Survey of SDN traffic flow classification approaches

UDAY DESHPANDE[1]
RAJESH N[2]
SHASHANK DHANANJAYA[3]

National Institute of Engineering,
Department of Information Science and Engineering,
Manandavadi Road, Mysuru, Karnataka, India PIN - 570008
[1]4NI19SCN_A@NIE.AC.IN
[2]https://nie.ac.in/faculty/rajesh-n/
[3]https://nie.ac.in/faculty/shashank-dhananjaya/

Abstract. Software-Defined Networking (SDN) is a flexible architecture approach to networking, offering versatility in the degree of control operators may choose to have of their networks, be it with custom-made off the shelf components or customized made-to-order network elements [47]. As the demand for cloud based applications are on the rise, SDNs also need to undergo modifications to suit the hosted application needs on a dynamic time frame. A key component of managing these changes is effective Traffic Engineering (TE) of the flows typically experienced in normal operational scope of the data center network (DCN). Primary interest of an operator would be to ensure SLA adherence of all applications hosted in the network. This would include effective classification of application flows and their appropriate routing, so that there is no loss of data and connectivity. In this paper, a survey of various mechanisms for traffic flow classification has been summarized with an aim to provide an overview for the beginners in SDN TE research area. The approaches are also evaluated against multi-dimensional parameters to aid operators in opting for the best possible TE approach according to their network needs. Recent advances in switch software and hardware development have also been considered to provide a holistic overview to researchers in this field.

Keywords: SDN, Traffic Engineering, Flow Classification.

(Received May 1st, 2021 / Accepted May 31st, 2021)

1 Introduction

SDN paradigm has been an evolutionary force in serving the demand for massive-scale datacenters. It is meeting the need for more efficient and intelligent network management systems. SDN decouples the packet forwarding mechanism (data plane) from the controller decisions (control plane) [18]. With this decoupling, SDN provides an open programmable interface between the control plane and the data plane, allowing application development for dynamically controlling and managing connectivity amongst network elements.

Lan et.al. [24] provides the taxonomy and nomenclature for analysing traffic patterns in datacenters. Traffic flows are classified based on their throughput size (Elephant or Mice), duration (Tortoise or Dragonfly), rate (Cheetah or Snail) and burstiness (Porcupines or Stingrays) rodriguez2012quality,rodriguez2016video. For the purpose of this survey, only throughput size classification of flows is considered. Certain transactions to applications (e.g., web search, DB commits) are made from multiple endpoints that require timely delivery to satisfy the application SLA requirements. These generate several small traffic flows (Mice). Mice flows are usually associated with delay-sensitive and bursty applications, such as Voice over IP. On the other hand, bulk transfers (e.g. MapReduce, FTP, 4K video streaming) utilize high and sustained network bandwidth (Elephants) while require minimal packet delivery delays [20, 44, 45, 4, 43, 42, 2, 3, 36, 41, 30]. Past studies on live datacenter traffic patterns [23, 8] indicate that the number of elephant flows is less than 10% of all flows flowing within the DCN but accounts to 80% of the payload volume. These large flows tend to increase load on the switch buffers, thereby inducing latency to mice flows that share the same buffers [23].On the other hand, due to their relatively short size payloads, Mice flows are short-lived, but they represent 90% of the entire flows [23, 8]. These flows usually trigger the continuous updating of switching tables, which may lead to increased overheads. This ecosystem of mice and elephant traffic flow patterns presents a DCN TE challenge. Every network switch has limited storage space for storing forwarding rules (i.e., flow entries). It is very probable that the switch may not have enough storage [16, 37, 22, 52, 55, 31, 40, 39, 38] to accommodate additional rules without evicting 'stale' entries, especially when a burst of mice packets arrive at a switch in midst of an ongoing elephant flow [8]. This process will lead to additional overhead packet delays to all flows currently handled by that switch.

Several approaches deal with the optimizing of mice and elephant flowsŕouting in SDN-based DCNs. Wang et.al. [56] have collated an useful survey on the various Elephant Flow detection methods. However, the survey is limited in two areas (i) not covering for researches on Mice flow detection and (ii) not evaluating flows against various network parameters. This paper aims to expand the survey to cover these two areas. Researchers primarily focus on Elephant flows, and in many cases Mice flows are not just of secondary importance, but left to the default options available in the network elements. The paper attempts to provide details on this importance, as well as provide a view of multiple researchers who have focused on prioritized classification both the flow types, leading to better flow control in the DCNs. In addition, recent methodologies and approaches are summarized. The rest of this paper is organized as follows: Section 2 describes the background of SDN, OpenFlow Switch, flows in DCNs and need of classification and routing. Sections 3, 4 and 5 lists various classification approach categories based on: (i) flow-characteristics (ii) flow statistics and (iii) Hardware-software modification. Section 6 provides the comparison of various approaches and the various issues of existing approaches and concludes the paper.

2 Background

In this section, terminologies relevant to DCN setup and operations are presented. Table 1 provides a list of abbreviations used throughout the paper. Specific discussions on SDN Architecture, the OpenFlow protocol, methodologies of flow statistics collection and a summary of traffic flow scheduling are provided in subsections 2.1, 2.2, 2.3 and 2.4 respectively. This background is essential for understanding the various approaches listed in this survey.

Table 1: List of Abbreviations

Acronym	*Description*
DCN	Data Center Network
ECMP	Equal Cost Multi Path
FCT	Flow Control Table
ML	Machine Learning
NE	Network Element
SDN	Software Defined Networking

2.1 SDN Architecture

SDN [18] is go-to network system paradigm for DCNs. The network functionality is separated into the data plane and control plane. The switches/routers that forward traffic form the data plane while the software logic that that decides how the traffic flow should be handled through these devices forms the control plane. This segregation enables a system administrator to control network behaviour from a single high-level control program. In a DCN, deployment of SDN helps manage the variety of network problems, as well as enabling granular level control or analysis of packets that pass through the switches. The controller in an SDN based network manages network functions such as routing, encryption, firewall, gateways, Network Address Translation (NAT), Deep Packet Inspection (DPI) etc.

As seen in Fig 1, the data plane traffic is limited to switches which conform to routing functionality by referencing the routing tables. Control plane traffic involves queries from switches (north bound), instructions and updates (south bound) between the controllers and the switches take place regularly, as well as communication between controllers (east-west bound). OpenFlow based switches transmit the request to the controller, and the controller receives the request and applies the action on those packets. In Fig. 1, there is a distinct dedicated connection between every SDN-enabled switch and the controller. This is typical of DCNs where the physical distance between controller and devices is small.

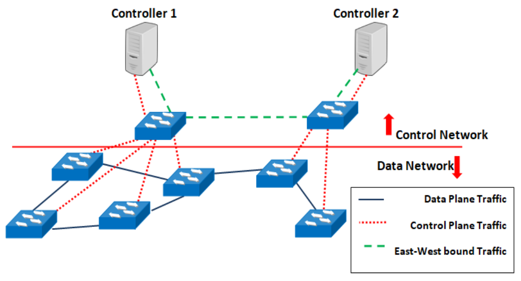

Figure 1: SDN Architecture

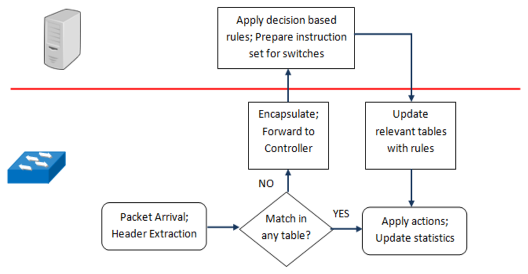
Apply decision based rules; Prepare instruction set for switches
Encapsulate; Forward to Controller
Update relevant tables with rules
NO
Packet Arrival; Header Extraction
Match in any table?
YES
Apply actions; Update statistics

Figure 2: Simplified OpenFlow workflow

2.2 OpenFlow Protocol

SDN-enabled devices use OpenFlow protocol for communication [49] between a switch and a controller. OpenFlow is a communication protocol defined between an SDN controller and OpenFlow switches, allowing the direct access (via user APIs) to the data plane of network elements and the packets' routing path through those network elements. ONF [18] provides the OpenFlow Switch Specification [19], based on which a group table and one or more flow tables are defined in an OpenFlow switch to perform packet forwarding. With OpenFlow, flow entries can be added, modified or deleted either reactively (i.e., as a response to incoming packets) or proactively (i.e., in response to the flow entry timeout) [19].

Fig 2 depicts the simplified OpenFlow workflow at the switch. Packets are routed in the network as per the rules in the respective switch flow tables. In case of an unknown packet, it is sent to the controller for verification. Multiple components such as match fields, priority, counters, instructions, timeouts, cookies, and flags are employed in a flow entry during the packet matching process. These fields and prioritized components are tupled to uniquely identify each entry in a specific flow table. This includes operations such as forwarding the packet to the controller, dropping the packet, or passing it to another table [19]. Decision making rules are applied at the controller end to determine the final destination of the packet (Drop the packet or Forward it)

As switches forward packets based on flow entries in an SDN, OpenFlow uses PacketIn, FlowMod, PacketOut and FlowRemoved control messages to create modify and delete flow entries. The PacketIn message mainly implements the function of sending the packet to the controller. This is done usually under 2 conditions (i) packet rules state that it must be sent to controller or (ii) no flow entry in the tables. When the controller receives a PacketIn message, it will create a corresponding flow entry for the flow which the packet belongs to. This generated flow entry can be added to the target switch through the FlowMod message. Along with the add functionality, FlowMod can be used to modify or delete a flow entry. Buffered packets will usually be processed via a Packet-out or Flow-mod message from the controller. If the flow is not active after the idle timeout, or the duration of flow entry exceeds hard timeout, the switch will delete the corresponding flow entry and send a FlowRemoved message to the controller [19].

2.3 Flow statistics Collection

In order for the controller to accurately obtain the network working status with minimum possible time lag, OpenFlow provides flexible mechanisms for management information collection. Flow statistics Collection is of two kinds (i) Active involving request and response messages and (ii) Passive involving triggering and FlowRemoved messages [19] In active flow statistics collection, when the controller needs to query the flow statistics of the target flow, it sends a flow-statistics Request message to the relevant switch. There are two types of Requests messages (i) individual request, where the request is treated as a standalone request needing a standalone response or (ii) aggregate request, where the request will be part of an ongoing response aggregation and is sent after a given time interval. The switch will send back a relevant flow-statistics Response message back to the requesting Controller.

In passive flow statistics collection, there is a threshold limit set by the controller on the number of flow entries in a switch. When the flow-statistics reaches the specified threshold of the corresponding flow entries, the switch will trigger a flow-statistics triggering message to the controller. In addition, when the FlowRemoved messages are triggered by the switch, the collector gets the relevant flow-statistics information of that entry. These help realize a low cost flow-statistics collection at the controller.

2.4 Traffic Flow scheduling in SDN based DCNs

DCNs typically have deployment of a large number of network policy rules in order to provide quality network and infrastructure services to individual users and cloud applications. These policies are enforced by a collection of Quality-of-Service (QoS) rule-sets such as network security and traffic engineering. Depending on the scale of the DCN, the number of rules may reach hundreds or thousands on a busy schedule. A switch accumulating too many routing rules will lead to scalability issues as it increases the search time that it takes to query the specific flow rule. Such accumulation in time may lead to the dropping of time constrained mice flows.

Thus, in distributed systems, flow scheduling approach is crucial to allocate bandwidth to appropriate switches. Currently, most datacenters use Equal-cost multi-path (ECMP) algorithm is a standard routing approach that takes flows along equivalent paths [64]. ECMP uses static rules by dynamically compiling and installing paths to the elephants and routing of mice. But if there are dynamic changes in the network state, this approach has a few shortcomings i.e. (i) Intricacy of repeated update of the routing rules, (ii) Redundant path problem, (iii) Uneven flow sizes (more mice and lesser elephants) and (iv) Large number of point-to-point routing updates.

Although SDN-enabled DCNs can utilize networking resources on-the-go, it is challenging to determine which forwarding rules should remain in the switch memory and which should be controller processed. The objective here is to minimize information exchanged between the switches and the controller [49]. The increase of various machine learning techniques has motivated networking researchers to apply these techniques in networks. With the data available through analysis, the controller is better enabled to automatically determine which forwarding rules should ideally remain long-term inside the switch memory. Thus, overall network control overhead is reduced to a large extent irrespective of the DCN traffic pattern. With proactive sampling, logging, and monitoring of the incoming packets in an SDN, the controller routing logic is augmented by various machine learning approaches which utilize the flow-characteristics and assign a suitable route for the flows requested by the switch, discussed in the next section.

3 Approaches to flow-characteristics based detection

The flows, through the flow classification, are detected according to markings or features extracted from the target flow. In the network, mice flows are characterized by having significantly less data but are time-sensitive (applications such as ARP, DNS etc), while elephant flows have significantly higher data (applications involving scientific computing, MapReduce etc). Based on such high-level differences, the flow can be classified as elephant or mice based on the markings extracted from the traffic flows.

3.1 Marking Identification

The key for marking identification based detection is to determine accurate markings. If many errors exist in the process of marking, the detection accuracy would be adversely impacted resulting in a series of detection errors. A very simple marker is to set the port ID (for e.g. port 20 FTP Data transfer, port 53 DNS ICMP etc) - i.e. classify all flows with Port 20 are elephant and all flows with 53 are mice. However, the accuracy of this marker is very low since most of the applications use random and indefinite transport ports.

Based on historical behavior of applications in a DCN, the network manager can obtain different application sets using stochastic machine learning techniques for clustering. The flows generated by the same application set belong to the same category of flows, which are different from other applications generating different flow kinds. Thus application sets can be categorized as elephant flows, high-probability elephant flows and low-probability elephant flows (or mice flows). Given the scale of application variety and deployments happening in a data center, it is not possible to make relevant changes at the application side, and hence the classification information is updated at the controller. EffiEye [57] employs this approach, with the controller determining the category of flows by extracting statistical features from the received packets. The drawback of this approach is that the accuracy and complexity of detection is dependant entirely on the classification of applications.

3.2 Flow Classification

Compared to Marking Identification methods, Flow classification methods have been widely studied. Roughan et.al. [46] have presented a framework and some results for classifying traffic into Class of Service based on measured traffic characteristics. Their method is based on statistics of the traffic which result from the way the application is used, leading to the understanding of how the application is actually used as against the expected notions - for e.g. HTTP used for a large amount of the streaming traffic (video conference), rather than the expected web browsing. Though the detection accuracy is greatly improved, the classification is not refined enough. The detecting speed is fast but does not provide high accuracy.

To improve accuracy, many machine learning methods are employed, such as Naive Bayes, k-means, C4.5 decision tree, SVM, KNN [61] are used in traffic classification. These approaches classify by measuring the flow features such as duration, treating it as a set of statistical values from the flow beginning to end. However, the arrival and departure of flows are very fast in data center [8], and it is required of the controller to spend some milliseconds to allocate resource for every new flow. The above flow-based approaches using ML methods get better accuracy but due of the latency involved, they lag in detecting flows real-time. Xiao et.al [61] proposes a real-time detection of the elephant flows. The approach is in 2 stages - first to detected suspected elephant flows based on multiple flow statistical thresholds and second to filter genuine elephant flows from the first stage based on C4.5 N-features decision tree metrics with cost-sensitive parameter matrix. This helps ensure timeliness of flow detection.

4 Approaches to Flow-statistics based detection

In the process of realization of flow-statistics based approaches, there are two factors to be considered (i.) maintaining real-time and efficient flow data collection and (ii.) setting appropriate threshold values in conformance to actual DCN environment. Cost is a key factor in deployment of flow-statistics based approach for flow detection in an SDN. Pre-determined threshold to classify a flow as elephant or mice is employed here. Depending upon the nature of flow data gathering, the approaches are categorized into Pull and Push approaches.

4.1 Data Pull methods

Here the controller requests for flow-statistics report from all relevant network elements (switches or routers) and they respond back with the corresponding information. Two different implementations in data pull are by Polling and Packet-based-sampling.

$Polling$: In this type, the collector periodically collects information of each flow, such as the number of received bytes within a given time interval, from all the data plane elements. Hedera [5] obtains the traffic change in the network through periodic of the target switch. Though higher polling frequency provides precise statistics, this will lead to high monitoring overhead which may impact normal flows. Also, in terms of flow entry volume, the table size can grow large very quickly (up to 60K per minute) which is not supported by a practical OpenFlow switch implementation. A tradeoff, with reduced monitoring while still retaining accuracy, is proposed with PayLess [12] having an adaptive flow-statistics polling approach. Polling frequency will be in line with the traffic volume - higher the volume higher the frequency. Alternatively, OpenTM [53] uses the routing information gleaned from the OpenFlow controller to mathematically model select the key switches from which to obtain flow statistics data, leading to load reduction on switching elements.

Though there has been a useful amount of optimization from the above approaches of PayLess and OpenTM, there is an acute need for further reducing the monitoring overhead in a very busy DCN. A very useful concept here is to create aggregations and detect hierarchy in the flows. For e.g. IP addresses in a DCN are allocated in a specific pattern and usually a set of IP addresses are marked for a certain type of application. Thus an elephant flow could correspond to an individual flow or an aggregation of multiple flows/connections that share some common property, but with themselves not being large flows. Lin et.al [26] utilizes Hierarchical Statistics Pulling with hierarchical aggregation of IP addresses as per the prefix. Aggregations are specified on one or more dimensions, e.g., source and destination IP address, source and destination port and protocol fields for IP flows. The ISP ideally needs to collect 100% of the flow statistics just for the 10% of the flows which are probable elephant flows. With a Hierarchical Statistics Pulling mechanism at the controller, the operation for each polling period in an edge switch will be to a.) Analyze aggregated statistics response for the whole flow space block (WFSB) if it exceeds pre-set threshold and make a request accordingly. b.) Repeat rounds (depth) of dividing the block into 4 equal-sized blocks to determine which among them exceeds threshold. Make calls accordingly c.)After 'depth' rounds, send only the individual stats requests for the blocks whose aggregate statistic replies are still over the threshold. This approach will effectively lead to lesser storage, but will increase the computational requirement of the controller.

Different from the above approaches is combining SDN measurements and inference techniques based on network tomography by Hu et.al. [21]. SDN is used to build the network tomography (TM estimation) and determine the "talky" (high network chatter) top-of-rack (ToR) pairs to locate server-to-server (potential) elephant flows in them. Once identified, data from flows originating to and from these ToRs are pulled and analyzed for presence of elephant flows. This approach suits well for a DCN with fixed resources and static applications, but may not be very effective in effective routing of mice flows.

Chao et al. [11] have devised FlowSeer, a rapid elephant flow detection method at the switch level that employs data stream mining. Statistics (such as IP addresses, max and minimum packet size etc.) from the first few packets of each flow are collected to train the stream classification models. Once identified, elephant flows are routed through least congested paths in order to improve the throughput. However, FlowSeer focuses only on maximizing the throughput, leading to lack of specific handling for mice flows. Poupart et al. [33] address the problem of reducing the flow completion time to a minimum. They propose a near-real-time flow size prediction for improving routing using multiple machine learning techniques such as Neural Networks, Gaussian Process Regression (GPR) and Online Bayesian Moment Matching (oBMM). The prediction is based on flow-statistics data collected from the starting packets including source and destination IPs, source and destination ports, protocol used and the size of the initial three packets. After predicting the flow size estimates, elephant flows are routed through the least congested paths. However, the forwarding rules are computed only when the flow arrives and the Flow Control Table (FCT) update is delayed by the computational time needed and rules installation.

Block Island (BI)-based elephant flow routing, proposed by Zhang et.al. [65], builds blocking islands to narrow searching space amongst the flows in a network based on the work load of switches. Then, the least cost rerouting path is discovered with the biggest bandwidth to route elephant flows, leaving other paths free for routing mice flows. This is termed DIFFERENtiated sChEduling (DIFFERENCE) approach where paths for elephant and mice flows are dynamically set up separately and mice flows are scheduled proactively installed weighted multipath routing algorithm based on terms of current link utilization ratio.

$Packet-based-sampling$: Sampling is a common technique employed to detect various flow types in Internet traffic. Statistical methods applied on the periodic capture of transmitted/received packets are used to infer the overall flow behavior. The key win here is the overall reduction of monitoring overhead, as only limited data is captured. sFlow [47] and Cisco NetFlow [13] are standard DCN specific applications used for monitoring large scale networks, especially SDNs. They employ simple uniform sampling and provide the user capability to upgrade / vary the sampling frequency. The basic approach of sampling is that it is very rare to miss an elephant flow, as long as many packets are generated. In many cases mice flows detections are left to the default ECMP algorithm.

Usage of wildcards in switch flow table entries are employed for mice flow detection in Mice Data Center Efficient Routing (MiceDCER) [6]. It relies on the Address Resolution Protocol (ARP) messages to indicate to the controller on rules that should be installed on the switch tables. MiceDCER relies on the relative positioning of switches for traffic routing and generating wildcard rules to save space in the switching tables. Instead of reading flow packets, the controller intercepts ARP messages to interpret the addresses and install the necessary rules in the switch tables, thus limiting the number of rules on the switch. MiceDCER also relies on the positioning of switches for traffic routing and combined with the wildcard rules applications, saves considerable overhead traffic within the network.

Although the overhead caused by sample collection is reduced, there is a tradeoff with detection accuracy. The mathematical statistical models employed may introduce large errors if data for a particular flow is limited. OpenSample [50] overcomes this by optimizing the analysis of sampled packets using the TCP/IP header sequence number to determine the flow size. This helps reduce the sampling latency from 1-5 seconds to around 100ms. Afek et.al. [1] employ packet sampling using hash matching and has comparison to detect elephant flows. Instead of storing the entire packet header, a hash of relevant fields is taken and compared with existing hashes to determine the nature of the flow. Repeated hash match indicate elephant flows and appropriate routing action is determined. Also, once a suspected elephant flow is detected, active-polling is introduced to get further detailed statistics of the flow to eliminate false positives.

4.2 Data Push methods

In the Push approach, the switch actively sends the flow-statistics information to the controller, either periodically or based on a trigger, without any specific flow-request. Compared with the Pull approach, pushing reduces the monitoring overhead in the DCN. However, this requires modification at switch level, usually software and sometimes hardware, leading to non-usability of custom off-the-shelf elements in the network and high cost of implementation.

To obtain timely flow-statistics of the entire network, FlowRadar [25] employs optimized NetFlow. As most flows traverse multiple switches, the algorithm leverages the redundancies across switches and ensures that flowsets are more compactly encoded based on IBLT (Invertible Bloom Filter Lookup Table). The key factor here is identification of the best work division between COTS switches (limited per-packet processing time) and the remote controller (higher computing resources). Each packet header is subjected to multiple hash functions and compared in IBLT to check if it is a new flow or existing flow. All flow records are stored at the switch within a limited storage space and exports the flow record codes to the collector periodically.

Wang et.al [59] propose an ant colony optimization (ACO) algorithm based on behavior pattern of an ant colony network - higher the pheromone content in a path, better the path for large flows. The system uses sFlow for flow detection, computes paths for elephant flows by an adaptive multi-path calculator algorithm and transmits mice flows through specified paths in the database. This approach targets reduction in route splits for elephant flows, while ensuring better than ECMP routing for mice-flows. Tang et.al [51] proposes ESCA (Efficient Sampling and Classification Approach) with two-phase Elephant Flow detection. In the first phase, a filtering flow-table with wildcards is employed to improve sampling efficiency and estimation of the arrival intervals. Time intervals are classified as Target Blocks (TB) and Gap Blocks (GB) which are non-overlapping. In the second, threshold is determined against 4 parameters - TB duration, GB duration, sampling interval of TB and GB. Samples are classified based on the minimum TB covering all elephant flows in the controller and modified rules are forwarded to the switch. Yang et.al [63] checks for similarities between related flows, with the traffic throughput data being subjected to smoothing, thresholding and windowing to determine a time slot for sampling. The data-slot having high flow is subjected to Euclidean distance measurement of APCA (Adaptive Piecewise Constant Approximation) representation algorithm to determine presence of heavy flows.

LUNA [62] is an SDN application which proactively installs the Switch forwarding rules on the initial / specific state of the network and user SLAs/requirements. An association rule is then created which is an if-else statement describing the relation between a user and the type of his typically requested flows. Using K-Means classification, LUNA analyzes user behavior by computing their corresponding association rules and routes each flow based on its class. Effiview [58] functionality allows user the control of flows to be pre-selected in the Open flow tables, and updates are triggered on the FlowRemoved messages. It is cost effective to implement, but it is dependent upon the setting of the threshold levels at the switch, which is not dynamic. Bi et.al [10] discusses a two-stage, adaptive elephant detecting system. The switch-based decision threshold is dynamically adapted based on the initial packet analysis for traffic behavior. The relationship between elephant threshold and traffic characteristic in data center network is analyzed. By adding Gauss White Noise channel data to the flow, the threshold is dynamically set by computing the intersection point of the two flow probability distribution curves - balancing the positive false rate and negative false rate of detection.

5 Network Element Hardware-software modification approaches

Solutions involving hardware software modifications at the switches or end hosts are available but they are all impacted the cost and nature of implementation. Two broad categories are: (i) Switch modifications and (ii) End host modifications. The former involves hardware and software modifications over and above OpenFlow on the switch element. The latter is about introducing monitoring software in end hosts of a DCN to track individual flows.

5.1 Switch modifications

Switch level modifications involve improving of switch software/hardware to primarily enable push mechanism of flow-statistic collection. Open flow supports setting up of stat-triggers on the switch, i.e. the switch will trigger some actions once a particular flow threshold is exceeded. and Helios [17] employs use of special optical switches, such as Glimmerglass 64-port optical circuit switch, to increase the throughput per switch. Devoflow [15] allows operators to target only the flows that matter. Since the flows are pre-identified, DevoFlow thus reduces the monitoring overhead between the control and data planes by only transmitting statistics for needed flows. Most mice-flows are handled in the data plane. However, this approach requires switch upgrade (ASIC changes) from the normal COTS switches available in the market. To reduce the implementation cost at the switch, Planck [35] from Rasley et.al. utilizes port mirroring mechanism to gather flow data. When port mirroring is enabled, traffic destined for a given port is mirrored to another monitoring port connected to a monitoring system. As the network traffic increases, the load on the mirroring port also increases leading to data drop of flows. This is akin to sampling with limited delay; thus a high rate of packet sampling can be achieved of the order of milliseconds. Rashid [34] proposes utilizing a new feature of OpenFlow 1.3, the 'drop' tracking mechanism in the Switch table. Packet dropping is considered as a congestion indicator. If the flow size is higher than the threshold and packets are getting dropped, it is treated as a sign of elephant flow and data is pushed to the controller. In the Controller, Sorted-Global First algorithm works on the existing flows collected, calculates the ECMP values for these paths and selects the lightest loaded path for routing. Wang et.al. [60] propose a hierarchical elephant flow detection based on the way Switch Ternary Content-Addressable Memory (TCAM) is employed in allocating memory for incoming flows. More the number of flows in a switch, higher the TCAM activity, and once it crosses a set threshold, data is pushed to the Controller. Controller has a global overview of the number of switches and a hierarchical map of DCN switches encountering the load. Flows are then routed accordingly. Liu et.al [27] propose deployment of deep residual learning models with AM-Softmax software on the switches to classify flows based on their characteristics. The time feature (inter-arrival time between flows), real-time features (flow duration, flow size) and packet-header features (Transport and N/W layer encapsulation headers) are extracted using AM-Softmax and subjected to deep learning methods such as Deep Residual Network, Random Forests, Convolution Neural Networks and Gated Recurrent Unit (GRU) employing recurrent neural networks to classify the flows. They focus here is the speed so that flows are classified with lesser than 10 packets of a particular flow. The disadvantage here is the incurred cost of the software, as well as the additional computation needed for the deep learning algorithms to be executed on the switch.

Madanapalli et.al. [28] propose modifications on commodity SDN hardware and software for elephant flow detection. The Openflow pipeline on the SDN switch consists of two tables - Table-0 consists of the reactive rules corresponding to the detected elephant flows and Table-1 contains proactive rules that forward and mirror bidirectional traffic. The proactive rules ensure that no PACKET_IN messages are pushed to the controller, reducing operational load on all network elements. Sivaraman et.al. [48] propose HashPipe, an elephant flow detection algorithm using emerging programmable data planes like P4 [32].P4 allows for new data plane concepts compared to OpenFlow in that instead of fixed data-switches, programmable chips("PISA" - Protocol Independent Switch Architecture) are used to provide better granularity on the flow control. HashPipe is an implementation of a sequence of hash tables in the SDN data plane, which retain flow identifiers as counters for heavy flows in the tables while evicting lighter flows over time. The hash of flow identifiers are 'Keys', which are stored along with the count of occurrence. This is modeled with Space Saving Algorithm in a series of piped has-tables, which allows for lighter keys to move along a series of tables and finally get evicted. Keys having higher counts are termed elephant flows and controller is informed for routing decisions. As all classification processing happens in the switch, there is a tremendous reduction in the overhead methods compared to other approaches. However, relative to space saving, HashPipe may evict a genuine elephant flow key (missing heavy items from the table) or it may allow numerous duplicate flow keys in the table, leading to lesser accuracy.

5.2 End host modifications

Two key advantages with end-host modifications for flow detection - first that the end host has computing resource capacity which can be utilized and the second is that early detection of elephant flows compared to the network. Since traffic captured by end-host is a small percentage of the overall network traffic, the flow detection in end-host is primarily flow-statistics based. With scalability and timeliness of elephant-flow detection as the key goals, Mahout [14] deploys a kernel patch on the end-host and uses a pre-set threshold to determine whether there is an elephant flow in the set of flows. All elephant flows are reported to the controller, which modifies the central traffic schedule and plan. To reduce the monitoring overhead, Mahout uses the differentiated service field (IP ToS) of the IP Packet header to mark elephant flows. These marked packets would be forwarded to the controller by the receiving switch for further analysis. Similar to Mahout, MicroTE [9], a Micro Traffic Engineering system, utilizes the top-of-the-rack (ToR) controller to aggregate and create a global view of network conditions and traffic demands at real time. OpenFlow is utilized to coordinate scheduling of traffic based on weighed ECMP within the network, and the flow degrades to ECMP when traffic is unpredictable. One of the challenges of these two approaches is that they are not capable of differentiating the invisible traffic generated by the VMs present in end-host of a DCN.

Due to virtualization techniques, when there are virtual machines deployed on an end-host, the traffic monitoring implies not just network traffic, but also generated virtual traffic. Depending upon the monitoring tools such as sFlow and NetFlow supported on Open v(Virtual) Switch (OVS), EMC2 (Edge Monitoring and Collector for Cloud) [29], employs monitoring of each hypervisor on the virtualized host, which adds another network layer in the form of a "vswitch" (virtual switch) forming the new "edge" of the network. Two constraints in this approach is that the data collected at each hypervisor may be very large to be sent to Flowcollector and precise recalibration of monitoring tool parameters (sFlow highlighted) to ensure no drop in performance. In an alternate approach, VirtMonE [7] employs OVS based elephant flow detection in a virtualized environment using depending time and volume threshold parameters at the network edge. Signal communication to other network elements happens in two stages: Notifying the underlay switching fabric by tunnel L3 header marker modification and Open vSwitch Database (OVSDB) update of the details of the detected elephant flows to be analyzed by the SDN controller.

MiceTrap [54] employs the end - host modifications primarily for mice-flows detection and handling. Mice Flows are aggregated based on destination entries of multi-path group type in a group table, then rerouted in a weighted multipath approach as against ECMP. Elephant flows are also detected based on the TCP socket and marked using a kernel-level shim layer. The mice-flow aggregations reduce the rules for traffic management, thus improving resource utilization and network scalability.

6 Conclusion

In a DCN, flow classification accuracy is critical for optimizing and scheduling network traffic. There is an ongoing body of research on classification methods for effective traffic engineering (TE) in a DCN. They have also thrown light upon various network dimensions during the flows in terms of performance, latency detection and control overhead. Typically data flows are classified as Elephant Flows (or Heavy Hitters) or Mice flows (small flows) based on their size. Incorrect flow classification at flow-level management can adversely impact system scalability; mice flows may often be blocked by elephant flows because of the incorrect flow scheduling. With the software-defined networking (SDN) paradigm applied in data centers; there is a clear distinction between the data planes and control planes, giving the advantage of centrally controlled network with better flow-level management.

Researchers have introduced various approaches combining the ability of various deep learning mechanisms to classify the flows based on multi-dimensional features. Current approaches on TE flow detection fall primarily into Flow characteristics based, Flow statistics based and Network Element Hardware-software modification. This paper elaborates and summarizes the three kinds of flow detection (for both elephant and mice flows) respectively. Tables 2 & 3 summarize the variety of SDN flow detection approaches along with an evaluation based on three criteria - Targeted priority of objectives (Minimize Congestion, Maximize throughput, Minimize Flow Completion Time and Minimize Network Element workloads), Classification (Approach, Type and Importance given to flow types) and Performance (Accuracy, Timeliness of detection and Cost of implementation). As observed, many approaches focus on thresholding and managing elephant flow classification primarily, leaving the mice flows to be handled by ECMP or other default inbuilt hardware mechanisms of the NE. Also, Packet Sampling approaches among the Flow statistics based methods and Switch Modifications among the NE modification methods offer a very high TE accuracy with medium cost.

A key observation with these approaches is that many of them are near-real-time, but cannot achieve the real-time requirements of DCNs, especially with those handling time sensitive flows such as VOIP calls. This is primarily due to setup of OpenFlow switches on COTS hardware which cannot be efficiently customized owing to implementation costs. In a move towards that direction, OpenFlow is now being replaced by Stratum [18], which is an open source silicon-independent switch operating system for SDNs. Open, minimal production-ready distribution white box switches which expose a set of next-generation SDN interfaces including P4Runtime and OpenConfig help in enabling interchangeability of forwarding devices and programmability of forwarding behaviors.

Taking into account of recent advances in SDN related hardware and software, DCN TE will show the below trends: (1) With expanding data-plane programmability capabilities, the onus of flow classification will move southbound instead of at the controller. The approach [48] already demonstrates this and it will become more robust going forward. (2) Improved machine learning algorithms with deep learning and reinforced learning models can help contribute in bringing in less monitoring overhead in the DCN, while bettering detection accuracy in existing networks. Overall a combination of approaches suiting the specific network will help in effective classification, routing, and hence robust TE practices in a DCN.

Table 2: Comparison of approach priorities of different approaches

Solution	-3***Category**		-3***Research**	[HTML]F2DDDC**Targeted priority of Objectives**			
				[HTML]F2DDDC [HTML]F2DDDC**Minimize Congestion**	[HTML]F2DDDC [HTML]F2DDDC**Maximize throughput**	[HTML]F2DDDC [HTML]F2D[HTML]F2DDDC**Minimize Flow Com- pletion Time**	[HTML]F2DDDC [HTML]F2DDDC**Minimize NE workloads**
Flow characteristics	**Marking Identification**		EffiEye [57]	Medium	Medium	Medium	Medium
	Flow classification		Roughan et.al.[46]	Medium	High	Medium	Medium
			Xiao et.al [61]	High	Medium	High	Medium
Flow statistics	**Pull method**	**Polling**	Hedera [5]	Medium	Medium	Medium	Medium
			Payless [12]	High	Medium	Medium	Medium
			OpenTM [53]	High	High	Medium	High
			Lin et.al [26]	Medium	Medium	Medium	Low
			Hu et.al. [21]	High	High	Medium	High
			FlowSeer [11]	Medium	High	Medium	Low
			Poupart et al. [33]	High	Medium	High	Medium
			Zhang et.al. [65]	High	Medium	High	Medium
		Sampling	MiceDCER [6]	Medium	Medium	High	High
			OpenSample [50]	Medium	Medium	High	Medium
			Afek et.al. [1]	Medium	Medium	High	Medium
	Push method		FlowRadar [25]	Medium	Medium	Medium	Medium
			Wang et.al [59]	Medium	High	High	Medium
			Tang et.al [51]	Medium	High	Medium	High
			Yang et.al [63]	High	High	Medium	Medium
			LUNA [62]	High	High	Medium	Medium
			Effiview [58]	Medium	High	Medium	Medium
			Bi et.al [10]	Medium	Medium	Medium	Medium
Network Element Hardware-Software modification	**Modifications at Switch**		Helios [17]	Medium	High	High	Medium
			Devoflow [15]	High	High	Medium	Medium
			Planck [35]	High	Medium	Medium	High
			Rashid [34]	High	Medium	Medium	High
			Wang et.al.[60]	High	High	Medium	High
			Liu et.al [27]	High	High	Medium	High
			Madanapalli et.al. [28]	High	High	Medium	Medium
			HashPipe [48]	Medium	High	High	Medium
	Modifications at End-Host		Mahout [14]	Medium	High	High	Medium
			MicroTE [9]	High	High	High	Medium
			EMC2 [29]	Medium	High	Medium	Medium
			VirtMonE [7]	Medium	High	Medium	Medium
			MiceTrap [54]	High	Medium	High	High

Table 3: Comparison among different flow classification approaches

	[HTML]FFFC9E**Classification**				[HTML]DBE5F1**Performance**		
	[HTML]FFFC9E	[HTML]FFFC9E	[HTML]FFFC9E**Importance**		[HTML]DBE5F1	[HTML]DBE5F1	[HTML]DBE5F1
-3***Research**	-2*[HTML]FFFC9E**Approach**	-2*[HTML]FFFC9E**Type**	**Elephant**	**Mice**	-2*[HTML]DBE5F1**Accuracy**	-2*[HTML]DBE5F1**Timeliness**	-2*[HTML]DBE5F1**Cost**
EffiEye [57]	Proactive	Threshold - Markers	Primary	Secondary	low - medium	not-real-time	high
Roughan et.al.[46]	Reactive	Threshold - QoS	Primary	Secondary	medium	not-real-time	medium
Xiao et.al [61]	Reactive	ML - C4.5	Primary	Secondary	high	not-real-time	medium
Hedera [5]	Reactive	ML - GFF	Primary	Secondary	high	not-real-time	high
Payless [12]	Reactive	ML - Adaptive	Primary	Secondary	high	not-real-time	high
OpenTM [53]	Reactive	ML - OF Routing	Primary	Primary	medium - high	not-real-time	medium
Lin et.al [26]	Reactive	ML - hierarchical aggregation	Primary	Secondary	medium - high	delayed	medium
Hu et.al. [21]	Proactive	Network Tomography	Primary	Primary	medium - high	delayed	medium
FlowSeer [11]	Reactive	ML - Stream Mining	Primary	Secondary	medium	not-real-time	medium
Poupart et al. [33]	Reactive	ML - Neural Networks	Primary	Secondary	high	delayed	high
Zhang et.al. [65]	Reactive	ML - Block Island	Primary	Secondary	medium - high	not-real-time	medium
MiceDCER [6]	Reactive	Wildcards, ARP read	Secondary	Primary	high	not-real-time	medium
OpenSample [50]	Reactive	TCP/IP Headers	Primary	Secondary	high	not-real-time	medium
Afek et.al. [1]	Reactive	Hash matching	Primary	Primary	high	not-real-time	medium
FlowRadar [25]	Reactive	IBLT	Primary	Secondary	medium	real-time	medium
Wang et.al [59]	Reactive	ML - ACO	Primary	Secondary	medium - high	real-time	high
Tang et.al [51]	Reactive	Threshold - Arrival block	Primary	Primary	medium	real-time	medium
Yang et.al [63]	Reactive	Threshold - Euclidean distance	Primary	Secondary	medium	not-real-time	medium
LUNA [62]	Proactive	ML - Kmeans	Primary	Secondary	medium	not-real-time	medium
Effiview [58]	Proactive	Threshold - Flow stats	Primary	Secondary	medium - high	delayed	low
Bi et.al [10]	Reactive	ML - Adaptive	Primary	Secondary	low - medium	real-time	medium
Helios [17]	Reactive	Optical switches	Primary	Secondary	high	near-real-time	high
Devoflow [15]	Proactive	ASIC upgrades	Primary	Secondary	medium - high	near-real-time	high
Planck [35]	Reactive	Port Mirroring	Primary	Primary	medium	near-real-time	medium
Rashid [34]	Reactive	Drop tracking	Primary	Primary	medium	near-real-time	medium
Wang et.al.[60]	Reactive	Threshold - TCAM upgrades	Primary	Primary	high	near-real-time	high
Liu et.al [27]	Reactive	Deep Residual Learning with AM-Softmax	Primary	Primary	high	near-real-time	high
Madanapalli et.al. [28]	Proactive	COTS OVSwitch modifications	Primary	Secondary	medium - high	near-real-time	medium
HashPipe [48]	Reactive	P4 with PISA	Primary	Secondary	medium	near-real-time	medium
Mahout [14]	Proactive	Threshold - IP ToS	Primary	Secondary	medium - high	near-real-time	medium
MicroTE [9]	Reactive	Threshold - ToR	Primary	Secondary	medium - high	real-time	medium
EMC2 [29]	Proactive	OVS updates	Primary	Secondary	medium - high	delayed	high
VirtMonE [7]	Reactive	OVSDB updates	Primary	Secondary	medium - high	delayed	high
MiceTrap [54]	Reactive	Marked HH flows; Weighed Multipath routing	Secondary	Primary	medium - high	near-real-time	medium

References

[1] Afek, Y., Bremler-Barr, A., Landau Feibish, S., and Schiff, L. Sampling and large flow detection in sdn. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 345–346, 2015.

[2] Affonso, E. T., Nunes, R. D., Rosa, R. L., Pivaro, G. F., and Rodríguez, D. Z. Speech quality assessment in wireless voip communication using deep belief network. *IEEE Access*, 6:77022–77032, 2018.

[3] Affonso, E. T., Rodríguez, D. Z., Rosa, R. L., Andrade, T., and Bressan, G. Voice quality assessment in mobile devices considering different fading models. In *2016 IEEE International Symposium on Consumer Electronics (ISCE)*, pages 21–22. IEEE, 2016.

[4] Affonso, E. T., Rosa, R. L., and Rodríguez, D. Z. Speech quality assessment over lossy transmission channels using deep belief networks. *IEEE Signal Processing Letters*, 25(1):70–74, 2017.

[5] Al-Fares, M., Radhakrishnan, S., Raghavan, B., Huang, N., Vahdat, A., et al. Hedera: dynamic flow scheduling for data center networks. In *Nsdi*, volume 10, pages 89–92. San Jose, USA, 2010.

[6] Amezquita-Suarez, F., Estrada-Solano, F., da Fonseca, N. L., and Rendon, O. M. C. An efficient mice flow routing algorithm for data centers based on software-defined networking. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2019.

[7] Bashir, S. and Ahmed, N. Virtmone: Efficient detection of elephant flows in virtualized data centers. In *2015 International Telecommunication Networks and Applications Conference (ITNAC)*, pages 280–285. IEEE, 2015.

[8] Benson, T., Akella, A., and Maltz, D. A. Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 267–280, 2010.

[9] Benson, T., Anand, A., Akella, A., and Zhang, M. Microte: Fine grained traffic engineering for data centers. In *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, pages 1–12, 2011.

[10] Bi, C., Luo, X., Ye, T., and Jin, Y. On precision and scalability of elephant flow detection in data center with sdn. In *2013 IEEE Globecom Workshops (GC Wkshps)*, pages 1227–1232. IEEE, 2013.

[11] Chao, S.-C., Lin, K. C.-J., and Chen, M.-S. Flow classification for software-defined data centers using stream mining. *IEEE Transactions on Services Computing*, 12(1):105–116, 2016.

[12] Chowdhury, S. R., Bari, M. F., Ahmed, R., and Boutaba, R. Payless: A low cost network monitoring framework for software defined networks. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–9. IEEE, 2014.

[13] Cisco. Cisco netflow, December 2020.

[14] Curtis, A. R., Kim, W., and Yalagandula, P. Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection. In *2011 Proceedings IEEE INFOCOM*, pages 1629–1637. IEEE, 2011.

[15] Curtis, A. R., Mogul, J. C., Tourrilhes, J., Yalagandula, P., Sharma, P., and Banerjee, S. Devoflow: Scaling flow management for high-performance networks. In *Proceedings of the ACM SIGCOMM 2011 Conference*, pages 254–265, 2011.

[16] de Almeida, F. L., Rosa, R. L., and Rodríguez, D. Z. Voice quality assessment in communication services using deep learning. In *2018 15th International Symposium on Wireless Communication Systems (ISWCS)*, pages 1–6. IEEE, 2018.

[17] Farrington, N., Porter, G., Radhakrishnan, S., Bazzaz, H. H., Subramanya, V., Fainman, Y., Papen, G., and Vahdat, A. Helios: a hybrid electrical/optical switch architecture for modular data centers. In *Proceedings of the ACM SIGCOMM 2010 Conference*, pages 339–350, 2010.

[18] Foundation, O. N. Open networking foundation, December 2020.

[19] Foundation, O. N. Openflow switch specification version 1.5.1, December 2020.

[20] Guimaraes, R. G., Rosa, R. L., De Gaetano, D., Rodríguez, D. Z., and Bressan, G. Age groups classification in social network using deep learning. *IEEE Access*, 5:10805–10816, 2017.

[21] Hu, Z. and Luo, J. Cracking network monitoring in dcns with sdn. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 199–207. IEEE, 2015.

[22] Jordane da Silva, M., Carrillo Melgarejo, D., Lopes Rosa, R., and Zegarra Rodríguez, D. Speech quality classifier model based on dbn that considers atmospheric phenomena. *Journal of Communications Software and Systems*, 16(1):75–84, 2020.

[23] Kandula, S., Sengupta, S., Greenberg, A., Patel, P., and Chaiken, R. The nature of data center traffic: measurements & analysis. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, pages 202–208, 2009.

[24] Lan, K.-c. and Heidemann, J. A measurement study of correlations of internet flow characteristics. *Computer Networks*, 50(1):46–62, 2006.

[25] Li, Y., Miao, R., Kim, C., and Yu, M. Flowradar: A better netflow for data centers. In *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, pages 311–324, 2016.

[26] Lin, C.-Y., Chen, C., Chang, J.-W., and Chu, Y. H. Elephant flow detection in datacenters using openflow-based hierarchical statistics pulling. In *2014 IEEE Global Communications Conference*, pages 2264–2269. IEEE, 2014.

[27] Liu, W.-X., Cai, J., Wang, Y., Chen, Q. C., and Zeng, J.-Q. Fine-grained flow classification using deep learning for software defined data center networks. *Journal of Network and Computer Applications*, 168:102766, 2020.

[28] Madanapalli, S. C., Lyu, M., Kumar, H., Gharakheili, H. H., and Sivaraman, V. Real-time detection, isolation and monitoring of elephant flows using commodity sdn system. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–5. IEEE, 2018.

[29] Mann, V., Vishnoi, A., and Bidkar, S. Living on the edge: Monitoring network flows at the edge in cloud data centers. In *2013 Fifth International Conference on Communication Systems and Networks (COMSNETS)*, pages 1–9. IEEE, 2013.

[30] Nunes, R. D., Pereira, C. H., Rosa, R. L., and Rodríguez, D. Z. Real-time evaluation of speech quality in mobile communication services. In *2016 IEEE International Conference on Consumer Electronics (ICCE)*, pages 389–390. IEEE, 2016.

[31] Nunes, R. D., Rosa, R. L., and Rodríguez, D. Z. Performance improvement of a non-intrusive voice quality metric in lossy networks. *IET Communications*, 13(20):3401–3408, 2019.

[32] P4. P4 software, December 2020.

[33] Poupart, P., Chen, Z., Jaini, P., Fung, F., Susanto, H., Geng, Y., Chen, L., Chen, K., and Jin, H. Online flow size prediction for improved network routing. In *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, pages 1–6. IEEE, 2016.

[34] Rashid, J. A. Sorted-gff: An efficient large flows placing mechanism in software defined network datacenter. *Karbala International Journal of Modern Science*, 4(3):313–331, 2018.

[35] Rasley, J., Stephens, B., Dixon, C., Rozner, E., Felter, W., Agarwal, K., Carter, J., and Fonseca, R. Planck: Millisecond-scale monitoring and control for commodity networks. *ACM SIGCOMM Computer Communication Review*, 44(4):407–418, 2014.

[36] Rodríguez, D. Z. and Bressan, G. Video quality assessments on digital tv and video streaming services using objective metrics. *IEEE Latin America Transactions*, 10(1):1184–1189, 2012.

[37] Rodríguez, D. Z., Carrillo, D., Ramírez, M. A., Nardelli, P. H. J., and Möller, S. Incorporating wireless communication parameters into the e-model algorithm. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:956–968, 2021.

[38] Rodríguez, D. Z., da Silva, M. J., Silva, F. J. M., and Junior, L. C. B. Assessment of transmitted speech signal degradations in rician and rayleigh channel models. *INFOCOMP Journal of Computer Science*, 17(2):23–31, 2018.

[39] Rodríguez, D. Z. and Junior, L. C. B. Determining a non-intrusive voice quality model using machine learning and signal analysis in time. *INFOCOMP Journal of Computer Science*, 18(2), 2019.

[40] Rodríguez, D. Z., Rosa, R. L., Almeida, F. L., Mittag, G., and Möller, S. Speech quality assessment in wireless communications with mimo systems

using a parametric model. *IEEE Access*, 7:35719–35730, 2019.

[41] Rodríguez, D. Z., Rosa, R. L., and Bressan, G. A billing system model for voice call service in cellular networks based on voice quality. In *2013 IEEE International Symposium on Consumer Electronics (ISCE)*, pages 89–90. IEEE, 2013.

[42] Rodríguez, D. Z., Rosa, R. L., Costa, E. A., Abrahão, J., and Bressan, G. Video quality assessment in video streaming services considering user preference for video content. *IEEE Transactions on Consumer Electronics*, 60(3):436–444, 2014.

[43] Rodríguez, D. Z., Wang, Z., Rosa, R. L., and Bressan, G. The impact of video-quality-level switching on user quality of experience in dynamic adaptive streaming over http. *EURASIP Journal on Wireless Communications and Networking*, 2014(1):1–15, 2014.

[44] Rosa, R. L., Rodríguez, D. Z., and Bressan, G. Music recommendation system based on user's sentiments extracted from social networks. *IEEE Transactions on Consumer Electronics*, 61(3):359–367, 2015.

[45] Rosa, R. L., Schwartz, G. M., Ruggiero, W. V., and Rodríguez, D. Z. A knowledge-based recommendation system that includes sentiment analysis and deep learning. *IEEE Transactions on Industrial Informatics*, 15(4):2124–2135, 2018.

[46] Roughan, M., Sen, S., Spatscheck, O., and Duffield, N. Class-of-service mapping for qos: a statistical signature-based approach to ip traffic classification. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 135–148, 2004.

[47] SFlow. Sflow, December 2020.

[48] Sivaraman, V., Narayana, S., Rottenstreich, O., Muthukrishnan, S., and Rexford, J. Heavy-hitter detection entirely in the data plane. In *Proceedings of the Symposium on SDN Research*, pages 164–176, 2017.

[49] Stallings, W. Software-defined networks and openflow. *The internet protocol Journal*, 16(1):2–14, 2013.

[50] Suh, J., Kwon, T. T., Dixon, C., Felter, W., and Carter, J. Opensample: A low-latency, sampling-based measurement platform for commodity sdn. In *2014 IEEE 34th International Conference on Distributed Computing Systems*, pages 228–237. IEEE, 2014.

[51] Tang, F., Li, L., Barolli, L., and Tang, C. An efficient sampling and classification approach for flow detection in sdn-based big data centers. In *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, pages 1106–1115. IEEE, 2017.

[52] Terra Vieira, S., Lopes Rosa, R., Zegarra Rodríguez, D., Arjona Ramírez, M., Saadi, M., and Wuttisittikulkij, L. Q-meter: Quality monitoring system for telecommunication services based on sentiment analysis using deep learning. *Sensors*, 21(5):1880, 2021.

[53] Tootoonchian, A., Ghobadi, M., and Ganjali, Y. Opentm: traffic matrix estimator for openflow networks. In *International Conference on Passive and Active Network Measurement*, pages 201–210. Springer, 2010.

[54] Trestian, R., Muntean, G.-M., and Katrinis, K. Micetrap: Scalable traffic engineering of datacenter mice flows using openflow. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 904–907. IEEE, 2013.

[55] Vieira, S. T., Rosa, R. L., and Rodríguez, D. Z. A speech quality classifier based on tree-cnn algorithm that considers network degradations. *Journal of Communications Software and Systems*, 16(2):180–187, 2020.

[56] Wang, B. and Su, J. A survey of elephant flow detection in sdn. In *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, pages 1–6. IEEE, 2018.

[57] Wang, B., Su, J., Chen, L., Deng, J., and Zheng, L. Effieye: Application-aware large flow detection in data center. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 794–796. IEEE, 2017.

[58] Wang, B., Su, J., Li, J., and Han, B. Effiview: Trigger-based monitoring approach with low cost in sdn. In *2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems*

(HPCC/SmartCity/DSS), pages 309–315. IEEE, 2017.

[59] Wang, C., Zhang, G., Chen, H., and Xu, H. An aco-based elephant and mice flow scheduling system in sdn. In *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, pages 859–863. IEEE, 2017.

[60] Wang, W., Yang, Y., and Wang, E. A distributed hierarchical heavy hitter detection method in software-defined networking. *IEEE Access*, 7:55367–55381, 2019.

[61] Xiao, P., Qu, W., Qi, H., Xu, Y., and Li, Z. An efficient elephant flow detection with cost-sensitive in sdn. In *2015 1st International Conference on Industrial Networks and Intelligent Systems (INISCom)*, pages 24–28. IEEE, 2015.

[62] Yahyaoui, H., Aidi, S., and Zhani, M. F. On using flow classification to optimize traffic routing in sdn networks. In *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE, 2020.

[63] Yang, L., Ng, B., and Seah, W. K. Heavy hitter detection and identification in software defined networking. In *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–10. IEEE, 2016.

[64] Zhang, H., Guo, X., Yan, J., Liu, B., and Shuai, Q. Sdn-based ecmp algorithm for data center networks. In *2014 IEEE Computers, Communications and IT Applications Conference*, pages 13–18. IEEE, 2014.

[65] Zhang, H., Tang, F., and Barolli, L. Efficient flow detection and scheduling for sdn-based big data centers. *Journal of Ambient Intelligence and Humanized Computing*, 10(5):1915–1926, 2019.