An Adaptive Threshold to Identify Elephants and Mice Flows in SDN

Dissertation of Master in Telematics Engineering

Eng. Alejandra Duque Torres

Advisor: Oscar Mauricio Caicedo Rendón, Ph.D.
Co-Advisor: Wilmar Yesid Campo, Ph.D.

University of Cauca
Faculty of Electronics and Telecommunications Engineering
Telematics Department
Line of Research in Advanced Services of Telecommunications
Popayán, Cauca, 2019

An Adaptive Threshold to Identify Elephants and Mice Flows in SDN

Eng. Alejandra Duque Torres

Dissertation submitted to the Faculty of Electronic
and Telecommunications Engineering of the
University of Cauca for the Degree of

Master in:
Telematics Engineering

Advisor: Oscar Mauricio Caicedo Rendón, Ph.D.
Co-Advisor: Wilmar Yesid Campo, Ph.D.

University of Cauca
Faculty of Electronics and Telecommunications Engineering
University of Cauca
Popayán, Cauca, 2019

Abstract

The volume and heterogeneity of computer network traffic are exponentially increasing. In consequence, managing the traffic that flows through the network is a challenge. Over recent years, an essential tool used in network traffic management is flow classification. In traffic flow classification a significant objective is to identify and classify flows that exhibit heavy-tailed or long-tailed distribution. An inferable observation from heavy-tailed distribution is that a very small percentage of flows carry the bulk of the traffic (in bytes). These flows are most commonly referred to as Heavy-Hitters (HHs).

One of the consequences of unsupervised (uncontrolled) forwarding of HHs is that it often leads to network congestion and, subsequently, to overall network performance degradation. The main motivation for HHs identification includes flow scheduling, QoS provisioning, and load balancing, especially applied to Data Centre Networks (DCNs). Therefore, the identification and classification of HHs remain to attract interest.

Most of the existing approaches to identify HHs are based on thresholds, *i.e.,* if the flow exceeds a predefined threshold, it will be marked as a HH; otherwise, it will be classified as a non-HH. However, these approaches present two significant issues. First, there is no consistent and accepted threshold that would reliably classify flows. Second, they use counters (duration, packets, and bytes), which their accuracy depends on how complete the flow information is. Thus, the goal of this master dissertation is to investigate the feasibility on using per-flow packet size distribution as an effective, in terms of Precision, Recall, F-measure, and ROC curve, approach for identifying HHs.

To achieve the raised goal, this master dissertation introduces a novel HHs identification approach based on per-flow Packet Size Distribution (PSD) and the Template Matching (TM). An extensive analysis of the approach was conducted on different dataset from DCNs. The results of the analysis have provided directions and evidence that corroborate the feasibility of using per-flow PSD and TM as an effective approach for identifying HHs. The approach proposed achieves up to 96% accuracy while using only the first 14 packets of a flow. Furthermore, this accuracy remains consistent throughout all classifications while existing approaches yield different accuracies for different flow size-based threshold

Keywords: Heavy-Hitters, flow, elephant, mice, data centre networks, packet size distribution, threshold, software-defined networking, template matching

Resumen

El volumen y la heterogeneidad del tráfico de la red aumentan exponencialmente. En consecuencia, administrar el tráfico que fluye a través de la red es un desafío. En los últimos años, la clasificación de flujos ha sido una herramienta esencial para la gestión de tráfico de red. Uno de los objetivos más importantes en la clasificación del flujo de tráfico de red, es identificar y clasificar los flujos que exhiben una distribución de cola larga o cola pesada. Una observación común de la distribución de cola pesada es que un porcentaje muy pequeño de flujos transporta la mayor parte del tráfico (en bytes). Estos flujos se conocen comúnmente como Heavy-Hitters (HHs).

Una de las consecuencias del reenvío no supervisado (descontrolado) de HHs, es que pueden dirigir a la congestión de la red y, posteriormente, a la degradación general del rendimiento de la red. La principal motivación para identificar HHs incluye la programación de rutas para flujos, el aprovisionamiento de QoS y el equilibrio de carga, especialmente aplicado a las Redes de Centros de Datos (Data Centre Network, DCN). Por lo tanto, la identificación y clasificación de HHs siguen atrayendo interés.

La mayoría de los enfoques existentes para identificar HH se basan en umbrales, *es decir,* si el flujo excede un umbral predefinido, se marcará como HH; de lo contrario, se clasificará como no HH. Sin embargo, estos enfoques presentan dos limitantes importantes. En primer lugar, no hay un umbral consistente y aceptado que clasifique de manera confiable los flujos. En segundo lugar, los enfoques basados en umbrales hacen uso de contadores (duración, paquetes y bytes), cuya precisión depende de qué tan completa sea la información del flujo. Por lo tanto, el objetivo de esta disertación de maestría es investigar la viabilidad del uso de la distribución del tamaño

de paquete por flujo, como un enfoque eficaz, en términos de precisión, recuperación, medida F y curva ROC, para identificar HH.

Para lograr el objetivo planteado, se presenta un nuevo enfoque de identificación de HHs basado en la Distribución de tamaño de paquetes (PSD) por flujo y la Coincidencia de plantillas (TM). Se realizó un análisis exhaustivo del enfoque en diferentes conjuntos de datos de DCN. Los resultados del análisis han proporcionado instrucciones y evidencia que corroboran la viabilidad de usar PSD y TM por flujo como un enfoque efectivo para identificar HH. El enfoque propuesto logra una precisión de hasta el 96% mientras usa solo los primeros 14 paquetes de un flujo. Además, esta precisión se mantiene constante ante todas las clasificaciones, mientras que los enfoques existentes presentan diferentes precisiones para diferentes umbrales basados en el tamaño del flujo

Palabras Claves: Heavy-Hitters, flujos, elefantes, ratones, redes de centros de datos, distribución de tamaño de paquete, umbral, redes definidas por software, concidencia de plantillas

Contents

List of Figures

LIST OF FIGURES

List of Tables

Acronym Lists

Acronym	English	Spanish
HH	Heavy-Hitter	Peso pesado
DCN	Data Centre Network	Red de Centro de Datos
PSD	Packet Size Distribution	Distribución de Tamaño de Paquete
TM	Template Matching	Coincidencia de Plantilla
SDN	Software-Defained Networking	Redes Definidas por Software
SDDCN	Software-Defined Data Centre Network	Red de Centro de Datos Definido por Software
QoS	Quality of Service	Calidad de Servicio
KDN	Knowledge-Defined Networking	Redes Definidas por Conocimiento
KDD	Knowledge Discovery in Database	Descubrimiento de Conocimiento en Base de Datos
(NBIs	North Bound Interface	Interfaz Limitada al Norte
SBIs	South Bound Interface	Interfaz Limitada al Sur
VM	Virtual Machine	Maquina Virtual
ToR	Top-of-Rack	Parte superior del estante
KP	Knowledge Plane	Plano del Conocimiento
AI	Artificial Intelligence	Inteligencia Artificial
ML	Machine Learning	Aprendizaje de Maquinas
SL	Supervised Learning	Aprendizaje Supervisado
UL	Unsupervised Learning	Aprendizaje No Supervisado
SSL	Semi-Supervised Learning	Aprendizaje Semi–Supervisado
RL	Reinforcement Learning	Aprendizaje por Refuerzo
DAM	Data Acquisition Module	Módulo de Adquisición de Datos
DANM	Data Analyser Module	Modulo de Analisis de Datos
APM	APplication Module	Modulo de Aplicación
KDP	Knowledge Discovery Process	Proceso de Descubrimiento de Conocimiento
DM	Data Mining	Mineria de Datos
PCAP	Packet CAPture	Captura de Paquetes
CSV	Comma-Separated Values	Valores Separados por Comas
pdf	Probability Density Function	Función de Densidad de Probabilidad
kde	Kernel Density Estimation	Estimación de Densidad del Núcleo
DTW	Dynamic Time Warp	Deformación Dinámica del Tiempo
TP	True Positive	Verdadero Positivo
TN	True Negative	Verdadero Negativo
FP	False Positive	Falso Positivo
FN	False Negative	Falso Negativo
TPR	True Positive Rate	Tasa Positiva Verdadera
FPR	False Positive Rate	Tasa Positiva Falsa
NN	Neural Networks	Redes Neuronales
GPR	Gaussian Processes Regression	Regresión de Procesos Gaussianos
oBMM	Online Bayesian Moment Matching	Coincidencia de Momento Bayesiano en Línea

Chapter 1

Introduction

1.1 Problem Statement

In the last few decades, Data Centre Networks (DCN) have become vital components of a wide range of applications, such as big data processing, cloud computing infrastructure, and multimedia content delivery. A traditional DCN comprises a very high number of network devices that support the seamless exchange of traffic between the virtual machines/servers and the Internet. These devices include switches that interconnect hosts, routers that forward the traffic, and gateways that act as a junction between the DCN and the Internet. The main DCN limitations have been scalability and growing management complexity.

To overcome these limitations, new networking paradigms such as Software-Defined Networking (SDN) have been introduced into DCNs. SDN enables network programmability and provides a flexible architecture for managing more efficiently the computer networks [1]. SDN separates the control plane from the data plane, enabling a logically centralised control of network devices [2]. By moving the control logic from the forwarding devices to a logically centralised device, namely, the Controller, the network devices become simple forwarders that are programmable through a standardised protocol such as OpenFlow [3]. Data Centres implemented using SDN are referred to as Software-Defined Data Centre Networks (SDDCNs).

One of the main benefits of SDDCN is the centralised view of the network and, most importantly, its traffic flows. However, the clear and well-defined benefits of central network traffic control cannot guarantee that the network performance will not degrade when traffic volume increases. Applications and services have increasing Quality of Service (QoS) requirements. In consequence, managing the traffic that flows through the network remains a challenge [4]. An essential tool to ensure the reliable operation of networks is traffic flow classification, which provides inputs for a variety of network managerial related activities [5]. In network performance maintenance a significant objective is to identify and classify flows that exhibit heavy-tailed or long-tailed distribution.

Examination of heavy-tailed distribution in flows has been an objective of several research efforts. An inferable observation from these works is that a very small percentage of flows carry the bulk of the traffic [6], [7]. These flows are most commonly referred to as Heavy-Hitters (HHs). One of the consequences of unsupervised forwarding of HHs flows is that it often leads to network congestion and, subsequently, to overall network performance degradation [6], [8].

Most HHs detection approaches are based on a *static* threshold [9]–[11]. The static threshold is usually specified at the start of the detection phase and remains unchanged until the end of the measurement or unless the detector is required to be reconfigured and restarted. Despite static-threshold based techniques have been shown to achieve promising results, it come short in taking into consideration the network traffic dynamics. To reflect the dynamic nature of the network in HHs detection, whenever the threshold needs to be changed, the classification must be restarted. However, network traffic condition changes are in-deterministic and rapid, making manual threshold adjustment impractical.

In addition, the existing threshold–based approaches share two significant issues. First, there is no consistent and accepted threshold that would reliably classify flows. Second, they use counters (duration, packets, and bytes), which their accuracy depends on how complete the flow information is. The identification of HHs provides inputs for a variety of network managerial related activities, such as flow scheduling, QoS provisioning, and load balancing.

Considering that HHs identification can improve the overall network performance in SDN, this dissertation highlights that such an identification process needs to be done in an early stage of the flow, that is, without complete information of the flow. This dissertation argues that the ability to predict which flow will have an impact size in an early stage allows may assist in improving network performance and minimising the overall network degradation. Therefore, this dissertation focuses on addressing the following research question:

How to predict what flow is going to be an HH or non-HH without its completed information?

1.2 Hypothesis

To address the research question stated in Section 1.1, this master dissertation raises the following hypothesis: using per-flow Packet Size Distribution (PSD) allows capturing the behaviour and dynamics of network traffic flow more accurately than the counters used by the threshold-based approach. In addition, the use of networking paradigms as Knowledge-Defined Networking (KDN) which takes advantages of the Artificial Intelligent concept, allows integrating behavioural models that can automatically detect patterns in data. For instance, in a per-flow PSD scenario, the uncovered patterns can be use to predict what flow is going to be HH or non–HH.

1.3 Goals

1.3.1 Main Goal

Introduce a mechanism to identify HHs in an early stage of the flow, that is, without complete information about the flow, aiming to improve network performance.

1.3.2 Specific goals

- Design a mechanism to HHs in SDN.
- Implement a prototype of the proposed mechanism.
- Evaluate the mechanism proposed regarding classification performance metrics (*i.e.*, Precision, recall, F-measure, ROC curve).

1.4 Contributions

The investigation about the feasibility of using per-flow PSD as an effective approach for identifying HHs led to the following major contributions

- A Knowledge-Defined Networking approach to identifying HH.
- A deeper analysis of the challenges for identifying heavy-hitter flows.
- An approach of Knowledge Discovery in Databases (KDD) for analysing the behavior of heavy-hitters flows.
- An heavy-hitter identification system based on per-flow PSD and template matching technique.

The above-mentioned contributions were reported to the scientific community through paper submissions to renowned conferences and journals (see Appendix A).

- **A. Duque-Torres**, F. Amezquita-Suarez, O.M. Caicedo Rendon, A Ordoñez, and W.Y. Campos, "An Approach based on Knowledge-Defined Networking for Identifying Heavy-Hitter Flows in Data Center Networks". Applied Science. 2019, 9, 4808. **PUBLINDEX:** *A2,* **JCR:** *2.17,* **JSR:** *Q2*

- **A. Duque-Torres**, A. Pekar, W.K.G Seah, and O.M.C Rendon, "Heavy-Hitter Flow Identification in Data Centre Networks Using Template Matching." Accepted, presented and to appear in the 44^{th} IEEE Conference on Local . **CORE:** *A*, **H-index:** *18* .

- **A. Duque-Torres**, A. Pekar, W.K.G Seah, and O.M.C Rendon, "Clustering-based analysis for heavy-hitter detection." Accepted and presented in the Asia Pacific Regional Internet Conference on Operational Technologies (APRICOT), Daejon, South Korea.

- **A. Duque-Torres**, A. Pekar, W.K.G Seah, and O.M.C Rendon, "Knowledge Discovery: A shed light on heavy-hitter detection." Submitted to the IEEE Transaction on Knowledge and Data Engineering. **PUBLINDEX:** *A1,* **JCR:** *8.1,* **JSR:** *Q1*

- A. Pekar, **A. Duque-Torres**, W.K.G Seah, and O.M.C Rendon, "Per-Flow Packet Size Distribution for Heavy-Hitter Flow Detection." Submitted to Compute Networks **PUBLINDEX:** *A1,* **JCR:** *7.1,* **JSR:** *Q1*

1.5 Methodology

Figure 1.1 depicts the phases of the scientific research process followed in this dissertation: Problem Statement, Hypothesis Construction, Experimentation, Conclusion, and Publication. In Problem Statement, the research question has been identified and defined. In Hypothesis Construction, the hypothesis and associated fundamental questions have been formulated. Furthermore, in such phase, the conceptual and technological proposals have been defined and carried out. In Experimentation, the hypothesis and evaluation results have been tested and analyzed, respectively. In Conclusion, conclusions and future works have been outlined. Note that Hypothesis Construction has been reffed after Experimentation and Conclusion. In Publication, papers for renowned conferences and journals have been submitted and published. This document was also written during such last phase.

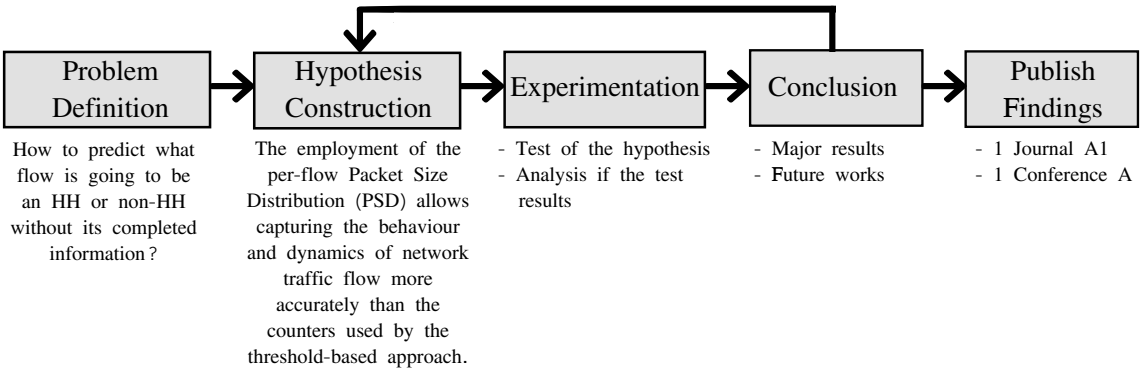


Figure 1.1: Thesis phases

1.6 Document Structure

This document has been divided into chapters described below

- **This introductory chapter** presents the problem statement, raises the hypothesis, exposes the goals, summarises the contributions,and describes the overall structure of this dissertation.
- Chapter 2 presents the **Background** of the main research topics touched in this dissertation. These topics include SDN, KDN, SDDCN, and HHs.
- Chapter 3 presents the **Related Work** that describes the related works closer to this dissertation. In addition, this chapter presents the challenges of the existing approaches for identifying HHs.
- Chapter 4 describes the **HHs identification system based on per-flow PSD**, the experiments conducted to test the system, discusses the corresponding results, and presents implementations highlights.
- Chapter 6 presents **Conclusions** and **Future work**. In this chapter is provided the main conclusion of the presented work and exposes implications.

Chapter 2

Background

The goal of this chapter is to present the background of the main research topics touched in this dissertation. In this way, first, this chapter starts discussing some important HHs aspects as its definition and mainly features. Second, this chapter presents the SDN paradigm and its architecture followed by a brief SDDCN overview. This chapter finished presenting the KDN concept.

2.1 Heavy-Hitter Flow

A flow as a set of packets passing an observation point during a specific time interval [12]. Packets sharing certain attributes belong to the same flow. Usually, such attributes are the source and destination IP addresses, source and destination port numbers, and the protocol identifier. Examination of various phenomena in flows that can be statistically described by heavy-tailed distribution has been an objective of several research activities [10], [13]. A common observation is that a very small percentage of flows carry the bulk of the traffic. These flows are often termed as HHs.

A heavy-tailed distribution assigns relatively high probabilities to regions far from the mean or median. It behaves differently from the distributions commonly used

in performance evaluation (e.g. exponential, normal, Poisson, etc.). For example, if $\alpha \leq 2$, then the distribution has infinite variance; if $\alpha \leq 1$, then the distribution has infinite mean. In consequence, as α decreases, an arbitrarily large portion of probability mass may be present in the tail of the distribution [14]. As such, heavy-tailed distributions give a significant probability of extremely large values being produced.

Since each HH exhibits a uniquely distinguishable phenomenon (e.g. heavy-tailed distribution), various schemes have been proposed for their classification. Usually, HHs can be classified in four different dimensions:

1. *Duration* – time elapsed between the first and last packet of the flow.
2. *Size* – total number of octets (bytes) transmitted.
3. *Rate* – is given by the size divided by the duration.
4. *Burstiness* – A burst is a group of consecutive packets with shorter inter-packet gaps than packets arriving before or after the burst of packets [15]. These packets can be from the same flow or from different flows.

Each flow type, based on its duration, size, rate and burstiness, can be classified into two groups – HH and non-HH. This dichotomy of the flow types is achieved using a threshold which varies depending on the classification scheme used. To emphasise the dichotomy between the characteristics, the flows within the individual classification schemes (categories) are usually termed with a zoological flair as follows [10]:

1. *Duration:* Flows that have a duration longer than a certain period of time d are *tortoises*. Flows with a duration less than or equal to d are *dragonflies*.
2. *Size*: Flows that have a size larger than s B (bytes) are *elephants*. Flows with a size less than or equal to s B are *mice*.
3. *Rate: Cheetahs* are flows with a rate greater than r Bps while *snails* are flows with a rate less than or equal to r Bps.

4. *Burstiness:* Flows with burstiness greater than b B are called *porcupines* while those with burstiness less than or equal to b B are *stingrays*.

Table 2.1 summarises the main characteristics of HH flows as described by Lan *et al.* [10]. In general, tortoise flows do not consume a lot of bandwidth. Elephants flows are long-lived and have a large size, but they are neither fast nor bursty. Cheetahs are small and bursty. The occurrence of porcupine flows is very likely due to the increasing trends in downloading large files over fast links.

Burstiness as a feature for HH classification is rarely used in SDDCN due to its computational complexity. Existing approaches identify HHs using the duration, size, and rate features, which can be all collected through the OpenFlow protocol. In the OpenFlow terminology, duration indicates the elapsed time the flow entry has been installed. Although this does not match exactly the above-mentioned definition of flow duration, it provides the same metric. Byte count represents the number of octets in packets matched by a flow entry and so it serves as the size feature. Both flow duration and byte count are default counters that can be computed on a per-flow basis, this enable to derive the rate using these two flow meters. While the main application of per-flow meters is to rate limit packets sent to the controller, they can be also used to measure the rate of the flow [16].

The HHs threshold-based classification approach and its gaps are later addressed in Chapter 3.

Table 2.1: Taxonomy of heavy-hitter flows as per [10]

Category	**Long-lived (dur.)**	**Large-size**	**Fast (rate)**	**Bursty**
Tortoise	Y	N	N	N
Elephant	Y	Y	N	N
Cheetahs	N	N	Y	Y
Porcupine	N	Y	Y	Y

2.2 Software-Defined Networking

SDN is an architecture flexible, scalable, efficient and adaptable to the changing needs of modern networks. Three elements characterize SDN [17], [18].

- The Control and Data planes are decoupled, *i.e.,* the control functionalities are removed from network devices that will become simple forwarding elements. The Control Plane can be called as the "network brain".
- The control functions are logically centralized, *i.e.,* the control logic is moved to an external entity called Controller. This Controller acts as an intermediary between the applications and the network devices. Furthermore, the Controller provides a global network view and the essential resources to facilitate the forwarding devices programming.
- The implementation of the control function in software, *i.e.,* the network is programmable by software applications running on top of the Controller that interacts with the underlying Data Plane.

The SDN architecture comprises four planes [19]–[21]: Data Plane, Control Plane, Application Plane, and Management Plane. They are described in detail below.

- The *Data Plane* includes the interconnected forwarding devices. These devices are typically composed of programmable forwarding hardware. Furthermore, they have local knowledge of the network, and rely on the Control Plane to populate their forwarding tables and update their configuration.
- The *Control Plane* consists of one or more NorthBound Interfaces (NBIs), the SDN Controller, and one or more SouthBound Interfaces (SBIs). NBIs allow the Control Plane to communicate with the Application Plane, and provide the abstract network view for expressing the network behavior and requirements. The Controller is responsible for programming the forwarding elements via SBIs. SBIs allow the communication between the Control Plane and the Data

Plane, providing: programmatic control of all forwarding operations, capabilities advertisement, statistics reporting; and event notification [17].

- The *Application Plane* is composed of network programs that explicitly, directly, and programmatically communicate their requirements and desired network behavior to the SDN Controller via NBIs.

- The *Management Plane* ensures the accurate network monitoring to provide critical network analytics. For this purpose, Management Plane collects telemetry information from the Data Plane while keeping a historical record of the network state and events [22]. The management plane is orthogonal to the control and data planes and typically operates at larger time-scales [22].

2.3 Software-Defined Data Centre Networks

A typical DCN comprises a conglomeration of network elements that ensures the exchange of traffic between machines/serves and the Internet. These networks elements include servers that manage workloads and respond to different requests, switches that connect devices, routers that perform packet forwarding functions, gateways that serve as the junctions between the DCN and the Internet [23]. Despite the DCNs importance, their architecture are still far from being optimal. Traditionally, DCNs use dedicated servers to run applications, resulting in poor server utilisation and high operational cost. To overcome this situation the emergence of server virtualisation technologies allows multiple virtual machines (VMs) to be allocated on a single physical machine. These technologies can provide performance isolation between collocated VMs to improve application performance and prevent interference attacks. However, server virtualisation itself is insufficient to address all limitations of scalability and managing the growing traffic in DCNs [23], [24].

Motivated by the limitations aforementioned, there is an emerging trend towards the use of networking paradigms as SDN in DCNs, also known as SDDCN. An SDDCN combines virtualised compute, storage, and networking resources with a

standardised platform for managing the entire integrated environment. Following Faizul Bari *et al.* [23], the major foundations of an SDDCN are:

- *Network virtualisation* – combines network resources by splitting the available bandwidth into independent channels that can be assigned or reassigned to a particular server or device in real-time.
- *Storage virtualisation* – pools the physically available storage capacity from multiple network devices. The storage virtualisation is managed from a central console.
- *Server virtualisation* – masks server resources from server users. The intention is to spare users from managing complicated server-resource details. It also increases resource sharing and utilisation while keeping the ability to expand capacity.

Figure 2.1 shows a SDDCN with a conventional topology. In this SDDCN, the Controller (or set of controllers) and the network applications running on it are

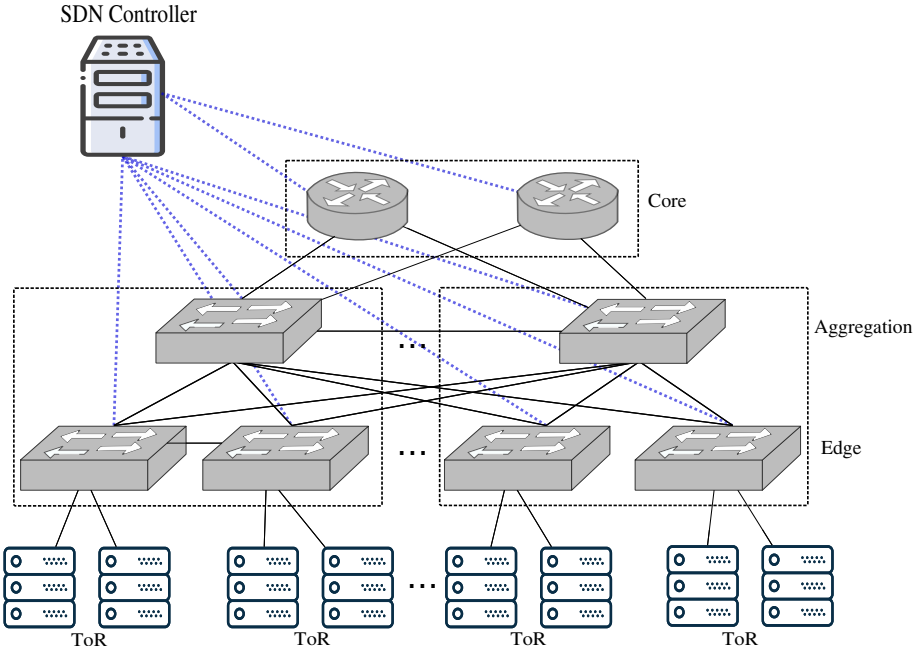


Figure 2.1: SDDCN structure with a conventional topology

responsible for handling the data plane. This plane includes a fat-tree topology that is composed by Top-of-Rack (ToR), Edge, Aggregation, and Core switches. ToR switch in the access layer provides connectivity to the servers mounted on every rack. Each aggregation switch in the aggregation layer (sometimes referred to as the distribution layer) forwards traffic from multiple access layer switches to the core layer. Every ToR switch is connected to multiple aggregation switches for redundancy. The core layer provides secure connectivity between aggregation switches and core routers connected to the Internet [25].

2.4 Knowledge-Defined Networking

2.4.1 Overview

In 2003, D. Clark suggested the addition of a Knowledge Plane (KP) to the traditional computers network architecture formed by the Control Plane and Data Plane. KP adopts the Artificial Intelligence (AI) [26] to perform tasks that are human intelligence characteristic, *i.e.,* systems with the abilities to reason, discover meaning, generalize, or learn from past experiences [27]. To achieve these abilities, KP proposed the use of Machine Learning (ML) techniques. By ML, KP offers advantages to the networking, such as automation processes (recognize-act), recommendation systems (recognize-explain-suggest), and data prediction. These advantages bring the possibility of having a smart network operation and management [22].

Although KP offers a new and a better way to operate, optimize and troubleshoot computer networks, its deployment in traditional networks was constrained because of two main shortcomings. First, KP needs to obtain a rich view and control over the network. In traditional networks, the switches and routers only have a partial view and control. Second, KP is responsible for learning the behavior of the network. To learn the network behavior, the network devices should have high capabilities of storage and computing.

Nowadays, the deficiencies above mentioned may be overcomed because, first, SDN

offers full network view via a logically centralized Controller. Furthermore, SDN improves the network control functions that facilitate the gathering of information about the network state in real-time [22]. Second, the capabilities of network devices have significantly improved, facilitating the gather of information in real-time about packets and flow-granularity [17].

2.4.2 Machine Learning

ML uses computers to simulate human learning and allows computers to identify and acquire knowledge from the real world, and improve performance of some tasks based on this new knowledge [28]. Formally, ML is a set of methods that can automatically detect patterns in data, aiming to use the uncovered patterns to predict future data, and, consequently, to facilitate the decision-making processes [27]. Overall, ML can be divided into Supervised Learning (SL), Unsupervised Learning (UL), Semi-supervised Learning (SSL), and Reinforcement Learning (RL).

SL focuses on modelling the input/output relationships through labelled training datasets. The training data consists of a set of attributes and an objective variable also called class [29]. Typically, SL is used to solve classification and regression problems that pertain to predicting outcomes such as traffic prediction [30], end-to-end path bandwidth prediction [31], and link load prediction [32]. Unlike SL, UL uses unlabeled training datasets to create models that can discriminate patterns in the data. UL can highlight correlations in the data that the Administrator may be unaware of. This kind of learning is most suited for clustering problems. For instance, flow feature-based traffic classification [33], packet loss estimation [34], and resource allocation [35].

SSL occupies the middle ground, between supervised learning (in which all training data are labelled) and unsupervised learning (in which no label data are given) [36]. Interest in SSL has increased in recent years, particularly because of application domains in which unlabeled data are plentiful, such as classification of network data using very few labels [37], network traffic classification [38], and verification networks [39] RL is an iterative process in which an agent aims to discover which actions lead

to an optimal configuration. To discover the actions, the agent is status aware of the environment and takes actions that produce changes of state. For each action, the agent can receive or not receive a reward. The reward depends on how good the action taken was [40]. RL is suited for making cognitive choices, such as decision making, planning, and scheduling. For instance, routing scheme for delay tolerant networks [33], and multicast routing and congestion control mechanisms [41].

2.4.3 KDN Architecture and Operation

The addition of KP to the traditional SDN architecture is called KDN [22]. The KDN architecture comprises four planes: Data Plane, Control Plane, Management Plane and Knowledge Plane.

- *Data plane* is responsible for generating metadata by the forwarding network devices. This metadata is extracted and transmitted to the Management plane.
- *Control plane* provides the interfaces to receive the instructions from the KP; then the Controller transmits the instructions to forwarding devices. Furthermore, the Control plane sends metadata to the Management plane about the network state.
- *Management plane* stores the metadata sent by the Control and Data planes. Furthermore, the Management plane provides a basic analysis of statistics per-flow and per forwarding devices to the KP
- *Knowledge plane* sends to the Controller one or a set of instructions about the what the network is supposed to do.

KDN works by employing a control loop. Formally, a control loop can be described as a system that is used to maintain the desired output, in spite of environmental disturbances. Overall, the components of a control loop include a Data Acquisition Module (DAM), Data Analyser Module (DANM), and APplication Module (APM) [42]. Figure 2.2 overviews the KDN control loop. In a high-abstraction level, the KDN control loop operates as follows [43], [44].

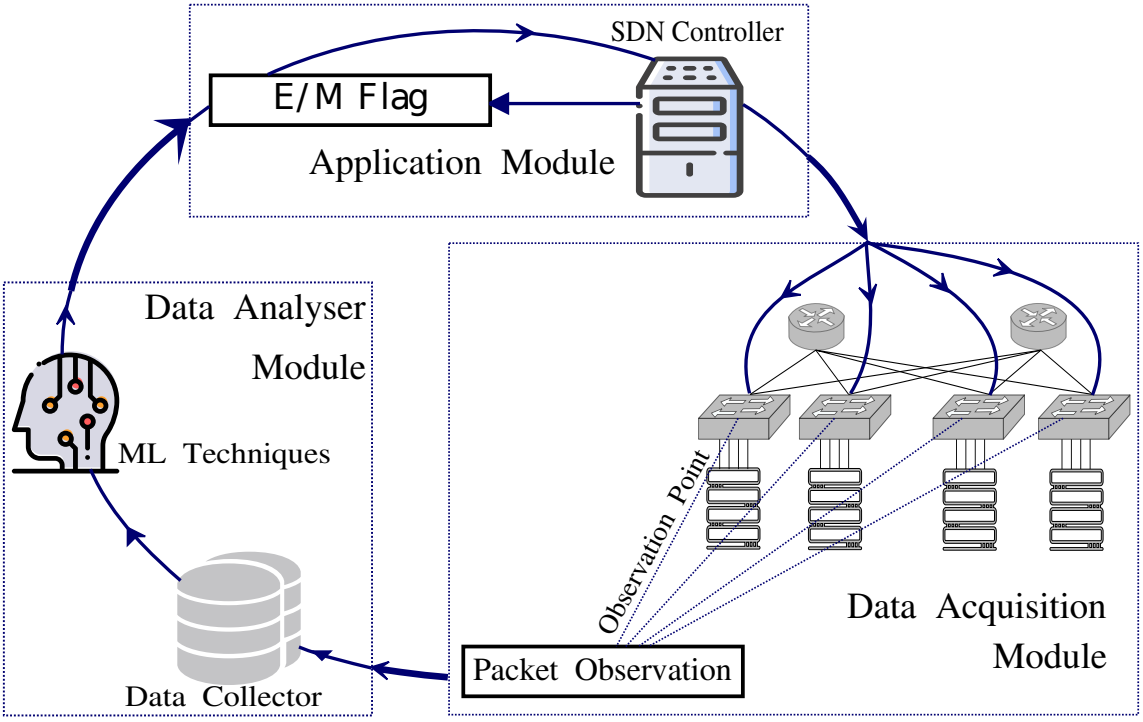


Figure 2.2: High-Level KDN Control Loop

1. *Forwarding Devices → Packet Observation.* *Packet Observation* performs packets capture from *Observation Points* in the network devices, *e.g.,* line card or interfaces of packet forwarding devices. Before starting to send the packets to the Data Collector, the packets can be pre-processed, trough sampling and filtering rules.

2. *Packet Observation → Data Collector.* In the *Data Collector*, the packets provided by DAM are organised and stored into flows. This Collector aims at gathering enough information to offer a global view of the network behaviour.

3. *Data Collector → ML Techniques.* Data Collector feeds the ML Techniques with current and historical data. Thus, the certain application or system can learn from the network behaviour and generate knowledge (*e.g.,* a model of the network).

4. *ML Techniques → Flag → SDN Controller.* The *Flag* eases the transition between the model-generated by the ML Technique sub-module and the control

of specific actions. Based on KDN control loop, this step can be either open or closed. If the Administrator is responsible for deciding on the network, the control loop is open. In this case, APM offers validations and recommendations, which the Administrator can consider when making decisions. For instance, in a control congestion case, the Administrator can query the model (*e.g.,* HHs traffic prediction model, HHs classification model, and link load model) to validate the tentative changes to the configuration before applying them to the network. In the closed control loop, the Administrator is not responsible for deciding on the network. In this case, the model obtained from ML Techniques sub-module can be used to automate tasks, since APM can make decisions automatically on behalf of the Administrator. Furthermore, the model can be used to optimise the existing network configuration [22]. For instance, the model can learn adaptively according to the traffic change, and find the optimal configuration to routing HHs and, thus, avoid congestion

2.5 Final remarks

In this chapter, the fundamental concepts of this dissertation were introduced. This dissertation considers such concepts for proposing an approach (see Chapter 4) that focuses on carrying out HHs identification in an effective way.

Chapter 3

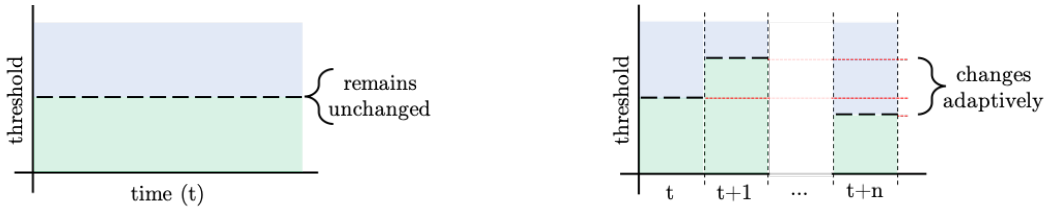
Heavy-Hitters Identification

This chapter provides an extensive study of the existing approach for identifying HHs and their challenges. Next, motivated by those challenges, this chapter presents a final discussion regarding HHs identification. Finally, this chapter finishes introducing the novel HHs identification approach using per–flow PSD and its main concepts.

3.1 Threshold-Based Heavy-Hitters Identification

The identification of HHs has two major processes: *flow measurement*, in which packets are organised into flows, and *flow marking* that assigns the 'HH' or 'non-HH' labels to the individual flows. Label assignment is based on a *threshold*. If a flow exceeds the threshold, it will be marked as a HH. Otherwise, a non-HH label will be assigned to the flow. There are two type of thresholds, static and adaptive threshold (see Figure 3.1). Irregardless of which approach is adopted, static or adaptive, the determination of the threshold is far from being obvious. In practice, several challenges can arise during both processes, the *flow measurement* and the *flow marking*.

The following subsections detail such challenges. In this way, first, It is discusses static and adaptive thresholds. Then, the feature selection and the threshold value



(a) A static threshold separates flows into HH and no–HH during the entire duration of classification

(b) An adaptive threshold is a sequence of different static thresholds that change over time

Figure 3.1: The comparison of static and adaptive thresholds

estimation challenges are detailed. Lastly, the challenges regarding the *flow measurements* are discussed.

3.1.1 Static and Adaptive Thresholds

Most HH detection approaches are based on a *static* threshold [9]–[11]. In the former, the static threshold is usually specified at the start of the detection phase and remains unchanged until the end of the measurement or unless the detector is required to be reconfigured and restarted (see Figure 3.1(a)). Despite static-threshold based techniques have been shown to achieve promising results, it come short in taking into consideration the network traffic dynamics.

Network traffic has a dynamic rather than a static character. Network traffic characteristic can change over time depending on various factors such as the time of the day, reconfigurations, failures or changes in the topology and instrumentation. To reflect the dynamic nature of the network in HHs detection, whenever the threshold needs to be changed, the classification must be restarted. However, network traffic condition changes are indeterministic and rapid, making manual threshold adjustment impractical.

A static threshold can also yield classification errors. Some studies have shown that variations in network traffic cause a number of false positive and false negative errors [45], [46]. Thus, although the static threshold would be suitable for the identification of flows at one point in time (in those specific conditions in which the

threshold was determined, e.g. t), it would be very likely unsuitable at another point in time (such as, $t + 1, \ldots, t + n, \ldots$). In consequence, approaches using a static threshold are considered to bring certain inaccuracy into HHs detection.

In an attempt to improve HHs identification, *adaptive* threshold adjustment mechanisms [45]–[47] have been proposed. Adaptive HHs detection aims to overcome the challenges of static techniques by adjusting the threshold in response to the changes in the network traffic. The changes that trigger threshold adjustment can be, for example, the size of the flow (in bytes) and the link bandwidth utilisation. The idea of adaptive HH detection is clear and well defined, however, it faces a problem, as shown in Figure 3.1(b).

Despite the threshold being adjusted to the changes in traffic behaviour, this change takes place at specific time intervals, for example, every 20 seconds. Between the changes (for 20 seconds) this threshold is still static. If a change occurs at the third second, the adaptive mechanism does not learn about this change for another 17 seconds. This delay can have a critical impact on some services such as DDoS detection that may rely, among others, on HH identification. In conclusion, adaptive threshold adjustment mechanisms can be considered as a continuous sequence of different static thresholds, and they still yield similar challenges as static threshold-based approaches.

3.1.2 Feature Selection and Threshold Estimation

HHs identification requires the knowledge of certain statistics about flows. The majority of existing approaches identify HHs using the duration, size, and rate features, mainly due to the fact that most measurement devices provide these flow statistics by default. Burstiness, on the other hand, is rarely used due to its computational complexity [6]. A common observation from related work is that there is no consensus about the threshold that reliably separates the flows into HHs and non-HHs.

Table 3.1 summarise some of the existing related work. From Table 3.1 it is evident that there is no consensus about what feature or set of features (e.g. size, duration, and rate) should be selected as well as what value(s) to assign to s, d, and r as

Table 3.1: Related work in the domain of SDDCN

Work	Citations	Year	Appeared In	Threshold	Value	Static	Adaptive
Curtis *et al.* [11]	326	2011	IEEE INFOCOM	Flow Size	128KB, 1MB and 100MB	✓	
Sivaraman *et al.* [56]	79	2017	Symposium on SDN Research	Number of packets	Not specified		✓
Chiesa *et al.* [57]	68	2016	IEEE/ACM Trans. Netw.	Bandwidth	10%	✓	
Trestian *et al.* [58]	32	2013	IFIP/IEEE IM	Rate	Not specified	✓	
Liu *et al.* [59]	24	2013	IEEE Commun. Lett.	Rate	1-10 Mbps	✓	
Lin *et al.* [60]	18	2014	IEEE GLOBECOM	Flow Size	100 MB	✓	
Bi *et al.* [45]	11	2013	IEEE GLOBECOM	Flow Size	Not specified		✓
Poupart *et al.* [61]	10	2016	IEEE ICNP Workshop	Flow Size	10 Kb to 1 Mb	✓	
Liu *et al.* [46]	2	2017	Wiley Int. J. Netw. Man.	Flow Size	Not specified		✓

thresholds for accurate HHs identification. In consequence, some techniques rely on size while other approaches use rate or duration as a threshold. Other features such as the number of packets transmitted in the flow are also used as a threshold. In addition, it is not uncommon to see the combination of two or more features. The combination of features is a common technique to increase the classification accuracy. However, if these features (for instance, duration and packet count) are not treated carefully, they can lead to an opposite effect, *i.e.,* a drop in the classification accuracy.

The threshold selection strategy of the advocates of one approach or the other is usually conditioned by the traffic and performance characteristics that are very specific to their networks. However, to date, *there is no generally accepted and widely recognised uniform threshold for HHs detection.* Indeed, different works use different thresholds without a detailed or systematic justification (reinforcement). A trend that is not uncommon to see through several works is that most papers cite previous works that also cited formerly published works, and at the end of the citation chain there is no reasoning for the selected threshold. Instead, they are provided on an 'as is' basis. Examples of such chains include [48], [49]⇒ [50]⇒[51]; [52] ⇒ [53] ⇒[54]; and [2], [55]⇒[6].

3.1.3 Flow Measurements Challenges

3.1.3.1 Effects of Sampling in HH Detection

To avoid network overload and latency, adaptive approaches such as [45], [46], [62] as well as static-threshold based techniques (see Table 3.1) utilise sampling for data collection, in particular, the sFlow protocol [63] which is an industry standard for monitoring high speed switched networks. However, packet sampling has a number of effects on HH identification that the approaches utilising it fail to take into consideration.

First, packet sampling extends the duration required to collect a minimum number of packets to reliably identify HHs. To classify a HH flow, a certain number of packets must be observed before a decision can be made. This number is usually kept at a minimum to achieve real-time classification (e.g. between ten and thirty). However, the packets belonging to various flows pass through the data plane in an indeterministic sequence. As a result, it is not uncommon to see that packets belonging to the same flow are captured with intervals of more than tens of seconds. Therefore, to achieve real-time classification, a time limit for per-flow packet collection is usually implemented. This way the classifier does not have to wait too long to see a specific number of packets before it makes a decision.

sFlow samples packets at the switch port uniformly at a random rate. With a sampling rate of 1/100, sFlow may skip a number of packets that belong to the same flow and the time specified to collect a minimum number of packets for HH classification may need to be extended. On the other hand, if time duration to collect per-flow packets is not extended, a classification may be performed on flows with not enough packets to make an accurate decision.

Second, packet sampling may completely miss some flows. With the advent of IoT, the number of non-HHs rapidly increases [64]. These flows, are typically short in their lifetime and account for a small volume of data, usually transmitted in only a few packets. When sFlow is used, these packets may be overseen (missed). In conclusion, sampling may also negatively affects the classifiers ability to capture

certain flows.

Third, sampling may also cause a change in the proportion of HHs/non-HHs in any given time period in a data centre. Assume that a 10 KB threshold and a sampling rate of 1/100 yield a 10% / 80% proportion of HHs / non-HHs. However, if the flows were identified on a per-packet basis, this proportion can change to 7% / 93%, demonstrating one consequence of sampling that is the number of bytes captured within a flow decreases and affects the distribution of the detected HH/non-HH flows in the data centre.

To quantify the consequence of sampling, the same dataset without sampling and with 1/100 sampling were processed. The results shown in Table 3.2 used the referenced dataset UNIV1 [6]. This dataset contains roughly twenty million packets collected in a university campus data centre network.Only the first fifteen million packets into flows were organised and only TCP and UDP packets were considered. For flow expiration, an idle timeout of 150 seconds was used. Lastly, the threshold

Table 3.2: Flow statistics of the UNIV1 [6] dataset

	Sampling	**OFF**	**ON**
Flows	Total no.	396,581	6,133
	HHs	59,919	706
	non-HHs	336,662	45,427
Packets	Min.	1	1
	Max.	1,188,676	11,899
	Mean	33	3
	Std. dev.	2,278	63
Bytes	Min.	31	29
	Max.	1,128,826,300	11,219,892
	Mean	21,603	1,847
	Std. dev.	2,003,929	57,279
Durat. [s]	Min.	0†	0.00131
	Max.	3,667.72	1,510.2
	Mean	7	6
	Std. dev.	118	22

†Single packet flow is deemed to have duration 0 sec.

used for HH classification was set to 10 KB [61].

Table 3.2 provides statistics for both the total dataset (Flows) as well as per-flow statistics (Packets, Bytes, and Duration). Table 3.2 clearly shows that one consequence of sampling is the loss of a considerable amount of packets. Ideally, these packets should belong equally to all the flows at a given period of time, but in reality, they constitute entire flows. Therefore, sampling brings certain bias into the measurements and so into the classification accuracy of HHs.

3.1.3.2 Complete Flow Information

Accurate HH identification supposed to be relatively simple if the HHs identification system has complete information of flows while is highly complex when performing real-time detection (*i.e.*, at any given time period only part of the flow is known, also often referred to as sub-flow [65]). In reality, however, even if complete flows are known, the selection of the flow feature(s) (see Section 3.1.2) and the determination of the threshold value(s) is still far off.

Using the UNIV1 dataset as a test case the following analysis was performed, first, the dataset was sorted based on their size (i.e., the total number of bytes) and plot them from the largest (left) to the smallest (right) as shown in Figure 3.2. Similarly, in Figure 3.3, the dataset was sorted and plot the flows based on the number of packets per flow.

Beyond the largest 50 flows, the sizes of the smaller flows differ marginally. If a threshold value of 10 KB is selected, almost 60K flows will be classified as HHs. Among these 60K flows, only a small proportion will be clearly larger than the threshold value. Flows with a size close to the threshold are almost indistinguishable in size among one another, increasing the probability of misclassification.

From Figure 3.2, it is evident various "steps" showing significantly larger differences in sizes between flows where an appropriate threshold value can be picked, *e.g.,* 100 MB [11], which clearly differentiates the flow that is greater than 100 MB from the next largest flow. In Figure 3.3, it is possible to see a similar phenomenon if we

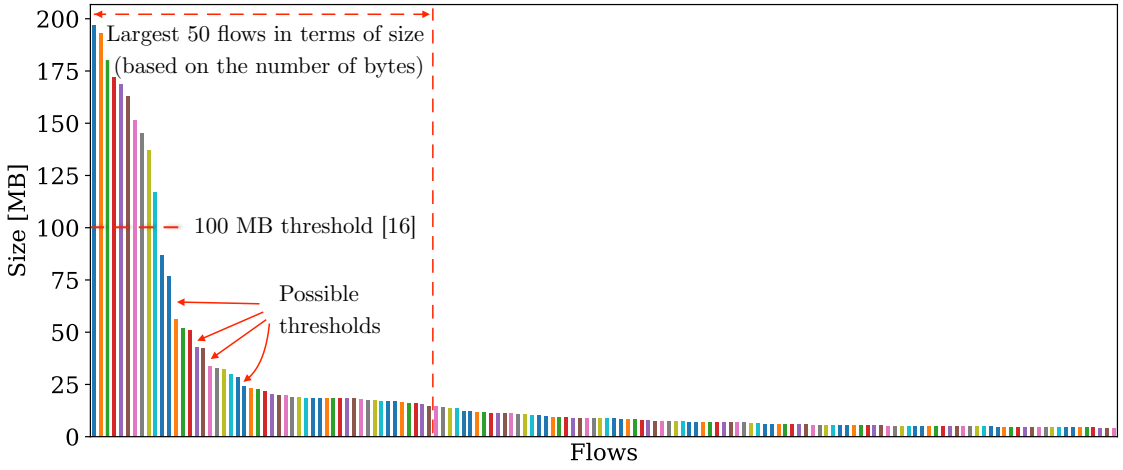


Figure 3.2: The largest 150 flows in the UNIV1 dataset. The flows are sorted by their size (from left to right)

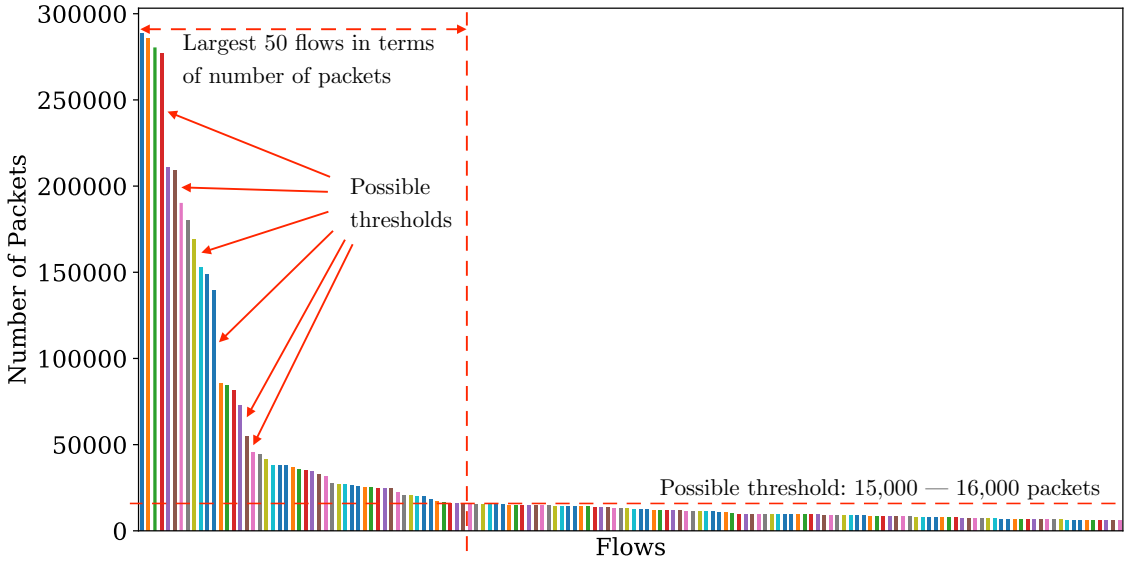


Figure 3.3: The largest 150 flows in the UNIV1 dataset. The flows are sorted by their number of packets (from left to right)

attempt to classify flows based on the number of packets. Beyond the first 50 largest flows, there is a very gradual decrease in the number of packets per flow, such that a threshold value of below 15K packets per flow can easily lead to misclassification. An intuitive conclusion would be to pick a threshold value located in one of the "steps".

3.2 Discussion

Traffic patterns of DC have changed over the last few years. Conventional data centre networks were designed to handle traffic between a data centre and the outside network (also often referred to as north-south traffic). However, the data-driven trend of digital businesses (the more the better) has led to a substantial increase in traffic within a data centre (i.e. east-west traffic). Big data analytics, micro-services architecture (a paradigm to break apart large monolithic applications into sets of small, discrete components), containers, virtual machines, and automation are all added contributors to the massive east-west traffic.

Considering the current trends in networking, a let-up in data expansion is unlikely. On the contrary, the changes in the traffic patterns will be amplified. To adequately support the demand and scale of the continuously increasing workloads, as well as business flexibility and agility, heavy-hitter traffic flow classification will continue to play a key role. However, as discussed earlier, HH detection in the current SDDCN environment is still challenging, especially with regard to traffic flow statistics collection and threshold estimation. Motivated by this, this dissertation formulate the following questions.

Is the threshold-based approach accurate for HH detection? The threshold classifies the flows from the perspective of the metering process rather than the entire network. A 10 KB threshold will result in a different number of HHs than when an 80 KB threshold is used. Obviously, a higher threshold will result in fewer HHs. But this does not mean that the flows classified as non-HHs are really not HHs. It is only the metering process that perceives the flows to be so. This yields certain a error-rate that is not uncommon to see across various HH detection approaches. Consider, for example, the UNIV1 dataset introduced in Section 3.1.3.1.

Table 3.3 shows the PSD of the TCP and UDP packets. From Table 3.3 it is possible to observe, that the packets with the highest occurrence have a size between 1,280 and 2,559 bytes while the average packet size within this range is around 1,377 bytes. On the other hand, Figure 3.4 illustrates the error-rate that one such a packet can cause.

Table 3.2 shows the results in the classification of 59.919 HHs and 336,662 non-HHs with a 10 KB as a threshold. If the classifier includes only one extra packet with a size of 1,377, a total number of 2,866 non-HHs are going to be classified as HHs. Conversely, if the classifier excludes (misses) only one such a packet, it will result in 1,981 HHs identified as non-HHs. In another interpretation, a threshold becomes a range rather than a single value. Flows with sizes within this range are so close to the threshold (10 KB) such that the line which separates them into HHs and non-HHs is very thin. This results in a miss-classification zone (error-rate) caused only by one large packet (of a size of 1,377 bytes).

Table 3.3: Packet size distribution of the UNIV1 [6] dataset

Packet Sizes [bytes]			Count	Average	Percent [%]
40	-	79	5,464,671	66.43	23.32 %
80	-	159	1,802,464	117.63	19.95 %
160	-	319	1,328,662	228.25	16.62 %
320	-	639	546,572	421.43	2.90 %
640	-	1,279	1,353,270	1,004.43	5.50 %
1,280	-	2,559	6,828,810	1,376.95	31.73 %
Total			17,324,449	592.32	100.00 %

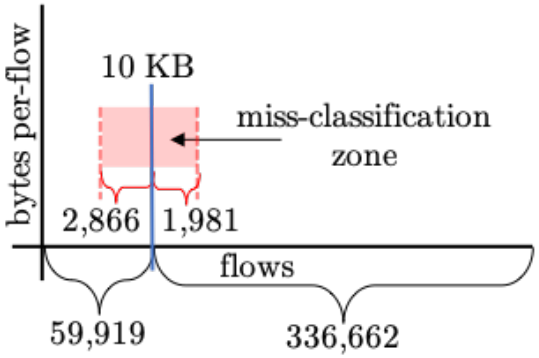


Figure 3.4: Miss-classification one large packet can cause

Is it necessary to identify HHs accurately? Whether the error rate is acceptable or not is a different question. If the main goal of HH identification is to achieve the highest possible accuracy, then threshold-based HH detection is probably not the best solution. However, from a different perspective, the motivation for HH

detection is to improve network performance while not overloading the components that facilitate the measurement/classification. In this case, maintaining network performance is more important than the accurate identification of HHs. Nonetheless, accurate HH identification is still required for some network management related tasks such as accounting purposes where operators want to identify flows that create loads on certain links. As such, accuracy plays a considerate role.

Is there a measure (metric) that could truly replace threshold-based HH detection? So far, the threshold-based approach is the most appropriate approach for HH detection. However, a recent traffic classification method that is gaining popularity is traffic classification using sophisticated statistical features. It is based on the assumption that, on a flow level, different applications exhibit various statistical features that make different types of traffic flows distinguishable. Statistical classification uses an underlying model to calculate the probability of a case belonging to a class. Besides its usefulness in determining the types of applications that generate the traffic, it is also efficient in identifying network attacks and anomalies. This dissertation believe that they capture the behaviour and dynamics of network traffic flows more accurately than the counters (duration, packets, and bytes) used by existing HH detection approaches, especially with respect to their heavy-tailed distribution.

3.3 Final Remarks

Despite the simple methodology of HH detection as well as the clear and well-defined benefits of a centralized network traffic view (control), existing approaches to identify HHs in SDN environments, it is still face challenges. The existing approaches to identify HHs are based on thresholds, *i.e.,* if the flow exceeds a predefined threshold, it will be marked as a HH; otherwise, it will be classified as a non-HH. The threshold can be static or adaptive.

The static threshold comes short in taking into consideration the network traffic character. On the other hand, an adaptive threshold is a sequence of static thresholds that change according to a temporal aspect and so also yields similar challenges as static threshold-based approaches. Furthermore, irregardless of which approach is

adopted, static or adaptive, the threshold–based approach presents two significant issues. First, there is no consistent and accepted threshold that would reliably classify flows. Second, they use counters (duration, packets, and bytes), which their accuracy depends on how complete the flow information is.

Chapter 4

A Threshold Estimation Based on Cluster Analysis

As discussed in the Section 3.1.3.2, identifying a flow is relatively simple when complete flow knowledge is available. However, the identification accuracy is challenging when at any given period only part of the flow is known or if there are too few packets in the flow. To overcome this challenge, a feature is required that captures the flow behaviour (HH or non-HH) well even if only a few packets are known in the flow. In this sense, it is well known that a flow level, different applications exhibit various statistical features (*e.g.,* PSD) that make different types of traffic flows distinguishable [30].

PSD captures the behaviour and dynamics of network traffic flows as accurately as the traditional counters (duration, packets, and bytes) that are used by existing HH identification approaches, especially with respect to their heavy-tailed distribution. The advantage of PSD over traditional counters is that it can perform reliably even if only limited information is available about the flow. An important consideration when analysing the viability of PSD for identifying HH flows is the minimum number of packets. This number represents a 'window' in which the PSDs are going to be analysed. The determination of such 'window' is named 'Ground Truth'.

This chapter provides an extensive study about how to get the ground truth for

identifying HHs using the PSD approach. In this way, a Knowledge Discovery in Database (KDD) method was used to solving the following research question, *It is possible to determine an optimal threshold or set of thresholds that would optimally separate the flows into HHs and non-HHs in a variety of network and traffic conditions?*. The chapter starts discussing KDD analysis. Then, the Six-steps Knowledge Discovery Process (KDP) is detailed. Lastly, a final discussion is presented.

4.1 Knowledge Discovery in Database Analysis

Statistical analysis has been shown to be highly efficient in tasks related to big data. Consequently, it is becoming widely-adapted by both the industry and academia to make an operational sense out of large datasets. Data analysis has three major components: *Knowledge Discovery in Data* (KDD), *Data Mining* (DM), and ML. These components often cause confusion as they generally employ similar (or identical) sets of tools and methods. KDD is used for the automated and systematic extraction of patterns representing knowledge implicitly stored or captured in large datasets, data streams, and massive information repositories. DM extracts patterns from data through ML algorithms. ML uses algorithms that can automatically learn the patterns in the data [27].

Following Fayyad *et al.* [66], this dissertation defines KDD as the multi-step process of discovering useful knowledge from the data and use DM as a particular step in this process. KDD has applications in several domains including computer networks, databases, and statistics [67]. In the networking domain, KDD is mainly used to comprehend massive traffic traces and find patterns that can help in tasks such as traffic classification and anomaly detection. One of the most commonly used methodologies is Cross Industry Standard Process for Data Mining (CRISP-DM) that was primarily used in the industry. In recent years, CRISP-DM has continuously gained popularity in the academic field, mainly due to the modifications that make it simpler to apply to academic research. Another methodology for KDD is Knowledge Discovery Process of Six-step (KDP) developed by Cios *et al.* [68]. KDP, as shown in Figure 4.1, consists of six steps and provides several

feedback loops that aim to improve the overall understanding of the analysed data. The description of the individual steps is as follows [68]:

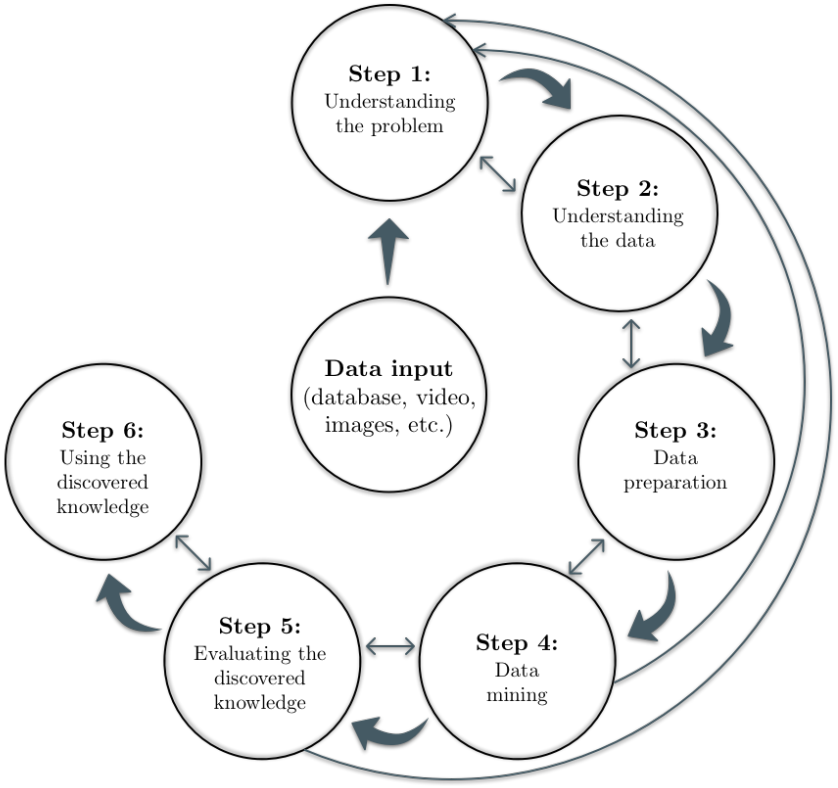


Figure 4.1: Knowledge Discovery Process Model

1. *Understanding the problem domain:* This initial step involves working closely with domain experts to define the problem and determine the project goals, identifying key people, and learning about current solutions to the problem. It also involves learning domain-specific terminology. A description of the problem, including its restrictions, is prepared. Finally, project goals are translated into DM goals, and the initial selection of DM tools to be used later in the process is performed.

2. *Understanding the data:* This step includes collecting sample data and deciding which data, including format and size, will be needed. Background knowledge can be used to guide these efforts. Data are checked for completeness, redundancy, missing values, plausibility of attribute values, etc. Finally,

the step includes verification of the usefulness of the data with respect to the DM goals.

3. *Data preparation:* This step concerns deciding which data will be used as input for DM methods in the subsequent step. It involves sampling, running correlation and significance tests, and data cleaning, which includes checking the completeness of data records, removing or correcting for noise and missing values, etc. The cleaned data may be further processed by feature selection and extraction algorithms (to reduce dimensionality), by derivation of new attributes (say, by discretization), and by summarization of data (data granularization). The end results are data that meet the specific input requirements for the DM tools selected in Step 1.

4. *Data mining:* This step involves using a specific ML algorithm (or a combination of them) to derive knowledge from pre-processed data. If the obtained results by the selected ML algorithm are unsatisfactoriness, requiring modification of the project's goals needs to be done.

5. *Evaluation of the discovered knowledge:* Evaluation includes understanding the results, checking whether the discovered knowledge is novel and interesting, interpretation of the results by domain experts, and checking the impact of the discovered knowledge. Only approved models are retained, and the entire process is revisited to identify which alternative actions could have been taken to improve the results. A list of errors made in the process is prepared.

6. *Use of the discovered knowledge:* This final step consists of planning where and how to use the discovered knowledge. The application area in the current domain may be extended to other domains. A plan to monitor the implementation of the discovered knowledge is created and the entire project documented. Finally, the discovered knowledge is deployed.

4.2 Six-step KDP-based Threshold Estimation

This section provides an exhaustible analysis for estimating an optimal threshold that can separate HHs from non-HHs. In this sense, a detailed description of the six-step KDP model applied to different dataset is provided.

4.2.1 Understanding the Problem Domain

In general, two HH detection approaches exist: static and dynamic threshold-based techniques. In the former, the threshold is usually specified at the start of the detection phase and remains unchanged until the end of the measurement or unless the detector is required to be reconfigured and restarted. In the latter, also often referred to as the adaptive approach, the threshold is adjusted in specific time intervals based on changes in network conditions such as the variation of the traffic load.

Irregardless of which approach is adopted, the determination of the threshold is far from being obvious. A common observation from related work is that there is no consensus about the threshold that reliably separates the flows into HHs and non-HHs. In addition, there is also an ongoing discussion regarding the feature or set of features (*e.g.,* size, duration, and rate) that should be used. It is useful to advise the reader that the problem was dentally explained in Section 3.1.

4.2.2 Understanding the Data

Four publicly accessible traffic traces collected from four different networks, stored in PCAP (Packet CAPture) format were used. To illustrate the packet size distribution, Table 4.1 provides a summary of the first ten million packets only for comparison as the datasets differ in size.

The traffic traces denoted as *UNIV1* and *UNIV2* were captured in university data centre networks and contain roughly 20 million packets (mainly TCP and UDP).

Table 4.1: Packet size distribution of UNIV1, UNIV2, CAIDA2016 and CAIDA2018

		TCP			UDP			Total		
Packet Size [B]	**Dataset**	Count	[%]	Average	Count	[%]	Average	Count	[%]	Average
40 to 70	UNIV1	2 614 663	37.97	65.77	203 256	13.99	66.02	4 077 529	40.78	65.49
	UNIV2	4412	72.42	68.04	111 982	1.13	68.54	237 744	2.38	65.86
	CAIDA2016	5 112 471	56.70	55.68	171 611	18.64	55.72	5 299 305	53.02	55.72
	CAIDA2018	1 770 745	21.53	55.36	415 214	25.23	55.70	2 212 466	22.16	55.49
80 to 159	UNIV1	398 815	5.79	112.15	542 922	37.36	117.84	1 093 850	10.94	115.84
	UNIV2	594	9.75	131.50	4 446 370	45.05	105.83	4 449 030	44.49	105.83
	CAIDA2016	530 109	5.88	97.02	240 413	26.12	125.26	786 351	7.87	105.87
	CAIDA2018	331 862	4.03	107.85	285 618	17.65	116.04	660 792	6.62	111.14
160 to 319	UNIV1	293 073	4.26	230.87	372 315	25.62	242.09	730 384	7.30	233.76
	UNIV2	336	5.52	220.90	619 348	6.28	192.74	619 728	6.20	192.76
	CAIDA2016	185 470	2.06	237.59	265 265	28.82	206.94	458 989	4.59	219.31
	CAIDA2018	297 903	3.62	254.00	125 710	7.77	226.09	437 883	4.39	244.46
320 to 639	UNIV1	310 571	4.51	447.95	25 405	1.75	417.10	346 474	3.46	445.85
	UNIV2	292	4.79	474.30	57 714	0.58	470.53	58 130	0.58	470.50
	CAIDA2016	274 319	3.04	502.00	33 812	3.67	491.80	310 219	3.10	500.60
	CAIDA2018	286 216	3.48	483.27	76 783	4.74	432.34	366 130	3.67	472.53
640 to 1279	UNIV1	790 563	11.48	1004.80	29 892	2.06	960.95	849 156	8.49	1005.77
	UNIV2	114	1.87	876.35	55 924	0.57	1030.12	56 140	0.56	1029.70
	CAIDA2016	365 397	4.05	1070.11	39 660	4.31	1047.44	408 903	4.09	1066.23
	CAIDA2018	346 840	4.22	1005.57	118 125	7.30	1044.11	473 519	4.74	1014.76
1280 to 2559	UNIV1	2 478 370	35.99	1483.37	279 513	19.23	1291.59	2 902 224	29.02	1465.33
	UNIV2	344	5.65	1512.71	4 578 640	46.39	1485.47	4 579 184	45.79	1485.48
	CAIDA2016	2 549 241	28.27	1473.25	169 656	18.43	1451.42	2 730 430	27.32	1471.52
	CAIDA2018	5 192 165	63.12	1483.68	597 074	36.89	1425	5 831 850	58.42	1477.51
Total	UNIV1	6 886 055	100	710.74	1 453 303	100	390.74	9 999 617	100	582.60
	UNIV2	6092	100	198.84	9 869 978	100	758.24	9 999 956	100	749.34
	CAIDA2016	9 017 007	100	517.31	920 417	100	433.62	9 994 197	100	509.13
	CAIDA2018	8 225 731	100	1021.20	1 618 524	100	674.91	9 982 640	100	959.00

From Benson *et al.* [6] is known that all traces were captured using a Cisco port span. To account for delay introduced by the packet duplication mechanism and for end host clock skew, the authors binned results from the spans into 10 microsecond bins. The datasets denoted as *CAIDA2016* and *CAIDA2018* are raw traffic traces captured on high-speed commercial backbone links and contain roughly 16 million packets (also mainly TCP and UDP). The CAIDA2016 dataset was collected by the 'equinix-chicago' high-speed monitor and contains a mix of 6.3M TCP, UDP and ICMP packets. The weight of each packed is defined as the size of its payload, not including the header. The CAIDA2018 was collected by the 'equinix-sanjose' high-speed monitor and contains a mix of 20.2M TCP, UDP and ICMP packets. The weight of each packet is defined as the size of its payload, not including

4.2.3 Preparation of the Data

A flow refers to a set of packets sharing some common properties that traverse an observation point during a certain period of time, usually defined by a *5-tuple*, namely, the source and destination IP addresses and port numbers, and the protocol identifier. Flow records contain traffic information such as flow features (*e.g.,* the total number of octets of all packets belonging to a certain flow) and the flow properties. The traffic traces (UNIV1, UNIV2, CAIDA2016, and CAIDA2018) were processed and organised into flow records using the flowRecorder tool [69]. This tool, written in Python, turns IP packets, either in the form of PCAP files or sniffed live from a network interface, into flow records and stores them in a CSV (Comma-Separated Values) file. Algorithm 1 shows pseudocode for the flow record that was executed for organizing packets into flows and extracts the information and features about the flows.

Algorithm 1 Flow Recorder

```
1: procedure FLOWRECORDER(i, f_ito, o)        ▷ i: trace input (pcap format), o: Output file name (csv format)
2:     function ISFLOWPRESENT(key, f_cache)
3:         if key in f_cache then
4:             return True
5:         else
6:             return False
7:     function ISEXPIRED(key, f_cache, f_ito, t_stp)
8:         if key in f_cache then
9:             t = t_stp − flow_cache[key][f_end]
10:            if t > f_ito then
11:                return True
12:            else
13:                return False
14:    function NEWID(f_id, f_cache)
15:        f_cache[f_id + f_start] = f_cache[f_id]
16:    function CREATEFLOW(f_id, f_cache, FlowFeatures)
17:        f_cache[f_id] = FlowFeatures
```

Flows composed only of packets sent from a single endpoint to another single endpoint are usually termed as unidirectional flows. When information about flows

Table 4.2: List of the flow features used for KDP

Feature name	Feature description	Direction F†	Direction B†
min_ps	Packet with minimum size in the flow	✓	✓
max_ps	Packet with maximum size in the flow	✓	✓
avg_ps	Average packet size	✓	✓
std_ps	Standard deviation of packet sizes	✓	✓
min_piat	Minimum packet inter-arrival time	✓	✓
max_piat	Maximum packet inter-arrival time	✓	✓
avg_piat	Average packet inter-arrival time	✓	✓
std_piat	Stand. dev. of packet inter-arrival times	✓	✓
octTotCount	Number of transmitted bytes in the flow	✓	✓
pktTotCount	Number of transmitted pkts in the flow	✓	✓
$flowDur$	Duration of the flow (in seconds)		
proto	Protocol identifier (e.g. TCP/UDP)		

† F: Forward; B: Backward

composed of packets sent in both directions between two endpoints are required (two unidirectional flows in opposite directions), bidirectional flows may be considered. flowRecorder supports the measurement of flow features in both unidirectional and bidirectional modes. Table 4.2 lists 22 statistical flow features supported by flowRecorder that were used for the analysis. As the table shows, each feature was computed in both directions except the flow duration and the protocol identifier. Depending on the properties of the observed (incoming) packets, either new flow records were created or the flow features of existing ones were updated.

Flow expiration is another key consideration [12]. A flow is deemed to be expired if no packets belonging to the flow have been observed for a certain period of time. This time is most commonly termed as *passive* or *idle time* (f_{ito}). Passive expiration of flows is required to prevent identifying two different flows as the same. Consider two hosts where one initiates many new TCP connections to the other. Each new TCP connection gets a new source port from a fixed range of ephemeral ports, generally incremented by one from the previous allocation. Over time, the TCP port numbers will roll over the range, starting from the smallest number again. In consequence,

Table 4.3: Flows size distributions per each dataset

Flow Size [KB/MB]	DataSet	TCP	[%]	UDP	[%]	Total	[%]
$f_s \leq 10KB$	UNIV1	74 915	25.48	175 029	59.54	249 944	85.02
	UNIV2	495	2.74	14 926	82.66	15 421	85.40
	CAIDA2016	354 299	78.44	79 050	17.50	433 349	95.94
	CAIDA2018	340 818	60.44	193 704	34.35	534 522	94.79
$10KB < f_s \leq 100KB$	UNIV1	38 775	13.19	195	0.07	38 970	13.26
	UNIV2	12	0.07	1 951	10.80	1 963	10.87
	CAIDA2016	13 986	3.10	543	0.12	14 529	3.22
	CAIDA2018	18 286	3.24	1 265	0.22	19 551	3.47
$100KB < f_s \leq 1MB$	UNIV1	4 446	1.51	86	0.03	4 532	1.54
	UNIV2	0	0	543	3.01	543	3.01
	CAIDA2016	2 566	0.57	429	0.09	2 995	0.66
	CAIDA2018	7 480	1.33	811	0.14	8 291	1.47
$1MB < f_s \leq 10MB$	UNIV1	457	0.16	17	0.01	474	0.16
	UNIV2	0	0	67	0.37	67	0.37
	CAIDA2016	721	0.16	74	0.02	795	0.18
	CAIDA2018	1 214	0.22	197	0.03	1 411	0.25
$f_s < 10MB$	UNIV1	46	0.02	4	0.0	50	0.02
	UNIV2	0	0	63	0.35	63	0.35
	CAIDA2016	32	0.01	5	0.0	37	0.01
	CAIDA2018	88	0.02	9	0.0	97	0.02
Total	UNIV1	118 639	40.4	175 331	59.6	293 970	100
	UNIV2	507	2.8	17 550	97.2	18 057	100
	CAIDA2016	371 604	82.9	80 101	17.7	451 705	100
	CAIDA2018	367 886	65.2	195 986	34.8	563 870	100

this creates a problem as a new flow will have the same five-tuple as an earlier flow. A common method to address this problem is to use the idle time.

Typical values for f_{ito} range from 15 *sec* to 300 *sec* [12]. A number of pilot measurements using all four traffic traces were performed. This pilot measurements were focused on the correct expiration without identifying two different flows as the same. The obtained results cover the entire range between $15sec$ and $300sec$. This dissertation uses a value that is from the middle of this range, specifically $150sec$. Using this idle time, the traffic traces were processed yielding four different datasets. Table 4.3 summarizes the flows size distribution obtained of the these traces.

4.2.4 Data Mining

Clustering is an essential tool in data exploration. Cluster analysis examines unlabelled data by either constructing a hierarchical structure or forming a set of groups. Data points belonging to the same cluster exhibit features with similar characteristics [70]. In a nutshell, clustering is about abstraction-discovering structure in collections of data.

In this analysis, it was decided to use K-means mainly because of its simplicity, speed, and accuracy [71]. In addition, several research works also report on its high efficacy when deployed on network traffic data [34], [72]. K-means is an iterative algorithm that seeks to cluster homogeneous or similar subgroups in data. If two data points are similar, they are considered as part of one cluster. The main goal of K-means is to group data points in one cluster to be as close to each other, *i.e.,* to minimise the distance between observations within one cluster, across all clusters. The similarity amongst the observations were determined via the Euclidean distance function [73] between data points. In addition, to provides a better data visualization, an unsupervised technique for reducing the number of dimensions was used. Principal Component Analysis (PCA) is a technique that performs linear transformations process resulting in primary components [74]. A PCA transformation was performed on the 22-dimensional datasets created by the employed features.

K-means requires the number of clusters as input. Several methods for determining the number of clusters exist including V-measure, Adjusted rank Index, and Homogeneity Score [75]. The V-Measure is defined as the harmonic mean of homogeneity and completeness of the clustering. The Rand index is a measure of the similarity between two data clustering. The Homogeneity Score is a metric which provides a confidence value of a cluster labeling given a ground truth [75].

The mentioned above methods are usually used with labelled datasets. Since the datasets used by this dissertation are not labelled, Silhouette score method is used. The silhouette score method does not require labelled data. In addition, the Silhouette method was also shown to be an effective approach for determining the number of clusters in data as well as for validation [76]. The Silhouette method provides a

Silhouette coefficient S_i which measures how well an observation is clustered while it estimates the average distance between the clusters [74]. The values of the measure range between -1 and 1, and the closer the values are to 1, the better the samples are clustered.

Per each dataset was computed the S_i values corresponding to the number of clusters (denoted by k), as shown in Figure 4.2. The number of clusters that yield a relatively strong basis for cluster analysis (i.e., $0.5 < S_i \leq 1$) is up to 15. With a higher number of clusters the S_i values drop significantly. All the Silhouette scores range between 0.5 and 1 with 10 scores (six for CAIDA2016 and four for CAIDA2018) having a value below 0.6; these values are relatively far from the optimal score of 1, and indicate a low basis for clustering. Among all the traffic traces, UNIV2 provides the most stable values that can eventually lead to more than 15 clusters. Candidate values for cluster analysis ranged between 2 and 9 as these cluster numbers got a Silhouette scores higher than 0.9.

The optimal value for k is given by the highest S_i score. In the analysed datasets, $k = 2$ corresponds to $S_i = 1$. However, using two clusters leads to a significant imbalance between them. This is evident from Figure 4.3 which shows the PCA representation – one cluster contains a large number of flows while the other extremely few. The specific numbers of this distribution is provided in Table 4.4.

Table 4.4: Flow distribution with $k = 2$

Dataset	**Class I**	**Class II**	**Total**
UNIV1	293 962	8	293 970
UNIV2	18 056	1	18 057
CAIDA2016	451 696	9	451 705
CAIDA2018	563 869	3	563 872

Such a significant imbalance can result in a deterioration of the classifier's performance, especially in the case of those patterns that belong to the less represented classes. Intuitively, to achieve a better distribution of flows between the individual clusters, it is needed to select a higher k value. Figure 4.2 shows that the Silhouette coefficients provide stable results for up to 9 clusters. Consequently, it is reasonable to select the number of clusters with S_i value closest to 1 yet segment the dataset

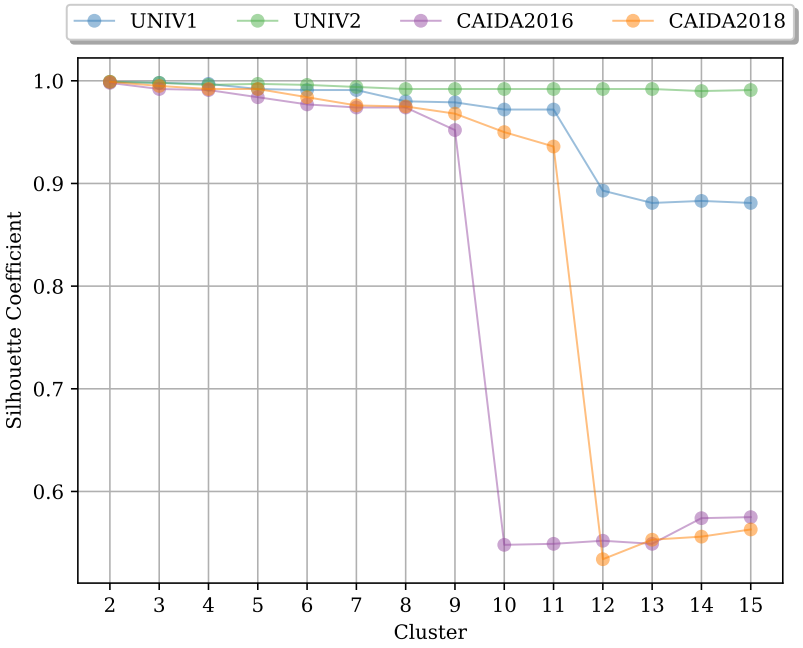
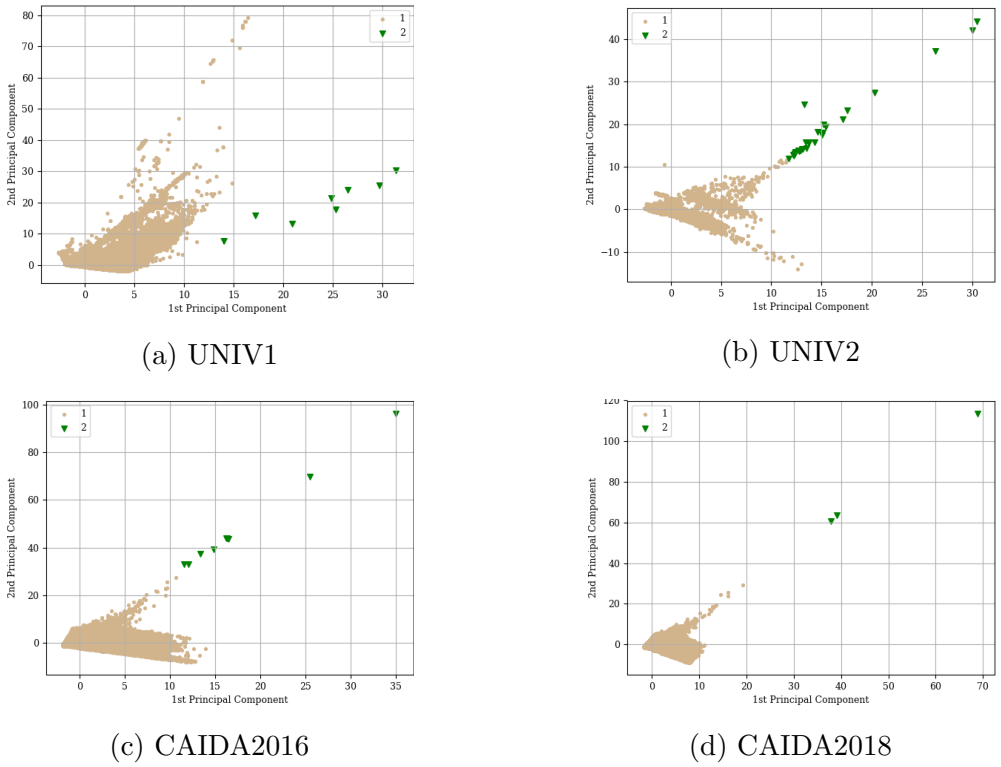


Figure 4.2: Silhouette coefficients for the datasets.

into as many classes as possible, *i.e.*, $k = 4$. The visualisation of the results along the two first principal components obtained with $k = 4$ is provided in Figure 4.4. As a complement to Figure 4.4, Table 4.5 provides further detail on the obtained results. The analysis focuses on the network traffic flow statistical features, in particular, the flow size, number of packets, and flow duration.

Table 4.5: Flows sizes, packet counts, and flow durations obtained using $k = 4$

			Flow Size [MB]			**Number of Packets**			**Flow Duration [s]**		
Class	Dataset	No. of Flows	Max	Min	Average	Max	Min	Average	Max	Min	Average
I	UNIV1	293 709	1.88	0.000032	0.008 47	11 394	1	16.09	2 643.73	0	6.76
	UNIV2	17 979	6.50	0.000064	0.002 12	18 180	2	103.81	455.32	0	33.87
	CAIDA2016	451 051	1.34	0.000028	0.004 21	33 043	1	16.78	20.202	0	3.23
	CAIDA2018	563 472	3.73	0.000029	0.009 48	14 258	1	12.4	22.40	0	2.39
II	UNIV1	6	194.11	145.39	167.37	281 168	149 047	219 806	2 643.13	151.97	1 061.68
	UNIV2	6	250.30	107.610	178.96	284 648	108 230	194 204	455.29	393.25	428.88
	CAIDA2016	7	42.65	31.14	37.60	28 465	20 775	25 357	20.19	15.8	15.86
	CAIDA2018	21	66.22	21.45	34.08	44 251	14 784	23 489	22.39	7.04	19.45
III	UNIV1	26	86.80	18.15	30.35	85 848	14 987	29 760.5	2 641.51	2.27	282.62
	UNIV2	32	95.63	32.53	57.94	118 060	32 590	61 127.50	454.52	48.34	402.615
	CAIDA2016	2	100.05	70.93	85.49	66 743	47 373	57 058	20.20	20.19	20.19
	CAIDA2018	376	19.81	3.76	7.46	14 658	2 513	5 162.06	22.40	0.26	17.25
IV	UNIV1	229	17.64	1.93	4.42	79 661	1 418	6 582.16	2 643.89	1.39	507.30
	UNIV2	36	28.87	7.081	13.10	40 486	8 256	17 376.44	454.52	48.34	402.61
	CAIDA2016	644	22.66	1.35	3.52	24 140	904	2 660.75	20.202	0.028	15.21
	CAIDA2018	3	277.62	143.45	191.33	190 733	98 909	130 531.66	22.29	21.09	21.89



(a) UNIV1

(b) UNIV2

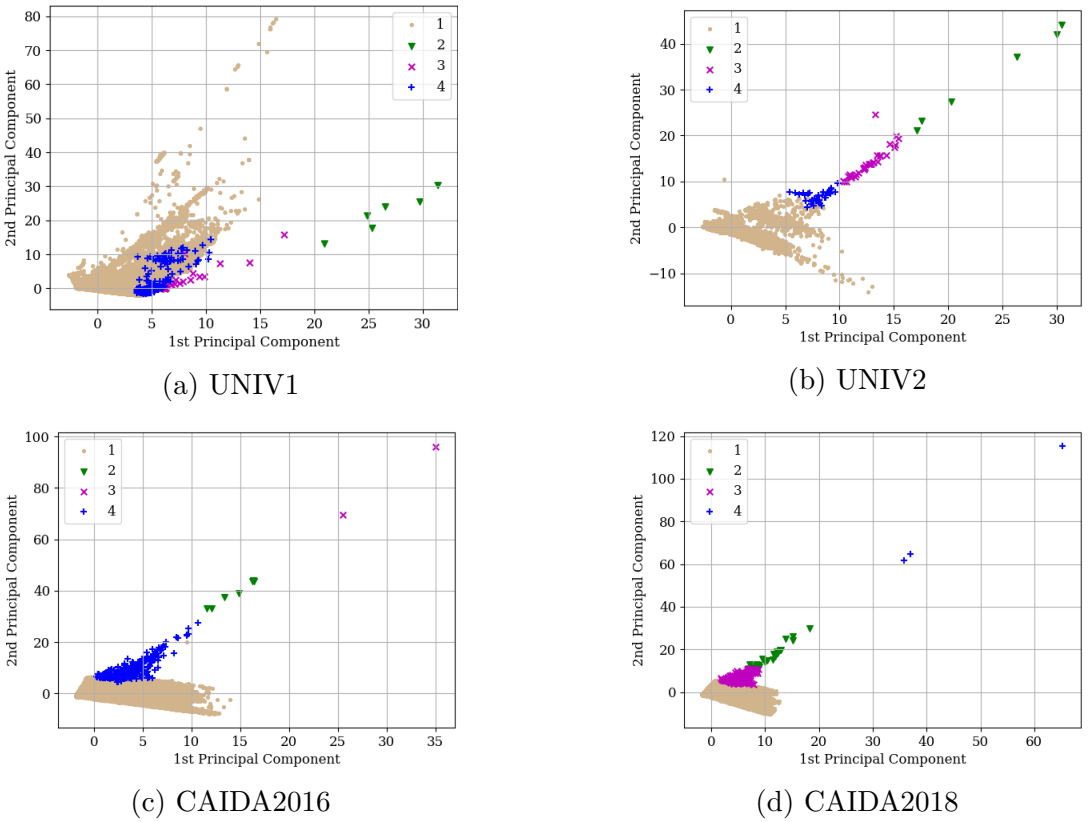
(c) CAIDA2016

(d) CAIDA2018

Figure 4.3: Visualisation of K-means along the two first principal components with $k = 2$

4.2.5 Evaluation of the Discovered Knowledge

Overall, class I contains the flows that are the smallest in size. UNIV2 seems to be an outlier as the obtained results significantly differ from the other datasets. For UNIV1, CAIDA2016, and CAIDA2018 the flow sizes range between 28 $B$and 3.73 MB while the average size is roughly 7.5 KB. The flow sizes for UNIV2 range between 64 $B$and 6.5 MBwhile the average size is 21.24 KB. In terms of packet counts, UNIV1, CAIDA2016, and CAIDA2018 have an average packet count of roughly 15 while in UNIV2 this value is 103.81. The difference between UNIV2 and the other datasets is significant. Table 3.3 shed some light on the reason for this difference. When compared to UNIV1, CAIDA2016, and CAIDA2018, UNIV2 have a large number of UDP packets within the 80 to 159 and 1280 to 2559 ranges. On the other hand, UNIV1, CAIDA2016, and CAIDA2018 have a large number of TCP packets in all the ranges, in contrast to UNIV2.



(a) UNIV1

(b) UNIV2

(c) CAIDA2016

(d) CAIDA2018

Figure 4.4: Visualisation of K-means along the two first principal components with $k = 4$

While $k = 4$ provides better results than clustering with $k = 2$, in terms of flow distribution, the imbalance is still noticeable. The results show class I as the most dominant class while class II to IV seem to be outliers. This distance is evident from both Figure 4.4 and Table 4.5. In general, classes II to IV can sometimes be considered as outliers caused by measurement errors or anomalies. However, this is not the case as they show a strong relationship with the flow size feature. Therefore, they are deemed as correct flows requiring the same attention as the flows in class I.

An important observation that can be derived from Figure 4.4 and Table 4.5 is that there is a class of flows that can be relatively simple to separate from the other flows, especially due to the large distance between them (i.e., the gaps between the groups in Figures 4.3 and 4.4) that make them clearly distinguishable. In UNIV1, UNIV2, CAIDA2016, and CAIDA2018, the flows that are relatively simple to classify belong

to classes II to IV. However, the classification of flows belonging to class I is not that obvious as it contains ambiguous flows, viz. flows that can be both HH as well as non-HH. Motivated by this, an analyse of class I in greater detail was performed.

The analysis of flows belonging to class I followed the same steps that were described in Section 4.1 and applied previously. Figure 4.5 shows that clustering with k higher than 2 yields S_i values not sufficiently close to 1. Therefore, only two clusters ($k = 2$) were used since the Silhouette analysis yields similar scores for all the datasets.

Table 4.6: Flows sizes, packet counts and flow durations for the ambiguous flows using $k = 2$

			Flow Size [MB]			**Number of Packets**			**Flow Duration [s]**		
Class	Dataset	No. of Flows	Max	Min	Average	Max	Min	Average	Max	Min	Average
I	UNIV1	293 002	0.456	0.000032	0.006 29	4 296	1	13.05	2 643.73	0	6.09
	UNIV2	17 979	6.5	0.000064	0.0214	18 180	2	103.81	455.32	0	33.87
	CAIDA2016	449 904	0.324	0.000028	0.0066	7 601	1	14.7	20.20	0	3.29
	CAIDA2018	562 273	0.922	0.000029	0.005 58	14 258	1	13.53	22.40	0	2.36
II	UNIV1	707	1.88	0.458	0.309	11 394	349	1 275	2 643.05	0.379	0.379
	UNIV2	36	28.87	7.08	13.10	40 486	8 256	17 376	453.71	16.06	339.58
	CAIDA2016	1 147	1.34	0.325	0.647	33 043	220	834	20.20	0.05	15.65
	CAIDA2018	1 199	3.734	0.923	1.83	9 008	620	1 356.28	22.403	0.030	15.52

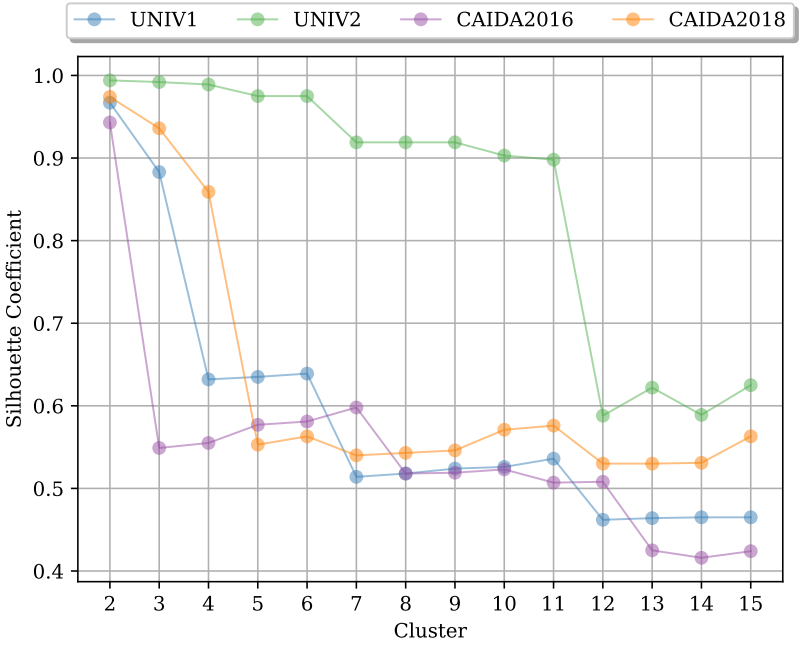


Figure 4.5: Silhouette coefficient scores for class I flows

4.3 Final Remarks

The question researched in this Chapter 4 is as follows: *Is it possible to determine a threshold or set of thresholds that would separate the flows into HHs and non-HHs in a variety of network and traffic conditions?* The answer to this question is: *no*. Based on the results obtained when analysing four data traces collected in four different networks, it can conclude that there is no threshold that could unambiguously classify HH flows.

4.3.1 Significance of Network and Traffic Characteristics

Table 4.6 shows as possibles thresholds for HH classification are as follow (see class I for UNIV1, CAIDA2016, and CAIDA2018 averages):

$$\begin{aligned} \textit{flow size } \theta_s &= 6KB, \\ \textit{packet count } \theta_{pkt} &= 14. \end{aligned}$$

When compared to existing approaches that evaluate their solution using the same datasets, it can conclude that neither θ_s nor θ_{pkt} match any of the thresholds used by related works. For example, Sivaraman *et al.* [56] use packet counts between 60 and 300, Poupart *et al.* [61] use flow size between 10 KB and 1 MB, and Chao *et al.* [47] use a rate of 10 $MBps$. Unlike what this dissertation have provided in this analysis, these related works do not provide any explanation nor justification for selecting the thresholds they use. As the results show, threshold selection without analysing the network and its traffic can lead to suboptimal results. For example, while in the case of UNIV1, CAIDA2016 and CAIDA2018, θ_s and θ_{pkt} are roughly the same and could be considered as optimal thresholds, they would be unsuitable when used in UNIV2 (the presented results show a flow size of $20KB$and a packet count of 104 packets as optimal thresholds for UNIV2). Therefore, works in the field of HH detection should *always* consider the network and its traffic when specifying the used thresholds.

4.3.2 Distribution of TCP and UDP Flows

As noted above, UNIV1 and UNIV2 were captured in data centres while CAIDA2016 and CAIDA2018 in backbone networks. Even though all the datasets contain the same number of packets that were organised into flows using the same f_{ito}, Table 4.3 shows that they yield a different amount of total flows. Another observable difference that can be related to these two network types is the distribution of TCP and UDP flows. UDP flows dominate in data centre networks but TCP flows account for most flows in backbone networks.

In terms of flow size distribution, 85% of the flows in UNIV1 and UNIV2 are smaller than $10KB$. The second most significant flow size in UNIV1 and UNIV2 accounts for roughly 12% of all the flows and falls in the range between $10KB$ and $100KB$. In CAIDA2016 and CAIDA2018, flows with a size less than $10KB$ account for approximately 95% of all the flows. Flows with a size between $10KB$ and $100KB$ account for around 3% of the total number of flows in these two data sets. Flows with a size above $100KB$ are not common for any of the four datasets.

4.3.3 Time Barrier

While analysing complete flows can help to obtain more insights into threshold determination for HH detection, real-time classification of flows is more important. Timely HH detection does not have complete flow information. Ideally, the flow should be classified based on the minimum number of octets (minimum size), packets or the shortest duration. In terms of flow duration, the obtained results above do not show a strong correlation. However, time is still required for real-time classification. This creates a challenge that has not been researched to date.

Chapter 5

Carrying Out Heavy-Hitter Identification using Packet Size Distribution

This chapter provides an extensive study about how to perform HHs identification using PSD. In this sense, this chapter starts introducing a novel approach to classify HHs using per-flow PSD and Template Matching (TM) technique. Next, this chapter presents an extensive evaluation of the approach proposed. Lastly, the final discussions are presented.

5.1 Heavy-Hitter Flow Identification Using Packet Size Distribution and Template Matching

As discussed in the Chapter 4, an important consideration when analysing the viability of PSD for identifying HH flows is the minimum number of packets. This number represents a 'window' in which the PSDs are going to be analysed. In this dissertation the threshold found by the KDP analysis performed are used.

$$\begin{aligned} \textit{flow size } \theta_s &= 6KB, \\ \textit{packet count } \theta_{pkt} &= 14. \end{aligned}$$

Using θ_s and θ_{pkt} defined above, it is possible to specify three regions of small non-HH flows, as shown in Figure 5.1. 'Region 1' contains extremely small non-HHs, less than $6KB$ and less than 14 packets. 'Region 2' also contains non-HHs, however, these flows can have more than 14 packets but still less than $6KB$ in size. 'Region 3' contains non-HHs as well, flows with less than 14 packets but more than $10KB$ in size. Lastly, the 'Region 4' contains flows where granularity is very significant as this region contains HHs, some of which are extremely large. The combination of features appears to be an appropriate technique to increase the classification efficacy and accuracy.

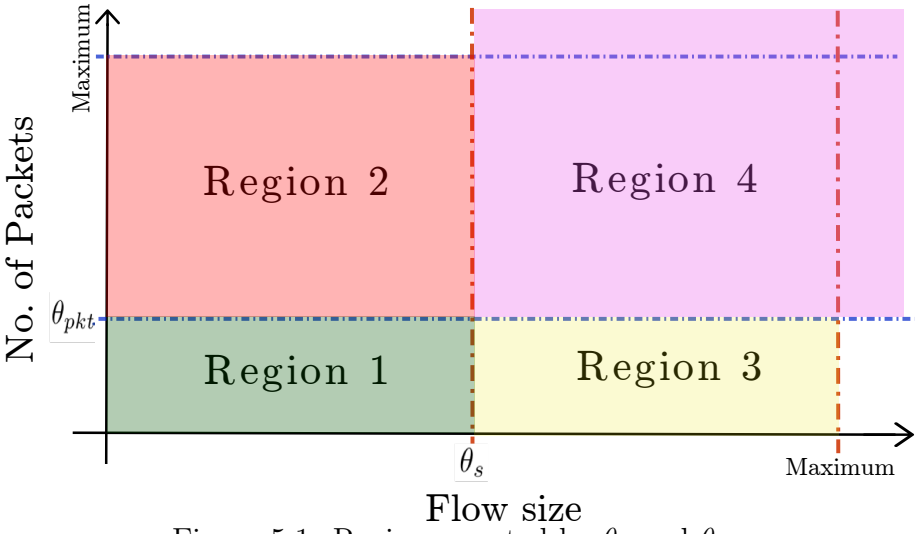
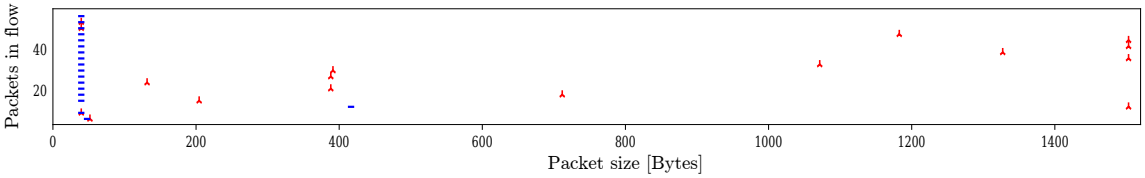
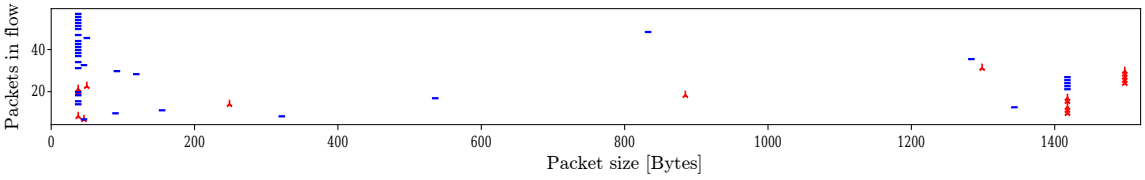


Figure 5.1: Regions created by θ_s and θ_{pkt}

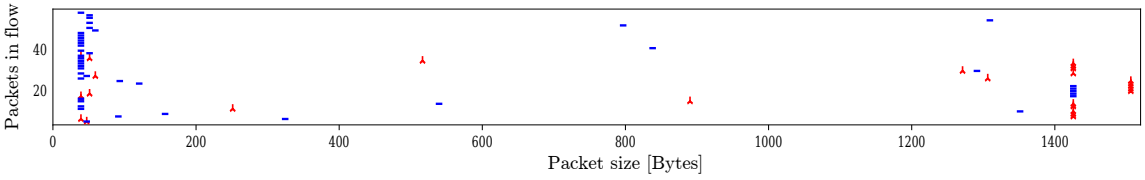
Figure 5.2 provides an example of the packet size distribution of an arbitrary flow belonging to each of the four regions. The flows in Regions 1 (Figure 5.2(a)) and 4 (Figure 5.2(d)) represent the minimums and maximums, *i.e.,* non-HHs and HHs, respectively. The difference in terms of per-flow PSD is clear and simple to recognise without the need for a detailed investigation of the flow. Regions 2 (Figure 5.2(b)) and 3 (Figure 5.2(c)) are, however, not that obvious as the flows exhibit similar characteristics in terms of PSD. The classification of these flows requires an in-depth investigation.



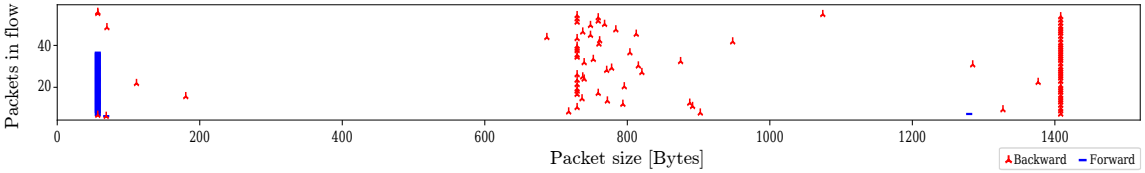
(a) PSD of an arbitrary flow from Region 1



(b) PSD of an arbitrary flow from Region 2



(c) PSD of an arbitrary flow from Region 3



(d) PSD of an arbitrary flow from Region 4

Figure 5.2: Flow samples with various per-flow PSDs

As Figure 5.2 shows, the per-flow PSD can be considered as a collection of points. To obtain a more efficient representation of PSD, the packet sizes can also be considered as a continuous random variable with different values. In this way, it is possible to generate a function which describes how such variable is distributed in a range given. The *probability density function* (*pdf*) is a statistical feature that is suitable for describing this phenomenon. It defines the probability distribution of a continuous random variable.

The continuous random variable representing each packet size belonging to a certain flow is denoted as x_{pkt}. The *pdf* provides the value of the function at any given x_{pkt}. It is important to emphasise that *pdf* does not directly provide the probability of

x_{pkt}. Instead, it yields the probability P that x_{pkt} will take on a value within a given interval $[a, b]$. This can be formally expressed as:

$$P[a <= x_{pkt} <= b] = \int_a^b pdf(x_{pkt})dx_{pkt}. \tag{5.1}$$

This dissertation introduce an approach to identify HHs based on per-flow PSD and TM. The PSD is able to capture the behaviour and dynamics of network traffic, and TM is used for pattern recognition [77]. A crucial point of TM when used in pattern recognition is to adopt an appropriate "measure" to quantify similarity given an *input* and a *template*. In the HHs identification demine, the *input* is a flow, which is expressed as

$$f_s = \sum_{i=1}^{N} x_{pkt}(i), \tag{5.2}$$

where x_{pkt} represents the size of packet belonging to a certain flow. The *templates* represent the flow size behaviours in a particular region, *i.e.,* a *pdf* class. The system proposed provides a similarity measure per-template given an input. The higher the similarity, the more probable is that the input (*i.e.,* a flow) belongs to the region.

Figure 5.3 shows the architecture of the proposed approach. It is composed of three main components: *Density Estimation*, *Template Generator*, and *Template Matching*. Each module is detailed below.

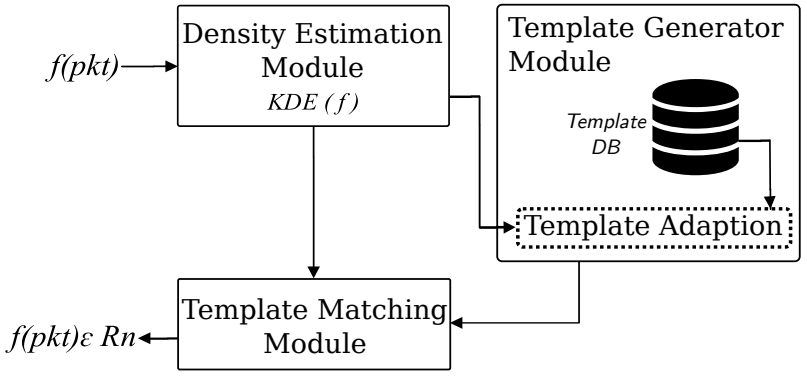


Figure 5.3: Architecture of the system prototype

5.1.1 Density Estimation Module

This module is responsible for estimating the *pdf* of the incoming flow. Several techniques exist to estimate the *pdf* of a random variable. In general, the density estimator can be classified into two types: *parametric* and *non-parametric* [78]. The parametric estimators need a fixed form or structure of the data and depend on the previous data point while the non-parametric estimators have no fixed structure and depend on all the input data points. This dissertation make use of Kernel Density Estimation (kde) to estimate $pdf(x_{pkt})$; formally expressed as:

$$kde(x_{pkt}) = \frac{1}{N} \sum_{i=1}^{N} K_h(x - x_i), \tag{5.3}$$

where K_h is a kernel function, and x represents each point of x_{pkt}. The kde is a non-parametric method in which does not require any preceding models. Moreover, in contrast to other non-parametric methods such as the histograms, its density estimates are smoother, continuous and differentiable [79].

This dissertation uses kde, in particular *Gaussian Kernel* to estimate the *pdf* since it has been shown to yield good performance under general smoothness assumptions. In addition, it has good performance when no additional knowledge of the data is available [70].

5.1.2 Template Generator

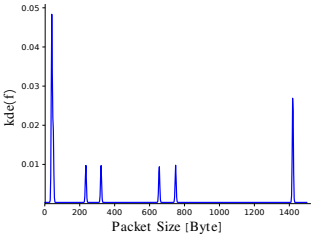
This module is composed of the *Template Database* and *Template Adaption* components. *Template Database* is responsible for storing the individual *templates*. Consider $R = [R_1,\ R_2,\ R_3,\ R_4]$ as the set of the four regions introduced in Figure 5.2 while $R_n = [f_{s_1},\ f_{s_2},\ ...,\ f_{s_N}]$ denotes an arbitrary region composed of a set of N flows.

Templates per each region are generated based on the TemplateGenerator procedure as per Algorithm 2. This procedure has three main functions: pktExtractor, kdeEstima-

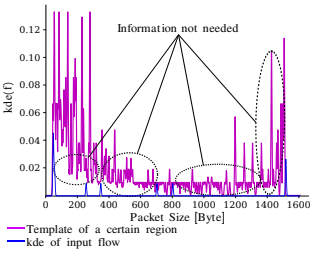
tion, and FinalTemplate. pktExtractor is responsible for extracting the first θ_{pkt} packets that are necessary to be collected before a flow can be classified. pdfEstimation computes the *pdf* estimation. The output of this function is a template for each flow in R_n. The prototype is implemented in Python, each template is a floating-point real value while these values (templates) are stored (per each region) as a list of floats.

Algorithm 2 TemplateGenerator

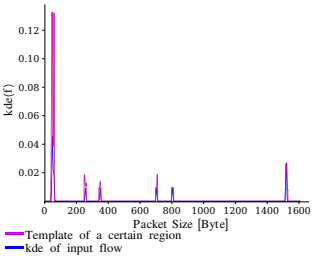
procedure TEMPLATEGENERATOR(i, θ_{pkt})
▷ *pd*: Python *pandas* library
  function FINALTEMPLATE(X_{kde}, Y_{kde})
    global mainT
    if len(mainT) ! = 0 **then**
      for j **in** range(0, len(Y_{kde})): **do**
        y_{aux} = mainT[y][i].max()
        y_{aux_2} = Y_{kde}[i].max()
        if $y_{aux} < y_{aux_2}$ **then**
          mainT = {'x': X_{kde},'y': y_{aux_2}}
        else
          mainT = {'x': X_{kde},'y': y_{aux}}
    else
      mainT = {'x': X_{kde},'y': Y_{kde}}
    return *Template*
  function PDFESTIMATION(y)
    kde = KernelDensity(kernel, bandwidth)
    kde.fit(Y_{kde}[:, None])
    return X_{kde}, Y_{kde}
  function PKTEXTRACTOR(i, θ_{pkt})
    for index, row **in** i.iterrows: **do**
      if index < len(i)−1 **then**
        flowIDaux = i[index+1, f_{key}]
        if flowIDaux ! = i[index, f_{key}] **then**
          data = i.$copy$()
          data.$drop$[data[f_{key}] != i[index−1, f_{key}]
          y = data.$head$(θ_{pkt})
          if len(y) == θ_{pkt} **then**
            X_{kde}, Y_{kde} = KDEESTIMATION(y)
            r = FINALTEMPLATE(X_{kde}, Y_{kde})
    return r
▷ Final template
  Template = PKTEXTRACTOR(i, θ_{pkt})



(a) Example of a *pdf* of a flow to be classified that is calculated by the *Density Estimation Module*



(b) Matching the *pdf* of an arbitrary flow and a template without adaptation



(c) Adapting the template by preserving only those the data points that are required

Figure 5.4: Template adaptation for efficient matching

The number of templates in R_n depends on the numbers of flows in the region. This, however, can lead to challenges in real-world applications (*e.g.,* in terms of resource, time, and memory constraints). To minimise the computing demands, a reduction the number of templates per each region was performed. More specifically, instead of maintaining hundreds of thousands of templates, the algorithm compute only one 'master' template per each region. To achieve this, the algorithm use the FinalTemplate function that computes the maximum value of each template in a particular region. As new flows are observed, new templates are generated and so the maximum values continuously update the final template.

Lastly, *Template Adaption* adapts the templates to the *pdf* of the observed flow (that is going to be classified). Figure 5.4(b) compares a *pdf* of a flow (see Figure 5.4(a)) calculated by the *Density Estimation Module* to a template of a particular region provided by the *Template Database*. From Figure 5.4(b) is evident that not all the data points are equally significant in terms of matching. In addition, the point-by-point comparison of each data point between the flow to be classified and the template is a computationally complex task especially with respect to memory and computing resources. To improve the performance, before the matching takes place, the template is adapted to the input flow so that only those data points that are required for accurate classification are preserved. An example of such adaptation is provided in Figure 5.4(c).

5.1.3 Template Matching Module

This module is responsible for performing a similarity measure between the *pdf* of the observed flows and the templates. In practice, this means that the *pdf* is compared with each 'master' template (one out of the four) in the template database and is classified into that class that yields the highest similarity measure. The similarity measure is usually calculated using distance or correlation metrics [77]. In this module, *Pearson correlation* and *Dynamic Time Warp* (DTW) similarity measures were used. The computational demand of obtaining Pearson correlation is relatively low, and the DTW is sensitive to a linear relationship between the variables [80].

Pearson correlation is a number between -1 and 1 that indicates how much two variables are linearly related. A correlation of -1 indicates that the input and the observed template have a perfect inverse linear correlation. A correlation of 0 means no linear relation. Finally, a correlation equal to 1 means that both the input and the template have a perfect direct linear correlation. DTW, on the other hand, is a distance measure which allows two-time series that are similar but locally out of phase to align in a non-linear manner [81]. DTW computation follows three major steps [81]:

1. Compare each point in $pdf(f)$ with every point in X_r, generating a matrix.
2. Work through the matrix and calculate the lowest cumulative distance for each cell. Subsequently, add the value to the distance of the focal cell.
3. The distance between the two signals is then calculated based on the most efficient pathway through the matrix.

Similarity measure between the input and each template is implemented based on Algorithm 3. The algorithm query the templates stored in the *Template Database*, and perform similarity measurement first using Pearson correlation and then with DTW. The Pearson correlation is computed using the Python `scipy` library [82] while DTW is calculated via the `DTAIDistance` library [83].

Algorithm 3 Match

1: **procedure** MATCH(f_{pkt}, T_n)
2: ▷ f_{pkt}: *pdf* of incoming flow
3: ▷ T_n: Template per-region
4: $T_n = [\ X_{r_1}, X_{r_2}, X_{r_3}, X_{r_4}\ ]$
5: corrResult [1] = pearsonr(X_{r_1}, f_{pkt}) ▷ Region 1
6: DWTResults [1] = dtw.warping(X_{r_1}, f_{pkt})
7: corrResult [2] = pearsonr(X_{r_2}, f_{pkt}) ▷ Region 2
8: DWTResults [2] = dtw.warping(X_{r_2}, f_{pkt})
9: corrResult [3] = pearsonr(X_{r_3}, f_{pkt}) ▷ Region 3
10: DWTResults [3] = dtw.warping(X_{r_3}, f_{pkt})
11: corrResult [4] = pearsonr(X_{r_4}, f_{pkt}) ▷ Region 4
12: DWTResults [4] = dtw.warping(X_{r_4}, f_{pkt})

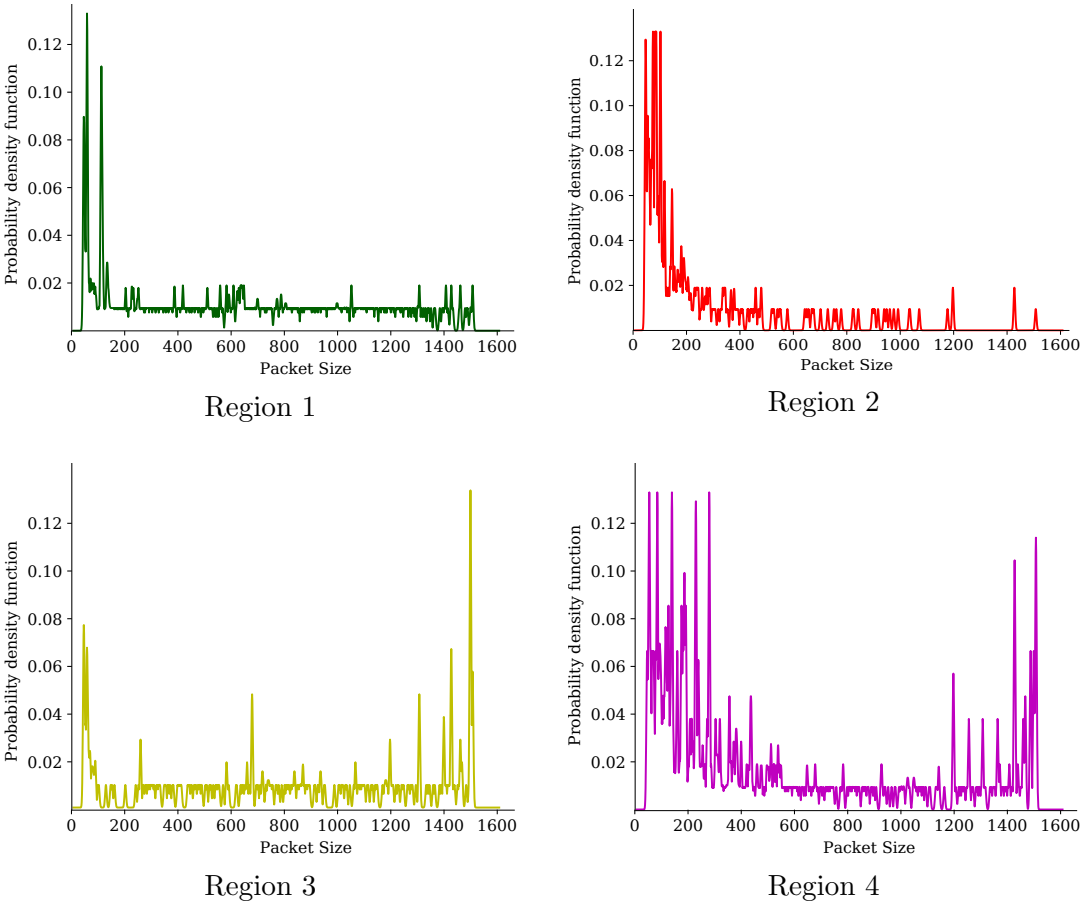
5.2 Evaluations and Results

This section provides an extensive evaluation about the feasibility of using per-flow PSD and TM approach for carrying out HHs identification. In this way, first, the dataset used and the setup are detailed in Section 5.2.1. Then, in Section 5.2.2, a validation using different performance measures are presented. Finally, in Section 5.2.3, a comparison with another approaches is performed.

5.2.1 Dataset

In order to evaluate feasibility of using per-flow PSD and TM approach for carrying out HHs identification. The UNIV1 dataset was used. The traffic traces (UNIV1) were processed and organized into flow records. It is useful to advise the reader that the UNIV1 dataset was dentally explained in Section 4.2.2. This dataset contains a sample of 34 788 flows — 13 457 belonging to R_1, 6894 belonging to R_2, 391 belonging to R_3, and 14 046 belonging to R_4. This dataset was split into training and test sets with a 70/30 ratio while cross validation was also used to increase the effectiveness of the model. To compute the templated was used the training set. To create the templates for the regions, as shown Figure 5.1, $\theta_{pkt} = 14$ and flow size $\theta_s = 6KB$ were also used. It is important to highlight that those values were determined via

KDP analysis in Section 4.1. Figure 5.5 shows the templates generated.



Region 1 Region 2

Region 3 Region 4

Figure 5.5: Template per-region for UNIV1

5.2.2 Performance measures

The performance measures in this paper are given by five metrics derived from the *Confusion Matrix*, viz., *True Positive Rate*, *False Positive Rate*, *Accuracy*, *Precision*, and *f-measure*. The confusion matrix is a specific table layout that allows visualisation of the performance of a certain classification system. Each row of the matrix represents the instances in a classified class while each column represents the instances in an actual class. Let A denoted the instances in an actual class and A' the instances in a classified class. Then each reference standard metric is expressed

as a function of the relationship between the *True Positive* (TP), *True Negative* (TN), *False Positive*, (FP) and *False Negative* (FN).

Table 5.2 shows a typical confusion matrix of a binary classification. The TP represents a successful classification, *e.g.,* a particular sample belonging to R_n, which was classified in class R_n. TN depicts successful classification in which the sample did not belong to R_n and was classified as Non-R_n. FP refers to an unsuccessful classification, *e.g.,* a sample belonging to Non-R_n was classified as R_n. FN represents an unsuccessful classification as well. In this case, a sample belongs to R_n is classified as Non-R_n .

Table 5.2: Confusion Matrix for R_n

A / A'	R_n	Non-R_n
R_n	TP	FP
Non-R_n	FN	TN

The classification system has been trained to distinguish between R_1, R_2, R_3, and R_4. The performance measures were conducted per each region. Table 5.3 reports the confusion matrix for each region. Also, the performance measures are summarised in Table 5.4.

The *True Positive Rate* (TPR) (or recall) is the ratio of successful classifications in a certain region. It is computed by the following equation:

$$TPR = \frac{TP}{TP + FN}. \tag{5.4}$$

The results obtained indicate that the region with highest TPR is Region 3, with TPR value of 0.9847, and the reason behind this is related to the number of flows belonging to this region. In contrast to the other regions, Region 3 contains very few flows, roughly 391. It is likely that, when the templates were generated, most of the flows belonged to Region 3. This results in a better representation of the flow behaviour. The region having the lowest TPR is the Region 2 with 0.8860. This can be regarded as the opposite of Region 3, *i.e.*, when the templates were generated,

Table 5.3: Confusion matrix per-region

Region 1

A' \ A	R_1	Non-R_1
R_1	12 930	780
Non-R_1	527	20 080

Region 2

A' \ A	R_2	Non-R_2
R_2	6108	667
Non-R_2	786	27 227

Region 3

A' \ A	R_3	Non-R_3
R_3	385	919
Non-R_3	6	33 478

Region 4

A' \ A	R_4	Non-R_4
R_4	12 955	44
Non-R_4	1091	20 698

Table 5.4: Performance measures for UNIV1 dataset

	TPR	**FPR**	**Accuracy**	**Precision**	**f-measure**
$\boldsymbol{R_1}$	0.9608	0.037	0.9619	0.9431	0.9519
$\boldsymbol{R_2}$	0.8860	0.024	0.9582	0.9018	0.8937
$\boldsymbol{R_3}$	0.9847	0.027	0.9734	0.2952	0.9628
$\boldsymbol{R_4}$	0.9223	0.002	0.9673	0.9966	0.9580
Mean	0.9385	0.020	0.9623	0.7841	0.9362

few flows belonged to Region 2. Overall, the proposed approach achieves a TPR of 0.9385 which means the approach has high sensitivity.

The *False Positive Rate* (FPR) is the ratio of unsuccessful classifications in a certain region, with the best value being zero. FPR is computed by the number of all FPs, divided by the total number of TNs and FPs:

$$FPR = \frac{FP}{TN + FP}. \tag{5.5}$$

The best FPR value is 0.002 which corresponds to Region 4. The highest value obtained is for Region 1 with 0.037. Notwithstanding this region has the highest value, it still is near to zero, which indicates good performance. The general result obtained shows an average of 0.020 which means that the approach has a low false positive rate.

The *Accuracy* is the ratio of successful predictions made to both classes and is calculated as the total number of two correct predictions divided by the total number of the dataset. It is computed by the following equation:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{5.6}$$

The best accuracy result is 1, whereas the worst is 0. Region 3 has the accuracy of 0.9734, while Region 2 is the region with lowest value, 0.9582. The reason is the same as that for TPR. The approach achieves an average accuracy of 0.96

Precision, also known as the positive predictive value, is the ratio of correct classification. Formally, the precision shows the ratio of correctly classified positive values to the total classified positive values. This metric highlights the correct positive predictions out of all the positive predictions. It is calculated as the number of correct positive predictions (TP) divided by the total number of positive predictions (TP + FP):

$$Precision = \frac{TP}{TP + FP}. \tag{5.7}$$

There is a natural trade-off between TPR and precision, which is why the region that performs best on one measure is not usually the one that perform best on the other measure. This is the case of Region 3. The precision value for Region 3 is 0.2952, while the other regions' values are over 0.9018. Such a low value for the Region 3 indicates that flows belonging to the other regions tend to be classified as Region 3 mostly. The overall precision is 0.7841 giving the per-flow PSD approach an acceptable false positive rate.

The *f-measure* statistic (or F1 score) considers both the TPR and precision of a classifier to measure its quality:

$$\text{f-measure} = 2 \times \frac{TPR \times Precision}{TPR + Precision}. \tag{5.8}$$

An *f-measure* score reaches its best value at 1 and worst value at 0. A low *f-measure* score is an indication of both poor precision and poor recall. The average of both recall and precision of the per-flow PSD approach is high. Thus, the intuitive *f-measure* score for the proposed approach is high, nearly 1.

5.2.3 Performance comparison

The obtained results were compared with classification techniques used by Poupart *et al.* [61]. Poupart *et al.* [61] report the TPR obtained using *Neural Networks* (NN), *Gaussian Processes Regression* (GPR), and *Online Bayesian Moment Matching* (oBMM) to classify HHs.

NN is a series of algorithms that endeavours to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. Also, NN can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria. Furthermore, NN is considered as one of the most flexible predictors in the sense that it can approximate any function when using sufficiently many nodes and data [27]. The GPR models are non-parametric kernel-based probabilistic models that are belonging to supervised learning. The GPR calculates the probability distribution over all admissible functions that fit by using the training data [65]. oBMM is a tractable online Bayesian learning algorithm for learning mixture models using the method of moments [61].

Figure 5.6 shows the TPR of each algorithm, and per-flow PSD using TM approach which is denoted as TM. It is evident that the proposed approach achieves only comparable results. However, there are some significant considerations.

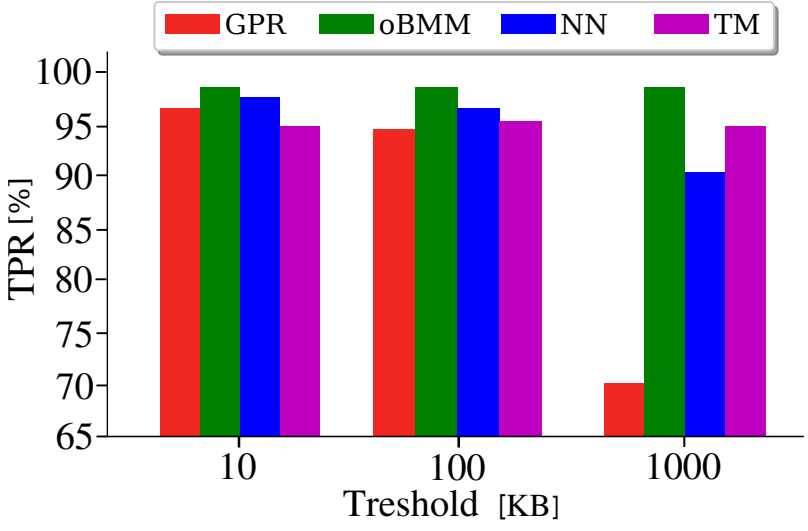


Figure 5.6: True positive rate comparison

1. In this comparison, it is performed only TPR results as Poupart *et al.* [61] does not provide any information on the other measures. As such, the comparison is only partial while it is also likely that the thresholds selected by Poupart *et al.* were selected optimally for their purposes.

2. GPR and NN cannot maintain the same performance when the threshold is changed. The per-flow PSD using TM approach achieves the same performance in different thresholds.

3. The NN and oBMM approaches tend to be affected by class imbalances more than the Gaussian process, which explains why their accuracy often suffers as the classification threshold increases. In the proposed approach, the imbalance in the regions is not a problem, since each flow is analysed in the same packet window. Also, the template is generated by the same packet window.

4. Considering Table 5.4, the achieved results are promising. Per-flow PSD is capable of capturing the behaviour and dynamics of traffic flows and as such, it is suitable for HH detection. Using TM, it can achieve a classification accuracy of 96% and higher per each class.

5.3 Final Remarks

In this chapter provides an extensive study about how to carry out HHs identification using per-flow PSD and TM. Also, the chapter provides an exhaustible evaluation of approach proposed. In this way, first, the evaluation performed shows the usefulness of per-flow PSD in describing the flow behaviour. Second, using the PSD and TM to classify, it possible achieve an overall accuracy of 96% in early stage of the flow. Third, this approach does not require any modification to the applications or end hosts and it provides an indication of which flow is leading to be a HH upon the start of each flow. The ability to predict which flow will have an impactful size in an early stage allows improving several activities related to the network such as routing mechanisms, QoS provisioning, and so on. In particular, for the routing mechanisms, it helps to avoid congestion and to mitigate the need for load balancing. However,

the time required to collect enough flow details for reliable classification can still be a challenge and requires further research.

Chapter 6

Conclusions

This chapter starts summarising the research work carried out in this dissertation. Then, it provides the answers for the fundamental questions that guided the verification of the hypothesis defended in this dissertation. Afterwards, the chapter overviews the main contributions achieved when conducting such verification. The last section outlines directions for future work.

6.1 Answers for the fundamental question

At first, this dissertation defined the following question: **How to predict what flow is going to be an HH flow or non-HH without its completed information?**

To get the answer, this dissertation presented the investigation carried out to verify the hypothesis: **using per-flow PSD allows capturing the behaviour and dynamics of network traffic flow more accurately than the counters in the early stage of the flow.** Based on the hypothesis, this work proposed a novel HH identification approach based on PSD and TM.

This dissertation also displayed the reference implementation of the proposed approach as well as an extensive evaluation and analysis about effectively carrying

out HHs identification in SDDCNs. The evaluation performed shows, first, the usefulness of per-flow PSD in describing the flow behaviour. Second, the approach proposed can achieves up to 96% accuracy while using only the first 14 packets of a flow. Furthermore, this accuracy remains consistent throughout all classifications while existing approaches yield different accuracies for different flow size-based thresholds.

This approach does not require any modification to the applications or end hosts and it provides an indication of which flow is leading to be a HH upon the start of each flow. The ability to predict which flow will have an impactful size in an early stage allows improving several activities related to the network, such as routing mechanisms, QoS provisioning, and so on. In particular, for the routing mechanisms, it helps to avoid congestion and to mitigate the need for load balancing.

6.2 Future work

Although this dissertation has achieved promising results in the experiments performed, this dissertation does not study the impact of time. Time is a critical factor in HH detection. The packets belonging to various flows pass through the data plane in an indeterministic sequence. As a result, it is not uncommon for packets belonging to the same flow to be captured with intervals of more than tens of seconds. In real-time HH detection, waiting for tens of seconds or longer is not acceptable.

For instance, the achieved accuracy for the approach proposed is up to 96%, when classifying UNIV1 with θ_s = 6, 10, 100 and 1000 KB and θ_{pkt} = 14 packets. However, for some flows, it takes up to 460 seconds to collect enough packets/bytes to meet these thresholds, which is unacceptable for timely classification. A time limit set for per-flow packet collection could resolve this problem simply. This way the classifier does not have to wait too long to see a specific number of packets before it makes a decision. However, this also raises degradation in accuracy due to too few packets captured per flow within the time duration. Real-time classification might be achieved but at the expense of collecting too few packets to make an accurate detection.

Bibliography

[1] B. J. van Asten, N. L. M. van Adrichem, and F. A. Kuipers, "Scalability and Resilience of Software-Defined Networking: An Overview," in *Computer Communications*, vol. 67, Elsevier, 2014, pp. 1–19.

[2] P. Xiao, W. Qu, H. Qi, Y. Xu, and Z. Li, "An efficient elephant flow detection with cost-sensitive in sdn," in *1st International Conference on Industrial Networks and Intelligent Systems*, ser. INISCom '15, Tokio, Japan, Mar. 2015, pp. 24–28.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," in *ACM SIGCOMM Computer Communication Review*, vol. 38, New York, USA, ACM, Mar. 2008, pp. 69–74.

[4] A. Pekár, M. Chovanec, L. Vokorokos, E. Chovancová, P. Feciľak, and M. Michalko, "Adaptive aggregation of flow records," in *Computing and Informatics*, vol. 37, Slovak Academy of Sciences, 2018, pp. 142–164.

[5] B. Li, J. Springer, G. Bebis, and M. H. Gunes, "A survey of network flow applications," in *Journal of Network and Computer Applications*, vol. 36, Elsevier, 2013, pp. 567–581.

[6] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of 10th Internet Measurement Conference*, ser. IMC '10, Melbourne, Australia: ACM, Nov. 2010, pp. 267–280.

[7] L. Yang, B. Ng, and W. K. G. Seah, "Heavy hitter detection and identification in software defined networking," in *Proceedings of 25th International Conference on Computer Communication and Networks*, ser. ICCCN '25, Waikoloa, HI, USA: IEEE, Aug. 2016, pp. 1–10.

[8] J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, A. R. Curtis, and S. Banerjee, "Devoflow: Cost-effective flow management for high performance enterprise networks," in *Proceedings of 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. HotNets '10, Monterey, California: ACM, Oct. 2010, pp. 1–6.

[9] T. Mori, M. Uchida, R. Kawahara, J. Pan, and S. Goto, "Identifying Elephant Flows Through Periodically Sampled Packets," in *Proceedings of 4th ACM SIGCOMM conference on Internet measurement*, ser. IMC '04: Taormina, Sicily, Italy: ACM, 2004, pp. 115–120.

[10] K.-C. Lan and J. Heidemann, "A measurement study of correlations of internet flow characteristics," in *Computer Networks*, vol. 50, New York, NY, USA: Elsevier North-Holland, Inc., Jan. 2006, pp. 46–62.

[11] A. R. Curtis, W. Kim, and P. Yalagandula, "Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection," in *Proceedings of 30th IEEE International Conference on Computer Communications*, ser. INFOCOM'11, Shanghai, China: IEEE, Apr. 2011, pp. 1629–1637.

[12] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow monitoring explained: From packet capture to data analysis with netflow and ipfix," in *IEEE Communication Surveys and Tutorials*, vol. 16, IEEE, 2014, pp. 2037–2064.

[13] N. Brownlee and K. C. Claffy, "Understanding internet traffic streams: Dragonflies and tortoises," in *IEEE Communication Magazine*, vol. 40, IEEE, 2002, pp. 110–117.

[14] M. E. Crovella and A. Bestavros, "Self-similarity in world wide web traffic: Evidence and possible causes," in *IEEE/ACM Transactions on Networking*, vol. 5, Dec. 1997, pp. 835–846.

[15] S. Shakkottai, N. Brownlee, and K. C. Claffy, "A study of burstiness in tcp flows," in *Proceedings of International Conference on Passive and Active Network Measurement*, ser. PAM'05, Berlin, Heidelberg: Springer, 2005, pp. 13–26.

[16] The Open Networking Foundation, *OpenFlow Switch Specification Version 1.5.1 (Protocol version 0x06)*, ONF TS-025, Mar. 2015.

[17] D. Kreutz and F. Ramos, "Software-Defined Networking: A Comprehensive Survey," in *Proceedings of the IEEE*, vol. 103, IEEE, Jan. 2014, pp. 14–76.

[18] H. Kim and N. Feamster, "Improving network management with software defined networking," in *IEEE Communication Magazine*, vol. 51, IEEE, Feb. 2013, pp. 114–119.

[19] A. I. Montoya-Munoz, D. M. Casas-Velasco, F. Estrada-Solano, A. Ordonez, and O. M. C. Rendon, "A yang model for a vertical sdn management plane," in *Proceedings of IEEE Colombian Conference on Communications and Computing*, ser. COLCOM'17, IEEE, Cartagena, Colombia: IEEE, Aug. 2017, pp. 1–6.

[20] F. E. Solano, A. Ordóñez, L. Z. Granville, and O. M. C. Rendon, "A framework for sdn integrated management based on a cim model and a vertical management plane," in *Computer Communications*, vol. 102, Elsevier, 2017, pp. 150–164.

[21] J. A. Wickboldt, W. P. de Jesus, P. H. Isolani, C. B. Both, J. Rochol, and L. Z. Granville, "Software-defined networking: Management requirements and challenges," in *IEEE Communication Magazine*, vol. 53, IEEE, 2015, pp. 278–285.

[22] A. Mestres, A. Rodriguez-Natal, J. Carner, P. Barlet-Ros, E. Alarcón, M. Solé, V. Muntés-Mulero, D. Meyer, S. Barkai, M. J. Hibbett, G. Estrada, K. Ma'ruf, F. Coras, V. Ermagan, H. Latapie, C. Cassar, J. Evans, F. Maino, J. Walrand, and A. Cabellos, "Knowledge-defined networking," in *SIGCOMM Computer Communication Review*, vol. 47, New York, NY, USA: ACM, Sep. 2017, pp. 2–10.

[23] M. Faizul Bari, R. Boutaba, R. Esteves, L. Granville, M. Podlesny, M. Rabbani, Q. Zhang, and M. F. Zhani, "Data center network virtualization: A survey," in *IEEE Communications Surveys Tutorials*, vol. 15, IEEE, Jan. 2013, pp. 909–928.

[24] F. Amezquita-Suarez, F. Estrada-Solano, N. L. S. da Fonseca, and O. M. C. Rendon, "An efficient mice flow routing algorithm for data centers based on software-defined networking," in *Proceedings of IEEE International Conference on Communications*, ser. ICC'19, Shanghai, China, May 2019, pp. 1–6.

[25] A. Yassine, H. Rahimi, and S. Shirmohammadi, "Software defined network traffic measurement: Current trends and challenges," in *IEEE Instrumentation & Measurement Magazine*, vol. 18, IEEE, 2015, pp. 42–50.

[26] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski, "A knowledge plane for the internet," in *Proceedings of Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '03, Karlsruhe, Germany: ACM, 2003, pp. 3–10.

[27] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[28] I. Portugal, P. Alencar, and D. Cowan, "The use of machine learning algorithms in recommend systems: A systematic review," in *Expert Systems with Applications*, vol. 97, Elsevier Ltd, 2018, pp. 205–227.

[29] S. Ayoubi, N. Limam, M. A. Salahuddin, N. Shahriar, R. Boutaba, F. Estrada-Solano, and O. M. Caicedo, "Machine Learning for Cognitive Network Management," in *IEEE Communications Magazine*, vol. 56, IEEE, 2018, pp. 158–165.

[30] T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," in *IEEE Communications Surveys Tutorials*, vol. 10, IEEE, 2008, pp. 56–76.

[31] N. Sadashiv and S. M. D. Kumar, "Cluster, grid and cloud computing: A detailed comparison," in *6th International Conference on Computer Science and Education*, ser. ICCSE, Singapore, Singapore: IEEE, Aug. 2011, pp. 477–482.

[32] P. Bermolen and D. Rossi, "Support vector regression for link load prediction," in *4th International Telecommunication Networking Workshop on QoS in Multiservice IP Networks*, Venice, Italy: IEEE, Feb. 2008, pp. 268–273.

[33] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," in *SIGCOMM Computer Communication Review*, vol. 35, New York, USA: ACM, Aug. 2005, pp. 217–228.

[34] J. Zhang, Y. Xiang, W. Zhou, and Y. Wang, "Unsupervised traffic classification using flow statistical properties and ip packet payload," in *Journal of Computer and System Sciences*, vol. 79, Elsevier, Aug. 2013, pp. 573–585.

[35] W. Iqbal, M. N. Dailey, and D. Carrera, "Unsupervised learning of dynamic resource provisioning policies for cloud-hosted multitier web applications," in *IEEE Systems Journal*, vol. 10, Dec. 2016, pp. 1435–1446.

[36] O. Chapelle, B. Schlkopf, and A. Zien, *Semi-Supervised Learning*, 1st. The MIT Press, 2010.

[37] F. Lin and W. W. Cohen, "Semi-supervised classification of network data using very few labels," in *Proceedings of International Conference on Advances in Social Networks Analysis and Mining*, ser. ASONAM '10, Odense, Denmark: IEEE Computer Society, Aug. 2010, pp. 192–199.

[38] F. Qian, G.-m. Hu, and X.-m. Yao, "Semi-supervised network traffic classification," in *AEU - International Journal of Electronics and Communications*, vol. 62, New York, USA: Elsevier, Jun. 2008, pp. 557–564.

[39] A. Shrivastav and A. Tiwari, "Network traffic classification using semi-supervised approach," in *International Conference on Machine Learning and Computing*, Bangalore, India: IEEE, Feb. 2010, pp. 345–349.

[40] A. L. Bazzan, "Opportunities for multiagent systems and multiagent reinforcement learning in traffic control," in *Autonomous Agents and Multi-Agent Systems*, vol. 18, Springer, Sep. 2009, pp. 342–375.

[41] R. Sun, S. Tatsumi, and G. Zhao, "Q-map: A novel multicast routing method in wireless ad hoc networks with multiagent reinforcement learning," in *Proceedings of IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*, vol. 1, Beijing, China: IEEE, Oct. 2002, pp. 667–670.

[42] P. Arcaini, E. Riccobene, and P. Scandurra, "Modeling and Analyzing MAPE-K Feedback Loops for Self-Adaptation," in *Proceedings of 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, Florence, Italy: IEEE, May 2015, pp. 13–23.

[43] J. Hyun and J. W. K. Hong, "Knowledge-defined networking using in-band network telemetry," in *19th Asia-Pacific Network Operations and Management Symposium*, ser. APNOMS'17, Seoul, South Korea: IEEE, Sep. 2017, pp. 54–57.

[44] A. Duque-Torres, F. Amezquita-Suárez, O. M. Caicedo Rendon, A. Ordóñez, and W. Y. Campo Muñoz, "An approach based on knowledge-defined networking for identifying heavy-hitter flows in data center networks," in *Applied Sciences*, vol. 9, MPDI, Nov. 2019.

[45] C. Bi, X. Luo, T. Ye, and Y. Jin, "On precision and scalability of elephant flow detection in data center with sdn," in *Proceedings of 32nd IEEE Global Communications Conference Workshops*, ser. GLOBECOM'13, Atlanta, GA, USA: IEEE, Dec. 2013, pp. 1227–1232.

[46] Z. Liu, D. Gao, Y. Liu, H. Zhang, and C. H. Foh, "An adaptive approach for elephant flow detection with the rapidly changing traffic in data center network," in *International Journal of Network Management*, vol. 27, Wiley, 2017, e1987.

[47] S. Chao, K. C. Lin, and M. Chen, "Flow classification for software-defined data centers using stream mining," in *IEEE Transactions on Services Computing*, IEEE, 2018, pp. 1–1.

[48] H. Xu and B. Li, "Repflow: Minimizing flow completion times with replicated flows in data centers," in *IEEE Conference on Computer Communications*, ser. INFOCOM'14, Toronto, ON, Canada, Apr. 2014, pp. 1581–1589.

[49] A. Munir, I. A. Qazi, Z. A. Uzmi, A. Mushtaq, S. N. Ismail, M. S. Iqbal, and B. Khan, "Minimizing flow completion times in data centers," in *Proceedings of IEEE Conference on Computer and Communications*, ser. INFOCOM'13, Turin, Italy: IEEE, Apr. 2013, pp. 2157–2165.

[50] C.-Y. Hong, M. Caesar, and P. B. Godfrey, "Finishing flows quickly with preemptive scheduling," in *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '12, Helsinki, Finland: ACM, 2012, pp. 127–138.

[51] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center tcp (dctcp)," in *SIGCOMM Computer Communication Review*, vol. 41, New York, USA: ACM, Aug. 2010, pp. 63–74.

[52] W. Cui, Y. Yu, and C. Qian, "DiFS: Distributed Flow Scheduling for adaptive switching in FatTree data center networks," in *Computer Networks*, vol. 105, 2016, pp. 166–179.

[53] X. Wu and X. Yang, "DARD: Distributed adaptive routing for datacenter networks," in *Proceedings of International Conference on Distributed Computing Systems*, Macau, China: IEEE, Jun. 2012, pp. 32–41.

[54] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: A scalable and flexible data center network," in *SIGCOMM Computer Communication Review*, vol. 39, New York, USA: ACM, Aug. 2009, pp. 51–62.

[55] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *Proceedings of 7th Conference on Networked Systems Design and Implementation*, ser. NSDI'10, San Jose, California: USENIX Association, 2010, pp. 19–19.

[56] V. Sivaraman, S. Narayana, O. Rottenstreich, S. Muthukrishnan, and J. Rexford, "Heavy-hitter detection entirely in the data plane," in *Proceedings of the Symposium on SDN Research*, ser. SOSR '17, Santa Clara, CA, USA: ACM, 2017, pp. 164–176.

[57] M. Chiesa, G. Kindler, and M. Schapira, "Traffic engineering with equal-cost-multipath: An algorithmic perspective," in *IEEE/ACM Transactions on Networking*, vol. 25, IEEE, Apr. 2017, pp. 779–792.

[58] R. Trestian, G. Muntean, and K. Katrinis, "Micetrap: Scalable traffic engineering of datacenter mice flows using openflow," in *Proceedings of 21st IEEE/IFIP International Symposium on Integrated Network Management*, ser. IM'13, Ghent, Belgium, May 2013, pp. 904–907.

[59] R. Liu, H. Gu, X. Yu, and X. Nian, "Distributed flow scheduling in energy-aware data center networks," in *IEEE Communications Letters*, vol. 17, IEEE, Apr. 2013, pp. 801–804.

[60] C. Lin, C. Chen, J. Chang, and Y. H. Chu, "Elephant flow detection in datacenters using openflow-based hierarchical statistics pulling," in *Proceedings of 33rd IEEE Global Communications Conference*, ser. GLOBECOM'14, Austin, TX, USA, Dec. 2014, pp. 2264–2269.

[61] P. Poupart, Z. Chen, P. Jaini, F. Fung, H. Susanto, Y. Geng, L. Chen, K. Chen, and H. Jin, "Online flow size prediction for improved network routing," in *Proceedings of 24th IEEE International Conference on Network Protocols*, ser. ICNP'16, Singapore, Singapore: IEEE, Nov. 2016, pp. 1–6.

[62] C. Wang, G. Zhang, H. Chen, and H. Xu, "An aco-based elephant and mice flow scheduling system in sdn," in *Proceedings of 2nd IEEE International Conferece on Big Data Analysis*, ser. ICBDA'17, Beijing, China: IEEE, Mar. 2017, pp. 859–863.

[63] P. Phaal and M. Lavine, "Sflow version 5," sFlow.org, Specification, Jul. 2004.

[64] Cisco and/or its affiliates, "White paper: The zettabyte era: Trends and analysis," Cisco Systems, Inc., Tech. Rep. C11-739110-00, Jul. 2017.

[65] T. Nguyen, G. Armitage, P. Branch, and S. Zander, "Timely and continuous machine-learning-based classification for interactive ip traffic," in *IEEE/ACM Transactions on Networking*, vol. 20, Piscataway, NJ, USA: IEEE Press, Dec. 2012, pp. 1880–1894.

[66] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "The kdd process for extracting useful knowledge from volumes of data," in *Communications of the ACM Magazine*, vol. 39, New York, USA: ACM, Nov. 1996, pp. 27–34.

[67] A. Metwally, D. Agrawal, and A. El Abbadi, "Efficient computation of frequent and top-k elements in data streams," in *Proceedings of the 10th International Conference on Database Theory*, ser. ICDT'05, Edinburgh, UK: Springer-Verlag, 2005, pp. 398–412.

[68] K. J. Cios, W. Pedrycz, R. W. Swiniarski, and L. A. Kurgan, *Data Mining: A Knowledge Discovery Approach*. Berlin, Heidelberg: Springer-Verlag, 2007.

[69] A. Pekar and A. Duque-Torres, *A Network Traffic Flow Feature Measurement Tool – flowRecorder*, version v1.1.2, Nov. 2018.

[70] S. Zhong, T. M. Khoshgoftaar, and N. Seliya, "Analyzing software measurement data with clustering techniques," in *IEEE Intelligent Systems*, vol. 19, IEEE, Mar. 2004, pp. 20–27.

[71] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms," in *Annals of Data Science*, vol. 2, Springer, Jun. 2015, pp. 165–193.

[72] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," in *Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data*, ser. MineNet '06, Pisa, Italy: ACM, 2006, pp. 281–286.

[73] L. A. kurgan and P. Musilek, "A survey of knowledge discovery and data mining process models," in *The Knowledge Engineering Review*, vol. 21, Cambridge University Press, 2006, pp. 1–24.

[74] C. Subbalakshmi, G. R. Krishna, S. K. M. Rao, and P. V. Rao, "A method to find optimum number of clusters based on fuzzy silhouette on dynamic data set," in *Procedia Computer Science*, vol. 46, Elsevier, 2015, pp. 346–353.

[75] Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu, "Understanding of internal clustering validation measures," in *IEEE International Conference on Data Mining*, Sydney, NSW, Australia, Dec. 2010, pp. 911–916.

[76] F. Wang, H.-H. Franco-Penya, and Kelleher, "An analysis of the application of simplified silhouette to the evaluation of k-means clustering validity," in *13th International Conference on Machine Learning and Data Mining*, ser. MLDM'17, New York, USA: Springer, 2017, pp. 291–305.

[77] A. Gorcin and H. Arslan, "Template matching for signal identification in cognitive radio systems," in *Proceedings IEEE Military Communications Conference*, ser. MILCOM, Orlando, FL, USA: IEEE, Oct. 2012, pp. 1–6.

[78] B. W. Silverman, *Density estimation for statistics and data analysis*. Routledge, 2018.

[79] M. Rousson and D. Cremers, "Efficient kernel density estimation of shape and intensity priors for level set segmentation," in *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention*, ser. MICCAI'5, Palm Springs, CA, USA: Springer, 2005, pp. 757–764.

[80] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise reduction in speech processing*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 1–4.

[81] J. Zhao and L. Itti, "Shapedtw: Shape dynamic time warping," in *Pattern Recognition*, vol. 74, Elsevier, 2018, pp. 171–184.

[82] E. Jones, T. Oliphant, P. Peterson, *et al.*, *SciPy: Open source scientific tools for Python*, Available: http://www.scipy.org/, 2001.

[83] W. Meert and T. V. Craenendonck, *dtaidistance*, version v1.1.2, Jul. 2018.

Chapter 7

Appendix A

The research work presented in this dissertation was reported to the scientific community through paper submissions to renowned conferences and journals. The process of doing research, submitting paper, gathering feedback, and improving the work helped to achieve the maturity hereby presented.

A.1 Papers: accepted and on reviewing

A.1.1 Accepted

1. **A. Duque-Torres**, F. Amezquita-Suarez, O.M. Caicedo Rendon, A Ordoñez, and W.Y. Campos, "An Approach based on Knowledge-Defined Networking for Identifying Heavy-Hitter Flows in Data Center Networks". Applied Science. 2019, 9, 4808.

    - **Type:** Journal – Applied Science
    - **Status:** Submitted, accepted and published
    - **Colciencias index:** A2
    - **JCR:** 2.71
    - **JSR:** Q2

2. **A. Duque-Torres**, A. Pekar, W.K.G Seah, and O.M.C Rendon, "Heavy-Hitter Flow Identification in Data Centre Networks Using Template Matching," accepted to the 44^{th} IEEE Conference on Local.

   - **Type:** Conference - 44^{th} IEEE Conference on Local
   - **Status:** Submitted, accepted and presented.
   - **core index:** A

3. **A. Duque-Torres**, A. Pekar, W.K.G Seah, and O.M.C Rendon, "Clustering-based analysis for heavy-hitter detection," presented in the Asia Pacific Regional Internet Conference on Operational Technologies (APRICOT), Daejon, South Korea.

   - **Type:** Conference
   - **Status:** Submitted, accepted and presented.

A.1.2 On Revision

1. **A. Duque-Torres**, A. Pekar, W.K.G Seah, and O.M.C Rendon, "Knowledge Discovery: A shed light on heavy-hitter detection."

   - **Type:** Journal – IEEE Transaction on Knowledge and Data Engineering
   - **Status:** Submitted
   - **Colciencias index:** A1
   - **JCR:** 8.71
   - **JSR:** Q1

2. A. Pekar, **A. Duque-Torres**, W.K.G Seah, and O.M.C Rendon, "Per-Flow Packet Size Distribution for Heavy-Hitter Flow Detection."

   - **Type:** Journal – Computer Networks
   - **Status:** Submitted

- **Colciencias index:** A1
- **JCR:** 7.71
- **JSR:** Q1

A.1.3 Other Publications

1. **Duque-Torres A**, Rodriguez-Pabon C, Ruiz-Rosero J et al. A new environmental monitoring system for silkworm incubators. F1000Research 2018, 7:248

    - **Type:** Journal – F1000Research
    - **Status:** Submitted, accepted and published
    - **Colciencias index:** A1
    - **JSR:** Q1

Chapter 8

Appendix B

The scripts developed and presented in this dissertation was reported to the scientific community through GitHub

B.1 Flow Recorder Tool

https://github.com/drnpkr/flowRecorder [69]

B.2 A Threshold Estimation Based on Cluster Analysis

https://github.com/aduquet/KDP-Clustering-HHs

B.3 HHs Flow Classification System using per-flow PSD and TM

https://github.com/aduquet/PSD-TM