Article

An Approach Based on Knowledge-Defined Networking for Identifying Heavy-Hitter Flows in Data Center Networks

Alejandra Duque-Torres ^{1,*} , Felipe Amezquita-Suárez ¹, Oscar Mauricio Caicedo Rendon ¹ , Armando Ordóñez ¹ and Wilmar Yesid Campo ²

¹ Telematics Engineering Group, Telematics Department, Universidad del Cauca, Popapán 190003, Colombia; cfamezquita@unicauca.edu.co (F.A.-S.); omcaicedo@unicauca.edu.co (O.M.C.R.); jaordonez@unicauca.edu.co (A.O.)

² Programa de Ingenieria Electronica, Universidad del Quindio, Armenia 630002, Colombia; wycampo@uniquindio.edu.co

* Correspondence: aduquet@unicauca.edu.co

Received: 2 October 2019; Accepted: 1 November 2019; Published: 10 November 2019

Abstract: Heavy-Hitters (HHs) are large-volume flows that consume considerably more network resources than other flows combined. In SDN-based DCNs (SDDCNs), HHs cause non-trivial delays for small-volume flows known as non-HHs that are delay-sensitive. Uncontrolled forwarding of HHs leads to network congestion and overall network performance degradation. A pivotal task for controlling HHs is their identification. The existing methods to identify HHs are threshold-based. However, such methods lack a smart system that efficiently identifies HH according to the network behaviour. In this paper, we introduce a novel approach to overcome this lack and investigate the feasibility of using Knowledge-Defined Networking (KDN) in HH identification. KDN by using Machine Learning (ML), allows integrating behavioural models to detect patterns, like HHs, in SDN traffic. Our KDN-based approach includes mainly three modules: HH Data Acquisition Module (HH-DAM), Data ANALyser Module (HH-DANM), and APplication Module (HH-APM). In HH-DAM, we present the flowRecorder tool for organizing packets into flows records. In HH-DANM, we perform a cluster-based analysis to determine an optimal threshold for separating HHs and non-HHs. Finally, in HH-APM, we propose the use of MiceDCER for routing non-HHs efficiently. The per-module evaluation results corroborate the usefulness and feasibility of our approach for identifying HHs.

Keywords: software-defined networking; knowledge-define networking; heavy-hitter; data center networks

1. Introduction

Today's data centres networks (DCNs) have become an efficient and promising infrastructure to support a wide range of technologies, network services and applications such as multimedia content delivery, search engines, e-mail, map-reduce computation, and virtual machine migration [1]. Despite the well-known DCNs capabilities, scalability and traffic management are still challenges. Both scalability and management are growing in complexity especially due to the exponentially increasing volume and heterogeneity of the network traffic. DCNs are based on networking paradigms as Network Virtualization and Software-Defined Networking (SDN) which are promising solutions to address these challenges [1]. In particular, SDN separates the control plane from the data plane, enabling a logically centralised control of network devices [2,3]. By moving the control logic from the forwarding devices to a logically centralised device, knowing as the Controller, network devices become simple forwarders

that are programmable through standardised protocols such as OpenFlow [4] and ForCES [5]. DCNs using SDN are referred to as Software-Defined Data Center Networks (SDDCNs) [6].

Centralized control of the network provides a flexible architecture for more efficiently managing the network traffic. However, this centralized control does not guarantee that the network performance will not degrade when traffic volume raises [7]. One class of traffic that constantly poses a challenge is Heavy-Hitter (HH) flows—large-volume flows that consume considerably more network resources than other flows combined [7–10]. One of the consequences of unsupervised forwarding of HHs is that it often leads to network congestion and, subsequently, to overall network performance degradation [7,8,11–14]. A pivotal task for handling HHs efficiently is their identification. In the literature, there are different approaches [15–19] that use per-flow statistics collection in SDDCNs for identifying HHs. Nevertheless, these approaches present both relatively high overhead and low granularity of traffic flow semantics. To avoid network overload and latency, other approaches [17,18,20] utilise sampling for data collection. Nonetheless, since sampling does not consider the packet size, large packets can be missed, resulting in a large error in the HHs identification. To identify HHs, switch-based and host-based approaches can also be used. However, switch-based can only be carried out by hardware modifications [21]. This fact contradicts the softwarization principles of SDN. In turn, despite the fact that the host-based approaches reduce the overhead on the network, they require modification of the operating system in each host leading to scalability issues [22–24]. It is noteworthy that all approaches afore-cited perform a threshold-based HH identification. However, they lack a smart system that efficiently identifies HH according to the network behaviour; it means they are unaware of traffic in the network.

In this paper, we propose a novel HH flow identification approach aiming at overcoming the lack above-mentioned and investigating the feasibility of using the Knowledge-Defined Networking (KDN) concept in the HH identification. KDN is a networking concept which takes advantage of Machine Learning (ML) techniques to improve several network management services. To the best of our knowledge, to date, there is no similar approach in the HH identification domain. Our approach is composed of three main modules: HH-Data Acquisition Module (HH-DAM), HH-Data ANALyser Module (HH-DANM), and HH-APplication Module (HH-APM). In particular, in HH-DAM, we present the flowRecorder tool [25] that is used for generating and organizing packets into flows records. Using this tool, we generated a dataset from publicly accessible university DCN traffic traces. In HH-DANM, we performed a cluster-based analysis to determine an optimal threshold that separates the flows into HHs and non-HHs. Through this analysis, we suggest the use of the flow size $\theta_s = 7$ KB, and packet count $\theta_{pkt} = 14$ as thresholds for HH classification. In HH-APM, we propose the use of MiceDCER to efficiently route non-HHs. MiceDCER results show that it can reduce the number of routing rules by installing wildcard rules based on the information carried by the Address Resolution Protocol (ARP) packets. In summary, the per-module evaluation results show the usefulness and feasibility of our KDN-based approach for identifying HHs smartly.

The rest of this paper is organised as follows: In Section 2 we give a brief overview of the fundamental aspects that our approach is built on. In Section 3 we review related work for identifying HH while in Section 4 we present our approach. Lastly, we conclude and discuss future work in Section 5.

2. Background

In this section, we give a brief background on the main domains our approach covers. We start with HH definition in Section 2.1. Then, Section 2.2 provides a brief overview of the SDN paradigm, its architecture, and main features. Section 2.3 presents a brief SDDCN overview while Section 2.4 describes ML. Finally, in Section 2.5, we introduce KDN-based.

2.1. Heavy-Hitter Flows

A flow is defined as a set of packets passing an observation point during a specific time interval [26]. 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. Studies about flow behaviour show that a very small percentage of flows carry the bulk of the traffic, especially in DCN. These flows are often termed as HH flows [27–29]. Generally, HHs can be classified by using different metrics based on its duration, size, rate and burstiness. Each flow 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 metric used.

Lan and Heidemann [28] provide a definition of flow types within each category with a zoological fair. For instance, flows that have a duration longer than a certain time period are *tortoises*, otherwise they are termed as *dragonflies*. Flows that have a size larger than s B (bytes) are *elephants*, in turn, *mice* are flows that have a size less than or equal to s . *Cheetahs* are flows with a rate greater than r Bps while *snails* are flows with a rate less than or equal to r . Flows with burstiness greater than b ms are called *porcupines* while those with burstiness less than or equal to b are *stingrays*. Table 1 summarises the main characteristics of HH flows as described by Lan and Heidemann [28]. 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. The rest of the paper uses the flow size feature for defining HHs.

Table 1. Taxonomy of Heavy-Hitter (HH) flows as per Lan and Heidemann [28].

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

The SDN architecture comprises four planes [30–32]: Data Plane, Control Plane, Application Plane, and Management Plane. 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 behaviour and requirements. The Controller is responsible for programming the forwarding elements via SBIs. An SBI allows communication between the Control Plane and the Data Plane by providing: programmatic control of all forwarding operations, capabilities advertisement, and statistics reporting [33]. The Application Plane includes network programs that explicitly, directly, and programmatically communicate their requirements and desired network behaviour to the SDN Controller via NBIs. Finally, the Management Plane ensures 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 [34].

2.3. Software-Defined Networking Data Centre Networks

A typical DCN comprises a conglomeration of network elements that ensures the exchange of traffic between machines/servers 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 [1]. Despite the importance of DCNs, their architecture is 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 [1,35].

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 virtualized compute, storage, and networking resources with a standardised platform for managing the entire integrated environment. Following Faizul Bari et al. [1], 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 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 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 [6].

Figure 1. SDDCN structure with a conventional topology.

2.4. Machine Learning

ML includes 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 [36]. In the networking context, some possibilities arise from using ML: (i) forecast behaviour in the network [37,38], (ii) anomaly detection [39,40], (iii) traffic identification and flow classification [17,41]; and (iv) adaptive resource allocation [42,43]. For more information about the ML potential in the networking field, we refer the reader to Ayoubi et al. [44] and Boutaba et al. [45].

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 [44]. Typically, SL is used to solve classification and regression problems that pertain to forecast outcomes. For instance, prediction of traffic [46], end-to-end path bandwidth [47], or link load [48]. 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 [49], packet loss estimation [50], and resource allocation [51].

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) [52]. Interest in SSL has increased in recent years, particularly because of application domains in which unlabelled data are plentiful, such as classification of network data using very few labels [53], network traffic classification [54], and verification networks [55]. 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 [56]. RL is suited for making cognitive choices, such as decision making, planning, and scheduling. For instance, the routing scheme for delay tolerant networks [50], as well as multicast routing and congestion control mechanisms [57].

2.5. Knowledge-Defined Networking

In 2003, Clark et al. [58] suggested the addition of a Knowledge Plane (KP) to the traditional computers network architecture formed by the Control Plane and Data Plane [33]. KP adopts Artificial Intelligence to perform tasks that are human intelligence characteristic, i.e., systems with the ability to reason, discover meaning, generalise, or learn from past experiences [36]. To achieve these abilities, KP proposed the use of ML techniques that offer advantages to networking, such as automation processes (recognise-act), recommendation systems (recognise-explain-suggest), and data prediction. These advantages bring the possibility of having a smart network operation and management [34].

Nowadays, the possibility of improving the way to operate, optimise and troubleshoot computer networks by using KP is nearer than fifteen years ago because of two main reasons. Firstly, SDN offers full network view via a logically centralised Controller. Furthermore, SDN improves the network control functions that facilitate the gathering of information about the network state in real-time [34]. Secondly, the capabilities of network devices have significantly improved, facilitating the gathering of information in real-time about packets, processing time, and flow-granularity [33].

The addition of KP to the traditional SDN architecture is called KDN [34]. It comprises four planes: Data Plane, Control Plane, Management Plane, and KP. The Data Plane is responsible for generating metadata by the forwarding network devices. The 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. In the Management Plane, the metadata from Control and Data Planes are collected and stored. The Management Plane provides a basic analysis of statistics per-flow and per-forwarding devices

to the KP. Finally, KP sends to the Controller one or a set of instructions about what the network is supposed to do [59].

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). In KDN, DAM comprises the Data and Control Planes, DANM contains both the Management Plane and the ML Module from KP, and the APM includes the Decision Module from KP [59].

3. Related Work

In SDN, the HHs identification has been addressed from different network places: Controller, Switch, and Host. The Controller-based approaches compare the flow size statistics with a static and predefined threshold to identify HHs. There are two approaches to obtain the flow size statistics: pulling, and sampling. In pulling, the central controller maintains statistics (e.g., packets, bytes, and duration time) provided by OpenFlow [15,19,60]. Also, the Controller maintains the network state by sending Read-State messages periodically to the Data Plane. Sampling reduces the original traffic data characteristics by extracting the representative traffic data part. The sampling of flows is a trade-off between data reduction and preserving the details of the original data [61].

In the literature, the switch-based approach moves the task of HHs identification from the Controller to the switches. This identification introduces new functionalities in switches to record flows size statistics. Then, the flow sizes are compared with a static and predefined threshold at the end-switch [2,62]. In Host-Based identification, when the measurement of a flow (e.g., socket buffer and flow size) exceeds a previously set threshold value, the detector determines if the flow is a HH or not [24,35,63]. It is important to highlight that the use of a static and pre-defined threshold offers a rapid identification but low accuracy when the traffic is dynamic and grows suddenly. Table 2 summarises the main shortcomings for the different approaches.

Table 2. Approaches for HHs identification shortcomings.

Approach	Related Works	Network Overhead	Processing Overhead	HW ¹	SW ²
Controller	Al-Fares et al. [15]	High	High	✗	✗
	Dias Knob et al. [60]				
	Xia and Ren [61]				
Switch	Peng Xiao et al. [2]	Medium	Low	✓	✗
	Cui, W et al. [62]				
	Peng, X et al. [2]				
Host	Trestian et al. [24]	Medium	Low	✗	✓
	Amezquita-Suarez <i>et al.</i> [35]				
	Afaq et al. [63]				

¹ Hardware Modifications, ² Software Modifications

In short, the HHs identification approaches that use per-flow statistics collection in SDDCNs [15–19] yield both relatively high overhead and low granularity of traffic flow semantics. To avoid network overload and latency, some approaches, such as [17,18,20], utilise sampling for data collection. Unfortunately, since sampling does not take into account the packet size, large packets can be missed, resulting in a large error in the HHs identification. Usually, the switch-based approaches can only be realised by hardware modifications [21]. This fact contradicts the softwarization principle of SDN.

On the other hand, despite that host-based approaches reduce the overhead on the network, they require modification of the operating system in each host leading to scalability issues [22–24].

4. KDN-Based Heavy-Hitter Identification Approach

This section details our approach. An overview is presented in Section 4.1. The architecture and modules are introduced and evaluated in Section 4.2.

4.1. System Overview

The current methods to identify HHs are threshold-based. However, such methods lack a smart system that efficiently identifies HHs according to the network behaviour. Hereinafter, we introduce an approach to overcome this lack and investigate the feasibility of using KDN to identify HHs. Figure 2 overviews our approach by a KDN control loop, including three modules: HH-DAM, HH-DANM, and HH-APM. Overall, HH-DAM sends packets captured to HH-DANM, which is responsible for storing and generating a network traffic state model. Finally, the Controller gets the awareness from HHs Flag sub-module in HH-APM and instructs the new network configurations to the forwarder devices. In a high-abstraction level, our approach operates as follows.

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, through sampling and filtering rules.
2. *Packet Observation* → *Data Collector*. In the *Data Collector*, the packets provided by the HH-DAM are organised and stored into flows. This Collector aims at gathering enough information to offer a global view of the network behaviour.
3. *Heavy-Hitters Data Collector* → *ML Techniques*. Data Collector feeds the ML Techniques with current and historical data. Thus, our approach can learn from the network behaviour and generate knowledge (e.g., a model of the network).
4. *ML Techniques* → *HHs Flag* → *SDN Controller*. The *E/M 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, HH-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 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 HH-APM can make decisions automatically on behalf of the Administrator. Furthermore, the model can be used to optimise the existing network configuration [34]. For instance, the model can learn adaptively according to the traffic change, and find the optimal configuration to routing HHs and, thus, avoid congestion.

Figure 2. HH Detection approach Overview.

4.2. Architecture and Modules

Figure 3 presents the architecture and modules of the KDN-based approach proposed for identifying HHs smartly. This architecture includes the four KDN planes namely, Data, Control, Management, and Knowledge. KP includes the ML techniques from the HH-DANM module aiming at generating a model for clustering HHs and non-HHs. The Control Plane is responsible for routing flows according to the decisions made by the HH-APM module. Furthermore, the Control Plane sends information to the Management Plane about the network state including flows statistics that are used in the clustering-driven analysis. In the Management Plane, the information from Control and Data Planes are collected and stored. The Management Plane carries out a basic analysis of statistics per flow and per forwarding devices by using the Data Collector from the HH-DANM module. The Data Plane is responsible for generating information by the forwarding network devices. This information is extracted and transmitted to the Management Plane. The Data Plane is formed by the network devices and the HH-DAM module. The following subsections detail the modules aforementioned.

Figure 3. HH Detection approach architecture.

4.2.1. Heavy-Hitters Data Acquisition Module

HH-DAM is responsible for performing two tasks. Firstly, it captures packets from some observation point in the network devices. That means a monitoring task needs to be carried out to collect and report network states in the data plane. Secondly, it generates a dataset for HH identification from the collected packets. In HH-DAM, since we do not implement the monitoring task, we build up the HH-identification dataset from a publicly accessible traffic trace collected in a university DCN, named UNIV1 [64]. UNIV1 was processed and organised into flow records by using the flowRecorder tool with an expiration time (f_{ito}) set to 15 s and 150 s. We wrote this tool in Python to turn IP packets, either in the form of PCAP (Packet CAPture) files or sniffed live from a network interface, into flow records that are stored in a CSV (Comma-Separated Values) file. Our flowRecorder supports the measurement of flow features in both unidirectional and bidirectional modes. Depending on the properties of the observed (incoming) packets, either new flow records were created or the flow features of existing ones were updated.

Tables 3 and 4 show the UNIV1 flows dataset size distributions obtained using flowRecorder with $f_{ito} = 15$ s and $f_{ito} = 150$ s. To get details and use this tool, we invite the reader to review [25].

Table 3. Flows size distribution obtained using $f_{ito} = 15$ s.

Flow Size	TCP	[%]	UDP	[%]	Total	[%]
$f_s \leq 10$ KB	89,261	33.07	180,581	57.2	269,842	85.47
$10 \text{ KB} < f_s \leq 100 \text{ KB}$	40,165	12.72	378	0.12	40,543	12.84
$100 \text{ KB} < f_s \leq 1 \text{ MB}$	4697	1.49	115	0.036	4812	1.52
$1 \text{ MB} < f_s \leq 10 \text{ MB}$	437	0.14	21	0.007	458	0.145
$f_s < 10 \text{ MB}$	43	0.013	3	0.001	46	0.14
Total	134,603	42.63	181,098	57.36	315,701	100

Table 4. Flows size distribution obtained using $f_{ito} = 150$ s.

Flow Size	TCP	[%]	UDP	[%]	Total	[%]
$f_s \leq 10$ KB	74,915	25.48	175,029	59.54	249,944	85.02
$10 \text{ KB} < f_s \leq 100 \text{ KB}$	38,775	13.19	195	0.07	38,970	13.26
$100 \text{ KB} < f_s \leq 1 \text{ MB}$	4446	1.51	86	0.029	4532	1.54
$1 \text{ MB} < f_s \leq 10 \text{ MB}$	457	0.16	17	0.01	474	0.16
$f_s < 10 \text{ MB}$	46	0.02	4	0.0	50	0.02
Total	118,639	40.4	175,331	59.6	293,970	100

Regarding the monitoring task, it is important to highlight that to avoid the SDN controller overload regarding traffic and processing caused by gathering statistics from a central point and aiming at getting better performance in collecting and reporting the network flows state, we plan in a future work to use Inband Network Telemetry (INT) [65] by using the Programmable, Protocol-independent Packet Processor (P4). P4 was created as a common language to describe how packets should be processed by all manner of programmable packet-processing targets, from general-purpose CPUs, NPU, FPGAs, and high-performance programmable ASICs [59]. The main advantages of P4 are: (i) protocol independence meaning that collecting network devices information should be protocol agnostic, (ii) target independence indicating programmers should be able to describe packet-processing functionality independently of the underlying hardware; and (iii) reconfigurability in the field highlighting programmers should be able to change the way switches process packets after their deployment in the network.

4.2.2. Heavy-Hitters Data Acquisition Module

HH-DANM is in charge of storing and generating a network traffic state model targeted to identify HHs smartly. To carry out this module, we performed a clustering-based analysis, on the

UNIV1 dataset, targeted to determine the optimal threshold that would separate the flows into HHs and non-HHs. We performed this analysis since there is no generally accepted and widely recognised uniform threshold for HHs detection. Indeed, different works use different thresholds without detailed or systematic justification. Some examples of such unjustified citation chains include Xu et al. [66], Munir et al. [67] $\Rightarrow$ Hong et al. [68] $\Rightarrow$ Alizadeh et al. [69]; Cui et al. [62] $\Rightarrow$ Wu et al. [70] $\Rightarrow$ Greenberg et al. [71]; and Al-Fares [15] Peng-Xiao et al. [2] $\Rightarrow$ Benson et al. [64]. Cluster analysis belongs to unsupervised ML techniques. It 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. In HH-DANM, we have decided to use K-means mainly because of its simplicity, speed, and accuracy. In addition, several research works report on its high efficacy when deployed on network traffic data [50].

There are several methods to determine the optimal number of clusters that K-means needs to operate, such as V-measure, Adjusted rank Index, V-score, and Homogeneity [72]. However, these methods are usually used with labelled datasets. Since our datasets are not labelled, we have decided to use the Silhouette score that 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 by Bishop [36] and Estrada-Solano [73]. In HH-DANM, we applied the Silhouette method by varying k from 2 to 15. This method uses a coefficient (S_i) that measures how well a point is clustered and estimates the average distance between clusters. The values of S_i range between -1 and 1. The closer the values are to 1, the better the points are clustered. As Figure 4 shows, in our analysis the S_i range is quite wide, with values that vary between 0.6 and the maximum, 0.99. In both $f_{ito} = 15$ s and $f_{ito} = 150$ s, the possible optimal k is between $k = 2$ to $k = 11$.

Figure 4. Silhouette coefficients for UNIV1 with $f_{ito} = 15$ s and $f_{ito} = 150$ s.

Both UNIV1 with $f_{ito} = 15$ s and $f_{ito} = 150$ s, got the highest S_i for $k = 2$. With $k = 2$, there is a significant distance between classes, as Figure 5b,f shows by the Principal Component Analysis (PCA) that performs linear transformations process resulting in primary components [36]. However, despite the promising S_i value and the evident distance between the classes, the imbalance between clusters is also evident, i.e., one cluster contains a large number of flows while the other, extremely few. The specific numbers of this distribution are provided in Table 5. This distribution can produce an important deterioration of the classifier performance in particular with patterns belonging to the less represented classes.

Figure 5. Visualisation of UNIV1 $f_{ito} = 150$ s and $f_{ito} = 15$ s along the two first principal components: (a) UNIV1 $f_{ito} = 150$ s; (b) UNIV1 $f_{ito} = 150$ s, $k = 2$; (c) UNIV1 $f_{ito} = 150$ s, $k = 5$; (d) UNIV1 $f_{ito} = 150$ s, $k = 10$; (e) UNIV1 $f_{ito} = 15$ s; (f) UNIV1 $f_{ito} = 15$ s, $k = 2$; (g) UNIV1 $f_{ito} = 15$ s, $k = 5$; (h) UNIV1 $f_{ito} = 15$ s, $k = 10$.

Table 5. Number of flows per class obtained using $k = 2$.

UNIV1	Class 1	Class 2	Total
$f_{ito} = 15$ s	315,694	7	315,701
$f_{ito} = 150$ s	293,962	8	293,970

Another promising S_i value is for $k = 5$, in which for both datasets it is almost 1. We applied K-means clustering with $k = 5$. Figure 5c,g show PCA with $k = 5$. Our analysis focuses on the relationship between statistical features of the flows, in particular, their size, number of packets, and duration. Table 5 summarises the relationship between them. While $k = 5$ 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 V seem to be outliers. This distance is evident from both Figure 5c,g as well as in Table 6.

Table 6. Flows size and number of packets obtained using $k = 5$ and $f_{ito} = 150$ s and $f_{ito} = 15$ s.

Class	UNIV1	Numb. of Flows	Flow Size (Bytes)			Number of Packets			Flow Duration		
			Max	Min	Avg	Max	Min	Avg	Max	Min	Avg
I	15 s	315,461	1.88×10^6	31	8.27×10^3	28,622	1	8.27×10^3	2643.73	0	2.85
	150 s	293,709	1.88×10^6	32	8.47×10^3	11,394	1	8.47×10^3	2643.73	0	6.67
II	15 s	6	180.23×10^6	137.52×10^6	154.93×10^6	206,026	149,047	154.93×10^6	511.61	151.9	277.6
	150 s	6	194.11×10^6	145.39×10^6	167.37×10^6	281,168	149,047	167.37×10^6	2643.13	151.9	1062
III	15 s	29	34.02×10^6	14.23×10^6	20.20×10^6	75,988	13,136	20.20×10^6	1.962	860.9	1.96
	150 s	32	32.95×10^6	11.81×10^6	18.89×10^6	33,972	11,738	18.89×10^6	2641.51	1.96	361.7
IV	15 s	5	76.86×10^6	42.73×10^6	55.80×10^6	84,413	34,243	55.80×10^6	504.84	8.65	2133
	150 s	6	86.80×10^6	42.73×10^6	60.97×10^6	85,848	34,243	60.97×10^6	2559.51	8.65	508.4
V	15 s	200	11.81×10^6	1.89×10^6	3.77×10^6	79,661	1418	3.77×10^6	2643.89	1.39	201.3
	150 s	217	11.81×10^6	1.89×10^6	3.77×10^6	79,661	1418	3.77×10^6	2643.89	1.39	201.3

We also performed the cluster analysis using $k = 10$. The new clusters seem to appear by splitting the old classes, in particular classes II to V. However, as Figure 5d,h show, the flows belonging to class I seems to keep the same shape and number of flows. This motivated us to analyse class I in detail. The analysis of flows belonging to class I followed the same steps that were applied previously. We have decided to use $k = 5$. Table 7 summarises the results that were obtained. Overall, the results show there is no clear threshold that separates flows into HHs and non-HHs. This is because the flow sizes have a diverse character that leads to more than two natural clusters. We stress that the threshold selection should include a detailed analysis of the network and its traffic. However, a pattern in the clusters I, II, IV, and V regarding flow size and the number of packets is evident, as Table 7 shows. Therefore, we suggest the use of the following thresholds for HH identification in traffic similar to UNIV1, Duque-Torres et al [74]: flow size $\theta_s = 7$ KB and packet count $\theta_{pkt} = 14$.

Table 7. Flows size and number of packets obtained using $k = 5$ for the class I.

Class	UNIV1	Num. of Flows	Flow Size [MB]			Number of Packets			Flow Duration		
			Max	Min	Avg	Max	Min	Avg	Max	Min	Avg
I	15 s	142,117	0.751	0.000031	0.00704	8143	1	14.3	2643.737	0	2.856
	150 s	130,430	0.628	0.000033	0.00683	11,394	1	13.09	2643.73	0	6.76
II	15 s	142,385	0.762	0.000033	0.00692	8143	1	14.2	511.61	151.97	277.65
	150 s	130,910	0.628	0.000032	0.00681	5 364	1	13.68	2643.13	151.97	1061.6
III	15 s	92	9.42	3.61	5.57	81,728	2982	8360	1.962	860.94	1.96
	150 s	162	8.31	2.78	4.33	81,728	2240	6793.8	2641.51	1.96	361.70
IV	15 s	805	3.52	0.74	1.47	27,737	536	1870.05	504.84	8.65	2132.9
	150 s	130,645	0.607	0.000031	0.00673	5908	1	13.6	2559.51	8.65	508.47
V	15 s	142,229	0.73	0.000031	0.007	5908	1	14.09	2643.89	1.39	201.25
	150 s	844	2.77	0.610	1.21	15,390	144	1645.37	2643.89	1.39	201.25

Aiming at corroborating the applicability of our proposal, we evaluated the time that HH-DANM spends to identify the flows in UNIV1 by using the above-defined thresholds. As such, we analysed the time required to get 14 packets and an accuracy of over 96%. Table 8 summarizes the results obtained, provides statistical information about the flow duration for the first 14 packets, and reveals diverse facts. Firstly, the majority of flows can be identified in a short time. In particular, this time is less than 0.9 s for 80% of flows for $f_{ito} = 15$ s and $f_{ito} = 150$ s. Secondly, approximately 95% of flows are classified in a time less than 6 s for $f_{ito} = 15$ s and $f_{ito} = 150$ s. Thirdly, roughly, for $f_{ito} = 15$ s and $f_{ito} = 150$ s, only 4% of flows are identified in a time higher than 16 s. Fourthly, approximately, for $f_{ito} = 15$ s and $f_{ito} = 150$ s, just 2% of flows are identified in a time higher than 23 s. Finally, some flows (worst cases) required up to 480 s to collect the required minimum volume of flow data and/or packets. Considering these facts, we argue that our proposal is applicable in real scenarios. Furthermore, it is possible to get less time by decreasing the accuracy; it is by establishing a trade-off between accuracy and identification time.

Table 8. Flow duration statistics of the first 14 packets in UNIV1.

$f_{ito} = 15$ s				$f_{ito} = 150$ s			
Percentile [%]	[%]	Percentile [%]	[%]	Percentile [%]	[%]	Percentile [%]	[%]
20	0.08	80	0.87	20	0.10	80	0.84
30	0.12	90	2.20	30	0.15	90	1.67
40	0.18	92	4.34	40	0.21	92	2.63
50	0.23	94	5.62	50	0.25	94	4.42
60	0.30	96	16.00	60	0.31	96	15.23
70	0.41	98	21.07	70	0.41	98	22.83
Count		6 122		Count		4 043	
Mean		1.752		Mean		3.666	
Std		5.807		Std		30.116	
Max		157.108		Max		480.636	
Min		0.003		Min		0.004	

Finally, in the evaluation of HH-DANM, we compared its True Positive Rate (TPR) with the provided by classification techniques used by Poupart et al. [75]. In particular, they used Neural Networks (NN), Gaussian Processes Regression (GPR), and Online Bayesian Moment Matching (oBMM) to classify HHs. Table 9 shows the comparison results, revealing that HH-DANM (using the suggested values, flow size $\theta_s = 7$ KB and packet count $\theta_{pkt} = 14$ [76]) achieves similar results to the approaches based on NN, GPR, and oBMM. However, there are some significant considerations. Firstly, unlike GPR and NN based approaches that do not hold their performance when the threshold changes, our approach achieves the same performance regardless of thresholds. Secondly, the approaches based on NN and oBMM tend to be affected by class imbalances more than GPR-based ones, which explains why their accuracy often suffers as the classification threshold increases. In HH-DANM, the class imbalance is not a concern since it uses the same number of packets to identify any flow.

Table 9. True positive rate comparison.

Threshold [KB]	GPR	oBMM	NN	HH-DANM
10	96.1	98.03	97.6	96.2
100	94.6	98.4	96.3	96.03
1000	70.2	98.1	90.4	96.1

4.2.3. Heavy-Hitters Application Module

HH-APM is responsible for sending instructions about what the network (i.e., the SDN controller) needs to do, when a HH or non-HH is identified. Once the HHs are identified, the Controller instructs

the forwarder devices to do a variety of network managerial related activities aiming at improving the overall network performance. To carry out HH-APM and routing the non-HHs identified, we propose the use of MiceDCER (Mice Data Center Efficient Routing) [35].

MiceDCER is an algorithm, proposed by our research group in a previous work [35], that efficiently routes non-HHs in SDDCN by assigning internal Pseudo-MAC (PMAC) addresses to the ToR switches and hosts. MiceDCER generates the PMAC of the switches that received the flow intercepted by the controller based on their position in the topology. PMACs are stored in a table that associates them with the corresponding actual MAC (Media Access Control) address of each switch. Also, MiceDCER reduces the number of routing rules by installing wildcard rules based on the information carried by the ARP packets. In a high-abstraction level, MiceDCER-based HH-APM performs three significant procedures: *Generation of initial rules for the edge switches*, *Intercepted message management*, and *Generation of table entries*.

Figure 6 shows the flowcharts of each procedure performed by MiceDCER. The *Generation of initial rules for the edge switches* is carried out by installing routing rules for the edge switches with the ARP field type. These initial rules allow the controller to intercept the ARP messages that arrive at the switch. To install rules on the switch tables, the algorithm performs the procedure *Intercepted message management*. In this procedure, a verification is carried out about the intercepted IP destination address of the ARP request message. If the controller does not recognise such address, then the controller instructs the switch that received the message intercepted to flood and instructs the other edge switches to flood with requests. If the controller recognises the IP address, it sends a reply ARP message back to the source host.

Figure 6. Flowcharts of the three major procedures performed by MiceDCER.

The *Generation of table entries* procedure has two major processes. Firstly, it verifies if the source IP address of the intercepted message is stored in the host PMACs table. Secondly, if the source IP (Internet Protocol) address is not stored, then this procedure proceeds to generate the PMAC and save it into the table, associating it with the source IP address. Finally, it is important to mention that MiceDCER asks the controller for updating the defined rules and generating new PMACs when topology modifications occur in the network.

Aiming at evaluating our MiceDCER-based HH-APM, MiceDCER was compared with IP-based and MAC-based routing. The evaluation was performed in a FatTree topology that consists of p pods, which are management units interconnected by the $(p/2)^2$ switches that make up the core layer. Each pod consists of $p/2$ edge switches and $p/2$ aggregate switches, and each ToR switch, connects to $p/2$ end hosts [63]. Therefore, a p -ary FatTree topology supports $p^3/4$ hosts.

Table 10 summarizes the number of rules generated by MiceDCER, IP-based and MAC-based routing when p is 16, 20, 24, 28, 32, 36, 40, and 48. The evaluation results reveal that, in the edge

layer, MAC-based and IP-based routing install more rules than our HH-APM running MiceDCER. Considering these results, we can conclude that MiceDCER reduces the number of rules per edge switch significantly when compared with other routing solutions. The results also reveal that, in the aggregate layer, MAC-based routing installs much more rules than MiceDCER. The IP-based routing installs approximately the double of rules than MiceDCER does. Thus, we can conclude that MiceDCER reduces the number of rules per aggregate switch significantly when compared with the MAC-based and IP-based routing. In the core switch layer, the results expose again that MAC-based routing installs more rules than MiceDCER. In turn, the IP-based routing installs about the same amount of rules as MiceDCER. We can conclude that MiceDCER reduces or at least generates the same number of routing rules to install in the core switches. For more information about MiceDCER, its design, implementation, and evaluation, we refer the reader to Amezcua-Suarez [35].

Table 10. Number of rules generated by MiceDCER, IP-based and MAC -based routing

p	Num. of Host	MiceDCER			IP-Based			MAC-Based		
		ES	AS	CS	ES	AS	CS	ES	AS	CS
16	1024	38	13	19	53	19	18	1365	512	512
20	2000	48	15	23	55	23	22	2666	1000	1000
24	3456	54	17	27	57	27	26	4608	1728	1728
28	5488	62	19	31	58	31	30	7317	2744	2744
32	8192	70	21	35	61	35	34	10,922	4096	4096
36	11,664	78	23	39	63	39	38	15,552	5832	5832
40	16,000	86	25	43	65	43	42	21,333	8000	8000
48	27,648	102	29	51	69	51	50	36,864	13,824	13,824

5. Conclusions

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 we discussed in the previous sections, HH detection in current SDDCN environment is still challenging, especially concerning traffic flow statistics collection and threshold estimation. Aiming at overcoming such challenges, we proposed a novel HHs identification approach based on the KDN concept. This concept takes advantage of ML techniques in SDN. In particular, this paper, firstly, presents a clear understanding of KDN concept. Secondly, it introduces the approach proposed for Heavy-Hitters identification as well as details and evaluates its modules. In the Heavy-Hitters Data Analyser module, we performed a cluster analysis using K-means for clustering the flows. We employed Silhouette analysis to determine the optimal number of clusters. Based on the obtained results, there is no single consistent threshold that separates flows into HHs and non-HHs. The flow sizes have a diverse character that leads to more than two natural clusters. We stress that threshold selection must include a detailed analysis of the network and its kind of traffic. In the Heavy-Hitters Application module, we present MiceDCER, an algorithm that efficiently routes non-HHs by assigning internal PMAC addresses to the edge switches and hosts. Our evaluation reveals MiceDCER significantly reduces the number of rules installed in switches and, therefore, contributes to reducing the delay in SDDCN. To sum up, the per-module evaluation results corroborated the usefulness and feasibility of our approach for identifying HHs.

As future work, we intend to perform non-threshold-based HHs identification. In this sense, we plan to offer a solution based on a per-flow packet size distribution for predicting in an early flow stage if it will be a HH.

Author Contributions: The authors contributed equally to this manuscript.

Funding: The authors would like to thank the Universidad del Cauca, Fundación Universitaria de Popayán, and Universidad del Quindío.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Faizul Bari, M.; Boutaba, R.; Esteves, R.; Granville, L.; Podlesny, M.; Rabbani, M.; Zhang, Q.; Zhani, M.F. Data Center Network Virtualization: A Survey. *IEEE Commun. Surv. Tutor.* **2013**, *15*, 909–928. [CrossRef]
2. Peng, X.; Wenyu, Q.; Heng, Q.; Yujie, X.; Zhiyang, L. An efficient elephant flow detection with cost-sensitive in SDN. In Proceedings of the 1st International Conference on Industrial Networks and Intelligent Systems, Tokyo, Japan, 2–4 March 2015; pp. 24–28.
3. Van Asten, B.J.; van Adrichem, N.L.M.; Kuipers, F.A. Scalability and Resilience of Software-Defined Networking: An Overview. *Comput. Commun.* **2014**, *67*, 1–19.
4. The Open Networking Foundation. OpenFlow Switch Specification Version 1.5.1 (Protocol Version 0x06). ONF TS-025. March 2015. Available online: <https://www.opennetworking.org/wp-content/openflow-switch-v1.5.1.pdf> (accessed on 2 October 2019).
5. Yang, L.; Dantu, R.; Anderson, T.; Gopal, R. *Forwarding and Control Element Separation (ForCES) Framework*; RFC Editor: Dallas, TX, USA, 2004.
6. Yassine, A.; Rahimi, H.; Shirmohammadi, S. Software defined network traffic measurement: Current trends and challenges. *IEEE Instrum. Meas. Mag.* **2015**, *18*, 42–50. [CrossRef]
7. Awduche, D.; Chiu, A.; Elwalid, A.; Widjaja, I.; Xiao, X. Overview and Principles of Internet Traffic Engineering. In Proceedings of the 21th IEEE International Conference on Computer Communications Workshops, New York, NY, USA, 23–27 June 2002; pp. 78–82, 357–362.
8. Benson, T.; Anand, A.; Akella, A.; Zhang, M. Microte: Fine grained traffic engineering for data centers. In Proceedings of the 7th Conference on Emerging Networking Experiments and Technologies, Tokyo, Japan, 6–9 December 2011; pp. 1–8.
9. Callado, A.; Kamienski, C.; Szabo, G.; Gero, B.P.; Kelner, J.; Fernandes, S.; Sadok, D. A Survey on Internet Traffic Identification. *IEEE Commun. Surv. Tutor.* **2009**, *11*, 37–52. [CrossRef]
10. Pekár, A.; Chovanec, M.; Vokorokos, L.; Chovancová, E.; Fecil’ak, P.; Michalko, M. Adaptive Aggregation of Flow Records. *Comput. Inform.* **2018**, *37*, 142–164. [CrossRef]
11. Sarvotham, S.; Riedi, R.; Baraniuk, R. Connection-level Analysis and Modeling of Network Traffic. In Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement, Burlingame, CA, USA, 1–2 November 2001; pp. 99–103.
12. Vokorokos, L.; Pekar, A.; Adam, N. Data preprocessing for efficient evaluation of network traffic parameters. In Proceedings of the 16th IEEE International Conference on Intelligent Engineering Systems, Lisbon, Portugal, 13–15 June 2012; pp. 363–367.
13. Pekar, A.; Chovancova, E.; Fanfara, P.; Trelova, J. Issues in the passive approach of network traffic monitoring. In Proceedings of the 17th IEEE International Conference on Intelligent Engineering Systems, San Jose, Costa Rica, 19–21 June 2013; pp. 327–332.
14. Hayes, M.; Ng, B.; Pekar, A.; Seah, W.K.G. Scalable Architecture for SDN Traffic Classification. *IEEE Syst. J.* **2018**, *99*, 1–12. [CrossRef]
15. Al-Fares, M.; Radhakrishnan, S.; Raghavan, B.; Huang, N.; Vahdat, A. Hedera: Dynamic Flow Scheduling for Data Center Networks. In Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation; USENIX Association, San Jose, CA, USA, 28–30 April 2010; pp. 19–25.
16. Farrington, N.; Porter, G.; Radhakrishnan, S.; Bazzaz, H.H.; Subramanya, V.; Fainman, Y.; Papen, G.; Vahdat, A. Helios: A hybrid electrical/optical switch architecture for modular data centers. *ACM SIGCOMM Comput. Commun. Rev.* **2010**, *40*, 339. [CrossRef]
17. Liu, Z.; Gao, D.; Liu, Y.; Zhang, H.; Foh, C.H. An adaptive approach for elephant flow detection with the rapidly changing traffic in data center network. *Int. J. Netw. Manag.* **2017**, *27*, e1987. [CrossRef]
18. Bi, C.; Luo, X.; Ye, T.; Jin, Y. On precision and scalability of elephant flow detection in data center with SDN. In Proceedings of the 32th IEEE Global Communications Conference Workshops, Atlanta, GA, USA, 9–13 December 2013; pp. 1227–1232.

19. Lin, C.; Chen, C.; Chang, J.; Chu, Y.H. Elephant flow detection in datacenters using OpenFlow-based Hierarchical Statistics Pulling. In Proceedings of the 33rd IEEE Global Communications Conference, Austin, TX, USA, 5–7 December 2014; pp. 2264–2269.
20. Moshref, M.; Yu, M.; Govindan, R. Resource/Accuracy Tradeoffs in Software-defined Measurement. In Proceedings of the 2nd ACM SIGCOMM Workshop on hot topics in software defined networking, Hong Kong, China, 12–16 August 2013; pp. 73–78.
21. Carpio, F.; Engelmann, A.; Jukan, A. DiffFlow: Differentiating Short and Long Flows for Load Balancing in Data Center Networks. In Proceedings of the 35th IEEE Global Communications Conference, Washington, DC, USA, 4–8 December 2016; pp. 1–6.
22. Curtis, A.R.; Kim, W.; Yalagandula, P. Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection. In Proceedings of the 30th IEEE International Conference on Computer Communications, Shanghai, China, 10–15 April 2011; pp. 1629–1637.
23. Liu, R.; Gu, H.; Yu, X.; Nian, X. Distributed Flow Scheduling in Energy-Aware Data Center Networks. *IEEE Commun. Lett.* **2013**, *17*, 801–804. [[CrossRef](#)]
24. Trestian, R.; Muntean, G.; Katrinis, K. MiceTrap: Scalable traffic engineering of datacenter mice flows using OpenFlow. In Proceedings of the 21st IFIP/IEEE International Symposium on Integrated Network Management, Ghent, Belgium, 27–31 May 2013; pp. 904–907.
25. “FlowRecorder. A Network Traffic Flow Feature Measurement Tool” [Online]. Available online: <https://github.com/drnprkr/flowRecorder2018> (accessed on 25 November 2018).
26. Aitken, P.; Claise, B.; Trammell, B. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. In RFC 7011 (INTERNET STANDARD), Internet Engineering Task Force; Cisco Systems, Inc.: San Jose, CA, USA, 2013. [[CrossRef](#)]
27. Brownlee, N.; Claffy, K.C. Understanding internet traffic streams: Dragonflies and tortoises. *IEEE Commun. Mag.* **2002**, *40*, 110–117. [[CrossRef](#)]
28. Lan, K.C.; Heidemann, J. A Measurement Study of Correlations of Internet Flow Characteristics. *Comput. Netw.* **2006**, *50*, 46–62. [[CrossRef](#)]
29. Smith, R.D. The dynamics of internet traffic: Self-similarity, Self-organization, and Complex phenomena. *Adv. Complex Syst.* **2011**, *14*, 905–949. [[CrossRef](#)]
30. Montoya-Munoz, A.I.; Casas-Velasco, D.M.; Estrada-Solano, F.; Ordonez, A.; Rendon, O.M.C. A YANG model for a vertical SDN management plane. In Proceedings of the IEEE Colombian Conference on Communications and Computing (COLCOM), Cartagena, Colombia, 16–18 August 2017; pp. 1–6.
31. Wickboldt, J.A.; de Jesus, W.P.; Isolani, P.H.; Both, C.B.; Rochol, J.; Granville, L.Z. Software-defined networking: Management requirements and challenges. *IEEE Commun. Mag.* **2015**, *53*, 278–285. [[CrossRef](#)]
32. Estrada-Solano, F.; Ordonez, A.; Granville, L.Z.; Rendon, O.M.C. A framework for SDN integrated management based on a CIM model and a vertical management plane. *Comput. Commun.* **2017**, *102*, 150–164. [[CrossRef](#)]
33. Kreutz, D.; Ramos, F.M.V.; Verissimo, P.E.; Rothenberg, C.E.; Azodolmolky, S.; Uhlig, S. Software-Defined Networking: A Comprehensive Survey. *Proc. IEEE* **2015**, *103*, 14–76. [[CrossRef](#)]
34. Mestres, A.; Rodriguez-Natal, A.; Carner, J.; Barlet-Ros, P.; Alarcón, E.; Solé, M.; Muntés-Mulero, V.; Meyer, D.; Barkai, S.; Hibbett, M.J.; et al. Knowledge-Defined Networking. *SIGCOMM Comput. Commun. Rev.* **2017**, *47*, 2–10. [[CrossRef](#)]
35. Amezcua-Suarez, F.; Estrada-Solano, F.; da Fonseca, N.L.S.; Rendon, O.M.C. An Efficient Mice Flow Routing Algorithm for Data Centers Based on Software-Defined Networking. In Proceedings of the IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–6.
36. Bishop, C.M. *Pattern Recognition and Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2006.
37. Sun, Y.; Yin, X.; Jiang, J.; Sekar, V.; Lin, F.; Wang, N.; Liu, T.; Sinopoli, B. Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction. In Proceedings of the conference on ACM SIGCOMM, ACM, Florianopolis, Brazil, 22–26 August 2016; pp. 272–285.
38. Chen, Z.; Wen, J.; Geng, Y. Predicting future traffic using Hidden Markov Models. In Proceedings of the IEEE 24th International Conference on Network Protocols, Singapore, 8–11 November 2016; pp. 1–6.
39. Niyaz, Q.; Sun, W.; Javaid, A.Y. A Deep Learning Based DDoS Detection System in Software-Defined Networking (SDN). *arXiv* **2016**, arXiv:1611.07400. [[CrossRef](#)]

40. Namdev, N.; Agrawal, S.; Silkari, S. Recent advancement in machine learning based internet traffic classification. *Proc. Comput. Sci.* **2015**, *60*, 784–791. [[CrossRef](#)]
41. Zhang, J.; Chen, X.; Xiang, Y.; Zhou, W.; Wu, J. Robust network traffic classification. *IEEE/ACM Trans. Netw.* **2015**, *23*, 1257–1270. [[CrossRef](#)]
42. Dong, M.; Li, Q.; Zarchy, D.; Godfrey, P.B.; Schapira, M. PCC: Re-architecting Congestion Control for Consistent High Performance. In Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation, Oakland, CA, USA, 4–6 May 2015; pp. 395–408.
43. Krupitzer, C.; Roth, F.M.; Vansyckel, S.; Schiele, G.; Becker, C. A survey on engineering approaches for self-adaptive systems. *Pervasive Mob. Comput.* **2015**, *17*, 184–206. [[CrossRef](#)]
44. Ayoubi, S.; Limam, N.; Salahuddin, M.A.; Shahriar, N.; Boutaba, R.; Estrada-Solano, F.; Caicedo, O.M. Machine Learning for Cognitive Network Management. *IEEE Commun. Mag.* **2018**, *56*, 158–165. [[CrossRef](#)]
45. Boutaba, R.; Salahuddin, M.A.; Limam, N.; Ayoubi, S.; Shahriar, N.; Estrada-Solano, F.; Caicedo, O.M. A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities. *J. Internet Serv. Appl.* **2018**, *9*, 16. [[CrossRef](#)]
46. Nguyen, T.T.; Armitage, G. A survey of techniques for internet traffic classification using machine learning. *IEEE Commun. Surv. Tutor.* **2008**, *10*, 56–76. [[CrossRef](#)]
47. Sadashiv, N.; Kumar, S.M.D. Cluster, grid and cloud computing: A detailed comparison. In Proceedings of the 6th International Conference on Computer Science and Education, Singapore, 3–5 August 2011; pp. 477–482.
48. Bermolen, P.; Rossi, D. Support vector regression for link load prediction. In Proceedings of the 4th International Telecommunication Networking Workshop on QoS in Multiservice IP Networks, Venice, Italy, 13–15 February 2008; pp. 268–273.
49. Lakhina, A.; Crovella, M.; Diot, C. Mining Anomalies Using Traffic Feature Distributions. *SIGCOMM Comput. Commun. Rev.* **2005**, *35*, 217–228. [[CrossRef](#)]
50. Zhang, J.; Xiang, Y.; Zhou, W.; Wang, Y. Unsupervised traffic classification using flow statistical properties and IP packet payload. *J. Comput. Syst. Sci.* **2013**, *79*, 573–585. [[CrossRef](#)]
51. Iqbal, W.; Dailey, M.N.; Carrera, D. Policies for Cloud-Hosted Multitier Web Applications. *IEEE Syst. J.* **2016**, *10*, 1435–1446. [[CrossRef](#)]
52. Chapelle, O.; Schölkopf, B.; Zien, A. *Semi-Supervised Learning*, 1st ed.; The MIT Press: Cambridge, MA, USA, 2010.
53. Lin, F.; Cohen, W.W. Semi-Supervised Classification of Network Data Using Very Few Labels. In Proceedings of the International Conference on Advances in Social Networks Analysis and Mining, IEEE Computer Society, Odense, Denmark, 9–11 August 2010; pp. 192–199.
54. Erman, J.; Mahanti, A.; Arlitt, M.; Cohen, I.; Williamson, C. Semi-supervised Network Traffic Classification. *SIGMETRICS Perform. Eval. Rev.* **2007**, *35*, 369–370. [[CrossRef](#)]
55. Shrivastav, A.; Tiwari, A. Network Traffic Classification Using Semi-Supervised Approach. In Proceedings of the 2nd International Conference on Machine Learning and Computing, Bangalore, India, 9–11 February 2010; pp. 345–349.
56. Bazzan, A.L. Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. *Auton. Agents Multi Agent Syst.* **2009**, *18*, 342–375. [[CrossRef](#)]
57. Sun, R.; Tatsumi, S.; Zhao, G. Q-MAP: A novel multicast routing method in wireless ad hoc networks with multiagent reinforcement learning. In Proceedings of the IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering, Beijing, China, 28–31 October 2002; pp. 667–670.
58. Clark, D.D.; Partridge, C.; Ramming, J.C.; Wroclawski, J.T. A knowledge plane for the internet. In Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Karlsruhe, Germany, 25–29 August 2003; p. 3.
59. Hyun, J.; Hong, J.W.K. Knowledge-defined networking using in-band network telemetry. In Proceedings of the 19th Asia-Pacific Network Operations and Management Symposium, Seoul, Korea, 27–29 September 2017; pp. 54–57.
60. Dias Knob, L.A.; Esteves, R.P.; Granville, L.Z.; Tarouco, L.M.R. SDEFIX—Identifying elephant flows in SDN-based IXP networks. In Proceedings of the IEEE/IFIP Network Operations and Management Symposium, Istanbul, Turkey, 25–29 April 2016; pp. 19–26.
61. Xia, J.B.; Ren, G.M. Survey on elephant flow identifying methods. *Control Decis.* **2013**, *6*, 801–807.

62. Cui, W.; Ye, Y.; Qian, C. DiFS: Distributed Flow Scheduling for adaptive switching in FatTree data center networks. *Comput. Netw.* **2016**, *105*, 166–179. [[CrossRef](#)]
63. Afaq, M.; Rehman, S.U.; Song, W.C. A Framework for Classification and Visualization of Elephant Flows in SDN-Based Networks. *Procedia Comput. Sci.* **2015**, *65*, 672–681. [[CrossRef](#)]
64. Benson, T.; Akella, A.; Maltz, D.A. Network Traffic Characteristics of Data Centers in the Wild. In Proceedings of the 10th Conference on Internet Measurement, ACM, Melbourne, VIC, Australia, 1–3 November 2010; pp. 267–280.
65. Kim, C.; Sivaraman, A.; Katta, N.; Bas, A.; Dixit, A.; Wobker, L.J. In-band network telemetry via programmable dataplanes. In Proceedings of the Symposium on SDN Research, Santa Clara, CA, USA, 17–18 June 2015.
66. Xu, H.; Li, B. RepFlow: Minimizing flow completion times with replicated flows in data centers. In Proceedings of the IEEE INFOCOM, Toronto, ON, Canada, 27 April–7 May 2014; pp. 1581–1589.
67. Munir, A.; Qazi, I.A.; Uzmi, Z.A.; Mushtaq, A.; Ismail, S.N.; Iqbal, M.S.; Khan, B. Minimizing flow completion times in data centers. In Proceedings of the IEEE INFOCOM, Turin, Italy, 14–19 April 2013; pp. 2157–2165.
68. Hong, C.Y.; Caesar, M.; Godfrey, P.B. Finishing Flows Quickly with Preemptive Scheduling. In Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, ACM, Helsinki, Finland, 13–17 August 2012; pp. 127–138.
69. Alizadeh, M.; Greenberg, A.; Maltz, D.A.; Padhye, J.; Patel, P.; Prabhakar, B.; Sengupta, S.; Sridharan, M. Data center TCP (DCTCP). *SIGCOMM Comput. Commun. Rev.* **2010**, *41*, 63–74. [[CrossRef](#)]
70. Wu, X.; Yang, X. DARD: Distributed adaptive routing for datacenter networks. In Proceedings of the International Conference on Distributed Computing Systems, Macau, China, 18–21 June 2012; pp. 32–41.
71. Greenberg, A.; Hamilton, J.R.; Jain, N.; Kandula, S.; Kim, C.; Lahiri, P.; Maltz, D.A.; Patel, P.; Sengupta, S. VL2: A Scalable and Flexible Data Center Network. *SIGCOMM Comput. Commun. Rev.* **2009**, *39*, 51–62. [[CrossRef](#)]
72. Liu, Y.; Li, Z.; Xiong, H.; Gao, X.; Wu, J. Understanding of internal clustering validation measures. In Proceedings of the IEEE International Conference on Data Mining, Sydney, NSW, Australia, 13–17 December 2010; pp. 911–916.
73. Estrada-Solano, F.; Caicedo, O.M.; Da Fonseca, N.L.S. NELLY: Flow Detection Using Incremental Learning at the Server-Side of SDN-based Data Centers. *IEEE Transactions on Industrial Informatics*. **2019**, *9*, 16. [[CrossRef](#)]
74. Duque-Torres, A.; Pekar, A.; Seah, W.K.G.; Rendon, O.M.C. Clustering-based Analysis for Heavy-Hitter Flow Detection. In Proceedings of the Asia Pacific Regional Internet Conference on Operational Technologies (APRICOT), Daejeon, Korea, 18–28 February 2019.
75. Poupart, P.; Chen, Z.; Jaini, P.; Fung, F.; Susanto, H.; Geng, Y.; Chen, L.; Chen, K.; Jin, H. Online flow size prediction for improved network routing. In Proceedings of the 24th IEEE International Conference on Network Protocols, Singapore, 8–11 November 2016; pp. 1–6.
76. Duque-Torres, A.; Pekar, A.; Seah, W.K.G.; Rendon, O.M.C. Heavy-Hitter Flow Identification in Data Centre Networks Using Packet Size Distribution and Template Matching. In Proceedings of the 44th IEEE Conference on Local Computer Networks (LCN), Osnabrück, Germany, 14–17 October 2019; pp. 1–8.

© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).