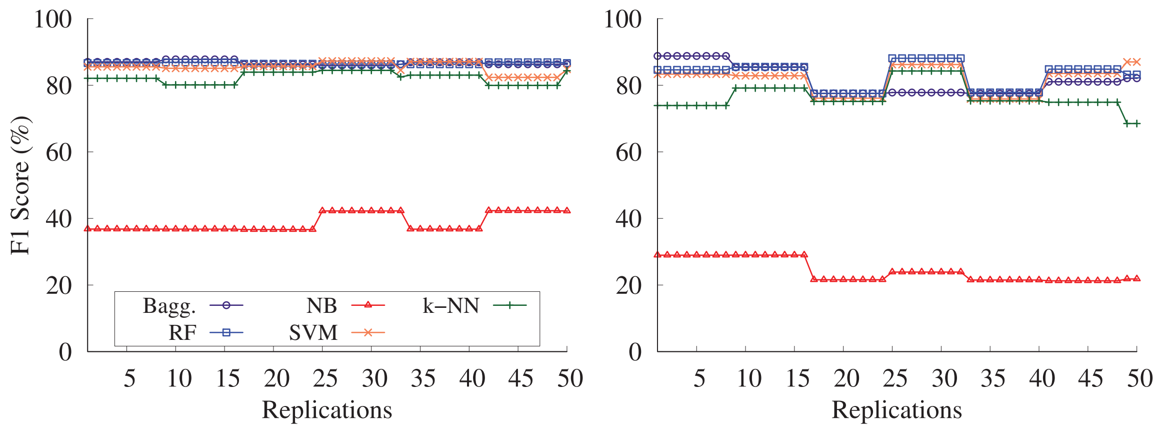


(a) PS Accuracy

(b) FS Accuracy

Figure 3.5: Accuracy of Holdout Method

As F1 score is the harmonic average of the precision and recall, we also present the results about these two metrics to further details. Moreover, recall highlights the FN, which is crucial for the healthcare context since it is better to identify a non-health application as a

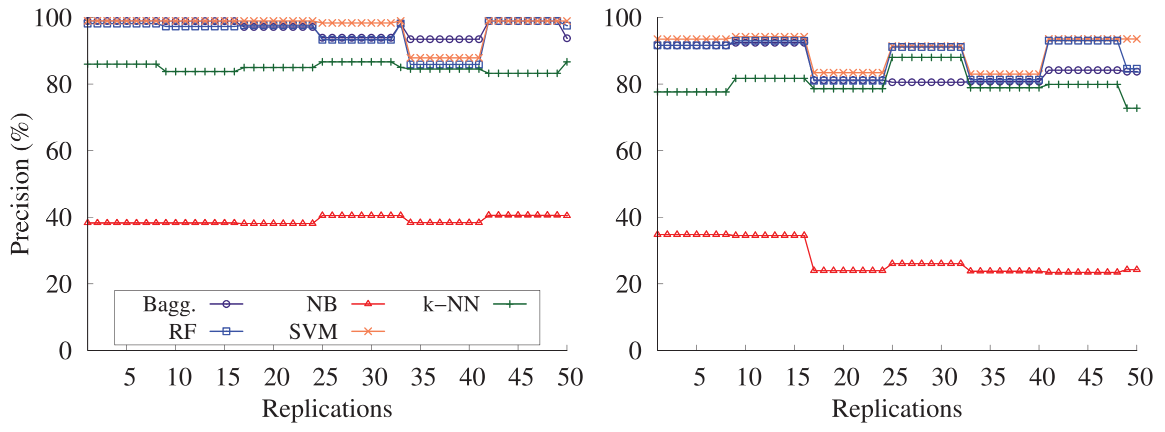


(a) PS F1 Score

(b) FS F1 Score

Figure 3.6: F1 Score of Holdout Method

health-related application than do not identify it. Figure 3.7(a) and Figure 3.7(b) show the precision results. For PS setting, RF, SVM and Bagg have reached high values ranging from 81% up to 100%, while k-NN and NB result in lower values. Figure 3.8(a) depicts a similar behavior for recall of RF, Bagg, k-NN, and SVM in PS setting, around 88%. The same four algorithms have achieved results with high variation in the FS setting (Figure 3.8(b)), where recall has presented low results because, in the FS scenario, the amount of samples in the training dataset is smaller. Once to compute the flow volume it is necessary to consider all packets of the flow, it causes a reduction in the amount of sample data. In contrast, the PS scenario considers the packet size of each network packet, having more sample data, which increase the performance.

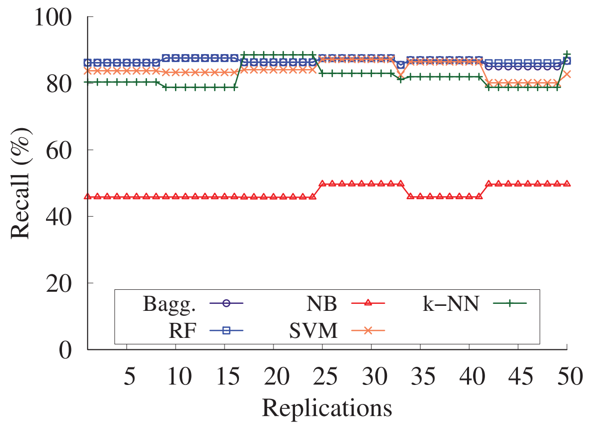


(a) PS Precision

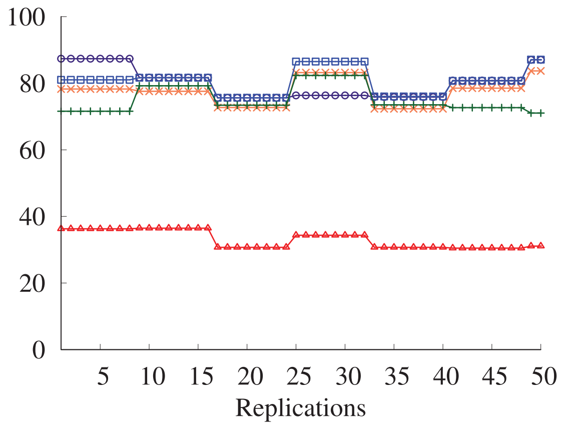
(b) FS Precision

Figure 3.7: Precision of Holdout Method

Figure 3.9(a) shows the total classification time of the five algorithms in the PS scenario, where k-NN and SVM are slower to process all the data. Both algorithms contain much computation. For instance, k-NN computes all distance neighbors from the new sample, which takes time. NB is faster than the other algorithms. FS scenario is faster than PS scenario in terms of classification time due to the small database to analyze (as mentioned before about small sample data). Figure 3.9(b) shows the classification time in FS scenario. Flow is the combination of several packets between the sender and receiver, which has an impact on the overall classification time taken. SVM takes the longest time to classify the traffic, while the other



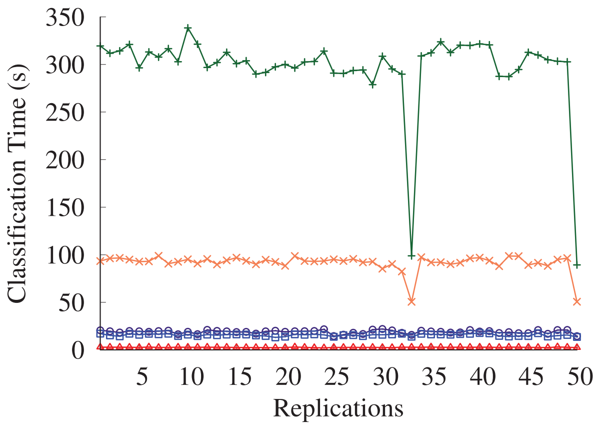
(a) PS Recall



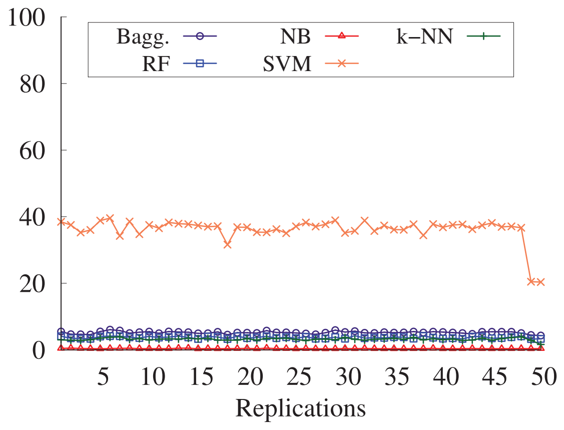
(b) FS Recall

Figure 3.8: Recall of Holdout Method

algorithms have achieved results less than 5 seconds each. When analyzing all the classifiers, RF presents the best overall performance in FS and PS scenarios. Bagg obtained high results around 90% in all performance metrics for PS, and NB did not present high performance, ranging from 20% to 50% in metrics results, not being appropriate for the s-health context.



(a) PS Classification Time



(b) FS Classification Time

Figure 3.9: Classification Time of Holdout Method

The analyses for the cross-validation method have followed the same five ML-algorithms (Bagg, NB, k-NN, RF, SVM). Table 3.5 shows the 9-fold (1-fold for each device) cross-validation method results for PS and FS settings considering all performance metrics. In both settings, RF and Bagg have achieved high results, around 90% in terms of accuracy, recall, precision, and F1 score. In terms of classification time, the fastest algorithm is NB. However, its classification performance was lower, reaching values around 30% and 20% of F1 score rate in PS and FS, respectively, not being suitable for the s-health context. In the FS setting, Bagg achieved great overall results since it based on ensembles techniques that combine the performance results of different algorithms. RF and SVM also achieve high performance results with recall around 81% and 78%, respectively. In summary, for FS and PS settings, RF and Bagg have presented the best overall results compared to other classifiers, especially in terms of recall rate. Therefore, both algorithms, RF and Bagg, are suitable for the s-health context, even with a small amount of data like in the FS setting. For more details, next, we present specific results for the wearable devices.

Table 3.5: 9-Fold Cross-Validation Results

	Classifier	Accuracy	F1 Score	Precision	Recall	Time
PS	Bagg.	**98.34%**	87.32%	98.05%	**87.68%**	214s
	NB	63.56%	37.03%	38.40%	45.95%	**8.11s**
	k-NN	98.17%	83.10%	93.49%	81.75%	2660s
	RF	**98.34%**	87.19%	97.39%	**87.68%**	149s
	SVM	98.29%	**87.44%**	**99.03%**	86.93%	1285s
FS	Bagg.	93.97%	**86.98%**	92.20%	**83.88%**	69s
	NB	38.66%	21.92%	24.27%	31.12%	**1.34s**
	k-NN	92.00%	75.87%	80.69%	73.47%	26s
	RF	**94.03%**	85.31%	92.50%	81.53%	38s
	SVM	93.09%	84.04%	**94.02%**	78.96%	1102s

3.4.4.0 Results from S-Health Application Fingerprinting

As the MOTIF method proposed the fingerprinting of different s-health applications, we present detailed results about the healthcare devices BlipcareBP and SleepSensor considering the best general classifiers of our previous evaluation: RF and Bagg. Table 3.6 and Table 3.7 shows the overall BlipcareBP and SleepSensor results, respectively. The BlipcareBP only generated traffic over 3 days (as discussed in Subsection 3.4.4), causing unbalanced data. Its network traffic is mixed with different types of traffic, including the traffic from the Laptop and Netatmo that generated much data, making it difficult to identify. Although the unbalanced data scenario, the MOTIF method reached 100% of accuracy in the 9 cross-fold validation method for the BlipcareBP. Both RF and Bagg achieved such accuracy in the PS setting. However, for the holdout method in the FS setting, results are lower, around 30% of accuracy, proving that the cross-fold validation method fits best in the s-health context. The holdout method requires more data samples for training than the 9-fold cross-validation method, while 9-fold uses a simple input file as training and testing, crossing the data for 9-fold times. Furthermore, RF and Bagg achieved 90% and 100% of recall in the cross-validation method for PS and FS, respectively.

Table 3.6: BlipcareBP Results

	Algorithm	Accuracy	F1 Score	Precision	Recall
PS	RF (9-fold)	**100%**	94.75%	90.05%	**100%**
	RF (Holdout)	94.30%	92.93%	**93.15%**	93.20%
	Bagg. (9-fold)	**100%**	**95.45%**	91.30%	**100%**
	Bagg. (Holdout)	94.76%	85.34%	78.40%	96.06%
FS	RF (9-fold)	50%	66.67%	**100%**	50%
	RF (Holdout)	14.96%	32.65%	14.96%	20.40%
	Bagg. (9-fold)	**75%**	**85.71%**	**100%**	**75%**
	Bagg. (Holdout)	38.70%	38.77%	32.65%	34.59%

In contrast, SleepSensor has generated more traffic than BlipcareBP (as discussed in Subsection 3.4.4), not being so challenging to identify. As shown in Table 3.7, RF and Bagg reached high values rate for all performance metrics, presenting then optimal overall results in all scenarios. As SleepSensor generated more traffic than BlipcareBP, the holdout method also presented high accuracy and recall, with values around 97%. However, the classification time of Bagg and RF (as shown in Table 3.5) reached values around 214 and 149s in PS settings. While for FS settings, the classification time reduced to 60s and 38s for Bagg and RF. Therefore, the FS setting in terms of time is better than PS for s-health, but improvements still are necessary. Based on the results from BlipcareBP and SleepSensor, we also have observed that cross-validation method jointly with RF and Bagg presented the best overall results performance for s-health identification, mainly for devices monitoring vital signs, as BlipcareBP.

Table 3.7: Sleep Sensor Results

	Algorithm	Accuracy	F1 Score	Precision	Recall
PS	RF (9-fold)	97.95%	98.89%	99.85%	97.95%
	RF (Holdout)	97.95%	98.89%	99.84%	97.95%
	Bagg. (9-fold)	97.95%	98.90%	**99.86%**	97.96%
	Bagg. (Holdout)	**98.02%**	**98.91%**	99.82%	**98.01%**
FS	RF (9-fold)	97.43%	**97.59%**	**97.75%**	97.43%
	RF (Holdout)	97.43%	97.02%	97.15%	97.08%
	Bagg. (9-fold)	**97.45%**	97.52%	97.59%	**97.45%**
	Bagg. (Holdout)	96.93%	96.90%	96.87%	96.93%

3.4.4.0 Discussion

In summary, the MOTIF method was able to fingerprint the s-health applications. In the results, two scenarios were considered, PS and FS, and two machine learning performance methods, the cross-validation and holdout. Once RF and Bagg presented the best results, the discussion follow their results related to the F1 Score. For the holdout method, results showed around 85% in the PS setting, and 80% in the FS. For the cross-validation method, results achieved around 87% in the PS setting, and 86% in the FS. The classification time in the holdout method reached values around 30s in PS and FS, while in cross-validation, it achieved values around 150s and 50s for PS and FS, respectively. Related to s-health devices, the BlipcareBP and SleepSensor were deeply analyzed. BlipcareBP generated the smallest traffic, and then we discuss its results. In this context, MOTIF, in the holdout method, presented values around 88% in the PS setting, and 35% in FS. In contrast, for the cross-validation method (9-fold), MOTIF reached values around 95% in the PS setting, and 80% in the FS. Based on the overall results, we noted that the holdout method is better than cross-validation in terms of time. However, cross-validation is better to fingerprint s-health application (BlipcareBP) than holdout. Furthermore, the FS setting contains a smaller amount of data than PS, which result in low classification time. In contrast, PS achieved better performance results than FS. Therefore, there is a trade-off between classification time and performance. For s-health, both metrics are crucial. Thus, the combination of the cross-validation method and FS setting is a solution since, by MOTIF method, they already reached optimal results. However, further studies are necessary to reduce the classification time.

3.5 SUMMARY

This chapter presented the MOTIF method, whose goals are to identify the traffic of smart healthcare applications. Thus, this chapter presented the related works and the main approaches on traffic and application identification. It also details the MOTIF method. Then, it presented the evaluation methodology, including dataset, feature extraction, ML algorithms, and performance metrics. Furthermore, this chapter presented and discussed the results from network traffic features and application identification. Although the unbalanced dataset, the MOTIF method identified with success the applications, where the cross-validation method achieved better results than the holdout method. Results showed 100% of accuracy and recall in the cross-validation method for the BlipcareBP device. For the SleepSensor device, MOTIF achieved optimal overall results in all scenarios. Based on the results, we noted that the cross-validation method fits in the s-health context, only needing further studies about the classification time.

4 AN ARCHITECTURE FOR THE PERFORMANCE MANAGEMENT OF S-HEALTH APPLICATIONS

The fifth-generation (5G) network will offer smart healthcare (s-health) treatments and allow efficient patient remote monitoring. S-health applications require strict network requirements, for instance, 99.999% of service reliability and 1 ms of end-to-end latency. However, it is a challenging task to manage network resources and applications towards such performance requirements. Hence, significant attention focuses on performance management as a way of searching for efficient approaches to adjust and tune network resources to application needs, assisting in achieving the required performance levels. Thus, this chapter presents PRIMUS, a performance management architecture that aims to meet the requirements of low-latency and high-reliability in an adaptive way for s-health applications. As network slicing is central to realizing the potential of 5G networks, PRIMUS manages traffic through network slicing technologies. Unlike existing proposals, it supports device and service heterogeneity based on the autonomous knowledge of s-health applications. Next sections present the related works, the PRIMUS architecture, PRIMUS performance evaluation, results, and open issues.

4.1 RELATED WORKS ON PERFORMANCE MANAGEMENT

Facing the issues and difficulties regarding to achieve high-reliability and low-latency, few approaches have been proposed in the last years (de Sousa et al., 2019). We have identified well-known mechanisms to attain application/service requirements in the literature, such as resource reservation, traffic shaping, traffic separation, scheduling (a queuing mechanism), resource allocation, and the most recent network slicing. Each mechanism presents advantages and limitations. Sometimes these mechanisms are combined to achieve specific requirements. We discuss these mechanisms under the perspective of their feasibility in the s-health context. Hence, in this section, we review the main approaches and explain their characteristics and drawbacks. A summary of the works is given in Table 4.1. We classify the works based on the considered approaches, the towards requirements, and implementation environments in the table.

4.1.1 Reliability on Healthcare Systems

Healthcare systems involves different type of applications and services, such as remote patient monitoring systems, surgical robots, telemedicine, and conventional healthcare. In the reliability engineering context, there are different directions for reliability analysis of a healthcare system: organizational, human, hardware, and software (Zaitseva, 2010). Organization and human factors consist of system organizations managing and the human operators running them. Organizational and Human Reliability Analysis (HRA) examines the probability of industrial accidents, device errors, medical errors, error in results interpretation, error in communication with the patient, and the risk factor of the final diagnosis. It then identifies appropriate countermeasures to prevent and reduce the linked risks. In contrast, some methods allow estimating hardware and software part of a system. In healthcare systems, human errors are considered as a separate problem. Hence, there are two directions for reliability analysis in healthcare systems. The first direction is the reliability estimation of technical system part (hardware and software mainly). The second one is the examination of human errors by HRA methods.

Table 4.1: Network Performance Management Works

Reference	**Approach**					**Requirement**			**Implementation**				
	Reso.Alloc.	Traf.Shaping	Traf.Separ.	Scheduling	Netw.Slicing	Reliability	Latency	Others	Simulation	Experiment	Mathemat.Mod.	Abstract Level	Virtualization
Clark et al. (1994)		✓						✓		✓	✓		
Lloret et al. (2017)			✓			✓	✓		✓				
Fu et al. (2018)			✓					✓				✓	
Uddin and Nadeem (2016)			✓					✓	✓			✓	✓
Vergütz et al. (2017)			✓	✓			✓		✓				
Li et al. (2014)				✓			✓		✓				
Petrov et al. (2018), Wang et al. (2018a)				✓		✓				✓	✓		
Ge (2019), Kalør et al. (2018)				✓	✓	✓	✓				✓		
Yilmaz et al. (2015), Karimi et al. (2019)				✓	✓	✓	✓		✓				
Kurtz et al. (2018), Bouzidi et al. (2020)	✓			✓	✓		✓		✓				✓
Sciancalepore et al. (2017b,a)		✓		✓	✓			✓	✓				
Salhab et al. (2019)	✓			✓	✓			✓		✓			
Richart et al. (2017)	✓			✓	✓			✓	✓				
Salvat et al. (2018),Marquez et al. (2019)	✓				✓			✓	✓	✓	✓		
Husain et al. (2018)					✓	✓	✓					✓	

The authors in (Zaitseva, 2010) measured the probability of a system healthcare component to fail. These components involve hardware, software, organizational, and human. For this, the authors considered a modern healthcare system as a complex system with different states (Multi-State System - MSS): completely failed, partially failed, partially functioning, and perfect functioning. In order to analyze the reliability and measure the probability to fail, they exploited mathematical tools of Multiple-Valued Logic function considering the structural importance of each component. Structural importance determines the proportion of working states of the system in which the working i-th component makes the difference between system failure and its working. Results pointed out the component that most influence in the system to fail is organizational. The authors in (Levashenko et al., 2016; Zaitseva et al., 2017) analyzed the healthcare system reliability addressing uncertain and ambiguous data since this is a typical problem in healthcare reliability engineering. The structural function used in MSS to model the healthcare system does not take into account ambiguity. Thus, the authors used fuzzy decision tree functions. These functions are widely used in data mining for analysis of uncertain and decision making in ambiguities. Results showed that the reliability estimation of healthcare systems is possible using fuzzy decision trees. However, the scenario considered is about human errors during laparoscopic surgery, abstaining from network, software, and hardware reliability.

In general, the researches about reliability engineering are related to human and organization errors. In this context, and to the best of our knowledge, there are no studies using reliability engineering on communication or wireless networks. Generally, the studies estimate the probability of a hardware component to fail, such as computational or some hospital equipment (Zaitseva, 2010). Moreover, most of the studies encompass hospital functioning, analyzing the work of nurses and physicians. For instance, there are studies about the reliability of nurses' work (Oster and Deakins, 2018). Hence, reliability engineering is little related to communication reliability in a network context, having few relationships with our research.

4.1.2 QoS and Resource Allocation Techniques

A couple of approaches have been introduced for a few decades, and they usually include resource reservation and traffic shaping. Resource reservation is the most basic and static approach, and it requires communication between the applications and the network, indicating the requirements. For instance, the well-known Resource ReSerVation Protocol (RSVP), employed by Integrated Services (IntServ) (Clark et al., 1994), signalizes the requirements for each device, allowing single receivers to change channels and optimize bandwidth. However, such approach is not scalable since it needs to install multiple states about the reservations in each router. Internet providers commonly use traffic shaping since it controls the amount of data and matches the network traffic to the client-hired profile, *e.g.*, it coordinates resource consumption by applications. When a given application affects the performance of others, its traffic is delayed. However, such approach is specific for clients or one application type, not being scalable and flexible.

Nowadays, traffic separation and scheduling mechanisms predominate (Li et al., 2014; Silva et al., 2016; Vergütz et al., 2017; Lloret et al., 2017; Petrov et al., 2018; Fu et al., 2018; Uddin and Nadeem, 2016). Traffic separation classifies the network traffic according to the application or device that generated it, such as Differentiated Services (Diffserv) (Chan et al., 2006). Scheduling follows a queue mechanism (*e.g.*, first in, first out) to transmit the packets according to some priority rule. Network traffic monitoring employs traffic separation since it analyzes the traffic to learn about application/device behavior. For instance, there are studies using machine learning techniques (ML-techniques) to manage network traffic in the 5G network and achieve requirements (Fu et al., 2018; Uddin and Nadeem, 2016). Hence, the authors have employed ML-techniques to learn about network traffic to adjust some resources and enable various policy-driven network management. However, they have proposed only an abstraction-level architecture (Fu et al., 2018). Although the authors have applied virtualization techniques to enable network management, they have considered only mobile applications from smartphones, not scalable for different scenarios (Uddin and Nadeem, 2016).

Moreover, in general, the studies rely on network technology. Some studies also employed queuing mechanism to give special treatment for packet with high priority. For instance, for IoT, Li et al. (2014) proposed a service-oriented scheduling model based on application and network layers classified into delay-sensitive and best-effort categories. Wi-Fi, by the Enhanced Distributed Channel Access (EDCA), modifies some parameters (*e.g.*, control window) to give priority in the medium access. EDCA has four access categories, including voice and video (Silva et al., 2016). However, nowadays, there is a more significant variety and diversity of types of applications, ranging from streaming or game to healthcare. Hence, such works do not support a wide range of applications. The authors in (Vergütz et al., 2017) created a new access category, especially to medical alerts. The results achieved around 200 ms of latency for medical alerts. S-health applications with restricted requirements do not support such latency. Therefore, improvements are crucial to achieve specific requirements.

In cellular network context, there are studies concerned about critical/sensitive applications (*e.g.*, alerts and mission critical). The authors in (Lloret et al., 2017) presented a 4G-5G architecture for s-health that classifies data vital into normal and abnormal by ML-techniques. Abnormal data receives a high priority. Petrov et al. (2018) presented a 5G virtualization-based architecture for end-to-end reliability of mission-critical traffic in an urban scenario. Through virtualization techniques, the architecture gives priority for all mission-traffic over other types of traffic. Results showed a cost since the high data rate of critical sessions brings degradation to other users' services. The authors in (Wang et al., 2018a) proposed a scheduler algorithm and filtered more relevant healthcare data by different levels of priority. Although the authors have reduced the loss rate, the latency is still high. Moreover, these studies did not consider the management and customization of abnormal or critical data from different users. The architectures are not adaptable and easy-customized for different scenarios.

In traffic separation, conventional technologies, such as virtual local area networks (VLAN), provide isolated networks for a specific application. However, in VLAN, it is difficult to control the availability and performance guarantee since the number of users operating in the network affects the entire system performance (Alliance, 2016). Therefore, the approaches mentioned above are still insufficient to accommodate the diversity of smart application requirements. Based on the literature, there is a diversity of network performance techniques due to unique network properties. Some applications cannot communicate their requirements, and the mapping between application requirements and network capabilities is not standard (Marquez et al., 2019). Hence, the appearance of some mechanism configurable between the applications and the network presents itself as a compelling alternative. Moreover, network virtualization can help achieve requirements since it controls virtual resources and supports the diversity of application, service, network control, and management (Alliance, 2016). The new paradigm, network slicing, is based on network virtualization. It has a considerable potential to achieve requirements over heterogeneous networks, especially for s-health applications that employ different network technologies in their communication (Khan et al., 2020).

4.1.3 Network Slicing Approach

The emergent network slicing technology has been used in numerous studies for network performance management. Unlike static and exclusive policies, network slicing creates slices (sub-networks) by virtualization techniques over the same physical infrastructure and allows users to share virtual network resources. There are main types of slices, such as URLLC and eMMB. S-health applications, such as remote surgery, require URLLC between wearables and infrastructure networks (5GPPP, 2015; Cisotto et al., 2020). However, network slicing impact on reliability and latency performance has been limited, and improving reliability methods have not been investigated on s-health (Afolabi et al., 2018). Generally, studies on network slicing implement some network slice type, following some priority queuing (Yilmaz et al., 2015; Kalør et al., 2018; Ge, 2019; Karimi et al., 2019), present an architecture (Lloret et al., 2017; Husain et al., 2018), or propose an end-to-end slice approach (Petrov et al., 2018; Salvat et al., 2018). Hence, this subsection presents studies about network slicing following these researches lines.

There are works combining network slicing with another performance management mechanism. For instance, the authors in (Ge, 2019) proposed a network slicing solution with priority queuing to provide ultra-reliability and low-latency in vehicular networks. They created three URLLC slices to control vehicle-state reports (*e.g.*, speed, and location), event-drive (*e.g.*, emergency), and user information. The first slice is used to provide the state of the vehicle, such as speed and location, to avoid collision. The second slice contains specific events, such as road information updates. The third slice is data from vehicle users. Hence, latency requirements distinguish the slices type. Results using URLLC improved reliability and latency, reaching values around 1% and 1ms, respectively. However, vehicle density still increased latency. The authors in (Kalør et al., 2018) used network slicing with priority queuing to manage the different requirements, such as reliability and high data rates. They followed three slicing schemes: static telegram allocation, shared telegram, and telegram overwriting. Static telegram allocation contains frames of one single application. Shared telegram has frames from several applications. Telegram overwriting contains frames overwrites by applications with higher priority (*e.g.*, alarm device frames can overwrite feedback control frames). Each slice consists of a queue with different priority. However, the concept of network slicing based on NFV and SDN was not proposed. The authors only introduced the idea of network slicing following priority queuing.

The authors in (Karimi et al., 2019) presented a resource allocation method and packet scheduling for a mixture of URLLC and eMBB traffic. The method explores the radio channel time-frequency variations. The urban 5G scenario contains users transmitting URLLC and eMBB packets, where 5 million of URLLC packet were transmitted in the simulations. For low load with 4-8 Mbps, the method reached 1.5 ms, and for load with 14 Mbps reached 4 ms. Although the authors achieved many improvements, they sliced the network on the physical layer, following channel time-frequency, which increase the complexity. Moreover, the authors did not evaluate the reliability rate. The authors in (Yilmaz et al., 2015) discussed the requirements on low-latency and high-reliability for a factory automation use case, where sensors and machines need to communicate with each other. Hence, the authors presented the challenges for mission-critical machine type communication. Network simulations were presented, achieving 3ms of latency. However, a perspective on how to construct an unified service system as well as understand the implications of critical traffic on serving other users is not available. Furthermore, network slicing and network virtualization were not used or mentioned. Hence, the authors did not implement a realistic network slicing since they did not use virtual network functions and virtual resources.

The authors in (Petrov et al., 2018) presented a 5G architecture for end-to-end reliability of mission-critical traffic in an urban scenario. Hence, the scenario contains several autonomous vehicles and ambulances (represent the mission-critical traffic). The NFV-based architecture prioritized all mission-traffic over other types of traffic. The authors analyzed the impact of supporting mission-critical traffic. Results showed that there is a cost since both high data rate of critical sessions and high velocity of the target user bring considerable degradation to the service of other user sessions. However, the solution is specific for 5G. Hence, it is necessary to evaluate this scenario with slices beyond-5G to optimize network deployment configurations in heterogeneous environments. The authors in (Kurtz et al., 2018; Bouzidi et al., 2020) presented a network slicing approach for critical communications on 5G and end-to-end delay. Based on NFV, the authors proposed a queuing driven approach focusing on different applications. However, it was only considered a smart grid as a critical communication, abstaining from healthcare applications. Moreover, although virtualization techniques improve flexibility and automation, some works pointed out that to efficiently manage network resources and meet performance requirements, network function virtualization techniques need mechanisms to predict performance degradation (Manousis et al., 2020). Network data analysis and packet classification also assists network functions to determine which action they should take for each incoming packet (Rashelbach et al., 2020). Therefore, improvements in virtualization techniques also are necessary to manage network resources efficiently.

The authors in (Sciancalepore et al., 2017b; Caballero et al., 2017; Marquez et al., 2019; Salvat et al., 2018) discussed the resource allocation in network slicing. Because each application has service customization, *i.e.*, each application assign to each slice fully dedicated resources, which brings significant consequences in terms of resource management. The authors in (Caballero et al., 2017) tailored the slices for each tenant, allowing flexible resource allocation. Each tenant communicates their preference to the infrastructure and have resources allocated according to them. However, it is crucial resource control mechanisms since tenant can monopolize resources, avoiding the use of resources for other tenants. The authors in (Sciancalepore et al., 2017b) proposed a traffic analyzes per network slice, an admission control decision algorithm, and a scheduler to give priority to traffic according to the SLA. Hence, the authors defined six traffic classes, where streaming traffic consist of the most priority traffic. However, this is not a real scenario since the current network traffic has more priority traffic than streaming.

The authors in (Salvat et al., 2018; Marquez et al., 2019) adopted a data-driven approach to quantify the efficiency of resource sharing. When instantiating a slice, the operator needs to allocate sufficient computational and communication resources to the associated slice. Hence, they adopted an overbooking policy, *i.e.*, the operator allocate more resource than have available for the users, believing that some user will no use all resource allocated. However, when some traffic not receive resource, the packet data is considered as best-effort traffic or is dropped. This approach is not suitable for smart healthcare scenario since critical applications can not have packets dropped. Moreover, if applications do not follow agreements like SLAs, some applications can request more resources than necessary, harming other applications.

Other studies propose architectures to achieve applications requirements and allocate resources. The authors in (Husain et al., 2018) presented a mobile edge architecture for IoT to customize required network resources. In the architecture, the network slicing acts in the gateway, as close to users as possible, to minimize latency. However, the authors did not present any results or implementation, only abstraction-level architecture. The authors in (Lloret et al., 2017) presented an architecture for smart continuous monitoring of patients with chronic diseases. The architecture has wearable devices collecting vital signs and transmitting to a smartphone through Bluetooth. Then, the smartphone sends the vital data to a database in the hospital through 4G and 5G networks. The authors employed 4G and 5G to evaluate in both networks. Moreover, ML algorithms classified data vital into normal and abnormal. Hence, when and abnormal data is detected, the architecture sends to the physician immediately an alert. To reduce the false positive, the physician receives the alert and verifies if it is genuinely an alert. Through network simulations, the packet loss rate were decreased in 5G, when compared with 4G. However, the authors did not consider network slicing technology.

Finally, there are studies combining network traffic monitoring and management network mechanisms to achieve QoS requirements (Fu et al., 2018; Uddin and Nadeem, 2016). These studies follow a similar idea of this thesis's objectives. The authors in (Fu et al., 2018) proposed to use machine learning algorithms to manage network traffic in 5G networks. The authors justified the use of ML-techniques because such techniques will enable better network performance, better reliability, and more adaptive systems by drawing new insights from the networks and predicting the network traffic conditions and the users' behavior, enabling smarter decisions. Hence, they proposed to use deep learning concepts from ML-techniques to learn about network traffic in order to adapt some resources. However, they proposed only an abstraction-level architecture.

Based on the literature, network slicing has the potential to brings enormous benefits for different applications because it provides resource allocation automatically based on application requirements. Moreover, it an end-to-end approach involving heterogeneous scenarios, networks, and requirements. Several studies are addressing the URLLC slice type, which involves critical applications with stringent requirements, *e.g.*, remote surgery, and autonomous vehicles (Ge, 2019; Kalør et al., 2018; Karimi et al., 2019). However, to the best of our knowledge, there is no study combining URLLC and smart healthcare. Moreover, generally, the studies address an individual characteristic of the network slicing concept, *i.e.*, some studies address queuing priority; other studies propose a resource allocation method. Nonetheless, there are few studies involving an end-to-end network slicing approach (Salvat et al., 2018). Hence, next we propose an architecture for performance management in the s-health context based on network slicing.

4.2 THE PRIMUS ARCHITECTURE

This section introduces PRIMUS, *an architecture for ada**P**table sma**R**t healthcare appl**I**cations require**M**ents based on network Reso**U**rce **S**licing*. Unlike the literature, PRIMUS adapts network virtual resources and routes according to the smart healthcare application demands. Hence, it identifies the applications and configures the network on-the-fly. We assume that PRIMUS receives as input the application identified in the MOTIF method (Vergütz et al., 2019), assisting our framework proposed in (Vergutz et al., 2020). Note that the MOTIF method instanced the FLIPER framework to identify the traffic applications. Connect the PRIMUS architecture with the MOTIF method and FLIPER framework is out of this thesis scope.

PRIMUS considers applications with critical requirements, such as s-health intensive care and remote surgery. Both applications require low-latency and high-reliability. Reliability in this work means successfully transmitting and delivering packets without exceeding the maximum latency supported by the application (5GPPP, 2015; Alliance, 2016). However, unlike the literature, PRIMUS does not refrain from other applications such as telemedicine (Wang et al., 2018b). Table 2.1 presents the main s-health applications and requirements (Alliance, 2016; 3GPP, 2018; Homma et al., 2019). In general, almost all s-health applications require reliable communications since patient health comprises a critical use case (Alliance, 2016). The most flexible applications, such as fitness and calorie counting, have no stringent requirements. Body temperature and diabetes monitoring consist of daily continuous monitoring applications. These applications collect specific vital signs and issue medical alerts when critical situations occur (*e.g.*, extreme body temperature values). They support up to 125 ms latency (Movassaghi et al., 2014). In contrast, telemedicine applications demand high data rates and low-latency (10 ms). Therefore, PRIMUS needs to meet these requirements automatically through network slicing. The next subsections detail the network model and the PRIMUS architecture.

Application Type	**Latency**	**Bandwidth**	**Reliability**
Telemedicine	$\leq$ 10 ms	$\geq$ 200 Mbps	-
Remote Surgery	$\leq$ 1 ms	10 Mbps	99.999%
Diabetes/Temperature	$\leq$ 125 ms	10 Mbps	-
Emergency Ambulance	$\leq$ 1 ms	10 Mbps	99.999%
Critical Care Monitor.	$\leq$ 1 ms	10 Mbps	99.999%
Remote Monitor.	$\leq$ 10 ms	$\geq$ 50 Mbps	-
Smart Medication	$\leq$ 125 ms	$\geq$ 20 Mbps	-

Table 4.2: S-Health and Performance Requirements

4.2.1 Network Model

PRIMUS takes into account a network model, as illustrated in Figure 4.1. The model encompasses the communication between smart devices and environments, such as smart home and smart hospitals, supporting new applications, particularly s-health. In the smart environments, the network infrastructure contains different devices generating network traffic to a gateway by wireless connection (*e.g.*, Wi-Fi, Bluetooth). The edge devices involve smartphones, augmented reality on video games, laptops, smartwatches from the smart home, ambulance, wearable, and sensors from the smart hospital. We can have usual devices in both smart scenarios, *e.g.*, smartphones. Each device contains applications running on it. In the core network, devices involve base stations, routers, switches, and others. Such network model supports network slicing and virtualization techniques (NFV and SDN).

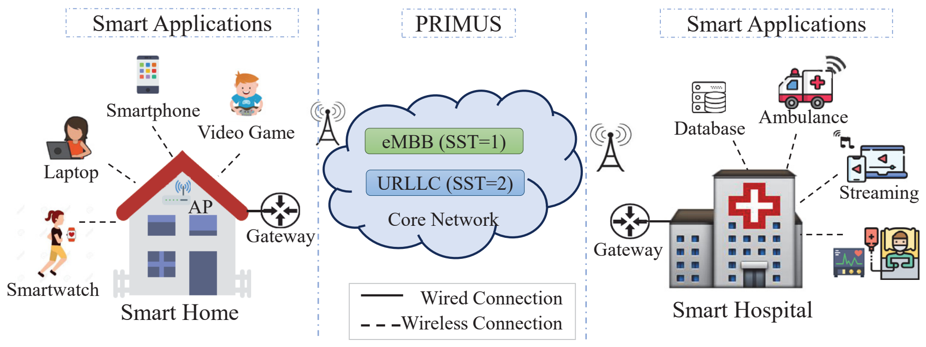


Figure 4.1: The PRIMUS Network Overview

Formally, let $N = (1, ..., n)$ be the set of tenants (*e.g.*, end-users) that generates network traffic from a given application app, where $App = (1, ..., app)$ is a set of applications. Hence, let $v_{n,s}(t)$ be the traffic of an application associated to a slice s at tenant n during instant t. Each network slice is responsible for traffic messages exchanged between hosts, considering application requirements. Hence, the slice s needs to achieve a set of requirements $Q = \{q_1, q_2, ..., q_n\}$ specific from the application app. A set of available network resources $R = (1, ..., r)$, such as bandwidth and computing, are necessary to achieve such requirements. Therefore, a given slice s is composed by virtual network resources $r_1, r_2..r_n$, network traffic $v_{n,s}(t)$ and an associated user n and application app that demands some requirement q. Moreover, each slice has one controller c, from the set of controllers $C = (1, ..., c)$, managing the resources. One slice can have more than one application and user associated at the same time. However, organizations have defined that at most eight network slices may serve one end-user at a time (3GPP, 2018; Alliance, 2016). Thus, let u_{num} be the respective user identification u using num network slices in parallel. When $num >= 8$, PRIMUS removes the user u from the slice s and reallocates the resources for another slice. In case an app uses more resources than allowed, or it is harming other app, PRIMUS defines a threshold duration time t_d. Hence, the app has a timing t_d until it makes the resource available. Otherwise, when exceeding t_d, PRIMUS removes the slice and the application associated, *i.e.*, it removes the associated traffic, application, and user, making resources available.

PRIMUS marks all slice s created with the Slice/Service Type SST and Slice Differentiator SD. The SST refers to the network slice behavior in terms of features and services. There are three main types of SST: *(1)* eMBB with high data rates, *(2)* URLLC with more critical data traffic, and *(3)* mIoT with massive amount of users and devices. Thus, s-health intensive care and remote surgery applications fit into the URLLC slice type. Based on this, the SST values are $SST = 1$ for eMBB, $SST = 2$ for URLLC, and $SST = 3$ for mIoT. Note that URLLC slices can not be removed because it involves critical applications. Thus, when occurring resource overloading, PRIMUS reallocates network resources or, in extreme situations, removes a mIoT or eMBB slice. The SD is optional information that complements the SST to differentiate among multiple network slices of the same SST. For instance, SD serves to identify extreme critical care applications from typical s-health applications. Finally, PRIMUS has full visibility and control of the end-to-end slice and its performance to maintain the overall requirements of the applications. PRIMUS computes a set of performance variables $P = (1, ..., p)$ (*e.g.*, packet loss rate and latency) to analyze the slice performance and create dynamic virtual routes. Such performance variables define the state of the physical structure for establishing or adapting network slices to meet predefined requirements.

4.2.2 The Architecture Details

Figure 4.2 illustrates the PRIMUS architecture that follows the modules: *1)* Decision Making, *2)* Network Slice Management, and *3)* Resource Management. PRIMUS considers the application context to create and allocate network slices. Thus, different tenants u (users) generate network traffic. Each one belongs to a use case and demands specific requirements Q. Tenant 1 (u_1), 2 (u_2) and 3 (u_3) refer to devices from industry 4.0, smart hospitals, and augmented reality, respectively. While tenant u_1 requires ultra-low-latency, tenant u_3 demands high bandwidth. Hence, PRIMUS needs to monitor the network traffic to establish and control the slice to support the requirements, especially the requirements of s-health. Based on the knowledge about the application context identified on MOTIF method, PRIMUS creates the slices routes and adapts resources through NFVs. It enables devices to logically and separately share numerous network services without overloading switches. Therefore, PRIMUS considers the highest virtualization level, coordinating devices, network functions, and resources with greater flexibility. The next subsections detail the three modules of PRIMUS.

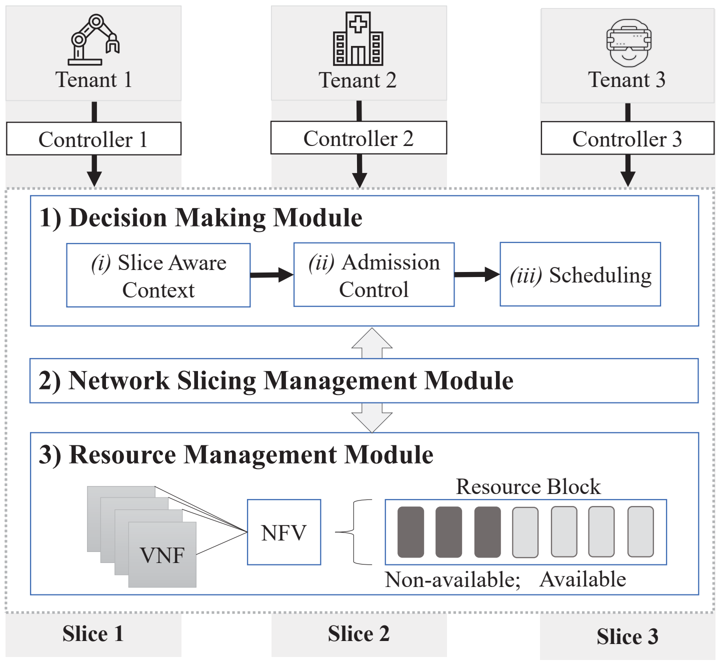


Figure 4.2: PRIMUS Architecture

4.2.2.1 Decision Making Module

This module takes decisions on slice configuration, slice admission, and slice scheduling following three steps: *(i)* Slice-Aware Context, *(ii)* Admission Control, and *(iii)* Scheduling. These steps define the creation of a new slice or the allocation of a pre-configured slice. There are pre-configured slices for s-health since urgent care application supports only 1 ms of latency. Thus, the slice-aware context step gets information about the application context and its requirements. Assuming the application identification, through announce control messages, PRIMUS informs the requirements demanded by each application (we detail control messages in Table 4.3). Thus, the first step analyzes and stores this information to make future decisions. The information about the requirements enhances the network resources utilization efficiency, once the infrastructure provider can get the network slice demands on the slice-aware context. The slice-aware context stores data regarding applications, requirements, and end-users. It stores information about identical applications but from different end-users since several users may generate traffic, simultaneously. In such situations, PRIMUS allocates equal or the same network slice to meet the same application needs. Each slice allows up to eight users per time. The resources are further pre-allocated following the slice-aware to provide the admission control policy.

The admission control step needs information about the available resource and priority policy. Thus, it offers information to the scheduling step to analyze the priority levels and, jointly to resource management, to know the available and allocated resources. Moreover, for the admission control policy, we assume that the resource demands are accurate. Thus, in the first moment, the admission control analyzes the network slice type, *i.e.*, URLLC, eMBB, or mIoT, to decide what network resource each slice will receive since URLLC has priority over eMBB and mIoT. The scheduling step stores and updates the priority information. As URLLC transmits critical network data, it receives the highest priority. eMBB and mIoT receive the second and third priorities, respectively. Hence, there are three priority queues: 1 - URLLC, 2 - eMBB, and 3 - mIoT. For URLLC, there are pre-configured slices to not increase the latency since slice creation and configuration take time.

4.2.2.2 Network Slice Management Module

The Network Slice Management Module comprises a fundamental component of network slicing, being responsible for managing and orchestrating network slices. It allows, configures, and determines slice policies, such as restricted access by a given user, and configures the network resources used by it. Hence, the module has visibility and control of end-to-end slices and the performance of each slice. Thus, it controls the overall functioning of the slices. Moreover, it controls the slice life cycle, *i.e.*, it controls the preparation, configuration, supervision, and deactivation phases, as illustrated in Figure 4.3

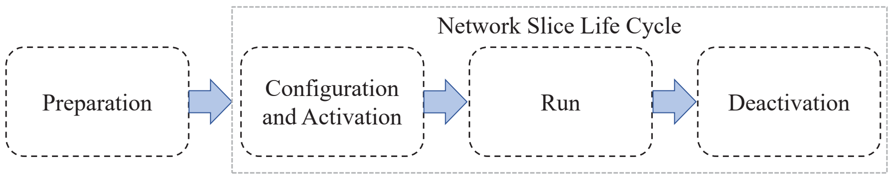


Figure 4.3: Network Slice Life Cycle

The preparation phase consists of creating and verifying the network slice, since it prepares the necessary network environment to encompass the network slice created, such as virtual network functions. Besides, it verifies if there are running slices of the same type (*e.g.*, eMBB and URLLC). It marks each slice with the Slice Differentiator *SD* to add extra information, if necessary. The configuration phase organizes and configures all shared and dedicated network resources. Then, it activates the slice. In the run phase, PRIMUS manages the network traffic transmission and observes the communication services and resources. If any application requirements are compromised, changes may be made, such as the virtual route used, slice topology or capacity, associating or disassociating virtual network functions. These changes are intended to improve slice performance and do not harm the application and the end-user. Finally, the deactivation phase makes available used network resources. Then, the created slice no longer exists (3GPP, 2018).

4.2.2.3 Resource Management Module

It manages the virtual resources of the network infrastructure and determines the association of resources to the slices. To be aware of available and utilized virtual resources, resource management communicates with the NFV, which controls the creation of virtual network functions. NFV controls the network's physical and virtual resources by analyzing and controlling VNFs, as illustrated in Figure 4.2. Hence, the PRIMUS allocates the necessary network resources for each application. This resource allocation comprises VNFs that establish better end-to-end routes. Applications with strict requirements, such as intensive care monitoring, are given priority over network resources. When there are insufficient or unavailable network resources, the resource management analyzes the slice types running to delete, share resources, or split traffic across different routes to free up new resources for the restricted application. The resource management computes performance metrics, such as packet loss rate and latency, to get information about NFV performance.

Finally, PRIMUS has two main types of control messages: announce and monitor, as shown in Table 4.3. Both control messages assist in controlling and establishing the slice. Announce messages inform the application type and its requirements based on the application identified. Note that the MOTIF and PRIMUS connection is not part of our scope. This message is essential to announce s-health applications that require URLLC slice type. In contrast, the monitor message updates the information about network resources availability. Thus, PRIMUS changes information about sharing, availability, and allocation of network resources through monitor control messages. The PRIMUS modules receive updates by such messages. Both control messages follow the TCP format, and they are exchanged between PRIMUS modules on a predetermined network port. Therefore, PRIMUS controls the network resources allocation and synchronizes the information.

Control Message	**Content**	**Description**
Announce	Application type and requirements	Inform if application is s-health
Monitor	Network resources (*e.g.*, resources use)	Inform about network resources availability

Table 4.3: PRIMUS Control Messages

4.3 PERFORMANCE EVALUATION

The PRIMUS architecture was evaluated on Mininet-WiFi Network Emulator (Fontes and Rothenberg, 2016) that supports virtualization techniques and network slicing. The emulator runs on the operational system Linux Ubuntu 18.04 LTS over a virtual environment (VirtualBox tool) with Intel Core i7 8500H processor and 8GB of RAM. The emulation comprises of an emulated wireless network infrastructure containing two switches, one access point, three wired devices, three wireless devices, and three servers, as shown in Figure 4.4. The network was virtualized and sliced based on Openflow rules by Libera Hypervisor (Yang et al., 2020), and the tenants slices were managed by ONOS based SDN controllers (Foresta et al., 2018). The six devices are divided among the three network service/slice types (*SST*): URLLC, eMBB, and Best Effort (BE). Each device generates synthetic traffic simulating the specific *SST* to a server by iPerf tool v2.0.5[1]. One way latency was captured by python scripts using the Scapy v3.7 library. We call each *SST* as its corresponding traffic, *i.e.*, URLLC, eMBB, and Best Effort slices for each traffic.

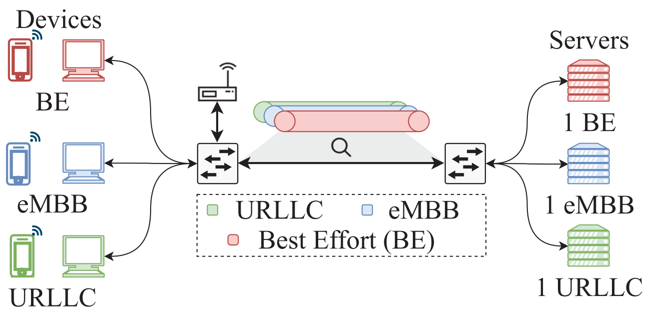


Figure 4.4: Emulation Scenario

The evaluation scenario allows the management of slices and bandwidth use. Hence, we have compared the network performance over two scenarios: *(i)* considering network slicing isolation and *SST* priority (called **NSI scenario**), and *(ii)* without slicing isolation (called **no-NSI scenario**). The NSI scenario implements the slicing isolation with one specific slice for each slice type (*SST*), following a priority queue where URLLC has the highest priority. In contrast, the no-NSI did not implement isolation. All traffic generated compete to access the same slice (one link). The network slices compete with each other because they share the same communication physical links. Hence, it was generated 1Mbits/s, 15Mbits/s, and 15Mbits for URLLC, eMBB, and BE, respectively. For BE traffic was created threads of traffic generation to increase the competition and the amount of traffic.

In the scenarios, by the iPerf tool, all devices start the workload generation together over 100s. As PRIMUS mechanism aims to achieve s-health requirements, we have considered network traffic similar to s-health applications. The characteristics of eMBB slice traffic correspond to telemedicine application, whereas the characteristics of the URRLC slice traffic corresponds to critical care monitoring, as shown in Table 4.2. At every 0.3 s interval, network traffic was generated. Moreover, we have generated best effort traffic to massive concurrent traffic to consider a more realistic scenario. Hence, the URLLC traffic has a lower throughput compared to the eMBB and BE throughput (Alliance, 2016). However, the URLLC requires stringent requirements than others slices. Thus, the tenant responsible for the URLLC service requested to network hypervisor for higher priorities in the processing of OpenFlow rules. We have computed the average TCP latency packet loss rates as the performance metric. The one way latency was computed by the difference between the timestamp of the message received by the server and sent by the host ($received\_timestamp - sent\_timestamp$). We have extracted the metrics from the three servers that receive the traffic generated by the devices. These metrics were measured using scripts in Python v2.7 with Scapy library. Each plotted result is an average of 33 repetitions for both scenarios. Results consider the latency metric.

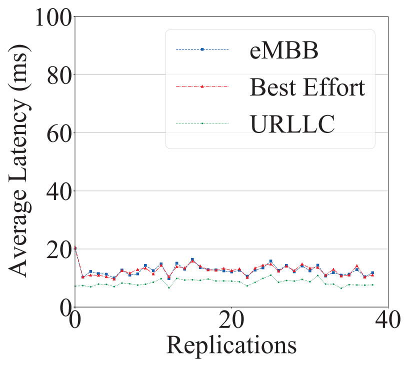
4.3.1 Results

The results comprise both scenario settings, NSI and no-NSI scenarios, with and without network slicing isolation. Moreover, in both scenarios was considered different values for bandwidth. The bandwidth evaluated was 100Mbit/s and 1Gbit/s aiming to evaluate network slicing and PRIMUS in low and high competitive resource scenarios. The results employed network traffic

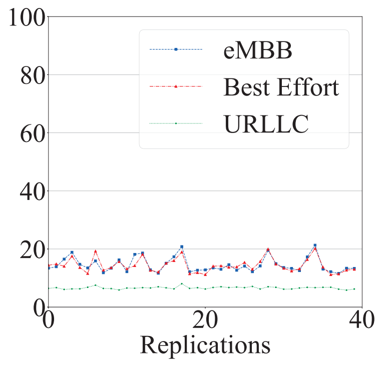
[1]iPerf Tool: https://iperf.fr/. Accessed on Dez/2021.

from 6 devices competing for the same slice (no-NSI) and separated on three network slices (NSI) in the emulated network environment. Network traffic from the same slice type competes to transmit the traffic in the slice. These scenarios assist in showing the network slicing efficiency on isolating and controlling the physical network to share the resources, even in stringent concurrent scenarios. Hence, results are presented and discussed following a critical performance analysis.

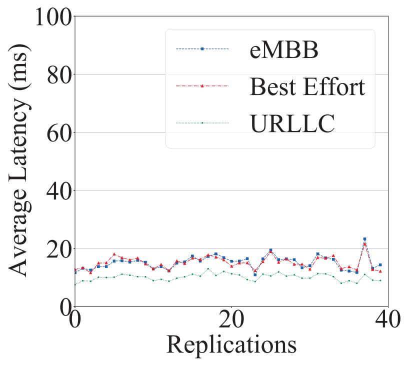
Figure 4.5 presents the results for latency on both scenarios. Particularly, Figures 4.5(a) and 4.5(b) show the average latency with 1Gbit/s of bandwidth for scenarios with and without network slicing isolation. For all slice types, the latency has reached values around 10 ms and 25 ms since there is enough bandwidth to transmit all network traffic. In the NSI scenario, the URLLC traffic has reached better latency results, such as 8 ms, than in the no-NSI scenario once the first one implements network slicing isolation and priority. eMBB and BE have reached values higher in NSI than in no-NSI. However, they have still presented low-latency values, *e.g.*, 15 ms and 18 ms. Figures 4.5(c) and 4.5(d) present the average latency with 100Mbits/s of bandwidth, *i.e.*, more congested scenarios. The average latency for both scenarios, NSI and no-NSI, achieved similar results as the 1Gbits/s analyses. However, PRIMUS has achieved lower and stable latency values for URLLC in NSI scenario, such as 5 ms.



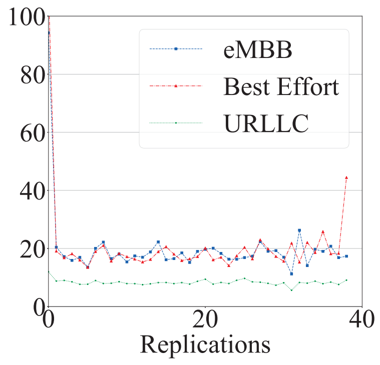
(a) no-NSI (1Gbits)



(b) NSI (1Gbits)



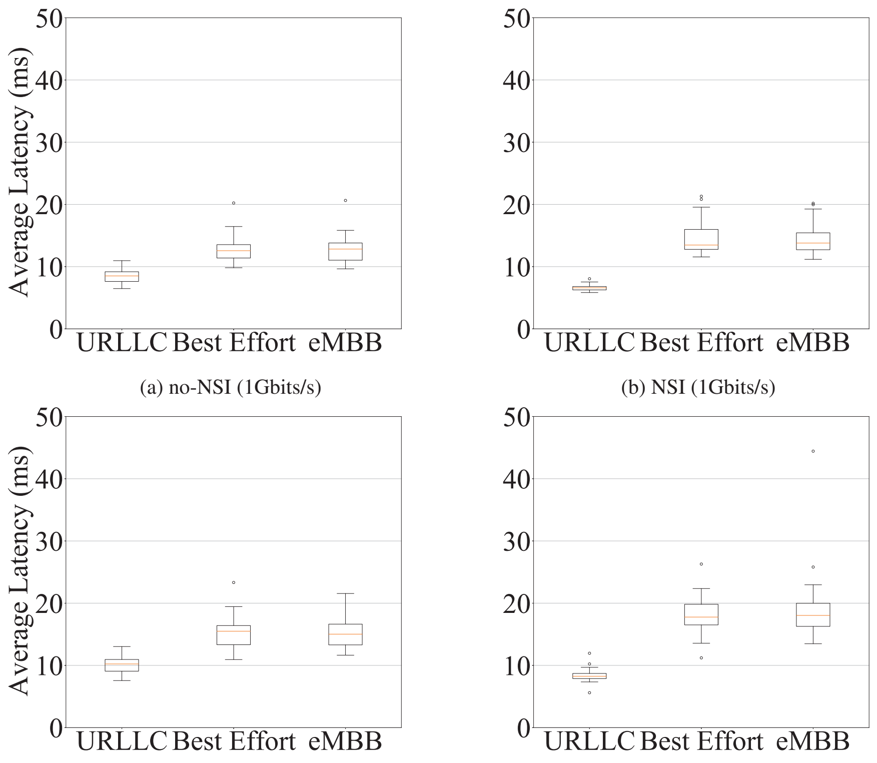
(c) no-NSI (100Mbits)



(d) NSI (100Mbits)

Figure 4.5: Average Latency

More details on the improvements in latency results are shown Figure 4.6, considering the same scenarios and bandwidth values. Particularly, Figures 4.6(a) and 4.6(b) show the average latency on NSI and no-NSI with bandwidth available, and Figures 4.6(c) and 4.6(d) present the average latency in congested scenario. While BE and eMBB traffic achieved latency values around 15 ms and 20 ms, URLLC reached 6 ms. In a stringent scenario without available bandwidth, it is clear the benefits for URLLC. In Figure 4.6(d), the URLCC traffic presented average latency values lower than 9 ms, while eMBB and BE reached values higher than 20 ms. Therefore, PRIMUS improved the average latency of URLLC, not harming considerable other network traffic. Moreover, URLLC did not present network packet losses.



(a) no-NSI (1Gbits/s)

(b) NSI (1Gbits/s)

(c) no-NSI (100Mbits/s)

(d) NSI (100Mbits/s)

Figure 4.6: Average Latency

Based on the results, PRIMUS was able to manage network slicing, network resources availability, and application requirements. Results have met low-latency even with different applications transmitting data simultaneously and stringent concurrent traffic. The high data rates transmitted on the physical link with restricted bandwidth and physical bottleneck did not interfere with PRIMUS performance. Although PRIMUS has achieved low-latency results, improvements are still necessary because the URLLC slice type demands only 1 ms of latency. However, such latency depends on efficient network physical infrastructure. Even in our network scenario, PRIMUS reached less than 125 ms of latency (value required by daily s-health applications). It is necessary to research latency since there is a delay for the controller to interact with other virtualized components. Moreover, differently from the literature, PRIMUS contributes to the implementation of a virtualized scenario with network slicing. The virtualized implementation is a potential solution to apply in the 5G environment since it uses network slicing. For future analysis, a realistic smart hospital scenario containing many devices can be considered to evaluate PRIMUS scalability. However, the network slices and network traffic management used in PRIMUS will assist in network traffic concurrence since, in our results, the slice isolation improved traffic management. Therefore, the results represent success in interrelating the set of network components by automatically analyzing and managing the network resources. This work contributes to the state-of-the-art offering a virtualized scenario implementation with network slicing, network traffic management, and improvements on performance requirements to support s-health. However, there are open challenges, such as reducing latency, analyze packet loss, improving reliability, and validating the scalability considering different combinations of various s-health applications and more devices.

4.4 DISCUSSIONS AND LIMITATIONS

The PRIMUS architecture followed network slicing with virtualized scenario for smart healthcare applications. Over the last years, there has been a significant number of attempts in developing architecture designs for smart applications, 5G, and IoT networks. However, there are outstanding problems and challenges in the network slicing and smart applications fields, mainly considering s-health applications requirements, such as ultra-low latency. Hence, several works proposed architectures to support smart applications demands. Nonetheless, the architectures still present problems and open issues regarding the implementation of network slicing and s-health. Therefore, this section presents the open issues, challenges and opportunities to point out future directions.

4.4.0.0 Reliability

High reliability for s-health is essential. It encompasses network communication without failures, data and services available as long as possible, and the support for service operation. For instance, unreliable network quality can cause a high rate of packet loss and errors, which can lead to an erroneous diagnosis by the health professional (De la Torre Díez et al., 2018). However, ensuring reliability is complex, and it is not straightforward. In this context, adopting a network slicing approach for s-health would be beneficial in two ways. First, it makes possible to adapt network resources and services when a network failure occurs, since network slicing has automation and isolation characteristics. Moreover, data analytic of network traffic provided by network slicing joint to fingerprinting techniques extract network behaviors to assist in resources allocation. However, there is an issue related to failure detection, because network slicing is not prepared to detect a network or data failure.

4.4.0.0 Performance

S-health applications require performance measured by bandwidth, loss rate, and end-to-end delay, which increase in a huge proportion the complexity. End-to-end delay is very important for critical care monitoring applications, supporting only 125ms of latency. Considering network slicing, critical life applications only support 1ms, being very difficult to reach such latency in traditional networks. Hence, virtual network slices with network resources will provide specific services to reach such latency. However, the slice deployment time and the slice transition time (*e.g.*, time is taken to change the applications for another slice) can be critical for some s-health applications. Such cases can be supported by special mechanisms that contribute to the minimization of these times. These mechanisms can include the creation of functions shared by multiple slices and/or deploying a priori certain slices and keeping them in a “frozen” state, reducing the use of resources. In order to assist end-to-end latency, it is important to fingerprint the network traffic to know the necessary services. However, two important issues related to s-health fingerprinting are the small amount of traffic and the collect ground-truth information. For the first one, it is necessary to fingerprint a s-health application with higher accuracy, even with small amount of traffic. For the second, in general, the ground-truth is used to fingerprint correctly the application. In the literature, studies applied the MAC address or payload inspection techniques to collect the ground-truth. Nonetheless, sometimes the devices MAC address are unavailable, and the payload inspection techniques violate the user’s privacy. Hence, it is indispensable advances on network slicing to assist fingerprinting and end-to-end delay.

4.4.0.0 Security

Security and privacy protection are certainly important for s-health applications, because they use location, personal, and context information of users. Smart health applications are susceptible to various types of attacks, such as patient tracking, side-channel attacks, and denial of service. The data obtained through network data analytic and fingerprint techniques (ML-algorithms) can be used by the attackers. Network traffic used to fingerprint a s-health device (*e.g.*, packet size) can be used by side-channel attacks to infer information about the users. In contrast, ML-algorithms can be used to solve challenges in security of network slicing such as to control the network slice behavior. Then, when the behavior changes, the system could trigger an alert. However, some attacker can mimic the s-health traffic behavior. In this case, authentication techniques jointly with fingerprinting would improve security. Another issue is related to network slicing isolation, since usually security is linked with the concept of isolation. An important opportunity refers to inter-slice isolation (Sathi et al., 2018). It would assist to guarantee an attack on a slice will not affect other network slices, and to avoid the attack propagation when using shared functions by the "cascade effect". There are works about network slicing related to security requirements (Li et al., 2017b; Ni et al., 2018), the authors proposed a service-oriented authentication framework for IoT services. The framework has a slice selection mechanism to select proper network slices for data forwarding, and hide the accessing service types of users. Moreover, it is expected that network slices will have different security levels and policies, once they could be managed by different providers. The challenge is how to enforce network slice security in the case that a network function or slice is compromised when infrastructure from different providers is used.

4.4.0.0 Cost

S-health applications have cost and complexity restrictions, once they can consume energy and memory from the devices. Hence, there are concerns related to the trade-off between the complexity and application latency. Although network slicing creates slices on demand and customized according to the needs of specific applications, the slice creation, management and isolation can increase in a huge proportion the complexity, and consequently the cost. It is expected to be huge the number of slices operating in network infrastructure, which make difficult the network management Nonetheless, the management issues grow not only with the number of slices, but also depends on their complexity. In some cases, the delegation of the slice management to industry application owner can be required. However, there is no doubt that the network slicing brings additional complexity and cost.

4.5 SUMMARY

This chapter presented PRIMUS, an architecture for performance management based on adaptable smart healthcare application requirements and network resource slicing. Different from the literature, PRIMUS adapts network virtual resources and routes according to the s-health demands. Hence, it identifies the applications and configures the network on-the-fly. The results obtained in a virtualized environment, supporting SDN, NFV, and network slicing, show the feasibility to achieve the performance requirements of s-health. PRIMUS achieved latency values less than 10 ms in URLLC slice type (refers to critical care monitoring), and less than 20 ms for eMBB (refers to telemedicine). The required 1 ms of URLLC latency can only be achieved with network physical infrastructure efficient, such as 5G networks. Hence, this work contributes to the state-of-the-art and research community by implementing a virtualized scenario considering network slicing, network performance management, and critical applications performance requirements, *i.e.*, s-health applications. Therefore, we concluded that network slicing is absolutely necessary to enable the smart healthcare envisioned scenario in 5G since network slicing with virtualization techniques will be key technologies to resolve heterogeneity issues.

5 AN EXPLORATORY ANALYSIS OF NETWORK TRAFFIC FROM AN ADVERSARIAL PERSPECTIVE

This chapter presents the IoTReGuard, an **IoT** Method to **Re**veal and **Guard** IoT Network Traffic Features. IoTReguard method aims to explore network traffic features to reveal the key ones and mask them to protect users' privacy. It innovates by quantifying the potential impact of feature exploration when exploited by adversaries on IoT network traffic. When the network traffic is intercepted and all features making up the traffic are fed into a PCA or FA, adversaries can discover the most relevant features of the traffic patterns to represent the captured data. Therefore, IoTReGuard, based on network traffic capture, reveals the key network traffic features through a feature exploration employing statistical techniques. Then, it masks network traffic data or specific features by obfuscation techniques to prevent traffic-based side-channel attacks. The following section presents an overview of related works on network traffic analysis and defenses against traffic-based side-channel attacks. Then, Section 5.2 details IoTReGuard method, and Section 5.3 presents a case study on IoT network traffic to highlight the obtained results and insights, and discusses the challenges and limitations in feature exploration and obfuscation approaches in IoT. Finally, Section 5.4 concludes the chapter.

5.1 RELATED WORKS ON NETWORK TRAFFIC ANALYSIS

IoT network traffic analysis through feature exploration and ML algorithms has been widely investigated with the aiming for improve the network performance (Vergütz et al., 2019; Vergutz et al., 2020), the removal of ambiguous traffic (Taylor et al., 2018), defending the network against attacks (Prates et al., 2020), and the characterization of IoT devices behavior (Sivanathan et al., 2018). For instance, the methods proposed by (Taylor et al., 2018; Vergütz et al., 2019; Vergutz et al., 2020) identified packet and flow-level data in mobile and IoT health applications. Through ML algorithms, the methods analyze network features, identify applications to remove ambiguous traffic and increase network performance. The frameworks proposed by (Prates et al., 2020; Sivanathan et al., 2018) performed fingerprinting operations. The former fingerprints IoT devices and presents a defense mechanism against traffic-based side-channel attacks. The latter uses packet and flow-level features to fingerprint IoT devices and to characterize the behavior of devices and users. These schemes can be further improved and made resilient under the existing threats on IoT devices by integrating their network traffic analyses with feature exploration and statistical analyses to discover and unfold the most relevant dimensions of the network traffic.

By analyzing the most relevant features, adversaries can exploit them to infer information that makes attacks more efficient (Prates et al., 2020). Without this knowledge, the adversary performs an attack less efficiently since it uses several network traffic features. For instance, assuming the most relevant feature is network packet time, the adversary would perform the attack mainly using timing features since they represent better the network traffic behavior, and such attacks may jeopardize user data privacy. The main challenge is to identify the relevant features from network traffic suitable for each network context. Thus, the following subsection presents the main works applying feature exploration for different purposes.

5.1.1 Network Traffic Feature Exploration

Most of the studies employ feature exploration and selection to reduce data dimensions, abandon irrelevant information, and improve traffic classification (Conti et al., 2018). For instance, PCA extracts the most relevant information from a set of redundant or noisy data. It offers detailed information on the sample representation of data acquired from the network. It creates new variables, called Principal Components (PCs), through a linear combination of the original features. PCs are uncorrelated to each other; hence, identifying PCs supports the removal of redundant information. As PCs may lead to entry points for the adversaries in IoT networks, these methods lead to the development of effective mitigation schemes against passive attacks. However, few works use PCA or feature selection to reveal relevant features to employ on network security or performance management. For instance, Ullah et al. (2021) proposed a new PCA-based method for intrusion detection systems, and Hoang and Nguyen (2018) used PCA for IoT network traffic anomaly detection since it reduces dataset complexity.

There are works using network traffic analysis to identify applications (Zhang and Årvidsson, 2012; Vu et al., 2018; Maxwell et al., 2019), IoT devices (Meidan et al., 2017), malicious activities (Maxwell et al., 2021; Hafeez et al., 2020), or information exposure by IoT devices (Ren et al., 2019). The authors in (Maxwell et al., 2019) compared different feature extraction techniques on network security context. They applied label encoding and MinMaxScaler on network traffic features (*e.g.*, packet size and host connection). Classifiers achieved different performances regarding the techniques since each one creates a different number of features. Thus, it is necessary to analyze the classifier used to decide the feature exploration technique. For instance, Random Forest achieved the best performance results since it is robust for many features and follow an ensemble approach. However, they considered string features, such as the host website name, and did not consider the exploration of relevant features.

The authors in (Vu et al., 2018) extracted network traffic features to identify typical traffic applications (*e.g.*, Email and P2P). They extracted packet length and applied it on deep learning algorithms, achieving successful application identification. However, the authors did not present an efficient data exploration involving the extraction of different features and feature selection algorithms. The authors in (Zhang and Årvidsson, 2012) explored flow-level and packet-level network traffic features to understand typical applications' differences (*e.g.*, Twitter and LinkedIn). They used PCA to learn the features that most distinguish the same type of applications (*e.g.*, social network applications). In the results, the packet size is relevant for some applications, while for others, the timing-based features. These results show the importance of feature exploration on traffic features. Thus, although the authors did not apply the insights for network management purposes, they achieved valuable insights on traffic exploration and pointed the importance of feature exploration for management purposes.

In the IoT context, Meidan et al. (2017) identified IoT and non-IoT devices by ML classifiers. They considered the ratio between incoming and outgoing bytes and the average time to live. By Random Forest and XGBoost, the authors achieved high performance on classification (around 80%). Maxwell et al. (2021) used network traffic features and machine learning algorithms to detect malware. However, the works did not consider feature exploration and selection. Hafeez et al. (2020) proposed an online device identification system based on unsupervised classifiers to detect malicious activities in smart environments. By studying the variance of each network traffic feature, the authors identified its contribution to the classification process. They applied a correlation-based feature selection algorithm to identify and remove any features that contained redundant information. They also calculated Pearson correlation to measure the dependencies among all features and discard one of any two strongly correlated features. Although they benefit from feature selection to improve the classification task, the authors did not analyze the feature impact on traffic-based attacks, abstaining from such attacks.

There are works employing network traffic analysis to perform passive attacks. In (Apthorpe et al., 2018), the authors showed that the adversary observes changes in traffic rates to determine the timing of user activities. Most user interactions with smart home devices occur at discrete times or over short periods surrounded by extended periods of no interaction. User activities generally cause noticeable changes in device traffic rates while or shortly after they occur. These changes can be visible on bandwidth graphs and detectable by ML methods, allowing to infer information. The limited-purpose nature of most smart home IoT devices makes this possible. Once an attacker has identified the identity of a particular smart home device, it is often trivial to associate specific traffic rate changes with user activities and violate privacy.

Regarding privacy, Yao et al. (2019) performed an exploratory study on how people desire to protect their in-home privacy. The authors questioned 25 participants about privacy and security concerns. Most participants seem to worry about an adversarial connection to their network to collect data and invade their privacy. Thus, security and privacy improvements are needed. Moreover, we noted that there are few works exploring network traffic features in the literature. Once IoT data is complex, unstructured, and keeps increasing (Pekar et al., 2020), there are research opportunities to explore network traffic features to give insights into security management, especially when considering IoT users' privacy.

5.1.2 Defense Techniques Against Traffic-based Side-Channel Attacks

The ability to infer information about encrypted wireless communications via side-channels data (*e.g.*, packet and flow-level features) has been previously established (Atkinson et al., 2018; Papadogiannaki and Ioannidis, 2021), as have the privacy implications of persistently carrying personal devices and the potential security risks of particular applications and services. Given the limited-purpose nature of many IoT devices (*e.g.*, a smart power switch only function is turning on or off), it is often straightforward for an adversary to map changes in traffic rates to a particular user's actions. Recently, a number of techniques have been proposed for IoT device identification (Dyer et al., 2012; Sivanathan et al., 2018; Conti et al., 2018; Prates et al., 2019b; Vergütz et al., 2019). These techniques mainly rely on extracting fingerprints from network traffic generated by IoT devices and using them to identify the type of IoT devices. However, research has shown that it is also possible to identify IoT devices, exhibiting malicious activity, by analyzing their network traffic (Dyer et al., 2012; Apthorpe et al., 2017a, 2018; Pinheiro et al., 2020; Yu et al., 2021). Traffic analysis attacks (also known as traffic-based side-channel and traffic-based attacks) have been used to track in-home activities, users' activities, detect smartphone usage, others. Note that we use the three attack names similarly; however, for readability, henceforth using only traffic-based side-channel attacks.

The authors in (Apthorpe et al., 2017a; Chaddad et al., 2018; Atkinson et al., 2018; Apthorpe et al., 2018; Prates et al., 2019b) discussed the risks for users privacy and presented ways to mask side-channel data (metadata such as packet size). For instance, in (Atkinson et al., 2018), the authors demonstrated how encrypted traffic pattern analysis could allow a remote observer to infer potentially sensitive data passively and undetectable without any network credentials. The authors in (Apthorpe et al., 2017a) analyzed four strategies against such attacks. One strategy consists of preventing an adversary collect smart home network traffic by using a firewall. Thus, the firewall blocks the traffic from living the network home. However, some devices need Internet connectivity to work properly, such as Amazon Alexa. The second strategy considered masking DNS queries since it is possible to identify devices by analyzing them. Nonetheless, besides DNS queries, there is other network traffic metadata possible to identity devices. Another strategy is to tunnel all traffic through a VPN (a virtual private network). Thus, the network stream has the home gateway router's source IP address and the server's destination IP address. The adversary then would see all traffic as originating and terminating from a single pair of endpoints. Even with the VPN, the adversary could infer the devices by traffic metadata and ML algorithms. Finally, the authors presented and discussed traffic shaping (injecting traffic) as a counter-measure against the attacks. However, they did not implement the technique.

There are different masking approaches to prevent adversaries from exploring network traffic features (or side-channel data leaks). For instance, the authors in (Prates et al., 2019b, 2020) masked side-channel data by manipulating the timing of packets (*e.g.*, adding random time) of IoT devices. In (Liu et al., 2018), the authors employed obfuscation technique (*e.g.*, padding). In the literature, some of the most popular defenses against traffic-based side-channel attacks propose *(i)* to inject dummy (fake) traffic (Fu et al., 2003b; Dyer et al., 2012; Feghhi and Leith, 2018; Hafeez et al., 2019; Acar et al., 2020), and *(ii)* padding bytes to packets for a fixed or random size (Pinheiro et al., 2020; Chaddad et al., 2018, 2021). Fake traffic injection involves inserting additional fake (dummy) packets and transmitting them to conceal the real traffic. For instance, the authors in (Feghhi and Leith, 2018) masked web traffic injecting fake packets. In contrast, packet padding relies on changing the traffic characteristics to confuse the classifier and make it difficult to identify the original traffic. It consists of modifying the packet size and/or the inter-arrival times, using padding, fragmentation, and buffering. Padding and fragmentation are used to hide packet size information while buffering aims to hide the inter-arrival time information. There are also approaches combining both defense techniques (Datta et al., 2018; Barman et al., 2021). Next, we present the main literature works following both defense techniques.

5.1.2.0 Fake Traffic Injection Approach

The fake traffic injection approach confuses the adversary by adding randomized or fixed fake network packets. The authors in (Fu et al., 2003b) generated random fake packets by following statistical distributions to learn and generate similar traffic to the original one. The authors in (Feghhi and Leith, 2018) obfuscated web traffic by injecting dummy traffic. The authors used only packet timing information. However, they assume that the packets are padded to the same size, and the user has baseline defenses in place by routing traffic via tunneling. The authors in (Dyer et al., 2012) simulated scenarios in which an adversary has access to the timing, lengths, and direction of packets. They compared the performance of insertion of dummy packets and padding techniques such as random 255, linear, exponential, and others. Although the techniques reduced the attack performance, the authors showed that these mechanisms still fail to protect against attacks when analyzing other features besides packet length. Thus, they performed the attack considering only-bandwidth and only-burst and achieved good performance. Hence, besides the per-packet features typically used, other features such as total bandwidth are also damaging. Considering packet length and timing, the authors proposed the BuFLO obfuscator that sends fixed-length packets at a fixed interval. BuFLO reduced the identification accuracy to 27.3%. However, different applications, scenarios, and environments will present different behaviors from the feature perspective. For instance, the authors noted that the bandwidth feature was relevant for web pages but may not be relevant in other contexts. Thus, it is essential to analyze and explore features to understand the particularities of the environment.

The authors in (He et al., 2016, 2017) added fake network traffic in the transmission sequence randomly based on logistic regression. First, all devices send fake traffic with a fixed time window. Then, they analyze the network status, time, and density to send more fake traffic during a long or short interval. Once the real data is sent, the time window is recovered again. Nonetheless, the authors did not discuss different network traffic features that adversaries can use. Further, no expressive accuracy reduction in traffic identification was achieved. The authors in (Hafeez et al., 2019) demonstrated that an adversary, with access to the network traffic of a smart home network, can lead to the identification of the device types and some user interactions with IoT devices. The authors proposed a traffic morphing technique that shapes network traffic by sending fake traffic at a constant rate, based on the statistical proprieties of the real traffic, such as packet size, inter-arrival delay, and direction. Meanwhile, when an IoT device is inactive, the authors send dummy traffic representing device activity, so that an adversary cannot identify the real activity of the IoT device. While this approach is not very sophisticated, it points to the direction of defending against traffic analysis on IoT devices. The technique limits the ability of an adversary to perform attacks but does not explore the key features used in such attacks.

The authors in (Acar et al., 2020) evaluated a traffic-based attack in a smart environment and proposed a mitigation approach based on spoofed traffic generation to hide the device states. The attack evaluation considered a dataset containing typical user movements. The authors detected the device states (*e.g.*, on or off) and the user activity (*e.g.*, sleeping, walking) through ML algorithms. Random Forest achieved around 90% accuracy in detecting the states and activities. By injecting 90% of false data in the training data, classifiers reached only 15% of accuracy. However, there are few details about the false traffic generation methodology. Although the authors considered the feature selection algorithm, they did not analyze the relevance of the features. Thus, it is not possible to analyze the features and behavior of false data. The authors in (Apthorpe et al., 2018) proposed an IoT traffic shaping considering fake traffic generation. First, according to the original device trace, the technique generates fake traffic for each device to be injected when idle. The generator randomly chooses a period of recorded user activity traffic and replays it in the generated trace. Then, the technique shapes the traffic by choosing a fixed traffic pattern for the traffic rate and duration to all fake traffic have similar patterns.

The authors in (Yu et al., 2021) proposed a defense system to reduce the private information leaked through IoT device network traffic. They analyzed the user activities from a smart home to mask them. The system learns the signatures and injects traffic signatures. It also reshapes partial traffic. The traffic rate signature learning assists in distinguishing the genuine IoT traffic rate signatures from the fake injected traffic. Moreover, the authors analyzed the principle features from network traffic traces by feature selection algorithms (*e.g.*, PCA). From 10 features, PCA indicated to use 8 features: total traffic, duration, range, mean, standard deviation, skewness, and others. The system learned the signatures, compared correlation metrics on the original and injected signatures, and masked the user activities. However, the feature analysis did not consider different network traffic features (*e.g.*, timestamp, and a number of bytes), only considering the statistical proprieties of traffic rate (*e.g.*, mean and max). It is possible to perform side-channel attacks based on different features been relevant to their deep analysis.

5.1.2.0 Packet Padding Approach

In the literature, works related to packet padding employ several padding strategies to reduce the accuracy of classifiers. Maximum transmission unit (MTU) padding, for example, is a well-known mutation technique that consists of padding all the packets to the maximum payload size MTU (*e.g.*, 1500 bytes) (Dyer et al., 2012). Random padding is another technique that consists of randomly padding the packets, and Random 255 inserts a random value between 1 and 255 bytes (Diyanat et al., 2016). Linear padding increases the packet size to its nearest 128 multiples, while exponential padding increments the packet to the nearest power of two (Pinheiro et al., 2020). Mouse/elephants padding increases the packet sizes to 100 or 1500 bytes accordingly with a threshold. Values less than 100 are incremented to 100 bytes, while sizes greater than 1000 are increased to 1500 bytes. However, packet padding can result in a drastic overhead in the amount of data sent, causing performance degradation related to delay and bandwidth consumption (Dyer et al., 2012). Furthermore, some works proved that it is still possible to classify the traffic, even when implementing these techniques (Fu et al., 2003a).

Other padding techniques aim to confuse the classifier into classifying the target application traffic as another type. The authors in (Wright et al., 2009) proposed a padding technique that transforms one class of traffic to look like another class by applying convex optimization. Nevertheless, this method applies to devices with computational solid capabilities. The authors in (Chaddad et al., 2018) proposed a system to mobile fingerprint applications (*e.g.*, Facebook, YouTube) and mask the packet size adding random sizes in the flow, following a packet padding approach. The authors in (Chaddad et al., 2021) improved the masking traffic data considering the performance since the padding approach increases the packet size transmitted. As the system considers mobile applications, it models the packet size probability distribution of the applications. It mutates the packet length of the source app to the length of the target app. In other words, the padding approach selects the target app to mutate one app with each other. Thus, the probability distribution of the mutated target application fits into the second application's probability distribution to confuse the ML classifiers. However, this technique always needs previous learning on the target application traffic to mutate according to it. Further, these works did not explore the feature importance considering the application.

Some padding techniques rely on either independent link padding (ILP) or dependent link padding (DLP) (Datta et al., 2018). ILP forces traffic to fit a predetermined distribution by padding and fragmenting packets and sending cover packets. DLP also uses padding, fragmentation, and cover traffic but uses some information from the real traffic to set the schedule for packet sending. The two main parameters of ILP and DLP are packet sizes and timing. The authors in (Datta et al., 2018) manipulated packet sizes and inter-packet intervals based on predetermine probability functions to protect IoT devices against traffic attacks. Through sending and receiving functions, they combined payload padding and randomized fake traffic. The sending function performs traffic shaping by padding payloads, fragmenting payloads, and adding fake traffic according to each device, while the receiving function discards the fake traffic and recombined fragmented packets. The authors also forced constants to send rates for packet size and inter-packet intervals. According to their results, they were able to obfuscate some traffic patterns. However, the shaped traffic does not preserve user peak traffic, reveling user activities. The authors in (Dyer et al., 2012) showed that simply padding or fragmenting packet lengths to a constant size is not sufficient for hiding user activity since using other features (*e.g.*, total bandwidth) still achieves high performance (98% of accuracy) on traffic-based side-channel attacks.

The authors in (Pinheiro et al., 2020) proposed a dynamic packet padding technique based on software-defined networks (SDN) to mask IoT network traffic. The authors aimed to achieve an equilibrium between privacy and overhead. Thus, they created four padding levels: 100, 500, 700, and 900. Each level compares the packet number of bytes to increment the length, *i.e.*, the level numbers denote the minimum length of the packets that will belong to the respective level after padding. For instance, packets with lengths smaller than or equal to 100 are resized to 100 bytes. In this way, when the network is overloaded, they set up the minor level, and when the network is idle, they use the highest level. Hence, they increase privacy according to the network volume (the metric used to analyze the network overload). They compared the padding technique with other padding approaches (*e.g.*, MTU and random). Through ML algorithms (*e.g.*, Random Forest), their solution achieved high privacy with the 900 level since it is the highest level of padding. Random Forest decreased the IoT devices classification to $\approx$ 49% of accuracy in the 900 level, while the 100 level achieved 66% of accuracy. However, the solution did not change the padding level automatically. Moreover, they did not perform any feature exploration. The decision on using packet size was based on other works that used the same network traffic feature.

Table 5.1: A Summary Table of Related Works to IoTReGuard

Reference	**Feat.Exploration**	**Obfuscation Technique**		**IoT**
		Fake Traffic	**Padding**	
Zhang and Årvidsson (2012)	✓			
Hoang and Nguyen (2018)	✓			✓
Hafeez et al. (2020)	✓			✓
Ullah et al. (2021)	✓			
Fu et al. (2003b)		✓		
Dyer et al. (2012)		✓		
Feghhi and Leith (2018)		✓		
Hafeez et al. (2019)		✓		✓
Acar et al. (2020)		✓		✓
Yu et al. (2021)	✓	✓		✓
Apthorpe et al. (2018)	✓	✓		✓
He et al. (2016, 2017)		✓		✓
Datta et al. (2018)		✓	✓	✓
Fu et al. (2003a)			✓	
Wright et al. (2009)			✓	
Pinheiro et al. (2020)			✓	✓
Chaddad et al. (2021)			✓	
Barman et al. (2021)	✓	✓	✓	
Hussain et al. (2021)		✓	✓	

As this thesis is s-health context-aware, there are also considerable concerns in the literature related to traffic-based side-channel attacks on smart healthcare applications and wearable devices (Newaz et al., 2021). The smart home is becoming a locus for healthcare innovations since it gives the patients higher freedom and reduces societal costs. In smart home-based remote health systems, caregivers can access patients' health status in a timely manner and provide them with preventive instructions. Studies show that people, especially the elderly, regard their home as a sanctuary and therefore prefer to stay at home for medical treatments of chronic diseases (Liu et al., 2018). However, health-related devices such as smartwatches and blood-pressure monitors process, store, and communicate sensitive and personal information related to the user's health, lifestyle, habits, and interests, which breeds serious privacy concerns. For example, a glucometer measuring the blood sugar level and a sleep monitoring sensor recording the sleep conditions can potentially reveal whether the resident carries diabetes or depressive disorder, respectively. For privacy concerns, patients are inclined to restrict the access of these data to a limited group of people like their personal doctors.

The authors in (Barman et al., 2021) showed that adversaries can infer sensitive information from the metadata (*e.g.*, packet size) of these device communications. A passive adversary can benefit from traffic-based side-channel attacks to accurately recognize devices, MAC address, human actions, and fine-grained actions (*e.g.*, recording an insulin injection). The authors evaluated standard defense techniques against traffic-based attacks such as padding, delaying packets, or injecting fake traffic. They highlighted the need to design new defenses tailored to the wearable setting because the countermeasures against these attacks provide insufficient protection. Moreover, the authors analyzed the feature importance of Bluetooth wearable devices. The results showed timing-based information is relevant for discriminating among the devices. However, the feature importance is particular for devices, applications, and functionalities. For instance, continuous monitoring devices affect in small proportion the timing-based information compared to classic on/off devices. The authors in (Hussain et al., 2021) developed a framework to detect malicious traffic in IoT healthcare use cases. However, is not related to traffic-based attacks. The studies presented significant security advances. However, fingerprint applications/devices can be used for immoral purposes, such as to infer personal information. Hence, it is crucial to highlight the need of defenses for s-health.

In summary, some studies leverage traffic device classification for malicious purposes, mainly to infer valuable knowledge from devices or users (Conti et al., 2018; Skowron et al., 2020; Yu et al., 2021). Such attacks take advantage of the network features (*e.g.*, packet size, timestamp), leading ML algorithms to observe the traffic passively. Since most IoT devices work with sensitive or personally identifiable information, such as behavioral patterns, they lead to a vulnerable surface for a passive eavesdropper to extract sensitive information. With more valuable features about network traffic, adversaries boost the success and the impact of their attacks. Although the literature points out significant attention in the design of defenses against attacks, quantifying the impact of passive traffic-based attacks considering feature exploration calls for further investigation to improve the design of protections against them.

5.2 THE IOTREGUARD METHOD

This section introduces the IoTReGuard method, which reveals and masks network traffic features on smart environments, such as smart homes. The IoTReGuard method instantiates the FLIPER methodology to organize and structure network traffic data. Figure 5.1 displays the system overview of IoTReGuard based on two main steps that support each other: *(i) reveal features* and *(ii) guard features*. IoTReGuard reveals the key network traffic features through an exploration employing statistical techniques, *e.g.*, PCA and FA. These features, such as the packet size and timestamp, consist of data generated on network transmissions, and when this is submitted to statistical analysis, leaks information and enables the identification of rooms, the number of residents, and even possible visitors (Srinivasan et al., 2008). Thus, IoTReGuard masks network traffic data or specific features by obfuscation techniques to prevent traffic-based side-channel attacks. It preserves users' privacy in a smart home with diverse IoT devices. The following subsections describe the attack overview and IoTReGuard's main steps.

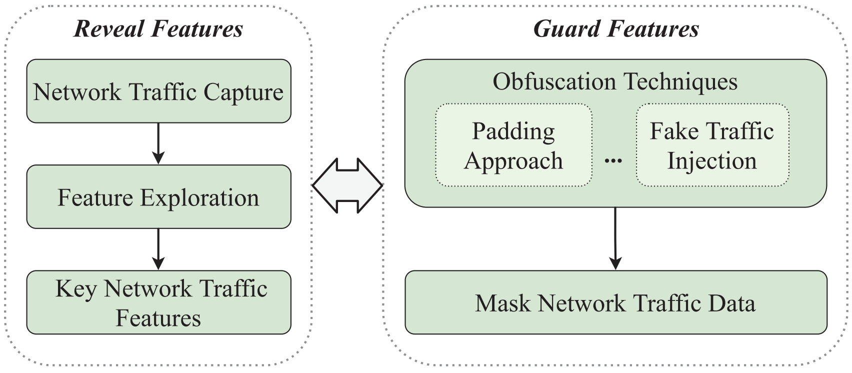


Figure 5.1: IoTReGuard Overview

5.2.1 Traffic-based Side-Channel Attacks Setting Overview

Many smart home devices have sensors that capture users' activities in their living spaces and transmit information about these activities on the Internet (Apthorpe et al., 2017b). IoT devices often perform particular functions, making them vulnerable to attackers seeking to infer details about potentially private user behaviors (Datta et al., 2018). Apthorpe et al. (2017b) showed that an adversary with access to IoT device network traffic detects when user events occur, even when all traffic is encrypted, simply by looking for peaks in traffic send and receive rates. User events differ across IoT devices and can reveal specific information about user activities. A user event for a sleep monitor, for example, usually indicates that a user has fallen asleep or woken up. If adversaries know device identities, they can deduce information about what users are doing inside their homes. However, an attacker inside or near a smart home environment can exploit these devices' innate wireless medium to extract sensitive information about the users and their activities, invading privacy (Acar et al., 2020).

Figure 5.2 shows a setting overview of a passive network traffic-based attack on a smart home, in which an adversary aims to infer user activities. We consider an adversary located physically within the wireless range of the targeted users' devices (Acar et al., 2020; Pinheiro et al., 2020). Hence, the adversary can install the sniffer in the gateway only once and manage it remotely. Alternatively, it could compromise a device inside the smart home remotely and turn it into a sniffer. In this way, the adversary may never need to be present. In all these cases, the adversary can eavesdrop on wireless network communications transmitted by the devices. These devices include cameras, motion sensors, temperature sensors, switches and triggers, lighting sensors, smartwatches, smartphones, laptops, and others. As a typical smart home setting, they transmit network traffic to the AP (access point), the AP transmits to the router and the router to the Internet. The adversary only needs to sniff the network traffic without interrupting passively. Therefore, the attacker may stay active long enough without being detected by the victim.

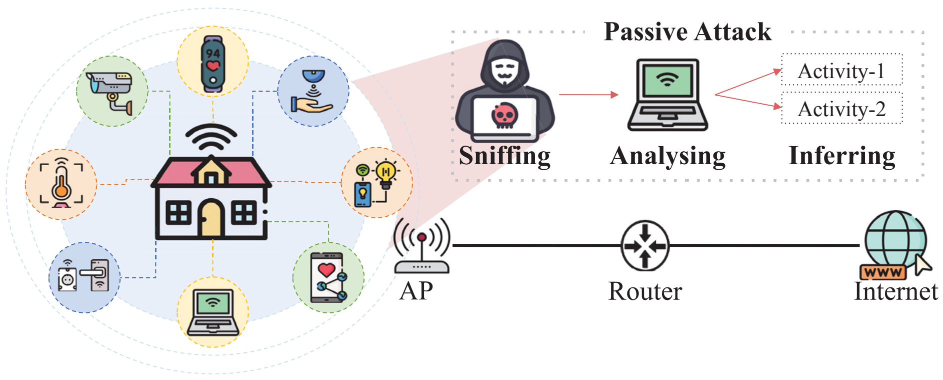
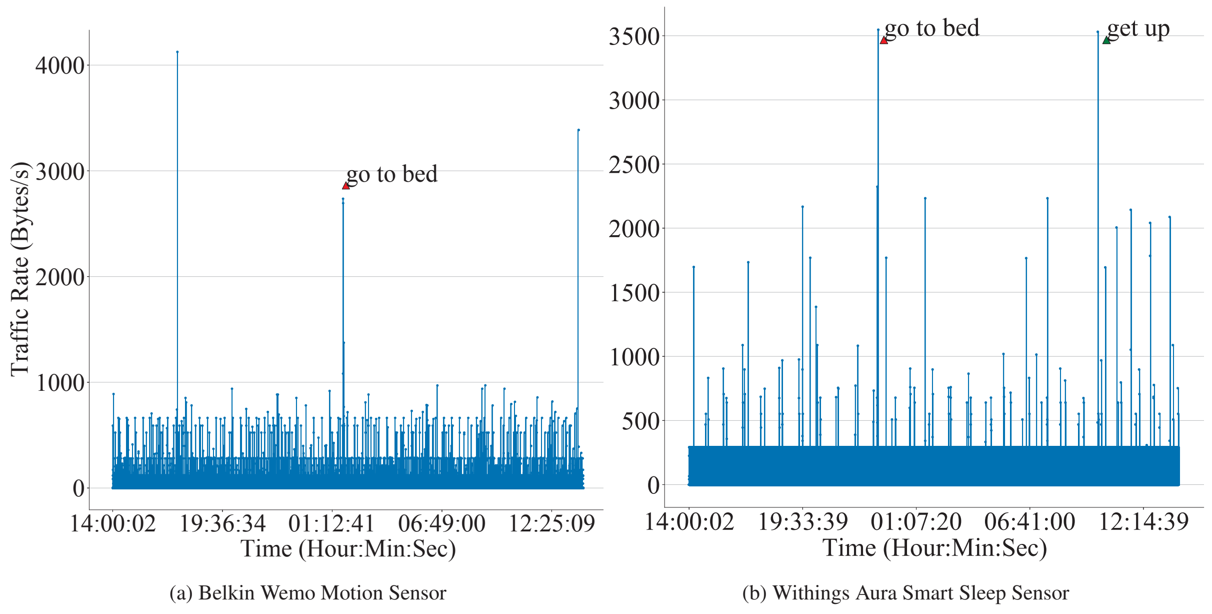
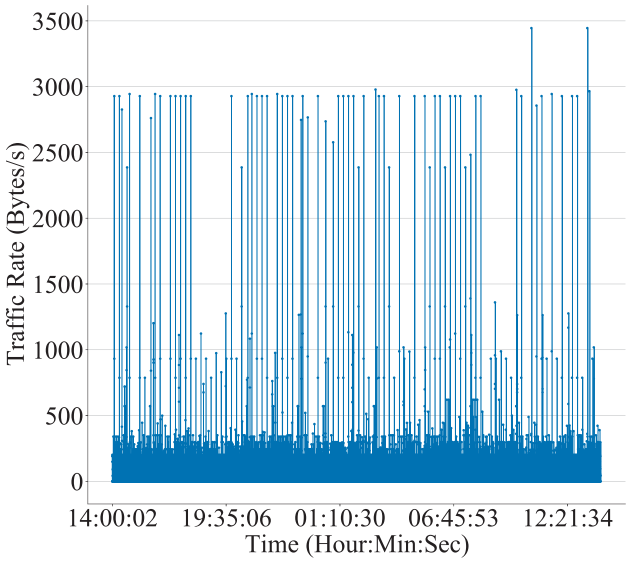


Figure 5.2: Setting Overview on Traffic-based Side-Channel Attacks

The adversary sniffs network traffic side-channel information such as packet size, the interval between packets, timestamp, and others (Pinheiro et al., 2020). Thus, the traditional means of encrypting packets are no longer feasible approaches from the privacy perspective since it only encrypts the packet payload. Using side-channel information, the adversary can build classifiers and infer user behavior. For instance, the packet size and timestamp in ML techniques enables inferring if an individual goes to bed, health conditions, assistant interactions, and others (Yu et al., 2021). Figure 5.3 presents the network traffic rate trace (in bytes/s) of 3 IoT devices over 24 hours (*hour:min:sec* format following the Brasília Standard Time) from a smart home. Figures 5.3(a) and 5.3(b) display significant peaks from the network traffic rate of a motion and sleep sensor, respectively. The relation between the time of each traffic rate peak from the sensors may indicate users go to bed around 01:05 a.m., and get up around 12:15 a.m. Figure 5.3(c) presents two peaks in traffic data rate around 12:20 a.m, showing interaction with the voice assistant near to the interactions with the sleep and motion sensors. Thus, by analyzing these 3 devices, the adversary can deduce the user went to bed late at night and woke up near noon, learning the users' daily routine.





(c) Amazon Echo

Figure 5.3: User Interactions over 24 hours increase IoT Devices Traffic Rate

Users' interaction with IoT devices, *e.g.*, talking to voice assistants, opening/closing doors, watching smart TVs, lends itself to straightforward attacks that detect changes in these metrics and associate them with changes in user activities (Yu et al., 2021). Network traffic peaks from these devices imply specific user activities without needing additional information (Datta et al., 2018). By combining rich sensors (*e.g.*, monitoring cameras, microphones, motion sensors) and Internet connectivity, these devices can expose extensive information about their users and their surrounding environment. Hence, traffic analysis is considered a significant threat to the users' privacy (Chaddad et al., 2021). Traffic analysis combined with ML algorithms and network feature exploration boosts the traffic-based attack impact by giving detailed information. Feature exploration serves as an essential data pre-processing step to transform network traffic data into useful feature vectors that optimize device identification performance (Maxwell et al., 2019). It reveals important information on network data behavior, such as selecting the most valuable features. However, adversaries take advantage of feature exploration to make data-driven decisions about what feature is most relevant to use in traffic-based attacks.

5.2.2 IoTReGuard Setting Overview

IoTReGuard method reveals and masks the key network traffic features on representing the network traffic behavior and inferring user activities. Hence, IoTReGuard method works based on side-channel features from the segment header. Figure 5.4 shows the system structure of IoTReGuard. It can be deployed either on a home router or home AP (as shown in Figure 5.4) that serves as a gateway for the home network. From the AP, IoTReGuard sniffs the network traffic to reveal key network traffic features. In this step, the adversary can still sniff the network traffic. Then, IoTReGuard takes additional steps to further obscure network traffic and user activities in-home privacy. It masks network traffic through obfuscation techniques such as the packet padding approach and injecting fake traffic. Therefore, the masking step of IoTReGuard makes unfeasible the adversary success on collecting useful features since the network traffic feature is masked. Thus, IoTReGuard identifies different IoT devices by side-channel information to reveal devices and users' behaviors and then masks this information to prevent passive attacks.

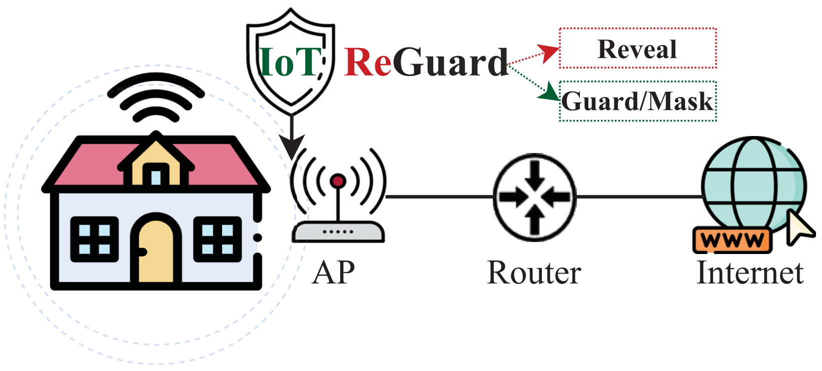


Figure 5.4: IoTReGuard Position in the Smart Home

5.2.3 Revealing Network Traffic Features

This section presents the *reveal features* step of IoTReGuard, *i.e.*, the feature exploration under the perspective of adversaries. Different from the literature, IoTReGuard highlights the possible damages of a thorough network traffic analysis by adversaries. For the sake of completeness, Fig. 5.5 shows an ML-based network traffic exploration and classification overview that involves five steps: data collection, feature pre-processing, feature selection, classification, and data-driven decision making. The first three steps compose the feature exploration. IoTReGuard focused on these steps from an adversarial standpoint. For feature exploration, it is necessary to collect network traffic data; hence, this work considers a smart home, containing various devices such as a camera, smartwatch, and ambient sensor that generate traffic. Hence, data input for feature exploration is network traffic containing data packets generated in an IoT setting. The next subsections detail the feature exploration steps.

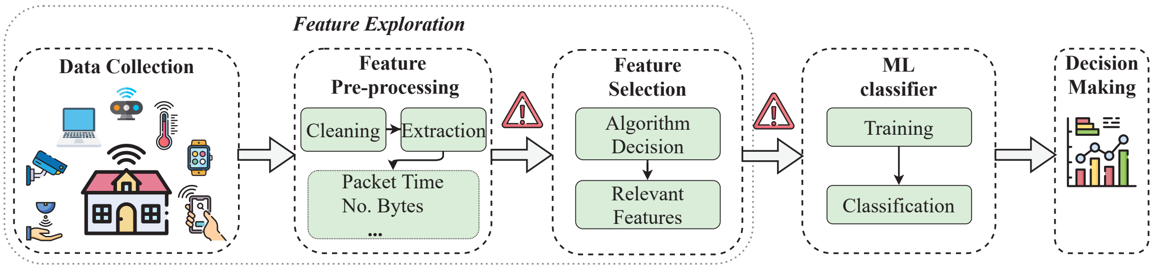


Figure 5.5: ML-based Network Traffic Classification Overview

5.2.3.1 Feature Pre-Processing from Network Traffic

In order to efficiently find out the meaningful features for traffic classification, it is essential to pre-process the collected network traffic through data cleaning and extraction. Cleaning the traffic capture by removing unnecessary traffic data and extracting network features is paramount. Feature extraction involves information from the segment header, such as segment time. Effective feature extraction has to enclose the output features as much as possible the information available from the collected network traffic. It is possible to extract features at the packet or flow level. Flow-level data is based on a 5-tuple structure: source/destination IP addresses, source/destination port numbers, and protocol (TCP/UDP). Analyzing the 5-tuple, flow-level data contains all network packets that belong to the same flow to extract features such as data volume per flow, number of packets per flow, and others. In contrast, packet-level considers features related to individual network packets, *e.g.*, packet size, timestamps, and others. Moreover, statistical primitives (*e.g.*, min, max, mean, and variance) computed from the features can increase data granularity and assist in device traffic classification.

Feature pre-processing output follows a time-series or packet-series format for flow-level and packet-level, respectively. For instance, the output of flow-level feature pre-processing consists of a time series containing packet size, packet time, and other traffic features. As traffic is generated by multiple devices, MAC address serves as a label to point out the association between traffic and a device. Through the output of feature pre-processing, it is possible to achieve network traffic feature information, which can be used by an adversary. Although the packet payload undergoes a cryptography process, network traffic features are extracted from the segment header. In general, the content of the payload is not deeply analyzed. Thus, even in the case of encryption, features can be extracted. Hence, adversaries can infer the user information and lead to security implications by only analyzing the network features (*i.e.*, network traffic-based attacks). Inferring user information have consequences such as learning users daily routines. Therefore, network features, when statistically align with human behavior patterns, may lead to privacy violations.

5.2.3.2 Feature Selection from Network Features

After feature pre-processing and extraction, it is necessary to analyze, select, and combine meaningful features. Fig. 5.6 shows the steps to discover the most relevant network traffic features: 1) the removal of features that are less significant, and 2) feature contribution analysis. It is necessary to normalize, discretize the features, and remove noise. The removal of features that are less significant can be achieved by using the variance of features. Low variance results in less significance of a feature to contribute to the characterization of the network traffic (Shen et al., 2020). Feature contribution indicates whether/how a particular feature plays a role in traffic classification. Thus, the relevant feature set is obtained by eliminating the features with low variance, as well as those redundant and less relevant.

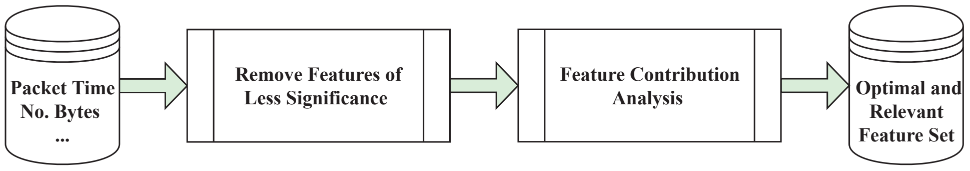

Packet Time
No. Bytes
...
Remove Features of Less Significance
Feature Contribution Analysis
Optimal and Relevant Feature Set

Figure 5.6: Feature Selection

Feature selection algorithms identify the most relevant features. For instance, PCA is a linear dimensionality reduction technique (Abdulhammed et al., 2019) that seeks to map data points from high dimensional space to a low dimensional space while keeping all the relevant linear structures intact; and FA looks for the fewer components which can explain the original features by identifying the latent correlations of features. Moreover, both analyses determine the key variables in a dataset that explain the differences in the observations. Hence, through PCA and FA analysis, an adversary can discover the key features that most represent the original network traffic. In other words, they assist in learning traffic behavior. Although such key features assist malicious adversaries to perform attacks, they also guide the design of security and privacy solutions by obfuscating relevant features and making unfeasible the discovery by adversaries.

Using only feature pre-processing and extraction, adversaries infer user behavior, which may lead to security breaches. However, feature selection algorithms improve data details by pointing out the most relevant and contribution features. Hence, they assist adversaries in performing traffic-based attacks and even zero-day attacks. Without this knowledge, the adversary performs an attack less efficiently since it uses several network traffic features. Thus, based on feature selection, the adversary learns the network traffic patterns and discovers the most significant feature, assisting in attack modeling. The adversary can achieve higher damage through the attack, improving the impact and reducing the complexity. For instance, assuming the most relevant feature is related to the network packet time, the adversary would perform the attack using mostly timing features since they represent better the network traffic behavior. Such attacks may jeopardize user data privacy and reveal sensitive information. In contrast, learn the key network traffic features can also assist defense mechanisms. For instance, instead of masking all network traffic data or specific features without knowledge about relevance, thorough feature exploration can mask only the most representative feature reducing time implementation and guaranteeing the obfuscation of essential and representative features. Therefore, by feature exploration of IoTReGuard is possible to make a data-driven decision on masking features.

5.2.4 Masking Network Traffic Features

The second step of IoTReGuard method, *guard features*, masks network traffic features to prevent traffic-based attacks and user activities inferring. Note that the IoTReGuard method follows two main steps that support each other: *(i) reveal features* and *(ii) guard features*, as shown in Figure 5.7. Hence, in the second step and based on the features revealed, IoTReGuard method employs obfuscation techniques such as the padding approach and injecting fake traffic. Note that IoTReGuard is dynamic since it supports different obfuscation techniques. The goal is to mask side-channel information in the best way possible. It is not in the scope of this work propose a new obfuscation technique. There are several obfuscation techniques in the literature (Yu et al., 2021; Prates et al., 2020; Yao et al., 2019); however, there are still open issues regarding these techniques such as the trade-off between privacy and overhead. Thus, IoTReGuard follows a dynamic approach making it feasible to employ different techniques.

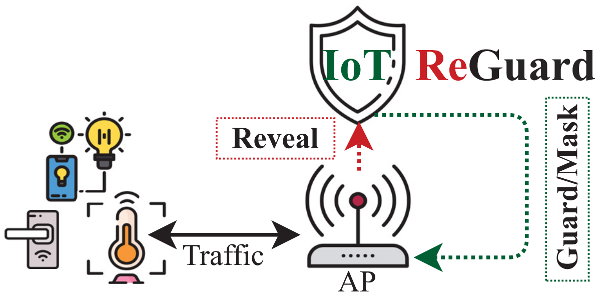


Figure 5.7: IoTReGuard Steps

In general, obfuscation techniques mask side-channel information by injecting or altering traffic data (Pinheiro et al., 2020; Yu et al., 2021). For instance, following the injection fake traffic approach, IoTReGuard generates network traffic based on the MAC address of the original IoT devices identified in the first step, feature exploration. In this way, the fake network traffic contains similar patterns to the original traffic since it considers the device type. Thus, it is difficult to distinguish the injected fake traffic from the real demands due to their behavior similarities. Moreover, network traffic injection usually follows the number of devices connected to the smart home or the original traffic size. In other words, it is possible to inject from 10% to 100% of fake traffic and shuffle with the original data. To analyze the masking performance, IoTReGuard compares classifiers performance results on identifying IoT devices. Therefore, first, it identifies devices only considering the real network traffic. Then, it adds the fake traffic and identifies the devices by classifiers. If the performance of the classifiers decreases, it means good masking and privacy protection performance.

In contrast, the padding approach eliminates existing network traffic patterns related to the packet size. Following this approach, IoTReGuard enables the modification of the number of bytes inserted in the packets protecting against traffic-based attacks. There are different padding approach strategies since each one increases a specific amount of bytes (Pinheiro et al., 2020). The more the number of bytes added to the network packet, the greater the masking performance because all packets will have the same size. The original values will change if we modify the packet size by following a simple strategy, such as adding 100 bytes to all packets. However, it is still possible to identify patterns in the sizes. Strategies based on random sampling are likely to create new patterns, especially when the sample space is small. Choosing the appropriate number of bytes to insert in the packets is challenging because excessive bytes lead to high overhead. In contrast, few bytes result in a weak privacy enhancement. Pinheiro et al. (2020) proposed a padding solution that dynamically modifies the packet size according to the volume of network traffic, without compromising the communication performance. Therefore, IoTReGuard follows the fake traffic injection approach and the padding approach based on (Pinheiro et al., 2020).

IoTReGuard is privacy-aware of user-health data since it comprises critical and sensitive data. Mainly, health data or health information obtained by analyses on side-channel data since the MOTIF method and the feature exploration of IoTReGuard use side-channel data to fingerprint s-health applications and IoT devices, respectively. In the smart healthcare context, the consequences of identifying users' behavior are more significant since malicious people can infer information about users' health. Therefore, it is crucial the development of protection techniques on side-channel data. Malicious people can infer the user information and violate security concerns, realizing the traffic side-channel attack. Infer information about user health can have serious consequences, *e.g.*, someone can alter the data. Thus, IoTReGuard aims to mask side-channel data to improve users' privacy.

5.3 CASE STUDY: REVEALING AND MASKING IOT TRAFFIC FEATURES

This section presents a case study on IoT network traffic to evaluate IoTReGuard method. The case study contains two main goals (*i.e.*, it follows the two main steps of IoTReGuard): *(1)* analyze the potential impacts caused by adversaries through feature exploration, *(2)* analyze and compare quantitatively different obfuscation techniques against traffic-based side-channel attacks. Throughout the text, we will call both analysis as *RevealFeat* and *GuardFeat*, respectively. The first analysis, *RevealFeat*, seeks to analyze the network traffic features through feature exploration in an adversarial perspective. Such feature exploration involves cleaning, pre-processing, extracting, and selecting network traffic features. The adversaries attempt to take advantage of ML's strengths during feature exploration to make data-driven decisions about what information (in network features) is most relevant to improve the attack performance. Thus, *RevealFeat* intention lies in highlighting the key network traffic features that can be used by adversaries on traffic-based side-channel attacks considering the application of feature exploration.

The second analysis, *GuardFeat*, aims to evaluate the effectiveness of common protections against traffic-based side-channel attacks. In this work, we compare two well-known techniques: *(i)* fake traffic generation (called FAKE-A), and *(ii)* packet padding (named PADD-A). Both techniques obfuscate features from the network traffic (*e.g.*, packet size), making it challenging to perform side-channel attacks. Note that the purpose of this section is not to discover a perfect defense since the cost might be unbounded due to the diversity and consumption restriction (*e.g.*, energy) of IoT devices. Moreover, a classifier properly trained can detect even small differences between two network traces. Ensuring that all network traces are perfectly indistinguishable from each other is infeasible. Therefore, we analyze the effectiveness of standard defenses and discuss their feasibility for the protection of IoT devices. Thus, next sections present the evaluation methodology, obfuscation techniques, and discuss the results in an adversarial perspective.

5.3.1 Feature Exploration Methodology

Both analysis, *RevealFeat* a *GuardFeat*, followed the same feature exploration methodology. Thus, the next subsections describe the dataset, feature exploration details, feature selection algorithms, and machine learning algorithms.

5.3.1.0 Dataset Characteristic

The case study follows a trace-driven approach considering the same dataset used in the MOTIF method evaluation (detailed in Chapter 3) since this work explores features on IoT environment. Therefore, the *IoT Traffic* dataset (Sivanathan et al., 2018, 2021) presents network traffic from 20 days, containing 20 packet capture (PCAP) files. Each one corresponds to 24h of the capture of the traffic generated by 30 consumer IoT and non-IoT devices. Different from the MOTIF method, in this case study, we considered all devices available in the dataset, totaling 31 devices (counting the gateway), since the dataset authors make available recently the entire dataset. Table 5.2 presents the entire dataset, including hubs, healthcare devices, monitoring cameras, switches and triggers, light bulbs, air quality sensors, electronics, and several non-IoT devices (*e.g.*, laptops).

The dataset serve as input for *RevealFeat* and *GuardFeat* analysis. Thus, we converted the PCAP files into comma-separated values (CSVs) by extracting network traffic features (*e.g.*, packet size, timestamp, and the number of packets) and the associated capture time. We filtered the devices by the MAC address and created the device label. Such labels assist the feature exploration and serve as input for the supervised ML-algorithms.

Table 5.2: Dataset - List of Devices in the Smart Home Environment (Sivanathan et al., 2018)

Category	**Device**	**MAC Address**	**Connection**	**Device Label**
Hubs	Amazon Echo	44:65:0d:56:cc:d3	Wireless	Assistant1
	Smart Things	d0:52:a8:00:67:5e	Wired	Device1
Healthcare Devices	Withings Smart scale	00:24:e4:1b:6f:96	Wireless	Scale1
	Blipcare Blood Pressure m.	74:6a:89:00:2e:25	Wireless	BlipcareBP1
	Withings Aura smart sleep s.	00:24:e4:20:28:c6	Wireless	Sleep1
Cameras	Netatmo Welcome	70:ee:50:18:34:43	Wireless	Camera1
	TP-Link Day Night camera	f4:f2:6d:93:51:f1	Wireless	Camera2
	Samsung SmartCam	00:16:6c:ab:6b:88	Wireless	Camera3
	Dropcam	30:8c:fb:2f:e4:b2	Wireless	Camera4
	Insteon Camera	00:62:6e:51:27:2e	Wired	Camera5
	Nest Dropcam	30:8c:fb:b6:ea:45	Wireless	Camera6
	Withings Smart Baby Monitor	00:24:e4:11:18:a8	Wired	BabyMonitor1
Switches and Triggers	Without Name	e8:ab:fa:19:de:4f	Wireless	Device2
	Belkin wemo motion sensor	ec:1a:59:83:28:11	Wireless	Motion1
	iHome	74:c6:3b:29:d7:1d	Wireless	iHome1
	TP-Link Smart plug	50:c7:bf:00:56:39	Wireless	Splug1
	Belkin Wemo switch	ec:1a:59:79:f4:89	Wireless	Wswitch1
Light Bulbs	Light Bulbs LiFX Smart Bulb	d0:73:d5:01:83:08	Wireless	LightBulb1
Air Quality Sensors	NEST Protect smoke alarm	18:b4:30:25:be:e4	Wireless	Smoke1
	Netatmo weather station	70:ee:50:03:b8:ac	Wireless	Weather1
Electronics	PIX-STAR Photo-frame	e0:76:d0:33:bb:85	Wireless	Picturef1
	HP Printer	70:5a:0f:e4:9b:c0	Wireless	Printer1
	Triby Speaker	18:b7:9e:02:20:44	Wireless	Speaker1
Non-IoT	Laptop	74:2f:68:81:69:42	Wireless	Laptop1
	MacBook	ac:bc:32:d4:6f:2f	Wireless	Laptop2
	MacBook/Iphone	f4:5c:89:93:cc:85	Wireless	Laptop3
	Android Phone	40:f3:08:ff:1e:da	Wireless	Phone1
	Android Phone	b4:ce:f6:a7:a3:c2	Wireless	Phone2
	IPhone	d0:a6:37:df:a1:e1	Wireless	Phone3
	Samsung Galaxy Tab	08:21:ef:3b:fc:e3	Wireless	Tablet1
	TPLink Router Bridge LAN	14:cc:20:51:33:ea	Wireless	Gateway

5.3.1.0 Details of Feature Exploration

To efficiently highlights the essential features in the *RevealFeat* and use them in *GuardFeat*, we extract network features from the segment header by Tshark[1] (Team, 2021). The packet payload is not included in the feature extraction. Thus, the input file for the analysis is composed of network features and the associated capture time. The capture time encompass the trace capture day from the dataset. Based on this, we normalize the capture time to an interval from 1 to 20 days. We extract network features following two evaluation scenarios: *(i)* the packet scenario (PS), where we employ the features related to the network packet size; and *(ii)* the flow scenario (FS), that we use the features related to the flow, such as flow duration, as shown in Table 5.3. A flow is the

[1]This work used the following Tshark commands: "-e ip.src -e ip.dst -e ip.proto -e frame.time_epoch -e tcp.len -e udp.length -e eth.src -e eth.dst, -e tcp.srcport -e tcp.dstport -e udp.srcport -e udp.dstport".

data-plane of packets between sender and receiver that shares key IP header information (*i.e.*, 5-tuple information). Flow duration is the sum of all packets timestamp.

Table 5.3: Network and Statistical Features considered in the Scenarios

Scenarios	**Network Traffic Features**	**Statistical Features**
Packet Scenario (PS)	Packet Size, Timestamp, Inter-packet-time	Mean, standard deviation, min, max, sum, median
Flow Scenario (FS)	Number of Packets, Packet Size, End and Start Timestamp, Inter-packet-time, Flow Duration	Mean, standard deviation, min, max, sum, median

Precisely, from the 20 pcaps, in the PS scenario we extract the packet size, timestamp, and inter-packet-time (IAT), while in the FS scenario we extract the number of packets, flow size, start-end timestamps, inter-packet-time, and flow duration. In this work, the inter-packet-time (IAT) consists in the time between two sequential network packets. The start timestamp consists of the time the first packet was transmitted, and the end timestamp corresponds to the time of the last packet was transmitted, both packets belonging to the same flow. Furthermore, in data cleaning, the analysis ignores packets destined to DNS servers, broadcast, and packets with size equal to 0 since they are packets without relevant content. The feature extraction and cleaning output consists in 20 CSVs files for each capture day (from 1 to 20 days) containing the packets network features (*e.g.*, IP and MAC addresses, packet size, timestamp), and the device label (Table 5.2). Note that categorical features, such as IP and MAC address, are only used to analyse the traffic behavior in the *RevealFeat*, not been used in the classifiers.

After extracted the network traffic features by Tshark, from the CSVs, we compute statistical features to increase the dataset details. By Numpy v1.20 (developers, 2021)and Pandas v1.2 (Pandas, 2021) libraries, we compute the following statistical features: *mean, standard deviation, min, max, sum*, and *median*. In the PS scenario to calculate the values for the statistical features, we group the packets belonging to the same device and create small samples of the network traffic features (samples size = 3). The samples have size 3 because the device with smallest amount of traffic was BlipcareBP (with 57 packets). Hence, we create small samples to not loss information from devices that generate small amount of traffic. Thus, every 3 network packets, we compute the statistical features and label according to the device MAC address. For the FS scenario, we compute the statistical features from the packets belonging to the same flow. After compute statistical features, each device has its own network and statistical features for the PS and FS scenarios. Then, in each scenario (PS and FS), we merge all devices features, including the device label, into one unique final file. This final file serve as input for both analysis, *RevealFeat* and *GuardFeat*. In *RevealFeat* the final file is used as input for the feature selection algorithms, and in *GuardFeat* serve as input for the obfuscation techniques and classifiers. Note that these statistical features increase the training and testing dataset, giving more feature examples for the classifiers.

5.3.1.0 Feature Selection Algorithms

The *RevealFeat* considers two well-known statistical methods: FA and PCA since the former aims to reveal the latent features whereas the other assumes correlation between features. They are multivariate statistical methods that analyze several variables to reduce a large dimension of data to a relatively smaller number of dimensions, components, or latent features (factors). PCA maps data points from a high dimensional space to a low dimensional space while keeping all the relevant linear structures intact (Abdulhammed et al., 2019). FA is an exploratory statistical technique based on the correlation matrix between features. It considers a new group capable of representing a set of highly correlated features, known as factors. There is one major contrast between the FA and PCA. Whereas FA represents only the variance shared among a set of variables, PCA represents the total variance (including variance unique to each variable, variance common among variables, and error variance) of a set of observed variables (Widaman, 1993; Brown, 2009). Both methods highlights the key features in data that explain the differences in the observations. Note that, in this work, the terms feature and variable are used as similar.

Network traffic features often involves a high dimension and correlated data since it includes many data. As PCA and FA are also considered as dimension reduction techniques, they are suitable for network traffic analysis (Abdulhammed et al., 2019). Therefore, in *RevealFeat*, PCA and FA follow two phases. PCA: 1) pre-processes data to normalize the mean and variance, 2) It computes co-variance matrix, eigenvectors, and eigenvalues. Thus, PCA converts the original data into new set of axes called PCs by keeping the most essential features of the original data. FA: 1) normalizes the network data and defines the number of factors, 2) It calculates the correlation matrix to correlate the features. Hence, FA converts the original data into factors containing the correlated features and keeping the most representative features (Li et al., 2017a). Feature exploration using PCA and FA follow scripts created in Python, version 3.8, and R, version 3.6 (R Core Team, 2021). In R, we use specific libraries for data analysis such as FactorMiner (Lê et al., 2008) and FactorExtra libraries (Kassambara and Mundt, 2021).

5.3.1.0 Machine Learning Algorithms

This work also consider supervised algorithms to classify the IoT devices. The available dataset contains the MAC address of each device; thus, we assume the adversary has several MAC address to perform the traffic identification. The MAC address serves as input to label devices. The *RevealFeat* and *GuardFeat* consider decision tree-based and ensembles supervised algorithms. Precisely, the analyses employ the Classification and Regression Trees (CART) as decision tree-based, the Extreme Gradient Boosting (XGBoost) and Bagging (Bagg) as ensemble algorithms, and Random Forest (RF) as decision tree ensemble. In general, these algorithms work well for classification problems and with unbalanced data (Hussain et al., 2020).

Decision tree-based algorithms classify by sorting samples according to their feature values. Each node in a tree represents a feature, and each edge denotes a value that the node can have in a sample. The samples are classified starting at the origin node and concerning their feature values. After constructing the tree, the algorithms classify the new samples with features and unknown classes. It starts the classification with the root nodes and proceeding on the path corresponding to the learned values (Al-Garadi et al., 2020). In RF, several decision trees are constructed and combined to acquire a precise and robust prediction model. RF consists of numerous trees constructed randomly and trained to vote for a class. It selects the most voted class as the final classification output. Thus, the classification output is the average of the results (Al-Garadi et al., 2020). RF uses an ensemble method due to the use of the voting rule. Ensemble learning combines numerous homogeneous or heterogeneous classifiers to produce a collective output and improve classification performance. Thus, it reduces variance and is robust to over-fitting. (Al-Garadi et al., 2020). Both bagging and boosting methods follow homogeneous classifiers. XgBoost is a gradient boosting classifier based on the boosting methodology. It follows a decision tree method with minor changes. It uses the weighted average for calculating the final predictions from the trees. XgBoost is a regularization technique since it improves overall performance and reduces over-fitting compared to the pattern gradient boosting classifier. Bagging Classifier is a derivation from Bootstrap Aggregation. It re-samples the dataset dynamically to train individuals classifier models in parallel. Each model is trained in a random, determined subset of the dataset containing all considered features. The output is the average prediction from the models. We implement the ML algorithms following scripts created in Python, version 3.8, and the Scikit-learn (version 0.24), Numpy (version 1.20), Pandas (version 1.2), and Scipy (version 1.6) libraries.

5.3.2 Obfuscation Techniques

In this case study, IoTReGuard employed two well-known approaches: injecting fake traffic (FAKE-A) and packet padding (PADD-A). Both obfuscation approaches inject or alter network traffic data belonging to the IoT dataset. Thus, the feature exploration, including cleaning, extraction and pre-processing network traffic features serve as input for the fake traffic injection and packet padding approaches. The fake traffic injection adds false data into the network capture of the dataset, while the padding approach alters the number of bytes from the devices network traffic capture. Both approaches was developed in Python version 3.8 (Python, 2021), including several libraries such as SciPy version 1.6, NumPy version 1.2 (developers, 2021), MatplotLib version 3.3, Pandas version 1.2 (Pandas, 2021), Scikit-Learn version 0.24 (Developers, 2021) for the ML algorithms, and Scapy Traffic Generator version 2.19[2]. To extract network traffic features from PCAPs files was used Tshark (Wireshark) version 3.2 (Team, 2021).

5.3.2.0 Fake Traffic Injection Details (FAKE-A)

The most simple and effective method to resist traffic-based side-channel attacks is to add fake traffic onto the transmission sequence to make the adversary unable to distinguish between fake and real network packets (He et al., 2016). From the MAC addresses and devices labels, IoTReGuard generates fake network traffic data. It creates one false MAC address for each real MAC address through the Python Generate MAC library (version 1.3) (Jack, 2021). The fake and real MAC addresses are similar because they are used to label devices and create network traffic behavior similar to the original. Hence, as the domestic IoT traffic dataset contains 31 devices, it was created 31 fake devices, totaling 62 devices (reals and fakes). Periodically, when the dataset contains more than 60 fake MAC addresses, we remove them from having a sizeable fake dataset. The fake devices receive random labels to use in the ML algorithms. Moreover, as the case study follows a trace-driven approach, the fake traffic generation follows a constant rate (*i.e.*, sending packets at a fixed frequency) to inject it into the IoT dataset.

Due to the limitations of the communication bandwidth, battery energy and computing power of IoT devices, the amount of fake data should be added as low as possible. We follow different levels of fake traffic injection to compare the difference between the small and large amount of fake network traffic on obfuscation efficiency. Table 5.4 presents the different levels of amount of traffic injected into the original IoT network traffic. 600-FAKE, 1000-FAKE, and 5000-FAKE generate 600, 1000, and 5000 network packets per fake device (or per fake MAC address). In contrast, the Random-FAKE adds a dynamic amount of packets ranging from 1000 to 10000 fake packets per device. We did not change the number of bytes of the fake packets. The Traffic Generator Tool adds a standard packet size for the fake traffic: 83 or 133 bytes. There are some real devices with similar packet sizes as shown in Table 5.8. Moreover, the fake packets follow the transport-layer TCP protocol; hence, the Traffic Generator Tool considers

[2]Scapy Traffic Generator project was removed by its maintainer after we implemented it on fake traffic injecting approach into IoTReGuard. Thus, we present the results using the library since we have already obtained the results and the implementation is working.

Acknowledge (ACK) packets as one of the fake packets generated. For instance, in the 600-FAKE level, each fake device generated 600 network packets, including ACKs. The feature extraction groups packets belonging to the same source and destination IP addresses. Therefore, the number of fake packets was reduced to 100, 200, and 1000 fake packets per device, respectively, for the 600-FAKE, 1000-FAKE, and 5000-FAKE levels.

Table 5.4: Levels Considered in the Fake Traffic Injection

FAKE-A Levels	**Amount of Fake Packets**	**Total Amount of Bytes**
600-FAKE	600 fake packets per device	1.7 MB
1000-FAKE	1000 fake packets per device	2.8 MB
5000-FAKE	5000 fake packets per device	14 MB
Random-FAKE	From 1000 to 10000 fake packets per device	135 MB

The entire IoT dataset contains 11.3 GB of network traffic data. Table 5.4 shows the total amount of bytes injected by each fake traffic generation level, not been a massive amount of fake traffic when compared to the amount of real network traffic. Moreover, in the network traffic generation, we create dynamic IP addresses for all fake devices. The devices communicate with the smart home gateway. Hence, the destination IP address receives the same IP address from the original gateway. In this way, the entire network traffic, including original and fake network traffic, communicates with the smart home gateway, following a typical IoT environment. We analyze all FAKE-A levels of amount of traffic in both FS and PS scenarios. Thus, we evaluate each FAKE-A level by injecting the fake MAC addresses, fake labels, and fake network packets into the original IoT dataset. In other words, the fake traffic is mixed with the original traffic.

5.3.2.0 Packet Padding Details (PADD-A)

In this thesis, we compare the fake traffic injection approach (FAKE-A) with the packet padding approach (PADD-A) proposed by (Pinheiro et al., 2020). The authors made the code available on GitHub[3], which made it easier to reproduce the results. The authors proposed a four-level padding strategy for IoT devices. As shown in Table 5.5, the four levels are named 100-PADD, 500-PADD, 700-PADD, and 900-PADD, where the numbers denote the minimum length of the packets that will belong to the respective level after padding. The length corresponds to the number of bytes (*i.e.*, packet size). The lower the level, the fewer the number of extra bytes inserted into the packets. The authors created such levels to achieve an equilibrium between privacy and overhead. In the lowest level (100-PADD), the lengths of the packets are kept close to the original values. As the padding level increases, the privacy and overhead also increase.

Each level of the packet padding technique follows specific rules. Packets with lengths smaller than or equal to 100 are resized to 100 bytes; packets with lengths between 101 and 200 are resized to 200 bytes; and packets with lengths between 201 and 300 are resized to 300 bytes. Thus, most packets have their lengths modified, but with less overhead than at higher padding levels (Pinheiro et al., 2020). Packets with length greater than or equal to 300 bytes and smaller than 999 bytes are incremented up to 1000 bytes; lengths smaller or equal to 500 are resized to 500; lengths between 500 and 900 are incremented up to 1000 bytes. Similar to the 500-PADD, the levels 700-PADD and 900-PADD compare the lengths according to the level

[3]Packet Padding Technique on GitHub. Available in: `https://github.com/janael-pinheiro/Adaptive_Packet_Padding_Approach_for_Smart_Home_Networks_A_Tradeoff_Between_Privacy_and_Performance`. Accessed on Feb/2022.

Table 5.5: Padding Rules for the Packet Length Modification (Pinheiro et al., 2020)

PADD-A Levels	**Original Length (l)**	**Obfuscated Length**
100-PADD	$l \leq 100$	100
	$100 < l \leq 200$	200
	$200 < l \leq 300$	300
	$300 < l < 999$	random from 301 to 1000
500-PADD	$l \leq 500$	500
	$500 < l < 999$	random from 501 to 1000
700-PADD	$l \leq 700$	700
	$700 < l < 999$	random from 701 to 1000
900-PADD	$l \leq 900$	900
	$900 < l < 999$	random from 901 to 1000
All-PADD	$999 \leq l \leq 1399$	random from 999 to 1400
	$1400 \leq l < 1500$	1500

number (700 or 900) and resized to the level number or randomly incremented up to 1000 bytes. Lengths between 999 and 1399 are incremented up to 1400 bytes; and finally, lengths greater than 1400 bytes are incremented to 1500 bytes. The authors considered the same IoT traffic dataset (detailed in Subsection 5.3.1) (Sivanathan et al., 2018). They noted that packets with lengths between 300 and 1500 bytes represent a small fraction of the total traffic. This behavior can be seen in Table 5.8. Due to this, the packet padding approach randomly modifies the lengths, *e.g.*, random from 500 to 1000 bytes or random to 301 to 1000 bytes. The additional number of bytes is incremented at the end of the payload from the network layer.

Pinheiro et al. (2020) considered the trade-off between privacy and overhead. Hence, the most appropriate level depends on the network traffic volume. When the network is overloaded, the level 100-PADD performs better because it generates less overhead. Otherwise, when the network is idle, it is more appropriate to implement level 900-PADD, which produces the best privacy improvements. We compare and test all levels of PADD-A to analyze the performance. As Pinheiro et al. (2020), we compare FAKE-A and PADD-A with well-established packet padding approaches, including: linear, exponential, mice/elephants, random 255, random, and MTU. These approaches are as follows: 1) linear increases the original packet length to the nearest multiple of 128 or MTU, whichever is smaller; 2) exponential rises the original length to the next power of 2 or MTU, whichever is smaller; 3) mice/elephants, in which packets smaller than 100 bytes are incremented to this value while the others are raised to 1500 bytes; 4) random 255 inserts randomly the number of bytes between 1 and 255 bytes; 5) random, in which the number of bytes inserted in the packets is chosen randomly from the difference between the original length and the MTU; and 6) MTU, in which packets have the same length as MTU, typically 1500 bytes (Dyer et al., 2012; Pinheiro et al., 2020).

5.3.3 *RevealFeat*: Feature Exploration Results

Fig. 5.8 shows the total number of packets of each device over the capture day. It contains IoT devices with less number of packets (*e.g.*, Splug, smoke sensor, BlipcareBP), IoT devices with more number of packets (*e.g.*, motion sensor, Baby Monitor, assistant), and non-IoT devices (*e.g.*, gateway, printer, smartphone, laptop, and tablet). Note that non-IoT devices comprise usual and well-known devices such as smartphones and tablets. Some IoT devices have reached small values for the number of packets, such as the BlipcareBP and Smart Plug (Splug). BlipcareBP presents a tiny amount of traffic since it only generated traffic for three days. In contrast, the camera reaches more than 10^4 number of packets and the tablet 10^6 packets, while other devices present similar behavior. This behavior indicates that the dataset contains unbalanced data (*i.e.*, dataset presents a diverse amount of data), which is a typical case for an IoT environment.

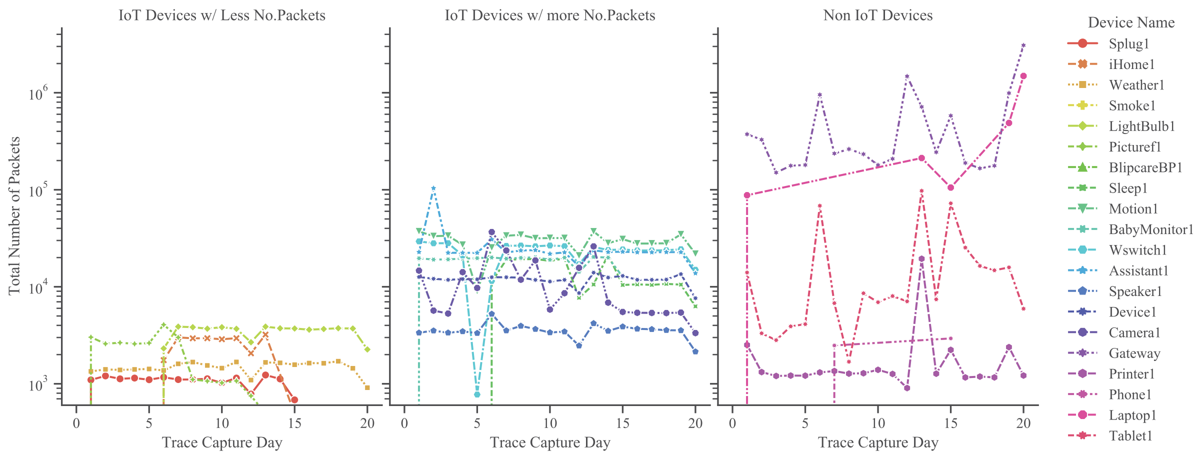


Figure 5.8: Network Traffic Behavior

Figure 5.9 shows the total number of packets and bytes (log) of the entire dataset, containing all devices. Different from Figure 3.4, Figure 5.9 contains 6 monitor cameras, 3 laptops, 3 smartphones, and others. Gateway, Belkin Wemo Switch (Wswitch), laptops, cameras, tablet, and Amazon Echo Assistant comprise devices with more amount of traffic. In contrast, smart thing devices, BlipcareBP, Smoke alert, Smart plug are the devices with a smaller amount of traffic. Such network traffic behavior is usual for the IoT environment since such IoT devices with a small amount of traffic did not generate continuous traffic, but only when some event happened. For instance, the smoke alert only generates traffic when someone is smoking near the device, while monitoring cameras generate continuous network traffic. Thus, it is harder to identify the behavior of small packet count. Feature exploration reveals device traffic patterns and highlights the key features that most represent such patterns and behavior. It assists in traffic and device identification. However, it also can be used by adversaries.

In the feature exploration (*RevealFeat*), we used the PCA and FA to highlight the most relevant network features from the IoT dataset. The number of PCs and Factors is less than or equal to the number of original features. As we follow two evaluation scenarios, PS and FS, in the packet scenario we consider the packet size (number of bytes), timestamp, number of packets, and IAT. In contrast, in the flow scenario we consider the number of packets, packet size (number of bytes), end and start timestamp, IAT, and flow duration (see Table 3.3). In this work, we use packet size and number of bytes to describe the size of network packets. For the PS scenario, Table 5.6 displays 4 PCs and 3 Factors results since this scenario contains four network traffic features. The number of factors is only three because the first three already contain the most representative features. The first component (PC1) heavily weights features related to the size of packets, achieving 0.70 of correlation between the number of packets and number of bytes. The second component (PC2) heavily weights features related to the time, with a high correlation (0.71) between timestamp and IAT. PC3 presents high variance for IAT, and the last component (PC4) for the number of packets, without strong correlation with another feature

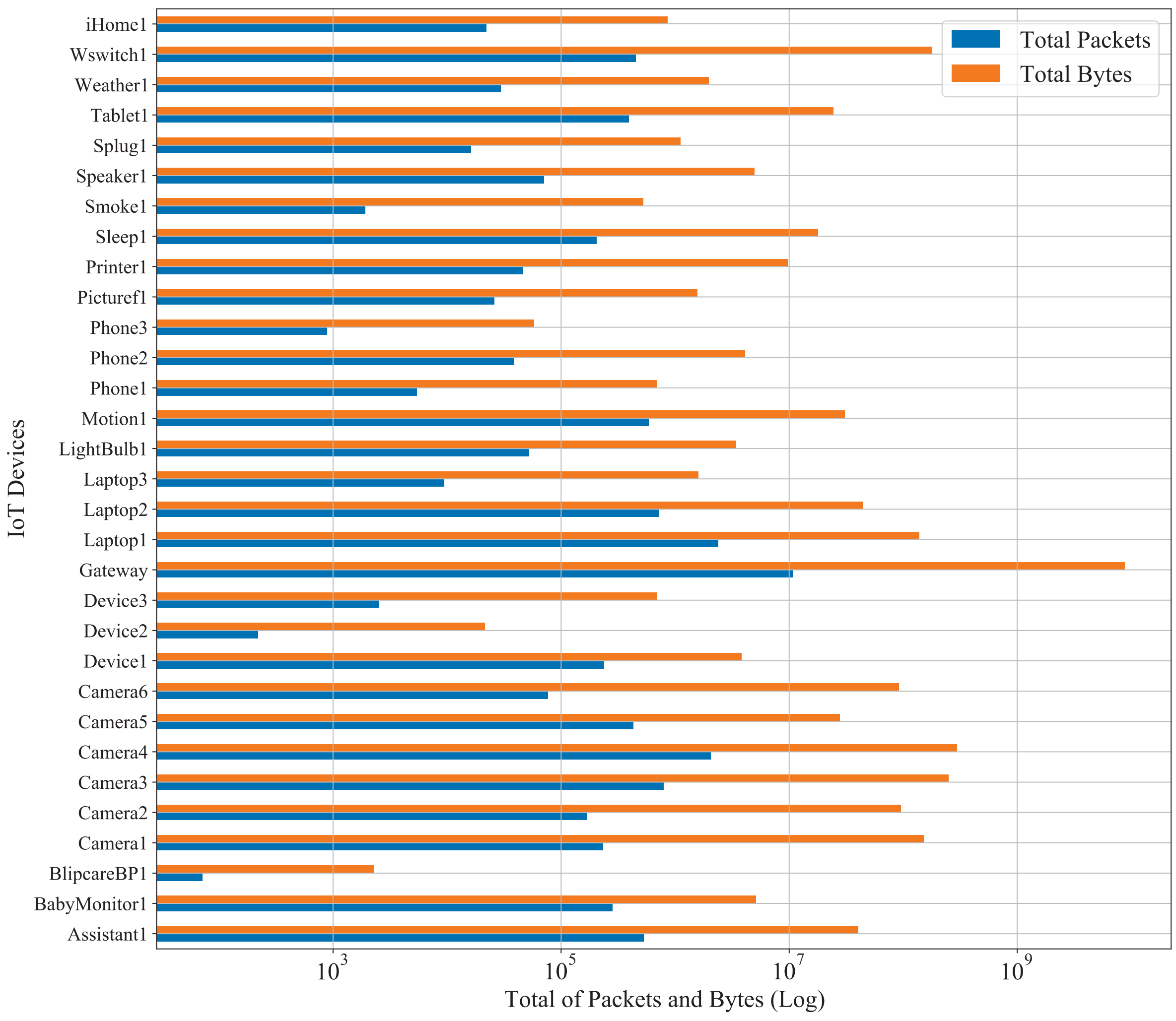


Figure 5.9: Total of Packets and Bytes generated by the Devices (Log)

since the correlation was achieved in PC1 and PC2. The proportion of variance and cumulative proportion under PCA assist in understanding the most representative PCs (and features). The proportion of variance analyzes how much each Factor or PC contains variance representing the original data. Closer to 100%, better to represent the data. The cumulative proportion quantifies the representation value. PC1, PC2, and PC3 represent 50%, 32%, and 17% of the original data, respectively, achieving 99% of cumulative proportion. PC4 only increase 1% of contribution, not been so representative. Hence, the three first PCs are enough to describe the original data.

Table 5.6: PCs and Factors Results from Network Traffic Features (PS Scenario)

Netw. Feature	**PCA**				**FA**		
	PC1	**PC2**	**PC3**	**PC4**	**F1**	**F2**	**F3**
No. Packets	**0.70**	-0.03	0.04	0.70	**0.99**	0.11	-0.07
No. Bytes	**0.70**	-0.04	0.10	-0.70	**0.99**	0.01	0.00
Timestamp	0.10	**0.69**	-0.71	-0.03	0.06	**0.60**	0.25
IAT	0.03	**0.71**	**0.70**	0.02	-0.04	**0.23**	0.57
Proportion of Variance	0.50	0.32	0.17	0.002	0.71	0.15	0.14
Cumulative Proportion	0.50	0.82	**0.99**	**1.00**	0.71	0.86	**1.00**

FA analysis presents similar results. For FA evaluation, it is necessary to run previous tests on the data to verify the dataset suitability and determine the "optimal" number of factors. A well-known and used test is the Kaiser-Meyer-Olkin (KMO) (DeCoster, 1998). KMO statistic is a variance proportion among variables that might be common variance: varies from zero to one, in which zero is inadequate, while close to one is adequate. Overall, KMO values equal to or above 0.50 are considered acceptable. We run KMO in data and achieve 50% as a result. Hence, our data is suitable. Moreover, to determine the number of factors, the Kaiser criterion analyzes the eigenvalues of each factor and cumulative variance. It suggests using a number of factors equal to the number of eigenvalues greater than one. We achieved eigenvalues higher than 1 (precisely, 2 and 1.28) for the two first factors, while F3 achieved 0.71. However, we decided to keep the F3 since it contributes to the cumulative variance proportion. The extraction technique most used is the PCA (Widaman, 1993). Thus, our FA is based on PCA to extract the variance. Finally, we used the varimax rotation from R to run FA. Table 5.6 presents the FA results. The first factor (F1) heavily weights features related to the size and number of packets (with 99% of correlation), second factor (F2) strongly correlates with time-related features (60% and 23% of correlation). F3 presents a high description for IAT. F1, F2, and F3 represent 71%, 15%, and 14% of the original network data, respectively, achieving 100% of cumulative proportion. Therefore, both analyses, FA and PCA, in PS presented the features related to the size of packets contribute more to representing the original data when compared to the other features.

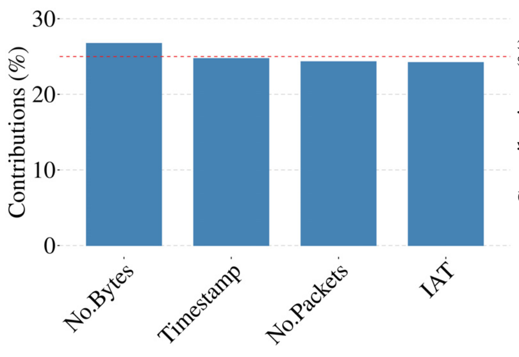
Table 5.7 displays 4 PCs and 3 Factors results for the FS scenario. Although this scenario contains six traffic features, it was unnecessary to create one PC for each feature since they are highly correlated. For instance, the first component (PC1) and factor (F1) strongly correlate with start and end timestamp features. The second component (PC2) and factor (F2) heavily weigh features related to the size and number of packets. Finally, PC3 and F3 achieve a high correlation between IAT and flow duration since both features are computed by time information. In comparison, PC4 did not present a correlation. PC1, PC2, and PC3 represent 33%, 33%, and 29% of the original network data, respectively, achieving 95% of cumulative proportion. PC4 only increase 4% of contribution, not been so representative. F1, F2, and F3 represent 33%, 33%, and 25% of the original network data, respectively, achieving 92% of cumulative proportion. Therefore, the first three components and factors have the most representative feature correlation and describe the entire dataset. In the adversary perspective, by analyzing the PCA and FA results, an adversary may decide to use only one feature from each combination since the results showed a high correlation, *e.g.*, the number of packets and start timestamp.

Table 5.7: PCs and Factors Results from Network Traffic Features (FS Scenario)

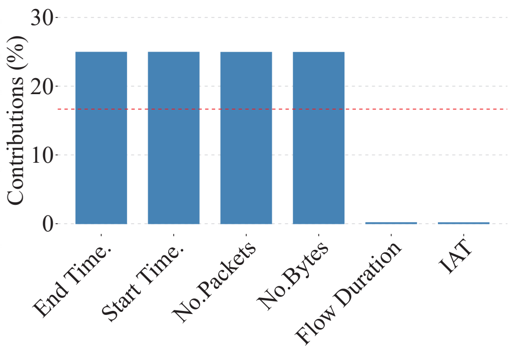
Netw. Feature	**PCA**				**FA**		
	PC1	**PC2**	**PC3**	**PC4**	**F1**	**F2**	**F3**
No. Packets	0.041	**0.70**	-0.007	0.013	0.058	**0.99**	-0.009
No. Bytes	0.041	**0.70**	-0.011	-0.019	0.057	**0.99**	-0.014
IAT	0.052	0.003	**0.70**	-0.70	0.074	0.004	**0.93**
Start Time	**0.70**	-0.041	-0.053	-0.0009	**0.99**	-0.058	-0.070
End Time	**0.70**	-0.041	-0.052	-0.0006	**0.99**	-0.058	-0.069
Flow Duration	0.054	0.009	**0.70**	0.70	0.076	0.012	**0.93**
Proportion of Variance	0.33	0.33	0.29	0.042	0.33	0.33	0.25
Cumulative Proportion	0.33	0.66	0.95	**0.99**	0.33	0.66	**0.92**

Figures 5.10(a) and 5.10(b) show the proportion of contribution of each feature for the PS and FS scenarios, and Figures 5.10(c) and 5.10(d) present the percentage of variance

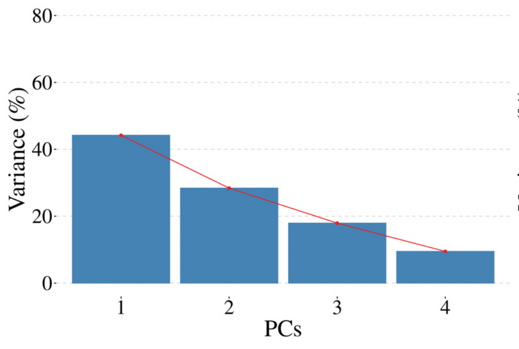
represented by each PC. The proportion of contribution highlights the most representative features in describing the dataset. For instance, Figure 5.10(a) presents a similar contribution for all features. Only the number of bytes (packet size) stands out over others. Thus, the number of bytes is the most representative feature for PS scenario because it contains more variance of data when compared to the others features. Figure 5.10(b) shows the same contribution for end and start timestamp, number of bytes, and number of packets. The percentage of contribution corroborates the results of the PCA and FA. Furthermore, Figures 5.10(c) and 5.10(d) confirm that only three PCs and Factors are enough to represent the original data. When the dispersion of the number of principal components until the curve of the individual variance of each component becomes flat or drops abruptly, it indicates that much variance was lost. Therefore, one should stop extracting components (it is similar for factors).



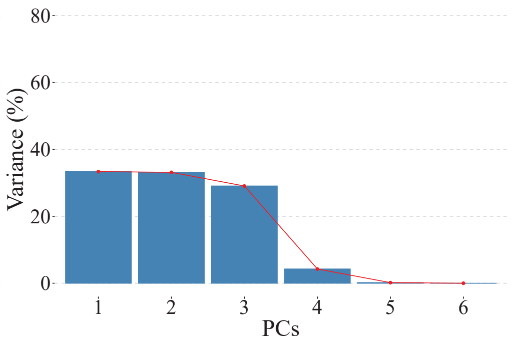
(a) Packet Scenario



(b) Flow Scenario



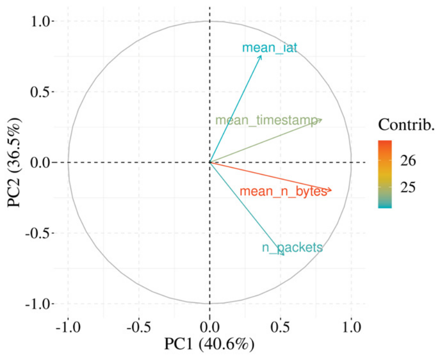
(c) Packet Scenario



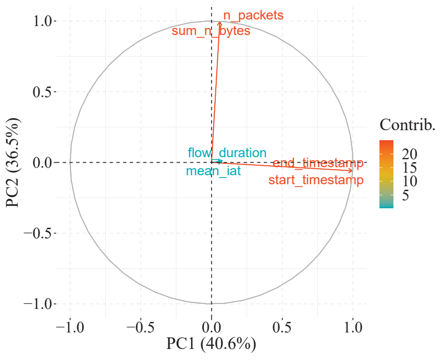
(d) Flow Scenario

Figure 5.10: Proportion of Contribution and Variance of Network Traffic Features

Through proportion of contribution, it is possible to point out that the key feature is the number of packets in the PS scenario. In the FS scenario, there are timestamps, the number of packets, and bytes. These key features are confirmed in Figures 5.11(a) and 5.11(b) since they present the features correlation and contribution under PCA. Considering the PS scenario, the number of bytes represents PC1, while PC2 reflects the number of packets and IAT due to the high variance of each one. PC1 is dominated by time-related features (start and end timestamp) in the FS scenario, while PC2 correlates the number of bytes and packets. Flow duration and IAT have presented the lowest contribution because they have values based on the timestamp values. In other words, it is most effective to consider the difference between timestamp and number of packets than IAT and number of packets due to the values distance and variance. Therefore, PCA and FA highlighted the key features: number of bytes and timestamp. Such a level of information offers rich insights into network traffic behavior and traffic classification. The PCA and FA results allow the selection of features with a higher contribution for further analysis.



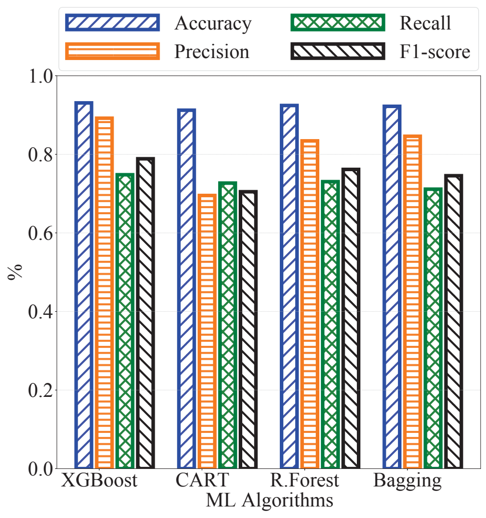
(a) Packet Scenario



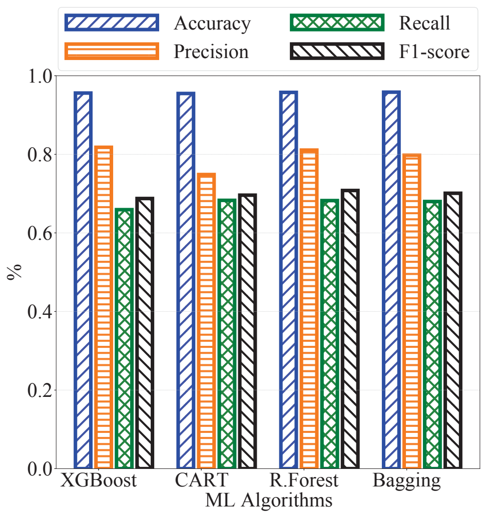
(b) Flow Scenario

Figure 5.11: Correlation of Network Features under PCA

PCA and FA revealed that the most relevant features involve the number of bytes for PS and timestamps, number of bytes, and packets for FS. To confirm features impact in classification, Figure 5.12 and 5.13 compare the results for accuracy, precision, recall, and F1-score using all set of features and only the key features for each scenario. XGBoost, CART, Random Forest, and Bagging were used to perform the comparison. Figures 5.12(a) and 5.12(b) show the ML algorithms results using all set of features and only key feature for PS. Using all set of features, the algorithms achieved around 78% of F1-score and more than 80% of precision. Considering only the main key feature, number of bytes, performance presented a small decrease because the training features reduced from four (timestamp, IAT, number of bytes, and packets) to only one feature. Although decreased performance, it was not so significant when comparing the ML results. Random Forest and Bagging achieved 88% and 92% of precision, respectively, using all features, while using only key features, they presented 81% and 79% of precision, respectively. Therefore, such results confirmed the relevance of key features on describing data.



(a) PS - All Set of Features

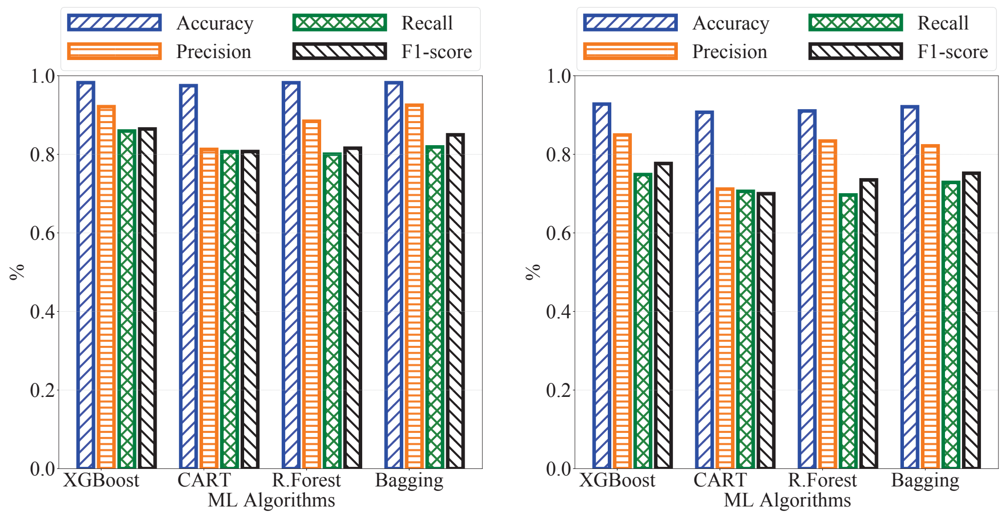


(b) PS - Key Features (No.of Bytes)

Figure 5.12: PS: Comparing the use of non-relevant feature through different ML Algorithms

Figures 5.13(a) and 5.13(b) show the ML algorithms results using all set of features and only key feature for FS. In both settings for FS, the ML algorithms presented similar results.

Regarding all features, the algorithms achieved around 75% and 84% of F1-score and precision, respectively. Considering only the key features (end and start timestamp, number of bytes, and packets), the algorithms presented around 74% and 80% of F1-score and precision, respectively. Hence, the IoT device identification still achieved a high performance only considering the key features revealed by PCA and FA. These results confirm the low relevance of IAT and flow duration presented in the feature exploration results since their use did not affect ML performance. Therefore, feature exploration and ML algorithm comparison indicate that number of bytes and timestamps features are the most relevant features, followed by the number of packets. However, network traffic-based attacks take benefits from this information to learn user routine and violate privacy. These attacks generally consider packet size (number of bytes) or timestamp as the network traffic feature (Prates et al., 2020). It is possible to infer information from a smart home routine by identifying and learning the device's behaviors.



(a) FS - All Set of Features

(b) FS - Key Features (Timestamps, No. of Bytes and Packets)

Figure 5.13: FS: Comparing the use of non-relevant feature through different ML Algorithms

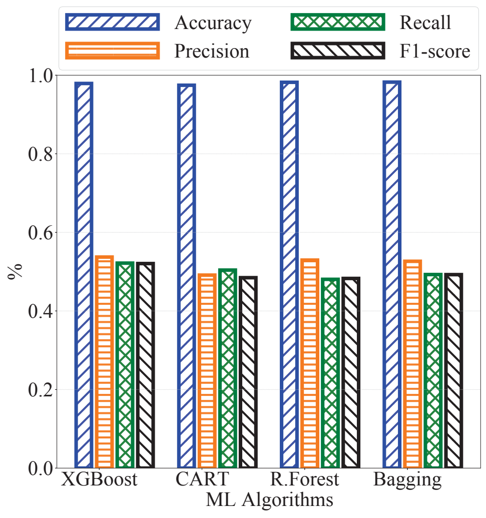
Table 5.8 shows the similarities and differences in the devices' packet size in order to detail the device behavior characteristics. Device2, a switch and trigger for smart home, generated the smallest packet size with a mean of 5 bytes and achieved a maximum of 105 bytes. In contrast, Wswitch1 (Belkin Wemo Switch Plug) presented the largest packet size achieving 682 bytes (mean). There are several devices with values 0 for median and mode because such devices did not send packets every day (*i.e.*, there are days without network traffic from them). For instance, BlipcareBP achieved 0 bytes for mode because it did not send network packets over 17 days. It is also possible to observe a slight difference between similar devices. For instance, the three traffic size categories (small, medium, and large) contain monitoring cameras. Some cameras generated more traffic than others due to the position of the camera in the environment, frequency of people passing by or movement, or even being turned off someday. However, is still possible to identify some pattern and learn the camera behavior and, consequently, the user behavior. Monitoring cameras with a tiny amount of traffic can indicate few or no people in the environment. Therefore, traffic-based attacks benefit from such device characteristics to learn the device behavior, and then, users' behavior. As this critical and private information is obtained by network traffic analysis, PCA and FA combined with ML algorithms can boost the attack's success and impact. Thus, defense mechanisms to mask the relevant features can make it unfeasible to infer sensitive information by the adversary (Prates et al., 2020).

Table 5.8: Devices Characteristics on Network Packet Size

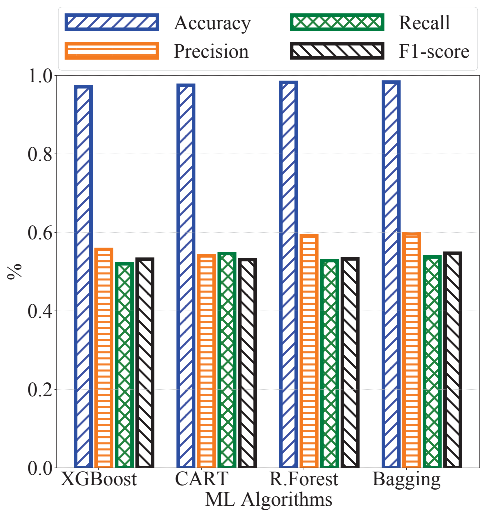
Traffic Size	**Device Label**	**No. of Bytes**			
		Median	**Mean**	**Mode**	**Max**
Small	Device2	0	5	0	105
	BlipcareBP1	0	7	0	53
	Device1	19	20	18	26
	Phone3	0	20	0	190
	Laptop3	0	23	0	471
	Phone1	0	29	0	299
	iHome1	0	38	0	156
	LightBulb1	70	52	70	72
	Camera6	0	68	0	1363
	Camera5	66	71	66	472
	Picturef1	107	73	0	126
	Laptop1	0	76	0	562
	Splug1	116	89	116	149
	Phone2	0	93	0	388
Medium	Weather1	101	103	100	111
	Laptop2	0	122	0	669
	Camera2	70	130	0	855
	Assistant1	123	138	122	217
	Camera4	120	138	108	393
	Sleep1	183	145	0	203
	Printer1	79	166	78	852
	Speaker1	205	209	204	250
	Motion1	174	224	172	579
	Tablet1	233	235	235	328
	BabyMonitor1	332	254	0	365
Large	Scale1	327	328	327	337
	Smoke1	374	365	374	404
	Camera3	429	448	424	663
	Camera1	379	460	39	1146
	Gateway	230	584	792	1381
	Wswitch1	711	682	711	719

5.3.4 *GuardFeat*: Feature Masking Results

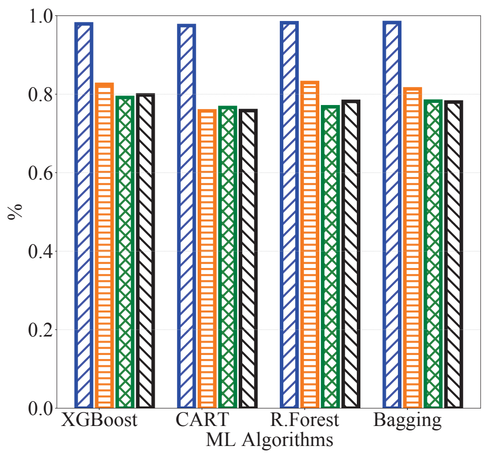
This section presents a quantitative comparison on network traffic feature obfuscation since IoTReGuard supports different obfuscation approaches. IoTReGuard method employed two approaches: injecting fake traffic (FAKE-A) and packet padding (PADD-A). Considering the *PS Scenario* and a holdout evaluation (testing and training datasets), Figure 5.14 displays the obfuscation results related to the fake traffic injection approach (FAKE-A). Without defenses against traffic-based side-channel attacks, classifiers achieved $\approx 82\%$ and $\approx 78\%$ of precision and F1-Score, respectively (see Figure 5.12(a)). Surprisingly, the 600-FAKE level obtained the best results on reducing the performance of the classifiers in the IoT device identification, as shown in Figure 5.14(a). As the FAKE-A sends additional traffic to hide real traffic, we expected the heavier the fake traffic, the higher the network traffic obfuscation. However, the obfuscation results showed the contrary. As many traffic samples are given to the supervised classifier, better will be its performance. A rich training dataset with large network traffic samples increases the probability of a classifier identifying the devices with success. Due to this, the Random-FAKE level presented high performance results ≈ 95% of F1-Score for all classifiers. Random-FAKE added from 1000 to 10000 fake network traffic packets per real device (135 MB of fake traffic). Compared to the IoT traffic dataset size (13 GB of real traffic), 135 MB is a small size value. However, it was enough to increase the results of the classifiers.



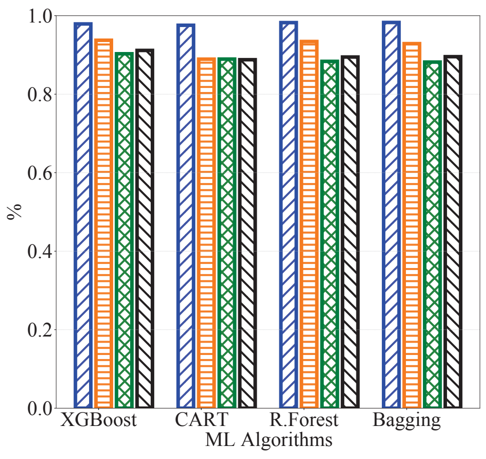
(a) 600-FAKE Level



(b) 1000-FAKE Level



(c) 5000-FAKE Level

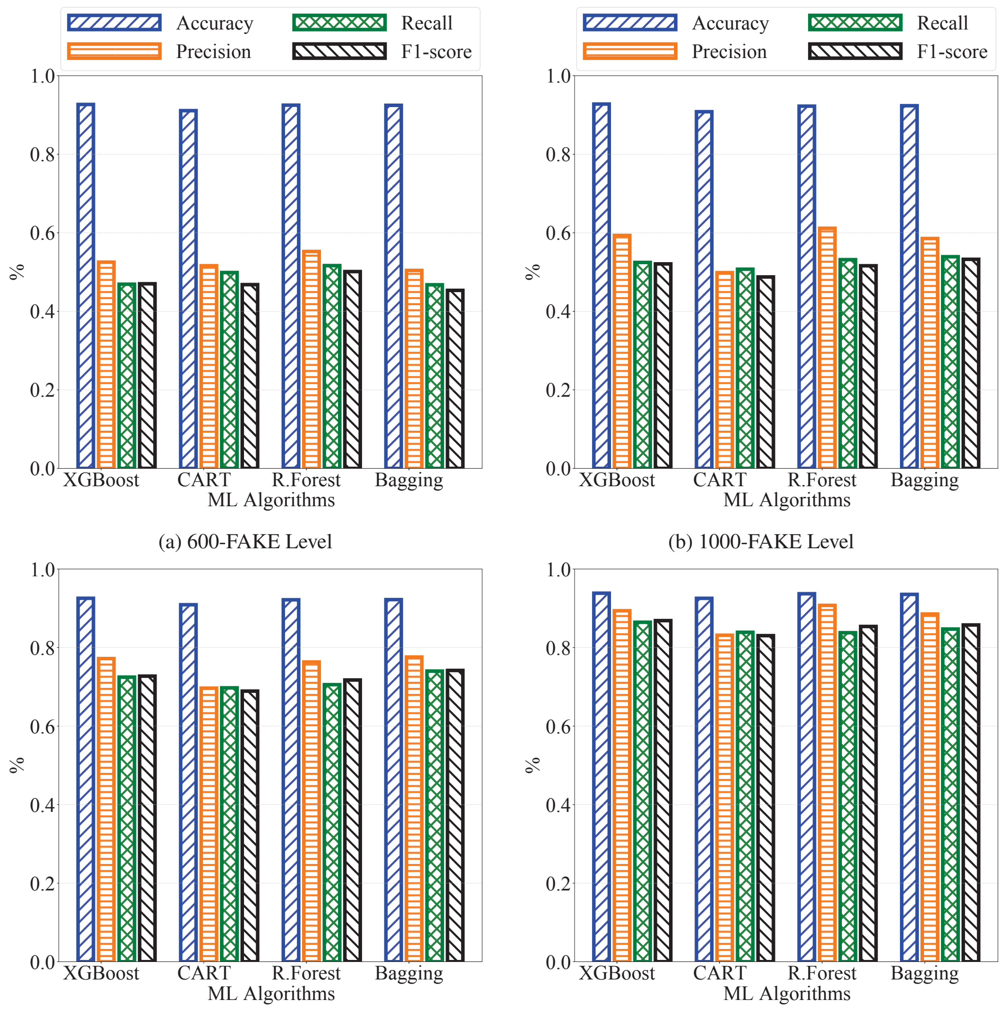


(d) Random-FAKE Level

Figure 5.14: FAKE-A Approach Results on *PS Scenario*

The 600-FAKE and 1000-FAKE levels achieved similar results on the network traffic obfuscation due to the amount of fake packets generated. As the FAKE-A follow the TCP protocol, the 600 and 1000 fake packets include ACK network packets. The feature extraction grouped packets belonging to the same source and destination IP addresses. Thus, the number of fake packets was reduced to 100 and 200 for the 600-FAKE, and 1000-FAKE levels, respectively. From the 31 fake devices created, the 1000-FAKE level was generated more 310 fake packets than the 600-FAKE level. The amount of fake traffic from both levels justifies the similar results on the classifiers. Nonetheless, it is possible to note a tiny improvement in the 1000-FAKE level classifiers results. Therefore, the 600-FAKE presented the best obfuscation results, which is good for the IoT environment since the less fake traffic added, the lower the network bandwidth overhead. Moreover, as presented in Table 5.8, most IoT devices generated a small number of traffic packets. Hence, to obfuscate typical IoT devices, a small amount of fake network packets is a potential approach to follow in defense mechanisms.

The FAKE-A obfuscation approach presented similar performance results for both *PS* and *FS Scenarios*. Figure 5.15 presents the FAKE-A results in *FS Scenario*. The 600-FAKE level also achieved the best results on masking the traffic from IoT devices. To evaluate the *FS Scenario*, we created dynamic port numbers and IP addresses to possibly create the flow 5-tuple (same source and destination IP addresses, and same source and port numbers). Once we generated fake traffic packets without changing any traffic feature value, the FAKE-A did not affect any specific network traffic feature considered in the scenarios. Then, it did not present different results for masking traffic features. For instance, in FAKE-A, we did not change the packet size values of the fake traffic. This justifies the similar results in both scenarios.



(a) 600-FAKE Level

(b) 1000-FAKE Level

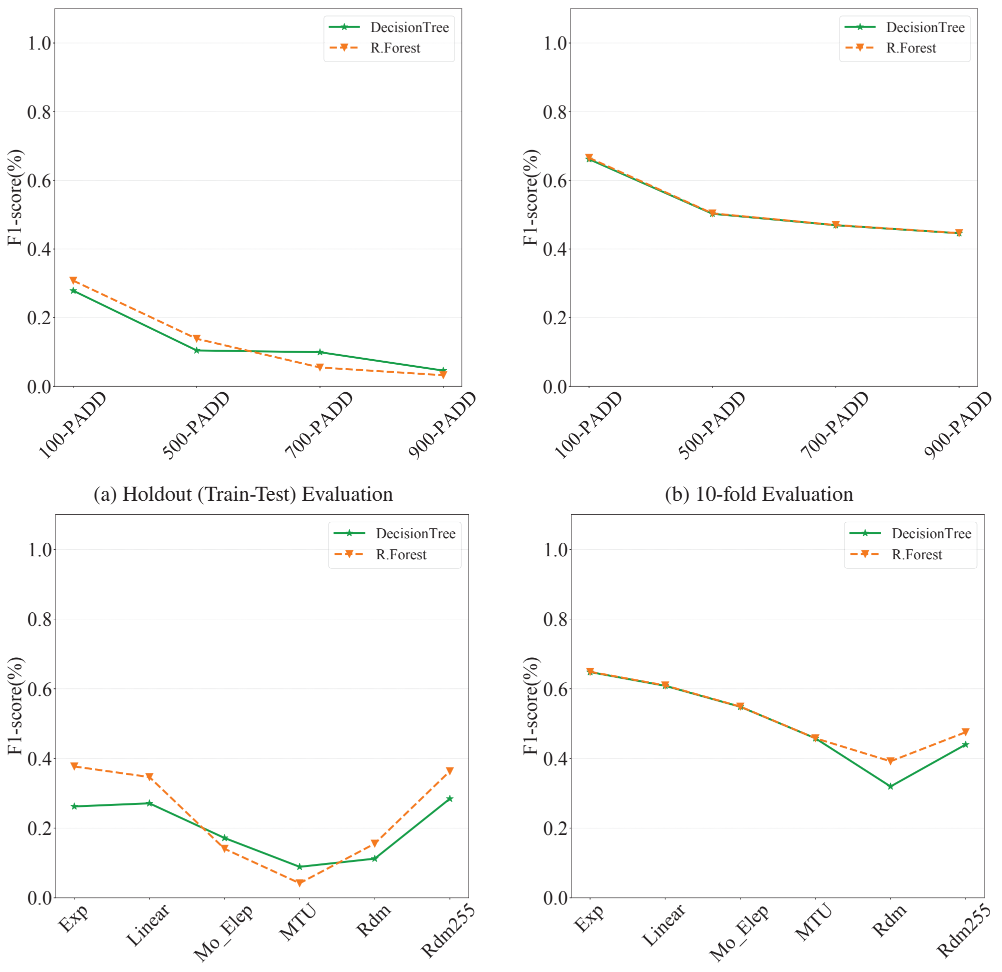
(c) 5000-FAKE Level

(d) Random-FAKE Level

Figure 5.15: FAKE-A Approach Results on *FS Scenario*

Once the PADD-A approach from (Pinheiro et al., 2020) followed only the network packet-level scenario in their analysis, the following results present the obfuscation performance considering only the *PS scenario*. Moreover, we considered decision tree-based classifiers as (Dyer et al., 2012; Pinheiro et al., 2020), including Random Forest, CART, and Decision Tree. Note that CART and Decision Tree were implemented using the Decision Tree Classifier from Scikit-Learn, which allows considering both classifiers as similar. The classifiers followed the holdout (split data into testing and training dataset) and 10-fold cross-validation evaluations. We defined parameter $k = 10$ for the cross-validation according to (Pinheiro et al., 2020).

Figure 5.16 shows results under the packet padding approaches considering ML algorithms also used by (Pinheiro et al., 2020). Figures 5.16(a) and 5.16(b) present the PADD-A proposed by (Pinheiro et al., 2020), while Figures 5.16(c) and 5.16(d) displays the well-established packet padding approaches (exponential, linear, mouse/elephants, MTU, random, and random 255). The holdout evaluation presented a better performance on reducing the success of the classifiers in identifying IoT devices. The 900-PADD and MTU achieved the best obfuscation results, reducing the F1-Score metric to $\approx 7\%$. This behavior was expected because the higher the packet padding level, the more uniform the packet sizes are, *i.e.*, probably all the network packets will have the same size. However, even the lowest level of packet padding, 100-PADD, reduced the performance of the classifiers to $\approx 32\%$, a significant performance degradation. One of the factors that may explain this performance degradation is that most IoT devices from the dataset generated packets with values around 100 bytes. Because of this dataset traffic behavior, 500-PADD also considerably reduces the performance of the classifiers ($\approx 15\%$), and the Random 255 approach achieved similar results to the 100-PADD.



(a) Holdout (Train-Test) Evaluation

(b) 10-fold Evaluation

(c) Holdout (Train-Test) Evaluation

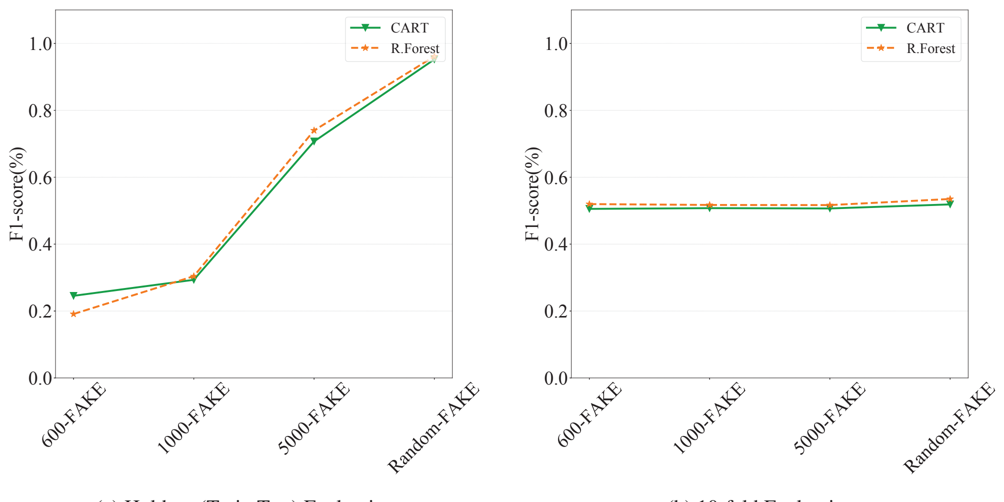
(d) 10-fold Evaluation

Figure 5.16: Packet Padding Approaches under *PS Scenario*

As the goal of a network traffic obfuscation approach is to reduce the performance of classifiers, the 10-fold cross-validation presented the worst results compared to the holdout evaluation. The 10-fold evaluation tests more data combinations since it splits the dataset into

10 groups to classify the devices. In contrast, holdout only tests the training dataset with the testing dataset. Thus, even padding the packet size of IoT traffic to difficult the classifiers, the 10-fold still achieved considerable classification results. The 100-PADD level and Exponential presented ≈ 62% of F1-Score. Even the 900-PADD level obtained 43% of F1-Score, similar to the 600-FAKE results. Therefore, the holdout evaluation with 900-PADD and MTU achieved the best obfuscation results. Nonetheless, they also increase the network bandwidth usage because the packet size has values around 1500 bytes. The highest value possible to a network packet. In contrast, the 100-PADD and 600-FAKE, and even the 1000-FAKE and 500-PADD, added fewer bytes in the network bandwidth, which is more feasible for IoT environments.

To be a fair comparison between the fake traffic injection (FAKE-A) and packet padding (PADD-A) approaches, we generate classification results for the FAKE-A only considering the packet size and its statistics properties (mean, standard deviation, min, max, sum, and median). The above-presented results of FAKE-A (Figure 5.14) considered all the network traffic features, including the timestamp. As (Pinheiro et al., 2020) only used packet size as a network feature. We removed the other features from the analysis. Note that the packet size was pointed out as one of the most important features by the feature exploration. Hence, the classification results without defense approach, considering only the packet size (see Figure 5.12(b)), achieved 69% and 70% of F1-Score in holdout evaluation for CART and R.Forest, respectively. In 10-fold cross-validation, the classifiers obtained ≈ 50% of F1-Score. Figures 5.17(a) and 5.17(b) present the FAKE-A approach results for holdout and 10-fold evaluations. The results in Figure 5.17(a) show that the FAKE-A using the 600-FAKE level can significantly reduce the performance of the analyzed classifiers. Compared to the results without defense approaches, the 600-FAKE in holdout reduced from 70% to ≈ 20% of F1-Score, a very similar result to the 100-PADD level from packet padding. The 1000-FAKE also reduced the performance of the classifiers.



(a) Holdout (Train-Test) Evaluation (b) 10-fold Evaluation

Figure 5.17: FAKE-A Results Considering only Size-based Network Traffic Features

In contrast, the 5000-FAKE and Random-FAKE improved the results of the classifiers because they increased the number of network traffic samples. Based on the results presented, we can conclude that the fake traffic injection approach in the IoT environment did not require a massive amount of fake traffic since the IoT devices generate a tiny amount of traffic. In the IoT Traffic Dataset, the device that generated more amount of packets was the gateway, which is not a typical IoT device. Typical IoT devices generally send small packet sizes, ≈ 50 or 80 bytes. Thus, adding a small amount of fake network traffic is a good approach to obfuscate the traffic from such devices. Such a conclusion is a good direction for the IoT environment since it comprises devices with low processing and energy power. Adding much fake traffic (Random-FAKE) or increasing the size of the packet to a considerable size (MTU and 900-PADD) can harm network bandwidth. Nonetheless, although the significant performance degradation presented by FAKE-A and PADD-A in the lowest levels, the adversary still can identify devices. Thus, a combination of both approaches can be a direction to improve the obfuscation of IoT devices.

5.3.5 Discussion and Limitations

The IoTReGuard method followed a feature exploration on network traffic, in an adversary perspective, and defense approaches against traffic-based attacks. Over the last decade, there has been a significant number of attempts in developing feature exploration for network traffic analysis and defense mechanisms against traffic-based side-channel attacks. However, there are outstanding problems and challenges in the feature exploration field, mainly considering adversities, such as traffic-based attacks. In this vein, several works proposed defense mechanisms against traffic-based side-channel attacks, in different contexts (*e.g.*, websites and IoT). Nonetheless, the defenses still present problems and open issues regarding network overhead. Therefore, this section presents the open issues, challenges and opportunities to point out future directions.

5.3.5.0 Feature Exploration Optimization

Currently, feature exploration is primarily a manual, time consuming task. Unfortunately, raw data pose no clear advice or insight into which variables must be focused on. Network data usually contain several features but not all of them contribute towards predictive modeling. Moreover, each ML model will respond differently to different types of features. Accordingly, a major challenge involves the decision making on feature exploration (Shen et al., 2020). In order to find the set of features that best suit to each ML model, it is necessary to test and analyze the classification or clustering results, requiring time and increasing the computational cost. It also costs time to find the best feature selection algorithm for specific classification problems (Li et al., 2017b). Hence, advances to optimize and facilitate these decisions can help develop ML processes, *e.g.*, development of a general method to remove redundant and irrelevant features that fit for different classification problems.

5.3.5.0 Streaming of Feature Exploration

In general, the feature exploration is processed offline since it collects, filters, and extracts network features. However, network traffic is real time, and entails dynamic and streaming data. In the worst case, the size of data are unknown or even infinite. Thus, it is not practical to wait until all data are available to perform feature exploration. Further, network administrators need to detect abnormal traffic in real-time and mitigate in time. Therefore, it is preferred to perform feature exploration streaming to adapt to the changes on the fly. Nonetheless, at the same time, an adversary also can boost the impact of its attack by monitoring the network traffic from the user home instantaneously. Thus, there is a trade-off between network monitoring and attack impact related to real-time monitoring. Feature exploration in data streaming adds the most recently arrived network data, extracts features to add to the set, until no new data show up anymore (Li et al., 2017b). However, dimensionality limits, especially in IoT environments, and processing time of the algorithms, require further investigation.

5.3.5.0 Security and User Privacy

Although feature exploration in network traffic analysis assists traffic classification, it opens breaches for security concerns (Hussain et al., 2020). Based on the presented results, one can achieve detailed information on network traffic from feature exploration. PCA and FA results have highlighted the features that better contribute in the representation of the network traffic. Such output gives insight on what feature use in device classification. On a malicious perspective, the adversary could use the most relevant features to perform the attack. However, by identifying the devices, the adversary can learn the device behavior and the user pattern. For instance, a movement sensor follows an on/off approach. It is on when it is generating traffic. Hence, by analyzing the sensor's traffic, the adversary can learn when someone is at home and the number of people inside the home. Therefore, this critical and sensitive information is obtained by network traffic analysis and feature exploration, improving the attack impact. Hence, traffic obfuscation mechanisms (*e.g.*, fake traffic injection and packet padding approaches) are essential to avoid such attacks (Prates et al., 2020). Such mechanisms mask the network features, making unfeasible to identify devices and infer sensitive information by the adversary.

5.3.5.0 Network Traffic Obfuscation

Traffic obfuscation can be employed to mask traffic from specific devices. For instance, the IoT device and the gateway can perform the network traffic obfuscation. In case an IoT device itself performs traffic masking, it will protect the device against any adversaries within the same network as IoT devices. Meanwhile, if every device performs traffic obfuscation individually, it will increase the overall bandwidth and energy consumption of IoT devices. It will also require significant effort to integrate traffic masking in IoT devices due to limited energy, memory, and processing power. In contrast, the smart home gateway performing the masking will protect all IoT devices. In general, the obfuscation aims to achieve a constant traffic rate and smooth the variations and peaks on the network traffic. The in-home user activities are identified by such peaks. Thus, if there are multiple IoT devices in the network, the volume of fake traffic needed to achieve a constant rate will be less than the total volume of fake traffic needed to mask background traffic of each device individually. In this way, it seems reasonable to develop defenses against traffic-based attacks tailored for network context and behavior.

Moreover, the fake traffic injection approach can add unnecessary delay in the original packets, even following a constant rate or a specific distribution. In the constant rate, the delay depends on the transmission interval. However, it is difficult to determine an appropriate transmission interval. The delay will vary with the time interval. When the time interval increases, the delay time will also increase. The amount of fake traffic and additional energy consumption will be increased significantly (He et al., 2016). It also can present patterns for traffic-based attacks using timing network features. A method following a random sequence to transmit fake traffic can assist on obfuscation. It also can add other patterns to the traffic. Hence, it is difficult to determine the most appropriate sequence on generating fake traffic. For the effect of privacy protection, the noise data should not be recognized. However, the noise can reduce the correct identification of the devices. The adversary will identify wrong behavior patterns. However, high latency negatively affects IoT device functionality because an additional delay in connectivity to cloud services can render IoT devices non-functional in some cases. One possible approach is prioritize real traffic over fake traffic to ensure that there is no additional latency and IoT devices can perform their normal functioning, as it can be critical, *e.g.*, health IoT devices.

5.3.5.0 Privacy versus Network Overhead

Several network traffic obfuscation approaches send additional traffic to hide real traffic or pad the packet size. Consequently, such approaches consume additional bandwidth. Depending on the network context, *e.g.*, IoT, such additional consumption can be critical. However, the bandwidth consumed by IoT traffic is relatively small due to the limited functionality of IoT devices. As shown in our obfuscation results, the amount of fake traffic needed to mask IoT traffic is very small. Even for IoT devices with high bandwidth consumption, such as some monitoring cameras and smoke sensors (Table 5.8), the amount of fake traffic necessary is small (from 330 to 600 bytes). The low levels from the padding and fake traffic injection approaches hide such traffic without increasing the amount of traffic in a huge proportion. Therefore, the fake traffic injected is reasonably limited and does not incur high additional costs. The packet padding approach, depending on the length of the packets, can add high additional costs. It is crucial, especially in the IoT context, to follow low levels of packet padding not to increase the number of bytes. Analyzing the network volume to send additional traffic when it is idle also can be a solution. However, the in-home user does not have enough knowledge to control when to send additional fake traffic, as proposed by (Pinheiro et al., 2020). Thus, it is necessary to evaluate the network bandwidth in an experimental scenario to analyze the cost of masking approaches and develop an automatic approach to change the amount of traffic according to the home network behavior.

5.3.5.0 Health-related Devices

The IoT environment contains health-related devices, such as a Blipcare Blood Pressure and a Sleep Sensor. In general, such devices generate a tiny amount of traffic. Thus, the low levels of the fake traffic injection and padding packet can protect their privacy. However, health-related devices require specific performance, such as low latency, to correctly work since they collect health and sensitive data. They do not support high additional costs and network overhead (Newaz et al., 2021). The costs of countermeasures against traffic-based attacks can be high (from tens to hundreds of additional exchanged data) for health-related device communications where energy consumption is crucial (Barman et al., 2021). Therefore, it is essential to highlight the need to design new defenses tailored to the health-related setting because the countermeasures against traffic-based attacks still provide insufficient protection. Although they yield a drop in the adversary's accuracy, it still is possible to identify some behavior and devices. The effectiveness of these defenses varies greatly with the adversarial task: although the defenses drop the attacker's accuracy, non-adapted defenses have small effect yet still incur high costs.

5.4 SUMMARY

This chapter presented the IoTReGuard method. The IoTReGuard method explores network traffic features to reveal the key ones and mask them to protect users' privacy. It innovates by quantifying feature exploration's potential impact when adversaries exploit it on IoT network traffic. Therefore, IoTReGuard, based on network traffic capture, reveals the key network traffic features through a feature exploration employing statistical techniques. Then, it masks network traffic data or specific features by obfuscation techniques to prevent traffic-based side-channel attacks. The results obtained by the feature exploration revealed the packet size is a key feature for IoT network traffic. Moreover, the IoTReGuard method revealed different traffic patterns for the IoT devices, allowing to get knowledge on device usage and users' daily routine. The IoTReGuard method reduced from 70% to $\approx$ 20% of F1-Score on identifying the IoT devices traffic in the masking traffic results. Compared to the literature's packet padding approach, the

IoTReGuard method and packet padding approach achieved similar obfuscation results in the lowest obfuscation levels (600-FAKE and 100-PADD). Once there is a trade-off between traffic obfuscation and network overhead, this level of obfuscation is interesting for the IoT environment. However, the total network traffic was not masked, keeping some traffic to the adversary. Thus, there are still open issues to be addressed. Therefore, we concluded that feature exploration reveals crucial information from the IoT environment. An adversary can use such information to perform attacks. However, this information also can be used to develop tailored defense mechanisms. Hence, network traffic analysis and feature exploration are crucial to improving the defense mechanisms against traffic-based attacks.

6 CONCLUSIONS AND FUTURE WORKS

This chapter summarizes and concludes this work. It also presents future works, publications, and participation in research projects. The objective is to reinforce the goals of this work and the contributions we have achieved and point out possible future directions for research community. Therefore, this chapter follows four sections. Section 6.1 presents the goals of this work and contributions, as well as the lessons learned. Section 6.2 discusses the future works and open issues for research community. Section 6.3 lists the publications and submissions related to this thesis. Finally, Section 6.4 presents the research projects participation.

6.1 GOALS AND CONTRIBUTIONS

The main goals of this work consist of structure and organizing network data efficiently through network traffic analysis and instrumentation to support other network actions. Such actions can involve network management and security. Thus, this work introduced the FLIPER methodology to analyze the network traffic data, structure it, and organize it to obtain valuable information to support data-driven decisions and network management operations (*e.g.*, performance and security). The FLIPER methodology comprises steps to perform an efficient network traffic analysis considering data instrumentation. We realized that efficiently analyzing data can provide valuable information for obtaining patterns and behaviors, assisting network management, improving network security, allowing specialized services, and others.

The main contributions of this work consist of the FLIPER methodology that efficiently analyses network traffic data to support network management. Thus, we presented the MOTIF method to identify legitimate traffic of health-related applications. MOTIF considers IoT devices' network traffic as input to identify different s-health applications. Optimal results on fingerprinting applications were reached; some test scenarios presented 100% of accuracy (Vergütz et al., 2019). To analyze the FLIPER support in the network performance management, we introduced PRIMUS architecture to assist network management through network slicing. As network slicing is central to realizing the potential of 5G networks, PRIMUS manages traffic through network slicing technologies. PRIMUS managed network slicing, network resources availability, and application requirements. The latency achieved less than 8 ms for URLLC. Considering a 5G scenario will be possible to achieve 1 ms for urgent care applications.

Finally, to analyze the support of network traffic analysis on network security, we presented the IoTReGuard method that analyzes and highlights the impacts of an efficient network traffic analysis from the adversary perspective. IoTReGuard method explores network traffic features to reveal the key ones and mask the traffic of smart home devices. Based on network traffic analysis and features exploration, the IoTReGuard method discovered the most important feature from a smart home scenario: packet size. Results showed that an obfuscation approach could mask only the key features without reducing the performance. Obfuscation approaches, packet padding, and fake traffic injection reduced the privacy violation caused by traffic-based attacks. They reduced from 80% to 20% the precision of the classifiers. Therefore, although few mentioned the FLIPER methodology in the methods and architecture, its steps serve as input for data-driven decisions. Hence, the steps of the FLIPER methodology offered support in different ways for the MOTIF method, the PRIMUS architecture, and the IoTReGuard method. The FLIPER methodology is the starting point for creating adaptive network solutions and offering assistance on data-driven decisions in traffic identification, network management, and security.

Therefore, regarding the research questions of this work and the results achieved, the answers are as follows: 1) Is it possible to create a broader methodology for network data instrumentation to support different purposes, such as network performance and security activities? Answer for question 1: Yes. The FLIPER methodology was instantiated by the MOTIF and IoTReGuard methods, supporting the network data instrumentation for different reasons. The FLIPER methodology was used in the MOTIF method to identify applications, while in the IoTReGuard supported the feature exploration and traffic obfuscation. The PRIMUS architecture also used the valuable information obtained by network data instrumentation to support the resource allocation. $1.A$) If we instrument network data efficiently and apply some intelligence, is it possible to know what application/device generates traffic? Answer for question $1.A$: Yes. The MOTIF method identified the traffic of different applications. $1.B$) This knowledge can assist network management and the achievement of smart application requirements? Answer for question $1.B$: Yes. Based on the traffic applications identified by the MOTIF method, the PRIMUS architecture adapted the network resources of s-health applications dynamically and decreased the latency. 2) Once an efficient network data instrumentation assists network management, can adversaries also benefit to violate users' privacy and security? Answer for question 2: Yes. The IoTReGuard method identified the traffic of IoT devices, allowing to infer users' activities and violate privacy. $2.A$) Network data instrumentation can also assist the tailored development of network security mechanisms? Answer for question $2.A$: Yes. Based on the relevant features pointed out by network data instrumentation and feature exploration, the IoTReGuard method masked only the relevant features and achieved good performance.

Finally, from our learning on network data instrumentation and feature exploration in the IoT context, and based on the literature review, an efficient network data instrumentation support and assist the following situations:

- Network data instrumentation offers the knowledge of what application or device generates the network traffic. It assists the special treatment according to the demands of the applications based on the knowledge on which device and application the traffic belongs;

- Network data instrumentation supports the data-driven decision on adaptive resources allocation. Resource allocation randomly or without knowledge on applications and traffic patterns generate the poor use of network resources, even letting some in an idle state;

- Network feature exploration reveals the most representative network traffic features assisting in obtaining useful patterns, behaviors, and others. It also supports the creation of tailored defense mechanisms;

- Network data instrumentation and feature exploration support data-driven decisions, *e.g.*, resources allocation according to traffic identification and obfuscation of specific features to protect users' privacy;

- Network data instrumentation can discover gaps or relevant features before adversaries, allowing the implementation of defense mechanism with antecedence;

- Network data instrumentation offers data-driven decisions for infrastructure providers prepare the infrastructure according to demands of applications, *e.g.*, applications requiring ultra-low latency and reliability did not support radio communication technologies since they can suffer noises, delays, and packet losses due to weather instability;

- Network data instrumentation supports the prior infrastructure preparation, *e.g.*, creating slicing or isolation for specific application traffic;
- Network data instrumentation allows the construction of diverse systems customized to devices, applications, and requirements due to offer previous and valuable information;
- Network data instrumentation supports the 5G network scenarios and uses cases by giving information related to the applications and demands. It allows to create of virtualized or physical infrastructure according to the demands, *e.g.*, specific for autonomous vehicles;

In summary, network traffic analysis and data instrumentation support several situations and network operations by offering valuable information for data-driven decisions. It supports network administrators, clients, infrastructure providers, network management, network security, and many others. It also supports IoT and the applications requirements, assisting in achieving the emergent and envisioned smart scenarios. Therefore, network traffic analysis and data instrumentation are the prior steps for the development of adaptive network solutions.

6.2 OPEN ISSUES AND FUTURE WORKS

This section presents and discusses the open issues and future directions under each contribution topic of this thesis. Thus, it discusses the open issues regarding the FLIPER methodology, MOTIF method, PRIMUS architecture, and the IoTReGuard method. It also offers future directions to the research community in order to persist the works. First, this thesis introduced the FLIPER methodology to support network actions based on our learning. Although the steps of FLIPER methodology were applied in different ways in the MOTIF and IoTReGuard methods, it is still important to apply and evaluate in other scenarios, such as in a 5G network scenario with smart applications. Moreover, evaluating the FLIPER methodology in a real and online environment is also interesting to analyze the time taken to perform the network traffic analysis and how it impacts smart and time-critical applications.

Considering smart health applications, the MOTIF method achieved good performance results in identifying the applications traffic. However, there is a trade-off between classification time and performance on fingerprinting s-health applications since both metrics are crucial. Once s-health critical applications require low-latency (*e.g.*, 1 ms), the classification time needs to be small. Furthermore, poor performance on fingerprinting results in not identifying critical applications can harm the QoS of such applications. We advocate for combining the cross-validation method based on the statistical properties of flow volume data since they have already reached optimal results by the MOTIF method. Nevertheless, further studies are necessary to reduce the classification time. Moreover, considering another dataset with IoT network traffic can assist in improving the MOTIF and IoTReGuard methods.

The PRIMUS architecture also contains concerns, such as the threshold duration time to drop some slice, the time taken to create a slice, heterogeneous networks, user mobility, and network slicing implementation. The network slicing concept is still in construction in the literature, requires several improvements to be implemented in the smart scenarios. For instance, research directions are necessary to determine the threshold duration time to drop the network slice since such threshold needs to consider each application's requirements and the healthy functioning of other slices. Another concern is related to heterogeneous network technologies along the end-to-end path. Network slicing has the end-to-end characteristic of using virtualization (SDN and NFV), which assist in addressing such heterogeneity. However, network technologies need to support virtualization. Finally, user mobility can also impact application latency and reliability. Therefore, it is necessary to evaluate PRIMUS architecture in a mobility scenario.

The IoTReGuard method revealed sensitive and relevant information from IoT devices. Analyzing the day capture and behavior makes it possible to learn user activities. However, feature selection and obfuscation approaches can increase the cost and time. IoT devices and applications did not support high costs since they are low-power devices with energy restrictions. Moreover, the obfuscation approaches can increase the network overhead. Although the literature addresses the trade-off between privacy and network overhead, it is still an open issue due to the task complexity. Finally, there are research opportunities to optimize feature exploration and testing in streaming data since it can be applied in online environments.

6.3 PUBLICATIONS AND SUBMISSIONS

This section presents the list of publications results from this thesis and manuscripts under reviews. It also presents the list of publications indirectly related to this thesis.

6.3.1 Directly Related

This section presents the list of publications resulted from this thesis.

- *An Architecture for the Performance Management of Smart Healthcare Applications*. A. Vergütz, N. Prates Jr, B. Schwengber, A. Santos, M. Nogueira. **SENSORS**, v. 20, p. 5566, 2020.

- *Reliability for Smart Healthcare: A Network Slicing Perspective*. A. Vergütz, G. Noubir, M. Nogueira. Special issue on "Big Data Intelligent Networking " - **IEEE Network Magazine**, July 2020.

- *A Method for Identifying eHealth Applications using Side-Channel Information*. A. Vergütz, I. Medeiros, D. Rosário, E. Cerqueira, A. Santos, and M. Nogueira. In **IEEE GLOBECOM 2019**, Hawaii, USA.

- *Um Método de Ofuscação para Proteger a Privacidade no Tráfego da Rede IoT*. B. V. dos Santos, A. Vergütz, M. Nogueira, R. T. Macedo. In **Anais do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC**, 2022, Brazil.

  The following manuscripts are in preparation for submission or are under review.

- *Data Instrumentation from IoT Network Traffic as Support for Security Management*. A. Vergütz, B. V. dos Santos, B. Kantarci, M. Nogueira. **IEEE Transactions on Network and Service Management**, Open Call 2022.

- *A Dynamic Method to Protect User Privacy Against Traffic-based Attacks on Smart Home*. B. V. dos Santos, A. Vergütz, M. Nogueira, R. T. Macedo. **IEEE GLOBECOM 2022**, Rio de Janeiro, Brazil.

- *Feature Exploration for Network Traffic Security: A Quantitative Adversarial Perspective*. A. Vergütz, U. Brezolin, B. Kantarci, M. Nogueira. **IEEE Internet of Things Magazine**, Open Call 2022.

6.3.2 Indirectly Related

This section presents the list of publications indirectly related to this thesis.

- *Um Método para Detecção de Vulnerabilidades Através da Análise do Tráfego de Rede IoT*. U. Brezolin, N. Prates Jr, A. Vergütz, M. Nogueira. In **Anais do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC**, 2022, Brazil.
- *Learning from Network Data Changes for Unsupervised Botnet Detection*. B. Schwengber, A. Vergütz, N. Prates Jr, M. Nogueira. **IEEE Transactions on Network and Service Management**, v. 1, p. 1-1, 2021.
- *A Method Aware of Concept Drift for Online Botnet Detection*. B. Schwengber, A. Vergütz, N. Prates Jr, M. Nogueira. In: **IEEE GLOBECOM 2020**, Taipei, Taiwan.
- *A Defense Mechanism for Timing-based Side-Channel Attacks on IoT Traffic*. N.Prates Jr, A. Vergütz, R. Macedo, A. Santos, M.Nogueira. In: **IEEE GLOBECOM 2020**, Taipei, Taiwan.
- *Análise de Vazamentos Temporais Side-Channel no Contexto de Internet das Coisas*. N. Prates Jr, A. Vergütz, R. Macedo, M. Nogueira. In **Anais do Workshop de Gerência e Operação de Redes e Serviços no Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - WGRS/SBRC**, 2019, Gramado, Brasil.
- *Um Mecanismo de Defesa contra Ataques Temporais Traffic Side-Channel no Contexto de Internet das Coisas*. N. Prates Jr, A. Vergütz, R. Macedo, M. Nogueira. In **Simpósio Brasileiro de Segurança de Redes - SBSeg**, 2019, São Paulo, Brasil.
- *Assessing Data Traffic Classification to Priority Access for Wireless Healthcare Application*. P. Resque, S. Costa, D. Rosário, E. Cerqueira, A. Vergütz, M. Nogueira, A. Santos. In **IEEE LATINCOM 2019**, Salvador, Brazil.

  The following manuscripts are in preparation for submission or are under review.

- *A Method for Vulnerability Detection by IoT Network Traffic Analytics*. U. Brezolin, A. Vergütz, M. Nogueira. **IEEE GLOBECOM 2022**, Rio de Janeiro, Brazil.

6.4 PARTICIPATION ON RESEARCH PROJECTS

Some parts of this thesis have contributions from the following research projects:

- HealthSense Project: The HealthSense was a cooperative project among the Northeastern University (USA), Federal University of Paraná (Brazil) and Federal University of Pará (Brazil). It was selected in the Brazil-USA joint public call organized by the RNP (Rede Nacional de Ensino e Pesquisa) and NSF (National Science Foundation). The project had two main goals: i) Analyze and explore characteristics of wearable devices, Apps and network stacks and ii) Proposes a resiliency technique, using the body's own conducting medium, to protect the transmission of the secret information. This thesis work is associated to the network traffic analysis and characteristics exploration. The MOTIF method was constructed during the HealthSense Project.

- GT-Arquimedes Project: The GT-Arquimedes is a cooperative project of technological transition among Federal University of Paraná (UFPR), Federal University of Minas Gerais (UFMG), and EarlySec Startup, and National Research Network (free translation from Rede Nacional de Pesquisa - RNP). The project aims to identify IoT vulnerabilities based on network traffic analysis.t comprises the devices identification and traffic obfuscation. Thus, the IoTReGuard was constructed during the GT-Arquimedes project.

REFERENCES

3GPP (2016). 3rd generation partnership project; technical specification group services and system aspects; feasibility study on new services and markets technology enablers; stage 1 (release 14). Technical report, 3GPP 5G Initiative.

3GPP (2018). 5G, system architecture for the 5G system (3GPP TS) 23.501 version 15.2.0 release 15. Technical report, 3GPP 5G Initiative.

5GPPP (2015). 5GPPP white paper on eHealth vertical sector. Technical report, 5G Infrastructure Association and others.

Abdellatif, A. A., Mohamed, A., Chiasserini, C. F., Tlili, M., and Erbad, A. (2019). Edge computing for smart health: Context-aware approaches, opportunities, and challenges. *IEEE Network*, 33(3):196–203.

Abdulhammed, R., Faezipour, M., Musafer, H., and Abuzneid, A. (2019). Efficient network intrusion detection using PCA-based dimensionality reduction of features. In *Proceedings of the International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–6. IEEE.

Acar, A., Fereidooni, H., Abera, T., Sikder, A. K., Miettinen, M., Aksu, H., Conti, M., Sadeghi, A.-R., and Uluagac, S. (2020). Peek-a-boo: I see your smart home activities, even encrypted! In *Proceedings of the Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec'20, page 207–218, New York, NY, USA. ACM.

Aceto, G., Ciuonzo, D., Montieri, A., and Pescapé, A. (2017). Traffic classification of mobile apps through multi-classification. In *Proceedings of the Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE.

Afolabi, I., Taleb, T., Samdanis, K., Ksentini, A., and Flinck, H. (2018). Network slicing and softwarization: A survey on principles, enabling technologies and solutions. *IEEE Communications Surveys and Tutorials*, 20(3):2429–2453.

Ajaeiya, G., Elhajj, I. H., Chehab, A., Kayssi, A., and Kneppers, M. (2018). Mobile Apps identification based on network flows. *Springer Knowledge and Information Systems*, 55(3):771–796.

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys and Tutorials*, 17(4):2347–2376.

Al-Garadi, M. A., Mohamed, A., Al-Ali, A. K., Du, X., Ali, I., and Guizani, M. (2020). A survey of machine and deep learning methods for internet of things (IoT) security. *IEEE Communications Surveys and Tutorials*, 22(3):1646–1685.

Alan, H. F. and Kaur, J. (2016). Can android applications be identified using only TCP/IP headers of their launch time traffic? In *Proceedings of the Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, pages 61–66. ACM.

Alcock, S. and Nelson, R. (2013). Measuring the accuracy of open-source payload-based traffic classifiers using popular internet applications. In *Proceedings of the Conference on Local Computer Networks (LNC)*, pages 956–963. IEEE.

Alliance, N. (2016). Description of network slicing concept, NGMN 5G P1 requirements and architecture, work stream end-to-end architecture. Technical report, NGMN 5G.

Aouini, Z., Kortebi, A., Ghamri-Doudane, Y., and Cherif, I. L. (2018). Early classification of residential networks traffic using C5.0 machine learning algorithm. In *Proceedings of the Wireless Days Conference (WD)*, pages 46–53. IEEE.

Apthorpe, N., Reisman, D., and Feamster, N. (2017a). Closing the blinds: Four strategies for protecting smart home privacy from network observers. *arXiv preprint arXiv:1705.06809*.

Apthorpe, N., Reisman, D., Sundaresan, S., Narayanan, A., and Feamster, N. (2017b). Spying on the smart home: Privacy attacks and defenses on encrypted IoT traffic. *arXiv preprint arXiv:1708.05044*.

Apthorpe, N., Yuxing Huang, D., Reisman, D., Narayanan, A., and Feamster, N. (2018). Keeping the smart home private with smart (er) IoT traffic shaping. *arXiv e-prints*, pages arXiv–1812.

Atkinson, J. S., Mitchell, J. E., Rio, M., and Matich, G. (2018). Your WiFi is leaking: What do your mobile apps gossip about you? *Elsevier Future Generation Computer Systems*, 80:546–557.

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, 54(15):2787–2805.

Avizienis, A., Laprie, J.-C., Randell, B., and Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33.

Azad, P., Navimipour, N. J., Rahmani, A. M., and Sharifi, A. (2019). The role of structured and unstructured data managing mechanisms in the internet of things. *Cluster Computing*, 23(2):1185–1198.

Baker, S. B., Xiang, W., and Atkinson, I. (2017). Internet of things for smart healthcare: Technologies, challenges, and opportunities. *IEEE Access*, 5:26521–26544.

Barman, L., Dumur, A., Pyrgelis, A., and Hubaux, J.-P. (2021). Every byte matters: Traffic analysis of bluetooth wearable devices. *ACM Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT)*, 5(2).

Bernaille, L., Teixeira, R., and Salamatian, K. (2006). Early application identification. In *Proceedings of the Conference on Future Networking Technologies (CoNEXT)*, pages 1–6. ACM.

Birman, P. K. (2005). *Reliable distributed systems: technologies, web services, and applications*. Springer Science and Business Media.

Boulos, M. N. K., Brewer, A. C., Karimkhani, C., Buller, D. B., and Dellavalle, R. P. (2014). Mobile medical and health apps: state of the art, concerns, regulatory control and certification. *Online Journal of Public Health Informatics*, 5(3):229.

Bouzidi, E. H., Outtagarts, A., Hebbar, A., Langar, R., and Boutaba, R. (2020). Online based learning for predictive end-to-end network slicing in 5G networks. In *Proceedings of the International Conference on Communications (ICC)*, pages 1–7. IEEE.

Brazil, F. S. o. (1988). Federal constitution of Brazil (free translation of constituição da república federativa do Brasil). *Brasília: Senado Federal, Centro Gráfico*.

Brown, J. D. (2009). Principal components analysis and exploratory factor analysis definitions, differences, and choices. *Statistics*, 13(1):26–30.

Burns, J., Cheng, A., Gurung, P., Rajagopalan, S., Rao, P., Rosenbluth, D., Surendran, A. V., and Martin, D. (2001). Automatic management of network security policy. In *Proceedings of the DARPA Information Survivability Conference and Exposition II. DISCEX'01*, volume 2, pages 12–26. IEEE.

Caballero, P., Banchs, A., de Veciana, G., and Costa-Pérez, X. (2017). Network slicing games: Enabling customization in multi-tenant networks. In *Proceedings of the International Conference on Computer Communications (INFOCOM)*, pages 1–9. IEEE.

Callado, A. C., Kamienski, C. A., Szabó, G., Gero, B. P., Kelner, J., Fernandes, S. F., and Sadok, D. F. H. (2009). A survey on internet traffic identification. *IEEE Communications Surveys and Tutorials*, 11(3):37–52.

Cao, J., Yang, Z., Sun, K., Li, Q., Xu, M., and Han, P. (2019). Fingerprinting SDN applications via encrypted control traffic. In *Proceedings of the International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, pages 501–515. USENIX Association.

CDW (2017). The future is data-driven, but IoT has its challenges. Available: `https://www.forbes.com/sites/cdw/2017/05/02/the-future-is-data-driven-but-iot-has-its-challenges/?sh=3fb796af1459`. Accessed February, 2022.

Chaddad, L., Chehab, A., Elhajj, I. H., and Kayssi, A. (2018). App traffic mutation: toward defending against mobile statistical traffic analysis. In *Proceedings of the Conference on Computer Communications (INFOCOM)*, pages 27–32. IEEE.

Chaddad, L., Chehab, A., Elhajj, I. H., and Kayssi, A. (2021). Optimal packet camouflage against traffic analysis. *ACM Transactions on Privacy and Security (TOPS)*, 24(3):1–23.

Chan, K. H., Babiarz, J., and Baker, F. (2006). Configuration guidelines for DiffServ service classes (RFC 4594). Technical report, Internet Engineering Task Force (IETF).

Cisco (2019). Cisco Joy. Available: `https://github.com/cisco/joy/wiki`. Accessed February, 2022.

Cisotto, G., Casarin, E., and Tomasin, S. (2020). Requirements and enablers of advanced healthcare services over future cellular systems. *IEEE Communication Magazine*, 58(3):76–81.

Clark, D., Braden, R., and Shenker, S. (1994). Integrated services in the internet architecture: an overview (RFC 1633). Technical report, Internet Engineering Task Force (IETF).

Conti, M., Li, Q. Q., Maragno, A., and Spolaor, R. (2018). The dark side (-channel) of mobile devices: A survey on network traffic analysis. *IEEE Communications Surveys and Tutorials*, 20(4):2658–2713.

da República, P. (2018). Lei geral de proteção de dados pessoais (LGPD). Available: http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/L13709.htm. Accessed February, 2022.

Dainotti, A., Pescape, A., and Claffy, K. C. (2012). Issues and future directions in traffic classification. *IEEE Network*, 26(1):35–40.

Datta, T., Apthorpe, N., and Feamster, N. (2018). A developer-friendly library for smart home IoT privacy-preserving traffic obfuscation. In *Proceedings of the Workshop on IoT Security and Privacy (IoT SP'18)*, pages 43–48. ACM SIGCOMM.

De la Torre Díez, I., Alonso, S. G., Hamrioui, S., López-Coronado, M., and Cruz, E. M. (2018). Systematic review about QoS and QoE in telemedicine and eHealth services and applications. *Springer Journal of Medical Systems*, 42(10):182.

de Sousa, N. F. S., Perez, D. A. L., Rosa, R. V., Santos, M. A., and Rothenberg, C. E. (2019). Network service orchestration: A survey. *Elsevier Computer Communications*, 142:69–94.

DeCoster, J. (1998). Overview of factor analysis. Retrieved from http://stat-help.com/factor.pdf. Accessed February, 2022.

developers, N. (2021). *NumPy*. Python3 package version 1.20.1. Accessed February, 2022.

Developers, S.-L. (2021). *Scikit-Learn: Machine Learning in Python*. Package version 0.24. Accessed February, 2022.

Ding, L., Liu, J., Qin, T., and Li, H. (2017). Internet traffic classification based on expanding vector of flow. *Elsevier Computer Networks*, 129(1):178–192.

Diyanat, A., Khonsari, A., and Shariatpanahi, S. P. (2016). A dummy-based approach for preserving source rate privacy. *IEEE Transactions on Information Forensics and Security*, 11(6):1321–1332.

dos Reis Fontes, R. and Rothenberg, C. E. (2019). Mininet-WiFi: Plataforma de emulação para redes sem fio definidas por software. In *Anais Estendidos do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 201–208. SBC.

Ducange, P., Mannarà, G., Marcelloni, F., Pecori, R., and Vecchio, M. (2017). A novel approach for internet traffic classification based on multi-objective evolutionary fuzzy classifiers. In *Proceedings of the International Conference on Fuzzy Systems*, pages 1–6. IEEE.

Dyer, K. P., Coull, S. E., Ristenpart, T., and Shrimpton, T. (2012). Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail. In *Proceedings of the Symposium on Security and Privacy*, pages 332–346. IEEE.

Erman, J., Arlitt, M., and Mahanti, A. (2006). Traffic classification using clustering algorithms. In *Proceedings of the Workshop on Mining Network Data (SIGCOMM/MineNet)*, pages 281–286. ACM.

FDA (2015). Mobile medical applications: Guidance for food and drug administration staff. Technical report, U.S. Department of Health and Human Services Food and Drug Administration.

Feghhi, S. and Leith, D. J. (2018). An efficient web traffic defence against timing-analysis attacks. *IEEE Transactions on Information Forensics and Security*, 14(2):525–540.

Fontes, R. d. R. and Rothenberg, C. E. (2016). Mininet-wifi: A platform for hybrid physical-virtual software-defined wireless networking research. In *Proceedings of the Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '16, pages 607–608. ACM.

Foresta, F., Cerroni, W., Foschini, L., Davoli, G., Contoli, C., Corradi, A., and Callegati, F. (2018). Improving OpenStack networking: Advantages and performance of native SDN integration. In *Proceedings of the International Conference on Communications (ICC)*, pages 1–6. IEEE.

Foukas, X., Patounas, G., Elmokashfi, A., and Marina, M. K. (2017). Network slicing in 5G: Survey and challenges. *IEEE Communications Magazine*, 55(5):94–100.

Frank, J. (1994). Machine learning and intrusion detection: Current and future directions. In *Proceeding of the National Computer Security Conference*, pages 1–6. NIST.

Fu, X., Graham, B., Bettati, R., and Zhao, W. (2003a). On effectiveness of link padding for statistical traffic analysis attacks. In *Proceedings of the International Conference on Distributed Computing Systems*, pages 340–347. IEEE.

Fu, X., Graham, B., Bettati, R., Zhao, W., and Xuan, D. (2003b). Analytical and empirical analysis of countermeasures to traffic analysis attacks. In *Proceedings of the International Conference on Parallel Processing*, pages 483–492. IEEE.

Fu, Y., Wang, S., Wang, C.-X., Hong, X., and McLaughlin, S. (2018). Artificial intelligence to manage network traffic of 5G wireless networks. *IEEE Network*, 32(6):58–64.

Ge, X. (2019). Ultra-reliable low-latency communications in autonomous vehicular networks. *IEEE Transactions on Vehicular Technology*, 68(5):5005–5016.

Goodell, J. and Kolodner, J. (2022). *Learning Engineering Toolkit: Evidence-Based Practices from the Learning Sciences, Instructional Design, and Beyond*. Taylor & Francis, New York, 1 edition.

Grajzer, M., Koziuk, M., Szczechowiak, P., and Pescapé, A. (2012). A multi-classification approach for the detection and identification of eHealth applications. In *Proceedings of the Computer Communication Networks*, pages 1–6. IEEE.

Grajzer, M. and Szczechowiak, P. (2012). Ehealth traffic detection and classification using machine learning techniques. In *Proceedings of the International Conference on eHealth, Telemedicine, and Social Medicine*, pages 151–154. IARIA XPS.

Hafeez, I., Antikainen, M., Ding, A. Y., and Tarkoma, S. (2020). IoT-KEEPER: Detecting malicious IoT network activity using online traffic analysis at the edge. *IEEE Transactions on Network and Service Management*, 17(1):45–59.

Hafeez, I., Antikainen, M., and Tarkoma, S. (2019). Protecting IoT-environments against traffic analysis attacks with traffic morphing. In *Proceedings of the International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 196–201. IEEE.

Hahn, D., Apthorpe, N., and Feamster, N. (2018). Detecting compressed cleartext traffic from consumer internet of things devices. *arXiv*.

Hameed, A. and Leivadeas, A. (2020). IoT traffic multi-classification using network and statistical features in a smart environment. In *Proceedings of the International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 1–7.

He, D., Ye, R., Chan, S., Guizani, M., and Xu, Y. (2018). Privacy in the internet of things for smart healthcare. *IEEE Communication Magazine*, 56(4):38–44.

He, J., Xiao, Q., He, P., and Pathan, M. S. (2017). An adaptive privacy protection method for smart home environments using supervised learning. *Future Internet*, 9(1):7.

He, J., Xiao, Q., and Pathan, M. S. (2016). A method for countering snooping-based side channel attacks in smart home applications. In *Proceedings of the International Conference on Communications and Networking in China*, pages 200–207. Springer.

Health Organization, W. (2013). *Global action plan for the prevention and control of noncommunicable diseases 2013-2020*. World Health Organization.

Hoang, D. H. and Nguyen, H. D. (2018). A PCA-based method for IoT network traffic anomaly detection. In *Proceedings of the International Conference on Advanced Communication Technology (ICACT)*, pages 381–386. IEEE.

Homma, S., Nishihara, H., Miyasaka, T., Galis, A., OV, V. R., Lopez, D., Contreras-Murillo, L., Martinez-Julia, P., Qiang, L., Rokui, R., Ciavaglia, L., and de Foy, X. (2019). Network slice provision models (internet-draft). Technical report, Internet Engineering Task Force (IETF).

Hou, R., Kong, Y., Cai, B., and Liu, H. (2020). Unstructured big data analysis algorithm and simulation of internet of things based on machine learning. *Neural Computing and Applications*, 32(10):5399–5407.

Huang, D. Y., Apthorpe, N., Li, F., Acar, G., and Feamster, N. (2020). IoT inspector: Crowdsourcing labeled network traffic from smart home devices at scale. *ACM Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(2).

Husain, S., Kunz, A., Prasad, A., Samdanis, K., and Song, J. (2018). Mobile edge computing with network resource slicing for internet-of-things. In *Proceedings of the World Forum on Internet of Things (WF-IoT)*, pages 1–6. IEEE.

Hussain, F., Abbas, S. G., Shah, G. A., Pires, I. M., Fayyaz, U. U., Shahzad, F., Garcia, N. M., and Zdravevski, E. (2021). A framework for malicious traffic detection in IoT healthcare environment. *Sensors*, 21(9):3025.

Hussain, F., Hussain, R., Hassan, S. A., and Hossain, E. (2020). Machine learning in IoT security: Current solutions and future challenges. *IEEE Communications Surveys and Tutorials*, 22(3):1686–1721.

Iana (2019). Internet assigned numbers authority. Available: `https://www.iana.org/`. Accessed February, 2022.

Islam, S. R., Kwak, D., Kabir, M. H., Hossain, M., and Kwak, K.-S. (2015). The internet of things for health care: a comprehensive survey. *IEEE Access*, 3(1):678–708.

ITU-T (2012). Framework of network virtualization for future networks, next generation network–future networks. In *Proceedings of the International Telecommunication Union*. ITU.

Jack, G. (2021). Library for generating MAC addresses. Available: `https://pypi.org/project/python-generate-mac/`. Accessed February, 2022.

Jain, R. (1990). *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. John Wiley and Sons, Digital Equipment Corporation Littleton.

Jenefa, A. and Moses, M. B. (2017). Multi level statistical classification of network traffic. In *Proceedings of the International Conference on Inventive Computing and Informatics (ICICI)*, pages 564–569. IEEE.

Kalør, A. E., Guillaume, R., Nielsen, J. J., Mueller, A., and Popovski, P. (2018). Network slicing in industry 4.0 applications: Abstraction methods and end-to-end analysis. *IEEE Transactions on Industrial Informatics*, 14(12):5419–5427.

Karimi, A., Pedersen, K. I., Mahmood, N. H., Pocovi, G., and Mogensen, P. (2019). Efficient low complexity packet scheduling algorithm for mixed URLLC and eMBB traffic in 5G. In *Proceeding of the Vehicular Technology Conference (VTC)*, pages 1–6. IEEE.

Kassambara, A. and Mundt, F. (2021). *Factoextra: Extract and Visualize the Results of Multivariate Data Analyses*. R package version 1.0.7.

Khalid, S., Khalil, T., and Nasreen, S. (2014). A survey of feature selection and feature extraction techniques in machine learning. In *Proceedings of the Science and Information conference*, pages 372–378. IEEE.

Khan, L. U., Yaqoob, I., Tran, N. H., Han, Z., and Hong, C. S. (2020). Network slicing: Recent advances, taxonomy, requirements, and open research challenges. *IEEE Access*, 8:36009–36028.

Khan, M. Z., Alhazmi, O. H., Javed, M. A., Ghandorh, H., and Aloufi, K. S. (2021). Reliable internet of things: Challenges and future trends. *Electronics*, 10(19):2377.

Kim, H., Claffy, K. C., Fomenkov, M., Barman, D., Faloutsos, M., and Lee, K. (2008). Internet traffic classification demystified: myths, caveats, and the best practices. In *Proceedings of the Conference on Emerging Network Experiment and Technology (CoNEXT)*, CoNEXT '08, page 11. ACM.

Kizza, J. M., Kizza, and Wheeler (2013). *Guide to computer network security*, volume 8. Springer.

Kreutz, D., Ramos, F., Verissimo, P., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76.

Kumar, R., Swarnkar, M., Singal, G., and Kumar, N. (2022). IoT network traffic classification using machine learning algorithms: An experimental analysis. *IEEE Internet of Things Journal*, 9(2):989–1008.

Kurose, J. and Ross, K. (2014). *Redes de Computadores e a Internet: uma abordagem top-down*. Pearson Education do Brasil Ltda, São Paulo, Brasil, 6 edition.

Kurtz, F., Bektas, C., Dorsch, N., and Wietfeld, C. (2018). Network slicing for critical communications in shared 5G infrastructures-an empirical evaluation. In *Proceedings of the Conference on Network Softwarization and Workshops (NetSoft)*, pages 393–399. IEEE.

Lê, S., Josse, J., and Husson, F. (2008). FactoMineR: A package for multivariate analysis. *Journal of Statistical Software*, 25(1):1–18.

Lee, S., Levanti, K., and Kim, H. S. (2014). Network monitoring: Present and future. *Computer Networks*, 65(1):84–98.

Leinwand, A. and Fang, K. (1993). *Network management: a practical perspective*. Addison-Wesley Longman Publishing Co., Inc.

Levashenko, V., Zaitseva, E., Kvassay, M., and Deserno, T. M. (2016). Reliability estimation of healthcare systems using fuzzy decision trees. In *Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 331–340. IEEE.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., and Liu, H. (2017a). Feature selection: A data perspective. *ACM Computing Surveys*, 50(6):1–45.

Li, L., Li, S., and Zhao, S. (2014). QoS-aware scheduling of services-oriented internet of things. *IEEE Transactions on Industrial Informatics*, 10(2):1497–1505.

Li, X., Samaka, M., Chan, H. A., Bhamare, D., Gupta, L., Guo, C., and Jain, R. (2017b). Network slicing for 5G: challenges and opportunities. *IEEE Internet Computing*, 21(5):20–27.

Liu, J., Zhang, C., and Fang, Y. (2018). EPIC: a differential privacy framework to defend smart homes against internet traffic analysis. *IEEE Internet of Things Journal*, 5(2):1206–1217.

Liu, Y., Zhang, S., Ding, B., Li, X., and Wang, Y. (2018). A cascade forest approach to application classification of mobile traces. In *Proceedings of the Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE.

Lloret, J., Parra, L., Taha, M., and Tomás, J. (2017). An architecture and protocol for smart continuous ehealth monitoring using 5G. *Computer Networks*, 129:340–351.

Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A., and Lloret, J. (2017). Network traffic classifier with convolutional and recurrent neural networks for internet of things. *IEEE Access*, 5:18042–18050.

Manousis, A., Sharma, R. A., Sekar, V., and Sherry, J. (2020). Contention-aware performance prediction for virtualized network functions. In *Proceedings of the Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '20, pages 270–282. ACM.

Marquez, C., Gramaglia, M., Fiore, M., Banchs, A., and Costa-Pérez, X. (2019). Resource sharing efficiency in network slicing. *IEEE Transactions on Network and Service Management (TNSM)*, 16(3):909–923.

Maxwell, P., Alhajjar, E., and Bastian, N. D. (2019). Intelligent feature engineering for cybersecurity. In *Proceedings of the International Conference on Big Data (Big Data)*, pages 5005–5011. IEEE.

Maxwell, P., Niblick, D., and Ruiz, D. C. (2021). Using side channel information and artificial intelligence for malware detection. In *Proceedings of the International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, pages 408–413. IEEE.

Mehta, P. and Shah, R. (2017). A survey of network based traffic classification methods. *Database Systems Journal*, 7(4):24–31.

Meidan, Y., Bohadana, M., Shabtai, A., Guarnizo, J. D., Ochoa, M., Tippenhauer, N. O., and Elovici, Y. (2017). ProfilIoT: A machine learning approach for iot device identification based on network traffic analysis. In *Proceedings of the Symposium on Applied Computing*, page 506–509. ACM.

Mijumbi, R., Serrat, J., Gorricho, J.-L., Bouten, N., De Turck, F., and Boutaba, R. (2015). Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys and Tutorials*, 18(1):236–262.

More, A. (2019). Smart healthcare market 2019 global industry analysis by key players, share, revenue,trends, organizations size, growth, opportunities, and regional forecast to 2022. Available: `http://www.theexpresswire.com/pressrelease/Smart-Healthcare-Market-Research-2019-Business-Opportunity-Global-Trend-Future-Growth-Key-Findings-and-Forecast-to-2022_10229596`. Accessed February, 2022.

Movassaghi, S., Abolhasan, M., Lipman, J., Smith, D., and Jamalipour, A. (2014). Wireless body area networks: A survey. *IEEE Communications Surveys and Tutorials*, (3):1658–1686.

Neshenko, N., Bou-Harb, E., Crichigno, J., Kaddoum, G., and Ghani, N. (2019). Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on internet-scale IoT exploitations. *IEEE Communications Surveys and Tutorials*, 21(3):2702–2733.

Newaz, A. I., Sikder, A. K., Rahman, M. A., and Uluagac, A. S. (2021). A survey on security and privacy issues in modern healthcare systems: Attacks and defenses. *ACM Transactions on Computing for Healthcare (HEALTH)*, 2(3):1–44.

Nguyen, T. T. and Armitage, G. (2008). A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys and Tutorials*, 10(4):56–76.

Ni, J., Lin, X., and Shen, X. S. (2018). Efficient and secure service-oriented authentication supporting network slicing for 5G-Enabled IoT. *IEEE Journal on Selected Areas in Communications*, 36(3):644–657.

NIST, C. (2018). Framework for improving critical infrastructure cybersecurity version 1.1. Technical report, National Institute of Standards and Technology.

Nogueira, M. (2009). *SAMNAR: A survivable architecture for wireless self-organizing networks*. PhD thesis, PhD thesis, Université Pierre et Marie Curie-LIP6.

OpenTeleHealth (2019). Opentelehealth: Remote patient monitoring. Available: `http://opentelehealth.com/`. Accessed February, 2022.

Ordonez-Lucena, J., Ameigeiras, P., Lopez, D., Ramos-Munoz, J. J., Lorca, J., and Folgueira, J. (2017). Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges. *IEEE Communications Magazine*, 55(5):80–87.

Oster, C. A. and Deakins, S. (2018). Practical application of high-reliability principles in healthcare to optimize quality and safety outcomes. *Journal of Nursing Administration*, 48(1):50–55.

Pacheco, F., Exposito, E., Gineste, M., Baudoin, C., and Aguilar, J. (2018). Towards the deployment of machine learning solutions in network traffic classification: A systematic survey. *IEEE Communication Surveys and Tutorials*, 21(2):1988–2014.

Paglialonga, A., Riboldi, M., Tognola, G., and Caiani, E. G. (2017). Automated identification of health apps' medical specialties and promoters from the store webpages. In *Proceedings of the E-Health and Bioengineering Conference (EHB)*, pages 197–200. IEEE.

Panda, S. and Panda, G. (2020). Intelligent classification of IoT traffic in healthcare using machine learning techniques. In *Proceedings of the International Conference on Control, Automation and Robotics (ICCAR)*, pages 581–585. IEEE.

Pandas (2021). *Pandas*. Python3 package version 1.2.2. Accessed February, 2022.

Papadogiannaki, E. and Ioannidis, S. (2021). A survey on encrypted network traffic analysis applications, techniques, and countermeasures. *ACM Computing Surveys (CSUR)*, 54(6):1–35.

Pekar, A., Mocnej, J., Seah, W. K. G., and Zolotova, I. (2020). Application domain-based overview of IoT network traffic characteristics. *ACM Computing Surveys*, 53(4).

Peng, L., Yang, B., Chen, Y., and Chen, Z. (2016). Effectiveness of statistical features for early stage internet traffic identification. *International Journal of Parallel Programming*, 44(1):181–197.

Petrov, V., Lema, M. A., Gapeyenko, M., Antonakoglou, K., Moltchanov, D., Sardis, F., Samuylov, A., Andreev, S., Koucheryavy, Y., and Dohler, M. (2018). Achieving end-to-end reliability of mission-critical traffic in softwarized 5G networks. *IEEE Journal on Selected Areas in Communications*, 36(3):485–501.

Pinheiro, A. J., de Araujo-Filho, P. F., Bezerra, J. d. M., and Campelo, D. R. (2020). Adaptive packet padding approach for smart home networks: A tradeoff between privacy and performance. *Proceedings of the Internet of Things Journal*, 8(5):3930–3938.

Prates, N., Vergütz, A., Macedo, R., and Lima, M. N. (2019a). Análise de vazamentos temporais side-channel no contexto da internet das coisas. In *Anais do Workshop de Gerência e Operação de Redes e Serviços (WGRS) no Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pages 157–170. SBC.

Prates, N., Vergütz, A., Macedo, R., and Lima, M. N. (2019b). Um mecanismo de defesa contra ataques traffic side-channel temporais na IoT. In *Anais do Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg)*, pages 1–8. SBC.

Prates, N., Vergütz, A., T. Macedo, R., Santos, A., and Nogueira, M. (2020). A defense mechanism for timing-based side-channel attacks on IoT traffic. In *Proceedings of the Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE.

Prithi., S., Sumathi., S., and C.Amuthavalli (2017). A survey on intrusion detection system using deep packet inspection for regular expression matching. *International Journal of Electronics, Electrical and Computer Systems (IJEECS)*, 6(01).

Python (2021). *Python documentation toolset*. Accessed February, 2022.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rashelbach, A., Rottenstreich, O., and Silberstein, M. (2020). A computational approach to packet classification. In *Proceedings of the Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '20, page 542–556. ACM.

Ren, J., Dubois, D. J., Choffnes, D., Mandalari, A. M., Kolcun, R., and Haddadi, H. (2019). Information exposure from consumer IoT devices: A multidimensional, network-informed measurement approach. In *Proceedings of the Internet Measurement Conference*, IMC '19, pages 267–279. ACM.

Richart, M., Baliosian, J., Serrat, J., and Gorricho, J.-L. (2016). Resource slicing in virtual wireless networks: A survey. *IEEE Transactions on Network and Service Management (TNSM)*, 13(3):462–476.

Richart, M., Baliosian, J., Serrati, J., Gorricho, J.-L., Agüero, R., and Agoulmine, N. (2017). Resource allocation for network slicing in WiFi access points. In *Proceedings of the International Conference on Network and Service Management (CNSM)*, pages 1–4. IEEE.

Rost, P., Mannweiler, C., Michalopoulos, D. S., Sartori, C., Sciancalepore, V., Sastry, N., Holland, O., Tayade, S., Han, B., Bega, D., et al. (2017). Network slicing to enable scalability and flexibility in 5G mobile networks. *IEEE Communications Magazine*, 55(5):72–79.

Saha, D. (2020). How the world became data-driven, and what's next. Available: `https://www.forbes.com/sites/googlecloud/2020/05/20/how-the-world-became-data-driven-and-whats-next/?sh=4b7508b657fc`. Accessed February, 2022.

Salhab, N., Rahim, R., Langar, R., and Boutaba, R. (2019). Machine learning based resource orchestration for 5G network slices. In *Proceedings of the Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE.

Salvat, J. X., Zanzi, L., Garcia-Saavedra, A., Sciancalepore, V., and Costa-Perez, X. (2018). Overbooking network slices through yield-driven end-to-end orchestration. In *Proceedings of the International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, pages 353–365. ACM.

Sarker, I. H. (2021). Data science and analytics: An overview from data-driven smart computing, decision-making and applications perspective. *Springer Nature Computer Science*, 2(5):377.

Sathi, V. N., Srinivasan, M., Thiruvasagam, P. K., and Chebiyyam, S. R. M. (2018). A novel protocol for securing network slice component association and slice isolation in 5G networks. In *Proceedings of the International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pages 249–253. ACM.

Sciancalepore, V., Cirillo, F., and Costa-Perez, X. (2017a). Slice as a service (SlaaS) optimal IoT slice resources orchestration. In *Proceedings of the Global Communications Conference (GLOBECOM)*, pages 1–7. IEEE.

Sciancalepore, V., Samdanis, K., Costa-Perez, X., Bega, D., Gramaglia, M., and Banchs, A. (2017b). Mobile traffic forecasting for maximizing 5G network slicing resource utilization. In *Proceedings of the International Conference on Computer Communications (INFOCOM)*, pages 1–9. IEEE.

Shen, M., Liu, Y., Zhu, L., Xu, K., Du, X., and Guizani, N. (2020). Optimizing feature selection for efficient encrypted traffic classification: A systematic approach. *IEEE Network*, 34(4):20–27.

Shen, M., Wei, M., Zhu, L., and Wang, M. (2017). Classification of encrypted traffic with second-order markov chains and application attribute bigrams. *IEEE Transactions on Information Forensics and Security*, 12(8):1830–1843.

Sherry, J., Lan, C., Popa, R. A., and Ratnasamy, S. (2015). Blindbox: Deep packet inspection over encrypted traffic. *ACM SIGCOMM Computer Communication Review*, 45(4):213–226.

Shlens, J. (2014). A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*.

Silva, R., Achir, N., Santos, A., and Nogueira, M. (2016). Avoiding collisions by time slot reduction supporting voice and video in 802.11 networks. In *Proceedings of the Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE.

Sivanathan, A., Gharakheili, H. H., Loi, F., Radford, A., Wijenayake, C., Vishwanath, A., and Sivaraman, V. (2018). Classifying IoT devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing*, 18(8):1745–1759.

Sivanathan, A., Gharakheili, H. H., Loi, F., Radford, A., Wijenayake, C., Vishwanath, A., and Sivaraman, V. (2021). Iot traffic traces. Available: `https://iotanalytics.unsw.edu.au/iottraces`. Accessed February, 2022.

Skowron, M., Janicki, A., and Mazurczyk, W. (2020). Traffic fingerprinting attacks on internet of things using machine learning. *IEEE Access*, 8:20386–20400.

Smith, L. I. (2002). A tutorial on principal components analysis. Technical report, Department of Computer Science, University of Otago, New Zealand.

Spreitzer, R., Moonsamy, V., Korak, T., and Mangard, S. (2018). Systematic classification of side-channel attacks: A case study for mobile devices. *IEEE Communications Surveys and Tutorials*, 20(1):465–488.

Srinivasan, V., Stankovic, J., and Whitehouse, K. (2008). Protecting your daily in-home activity information from a wireless snooping attack. In *Proceedings of the International Conference on Ubiquitous Computing (Ubicomp)*, pages 202–211. ACM.

Standaert, F.-X. (2010). Introduction to side-channel attacks. In *Secure integrated circuits and systems*, pages 27–42. Springer.

Statista (2016). Internet of things (IoT) connected devices installed base worldwide from 2015 to 2025 (in billions). Available: `https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/`. Accessed February, 2022.

Stoyanova, M., Nikoloudakis, Y., Panagiotakis, S., Pallis, E., and Markakis, E. K. (2020). A survey on the internet of things (IoT) forensics: challenges, approaches, and open issues. *IEEE Communications Surveys and Tutorials*, 22(2):1191–1221.

Tanenbaum, A. S. and Van Steen, M. (2002). *Distributed systems: principles and paradigms*. Prentice-Hall, Upper Saddle River, New Jersey 07458.

Taylor, V. F., Spolaor, R., Conti, M., and Martinovic, I. (2016). Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic. In *Proceedings of the European Symposium on Security and Privacy (EuroSandP)*, pages 439–454. IEEE.

Taylor, V. F., Spolaor, R., Conti, M., and Martinovic, I. (2018). Robust smartphone app identification via encrypted network traffic analysis. *IEEE Transactions on Information Forensics and Security*, 13(1):63–78.

Team, L. (2006). L7-filter classifier - application layer packet classifier for linux. Available: `http://l7-filter.sourceforge.net/`. Accessed February, 2022.

Team, W. (2021). *Tshark - Dump and analyze network traffic*. TShark (Wireshark) version 3.2.3.

Uddin, M. and Nadeem, T. (2016). TrafficVision: A case for pushing software defined networks to wireless edges. In *Proceedings of the International Conference on Mobile Ad hoc and Sensor Systems (MASS)*, pages 37–46. IEEE.

Ullah, I., Mengersen, K., Hyndman, R. J., and McGree, J. (2021). Detection of cybersecurity attacks through analysis of web browsing activities using principal component analysis. *arXiv preprint arXiv:2107.12592*.

Valenti, S., Rossi, D., Dainotti, A., Pescapè, A., Finamore, A., and Mellia, M. (2013). Reviewing traffic classification. In *Data Traffic Monitoring and Analysis*, pages 123–147. Springer.

Vergütz, A., da Silva, R., Nacif, J. A. M., Vieira, A. B., and Nogueira, M. (2017). Mapping critical illness early signs to priority alert transmission on wireless networks. In *Proceedings of the Latin-American Conference on Communications (LATINCOM)*, pages 1–6. IEEE.

Vergütz, A., Medeiros, I., Rosário, D., Cerqueira, E., Santos, A., and Nogueira, M. (2019). A method for identifying ehealth applications using side-channel information. In *Proceedings of the Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE.

Vergutz, A., Noubir, G., and Nogueira, M. (2020). Reliability for smart healthcare: A network slicing perspective. *IEEE Network*, 34(4):91–97.

Verma, S., Jain, K., and Prakash, C. (2020). An unstructured to structured data conversion using machine learning algorithm in internet of things (IoT). In *Proceedings of the International Conference on Innovative Computing & Communications (ICICC)*.

Vidya, C. and Vijaya Kumar, B. (2016). Reliability analysis in healthcare images applications. *Indian Journal of Science and Technology*, 9(34):181–197.

Vo-Huu, T. D., Vo-Huu, T. D., and Noubir, G. (2016). Fingerprinting Wi-Fi devices using software defined radios. In *Proceedings of the Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, pages 3–14. ACM.

Vu, L., Thuy, H. V., Nguyen, Q. U., Ngoc, T. N., Nguyen, D. N., Hoang, D. T., and Dutkiewicz, E. (2018). Time series analysis for encrypted traffic classification: A deep learning approach. In *Proceedings of the Symposium on Communications and Information Technologies (ISCIT)*, pages 121–126. IEEE.

Wang, K., Shao, Y., Xie, L., Wu, J., and Guo, S. (2018a). Adaptive and fault-tolerant data processing in healthcare IoT based on fog computing. *IEEE Transactions on Network Science and Engineering*, 7(1):263–273.

Wang, Q., Alcaraz-Calero, J., Weiss, M. B., Gavras, A., Neves, P. M., Cale, R., Bernini, G., Carrozzo, G., Ciulli, N., Celozzi, G., et al. (2018b). SliceNet: end-to-end cognitive network slicing and slice management framework in virtualised multi-domain, multi-tenant 5G networks. In *Proceedings of the International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pages 1–5. IEEE.

Westin, A. F. (1968). Privacy and freedom. *Washington and Lee Law Review*, 25(1):166.

WHO (2016). Global diffusion of ehealth: Making universal health coverage achievable. Technical report, Report of the third global survey on eHealth, World Health Organization (WHO).

Widaman, K. F. (1993). Common factor analysis versus principal component analysis: Differential bias in representing model parameters? *Multivariate behavioral research*, 28(3):263–311.

Wood, D., Apthorpe, N., and Feamster, N. (2017). Cleartext data transmissions in consumer IoT medical devices. In *Proceedings of the Workshop on Internet Things Security Privacy (IoT SandP)*, pages 7–12. ACM.

Wright, C. V., Coull, S. E., and Monrose, F. (2009). Traffic morphing: An efficient defense against statistical traffic analysis. In *NDSS*, volume 9. Citeseer.

Yang, G., Yu, B.-y., Jin, H., and Yoo, C. (2020). Libera for programmable network virtualization. *IEEE Communication Magazine*, 58(4):38–44.

Yao, H., Ranjan, G., Tongaonkar, A., Liao, Y., and Mao, Z. M. (2015). Samples: Self adaptive mining of persistent lexical snippets for classifying mobile application traffic. In *Proceedings of the International Conference on Mobile Computer Networks (MobiCom)*, pages 439–451. ACM.

Yao, Y., Basdeo, J. R., Kaushik, S., and Wang, Y. (2019). Defending my castle: A co-design study of privacy mechanisms for smart homes. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI'19, page 1–12. ACM.

Yilmaz, O. N., Wang, Y.-P. E., Johansson, N. A., Brahmi, N., Ashraf, S. A., and Sachs, J. (2015). Analysis of ultra-reliable and low-latency 5G communication for a factory automation use case. In *Proceedings of the International Conference on Communication Workshop (ICCW)*, pages 1190–1195. IEEE.

Yu, K., Li, Q., Chen, D., Rahman, M., and Wang, S. (2021). PrivacyGuard: Enhancing smart home user privacy. In *Proceedings of the International Conference on Information Processing in Sensor Networks*, page 62–76. ACM.

Zaitseva, E. (2010). Reliability analysis methods for healthcare system. In *Proceedings of the International Conference on Human System Interaction*, pages 211–216. IEEE.

Zaitseva, E., Levashenko, V., Kvassay, M., and Barach, P. (2017). Healthcare system reliability analysis addressing uncertain and ambiguous data. In *Proceedings of the International Conference on Information and Digital Technologies (IDT)*, pages 442–451. IEEE.

Zhang, Y. and Årvidsson, A. (2012). Understanding the characteristics of cellular data traffic. In *Proceedings of the SIGCOMM Workshop on Cellular networks: operations, challenges, and future design*, pages 13–18. ACM.