OpenFlow-Enabled User Traffic Profiling in Campus Software Defined Networks

Taimur Bakhshi and Bogdan Ghita
Center for Security, Communications and Network Research
University of Plymouth
Plymouth, United Kingdom
{taimur.bakhshi, bogdan.ghita}@plymouth.ac.uk

Abstract— **The recently emerging paradigm of software defined networking (SDN) predominantly employs the control-data plane OpenFlow protocol, offering centralized real-time programmability and monitoring of network devices. Effective SDN based traffic engineering using OpenFlow, particularly in campus networking however, requires sophisticated real-time user traffic visualization solution having minimum management overhead. To address the intuitive monitoring gaps in existing campus based SDN, the present paper proposes profiling campus user traffic to visualize real-time network workload and accurately provision resources. The design solely utilizes existing OpenFlow traffic measurements, subjected to k-means clustering to segregate users into different traffic classes (profiles) based on their application trends. To validate design feasibility, the present study derived six unique user traffic profiles from OpenFlow generated traffic statistics of a realistic campus switch over a two-week time frame. The derived profiles represented significant discrimination among user application trends and were further benchmarked for high stability (96.1-99.1%), to ascertain their viability for monitoring purposes. Additional simulation tests at varying user loads attributed minimum computational cost and low OpenFlow control overhead (4.02-4.96%) to the proposed approach, offering high scalability for real-time network monitoring and resource provisioning.**

Keywords—network monitoring; software defined networking; campus networking; user profiling; k-means clustering

I. INTRODUCTION

The emerging paradigm of software defined networking (SDN) offers real-time network programmability, separating network control logic from underlying network devices into distinct control and data planes respectively. The centralized SDN controller manages and monitors individual network devices through southbound control-data plane protocols [1], enabling real-time reconfiguration of the network fabric (e.g. campus wide switches) according to changing traffic priorities. One of the most prominent southbound protocols is the OpenFlow protocol [2], providing per-flow monitoring and management of OpenFlow compliant SDN switches through a sophisticated set of controller to device message exchanges. The OpenFlow protocol also caters for improving the individual service performance by guaranteeing quality of service through isolated application flow metering. However, isolated application performance, which is the default traffic optimization method in SDN, may not be suitable for all campus users, particularly the users using different set of applications to the ones optimized. Furthermore, conventional traffic monitoring approaches for segregated service prioritization may also fail to fully utilize the real-time programmability offered by SDN. An SDN specific traffic monitoring solution is, therefore, required giving operators the ability to visualize and account actual user activities and improve the end user experience. Profiling user traffic based on application trends may more accurately express user activities and aid administrators in aligning optimization solutions to the inherent campus user classes instead of individual applications [3]. To this end, the present paper proposes profiling user application trends and employing the extracted profiles as a means to characterize and monitor the campus workload. Utilizing per-profile traffic statistics the SDN controller can anticipate real-time traffic conditions based on changing profile memberships assisting operators in implementing user-centric policies. Our profiling methodology accounts user-generated flows (application usage) towards the campus servers, solely employing OpenFlow counters in network switches polled via the SDN controller. Furthermore, the design focuses on retrieving traffic statistics nearer to the user i.e. the edge or access switches to account for any local service (or application server) traffic which may not traverse the campus core. Using the existing OpenFlow control channel between the edge switches and the SDN controller, flow measurement as well as profile extraction remains centralized eliminating requirement for external flow accounting (NetFlow, IPFIX, etc.) and dedicated monitoring overlays. To validate the feasibility of our approach, the study collected traffic statistics from a virtual Open vSwitch [4] and Ryu SDN controller [5] instance connected to a realistic campus edge to profile user activity. User statistics were monitored over a two-week time frame and subjected to unsupervised k-means cluster analysis to segregate users into different classes based on their application trends. The scalability of the design was further evaluated by simulating several user profile loads in Mininet [6] to benchmark the monitoring message overhead as well as the computational cost associated with user profiling. The rest of this paper is organized as follows. Section II presents related work and background material. Section III details the proposed user profiling method. Section IV discusses the derived profiles also highlighting the scalability evaluation of the design. Section V gives a perspective on profiling based traffic engineering and final conclusions are drawn in section VI.

II. Related Work And Background

A. Related Work

OpenFlow traffic monitoring schemes have increasingly focused on using it for flow monitoring and control, while keeping the associated management overhead to a minimum. Isolani et. al [7] proposed interactive SDN management using OpenFlow to monitor network traffic and establish the effects on resource usage and control channel load by varying flow parameters such as the flow idle time out and installation of flow forwarding rules in switches. Using an adaptive switch polling frequency, van Adrichem et. al [8] optimized OpenFlow based traffic measurement accuracy while seeking to reduce the network and switch CPU overhead. Similarly, the polling frequency and selection of switch subsets for monitoring was evaluated in [9], balancing monitoring accuracy and the control overhead. Yu et al. [10] proposed using FlowSense, an OpenFlow monitoring solution relying on asymmetric messsages between switch and controller to keep the control channel overhead minimum while promising high monitoring accuracy. On a separate strand, anomaly and intrusion detection algorithms relying on OpenFlow based measurements have also been the focus of research efforts to harden SDN security while simultaneously providing traffic engineering solutions [11]. The aforementioned approaches, therefore, increasingly emphasize OpenFlow usage for network monitoring and simultaneously seek to minimize the monitoring traffic overhead. However, to the best of our knowledge, no previous work specifically focuses on leveraging OpenFlow based monitoring information to profile user behaviour in an SDN framework. As mentioned earlier, user traffic profling within an SDN context aims to understand real-time user behaviour and to help network administrators in visualizing user trends in subsequently implementing user-centric policies. The following sub-sections provide an overview of the OpenFlow protocol and the prominent k-means clustering algorithm before discussing the proposed profiling methodology.

B. OpenFlow Protocol

The OpenFlow protocol employs flow table pipeline processing in each individual network switch to forward traffic from source to destination. A set of controller-switch OpenFlow messages retrieve flow forwarding information from the SDN controller also allowing the controller to read and update switch status. The three prominent message types used are (i) *Controller-to-switch*, initiated by the controller to manage and inspect switch state, (ii) *Asymmetric*, initiated by the switch to notify of network events and (iii) *Symmetric*, initiated by either entity to *keepalive* the control channel [2]. Each message comprises of further sub-types for specific actions. A campus schematic representing the basic working principle of the protocol is depicted in Fig. 1. The first packet of an incoming user flow is matched against switch flow table entries prescribing further action such as forward or drop in case of a successful match. In case of a table-miss, the switch (default behaviour) forwards the first packet of the flow to the

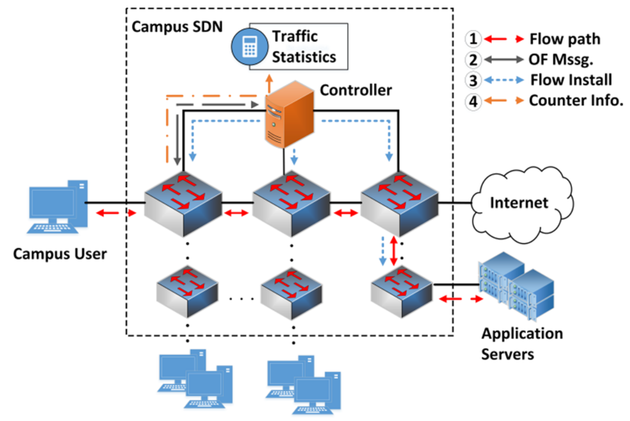

Fig. 1. OpenFlow protocol: control and monitoring information

controller to request forwarding information using *packet_in* message. The controller in turn has the prerequisite knowledge regarding location of the target desitnation, such as campus server(s) or the Internet gateway discovered during service intitiation. The controller sends a *packet_out* message to the respective switch and *flow_mod* messages to each switch along the destined path of the flow describing forwarding actions. The flow-entries, once installed in switch tables, conform to pre-set *idle_timeout* and *hard_timeout* values. The OpenFlow protocol also allows multi-part *read-state* messages to retrieve traffic statistics from switches. The controller, however, does not orchestrate flow-forwarding behaviour based on any collected statistical information on its own. Using the controller northbound programming interface (API), SDN frameworks such as Ryu [5] allow administrators to use the RESTful protocol to poll OpenFlow switch counters and utilize this information in a monitoring solution to manage the underlying network.

C. Unsupervised K-Means Cluster Analysis

The present work requires segregating user classes based on proportional variations in application trends using the statistics collected through OpenFlow based traffic monitoring. For this purpose, the prominent un-supervised k-means clustering technique was employed [12]. As evaluated in [13], compared to other methods such as hierarchical clustering, k-means is resource efficient and produces tighter clusters. The k-means algorithm minimizes a given number of vectors by choosing k random vectors as initial cluster centers and assigning each vector to a cluster as determined by distance metric comparison with the cluster center (a squared error function) as given in (1). Cluster centers are recomputed as the cluster member average and the iteration continues until the clusters converge or specified iterations have passed.

$$J = \Sigma_{j=(1,k)} \Sigma_{i=(1,n)} ||x_{ij} - c_j||^2 \quad (1)$$

In (1) above, c_j represents the cluster center, n equals the sample space size and k the chosen value for number of unique clusters. Hence, using k-means, n entities, translating for users in the present case, can be partitioned into k profiles. The present scheme used Everitt and Hothorn method [14] to determine the optimal number of user profiles (section IV).

III. Methodology

The proposed traffic profiling methodology comprises of two main components (i) OpenFlow traffic monitor and (ii) the traffic profiling engine.

The *traffic monitor* utilizes five primary OpenFlow message types presented in Table 1 to record the individual application usage of users connected to an OpenFlow compliant switch (such as Open vSwitch). The resulting monitoring information collected per flow is a seven-tuple record including the source and destination IP address and ports, duration of the flow, the number of packets matching the flow and the total bytes transferred before flow termination <SrcIP, DstIP, SrcPort, DstPort, Duration, Packets, Bytes>. Following record collection, the *profiling engine* collates traffic composition statistics per user as a percentage of user generated flows towards campus data sources, mapped according to their respective destination IP address(es). The aggregate traffic composition vectors are afterwards fed to a k-means clustering module, segregating users into different classes (profiles) based on their application trends. The resulting profiles are stored in a central database and continuously monitored to benchmark their stability with any deviation from pre-determined baseline values triggering re-profiling. The following sub-sections discuss the data collection setup, the traffic monitor and the profiling engine in detail.

A. Data Collection Setup & Traffic Monitor

To determine the feasibility of our traffic profiling approach, traffic records were collected from a realistic academic network segment consisting of approximately 42 users over two week duration between 15/02/2016 and 29/02/2016. In order to eliminate the impact of our study on production traffic, the setup used a Linux monitoring machine (VM1) running an Open vSwitch (SW1) and Ryu SDN controller instance connected to the departmental LAN as shown in Fig. 2. Port monitoring was enabled at the default gateway (SW2) to replicate all traffic to and from each user to the VM1 interface (virtual switch SW1). The mirrored traffic was processed through the Open vSwitch (SW1), however, not forwarded to any outside port, since the objective was solely the collection of user traffic statistics. The traffic monitoring application running on VM1 polled Open vSwitch (SW1) counters by issuing RESTful calls to the Ryu controller to collect per user flow statistics. All user machines used static IP addressing scheme to simplify accounting per user application usage from the collected OpenFlow records. Steps describing

TABLE I. OpenFlow Message Specification

Type	Message	OpenFlow Switch Conformance Spec. v1.3.1	Size
Flow forwarding and control	Packet_In	OFPT_PACKET_IN	160 bits+ first flow packet
	Packet_Out	OFPT_PACKET_OUT	160 bits+ update packet
Statistics, counter polling	Table_Stat	OFPC_TABLE_STATS	32.3 bytes
	Flow_Stat	OFPC_FLOW_STATS	448 bits
Switch event	Flow_Rem	OFPT_FLOW_REMOVED	352 bits

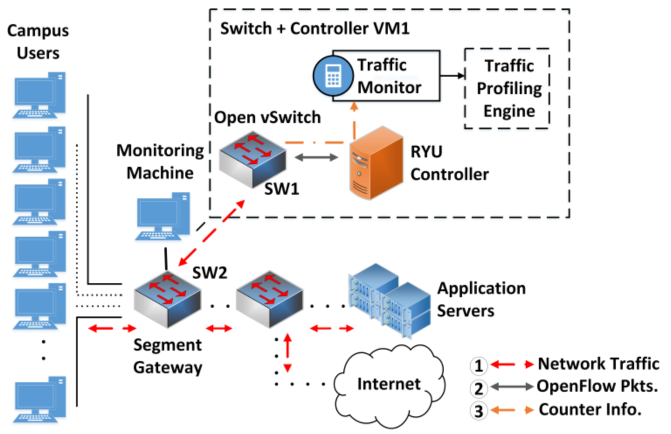

Fig. 2. Data collection setup: campus network segment

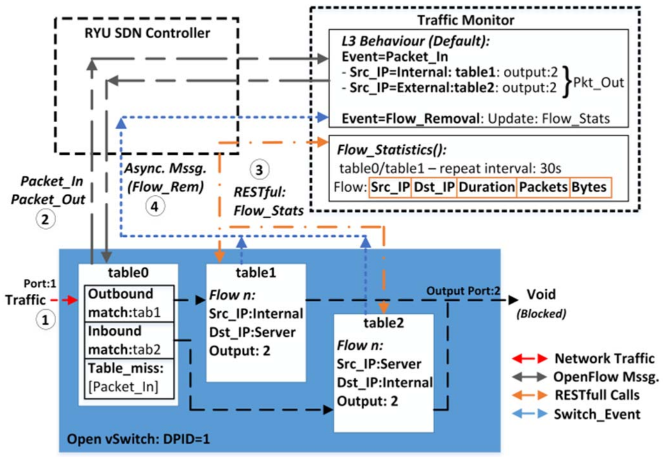

Fig. 3. Traffic monitoring schematic

the installation of flows in SW1, their subsequent updating, and user statistics collection are detailed as follows.

1) Flow installation: Standard Ryu based layer3 routing function and the traffic monitor application was implemented using the default OpenFlow behaviour with customized flow routing as shown in Fig. 2. The first packet of incoming (mirrored) traffic on SW1 *port1*, was matched against existing *table0* entries for processing. In case of a *table_miss*, an OpenFlow *packet_in* message consisting of the first packet of the flow was created and sent to the controller. On receiving *packet_in*, the controller created a *packet_out* message instructing SW1 to forward upstream LAN traffic via *table1* and downstream traffic (from campus servers and the internet) via *table2* out *port2*. Since the purpose of the experiment was data collection and not actual flow forwarding and OpenFlow does not prevent flow installation towards a blocked port, virtual *port2* on SW1 was set to blocking mode (sink). This resulted in the installation of respective flows in SW1 with subsequent flow packets negotiating *table1* or *table2* out *port2*, without consequences for live user traffic and generating per user flow statistics. The Open vSwitch SW1, therefore, installed and updated OpenFlow entries in each flow table as would be the case if directly connected as the LAN default gateway to forward user traffic.

2) Statistics collection: The traffic monitor made RESTful calls to the Ryu controller to determine the statistics for *table1* and *table2* entities. The controller in turn, fulfilled these polling requests by issuing OpenFlow *flow_stats* and *table_stats* messages to SW1, with the respective counter for each flow and table entry sent to the traffic monitor. Since RESTful is a non-subscription based API, the polling frequency of RESTful calls was manually set to 30 seconds, approximately half of the default *idle_timeout* value (60 seconds) to regularly generate statistics. In addition to frequent polling, flow completion and *idle_timeout* expiration triggered asymmetric *flow_rem* event message by SW1 to the controller, in turn updating the traffic monitor. The monitor collected per user seven-tuple record entries, collated every 24 hours and fed to the profiling engine to extract user profiles. The profiling engine design is discussed in next subsection.

B. Traffic Profiling Engine

The profiling engine design is depicted in Fig. 4. User traffic collected by the traffic monitor was classified by matching seven-tuple traffic records against source and destination IP addresses and ports used by the respective users and campus servers. To further account for replication in nature of user activities, and derive meaningful profiles, services were tiered into distinct categories depicted in Table 2. A unique webpage visit or service usage on a user machine could therefore, be defined by the vector u_i given in (2).

$$u_i = [e_i, g_i, v_i, c_i, p_i, h_i, r_i, w_i, z_i] \quad (2)$$

In equation (2) above, *i* uniquely identifies the user machine and the remaining entities represent the service usage percentage in accordance with Table 2. Network activity for a user u_1 over any given 24 hour time-bin, e.g. [29/02/2016], could therefore be represented by the application distribution in (3).

$$u_{1\,[29/02/2016]} = [43.2\ 15.1\ 0.2\ 9.6\ 4.1\ 2.1\ 18.4\ 4.4\ 2.9] \quad (3)$$

Once application distribution vectors per user machine were collected, the traffic profiling engine implemented an R programming library of the Hartigan and Wong k-means function [12], to extract user classes based on the application trends. As mentioned earlier, since the number of clusters (k)

TABLE II. APPLICATOIN TIERS

Application Tier	Website, Destination Port
Emailing (e)	Outlook, SMTP, POP3, IMAP
Storage (g)	Central storage (://Z Drive, FTP)
Streaming (v)	Podcasting, video content
Communications (c)	Office communications server
Enterprise (p)	Corporation information system, staff portal
Publishing (h)	Content management system (document, print)
Software (r)	Software distribution service
Web browsing (w)	External Internet traffic
Network utility (z)	DNS queries, Network Multicasts

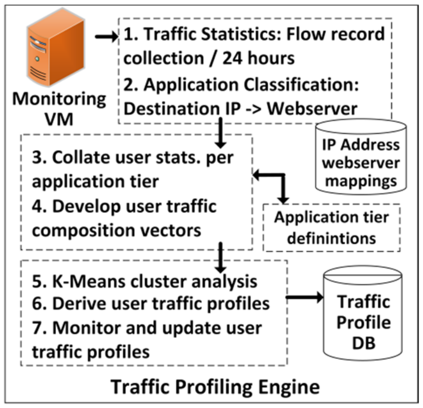
Monitoring VM
1. Traffic Statistics: Flow record collection / 24 hours
2. Application Classification: Destination IP -> Webserver
IP Address webserver mappings
3. Collate user stats. per application tier
4. Develop user traffic composition vectors
Application tier defininitions
5. K-Means cluster analysis
6. Derive user traffic profiles
7. Monitor and update user traffic profiles
Traffic Profile DB
Traffic Profiling Engine

Fig. 4. Traffic profling engine

directly influences the resulting number of profiles, the present scheme employed the graphical Everitt and Hothorn technique [14] plotting *within cluster sum of square values (wss)* against the number of clusters k. The resulting knee in the derived curve (plot) signifies an appropriate number of clusters that fit the input data. Furthermore, to benchmark the stability of the profiles, we evaluated user profile transitions every 24 hours over the two weeks of study. The extracted user profiles, profile stability evaluation, and computational cost of the design are described in the following section.

IV. USER TRAFFIC PROFILES

A total of approximately 7.8 million records were collected over the two week study and the corresponding user traffic composition vectors were afterwards subjected to k-means clustering. The plot of *wss vs. k* of the user traffic composition vectors is given in Fig 5. The profiling engine calculated the maximum within-cluster variance between each successive value of k, examined up to k=20 to evaluate the optimal cluster number [14]. As shown in Fig. 5, the variance between individual values is maximum until k=6, however, subsequent values of k (≥6) show minimum change in the successive overall variance (<0.05%). Therefore, for the present study, k=6 provided an optimal number of user profiles fitting the sample space used for further analysis. The profiling engine correspondingly marked the daily user traffic records with the individual cluster (profile) colour. The next subsection examines the resulting six user traffic profiles.

A. Extracted Profiles

The extracted user traffic profiles (for clusters k=6), highlighting the application trends as a percentage of user generated flows are depicted in Fig. 6. From a monitoring and

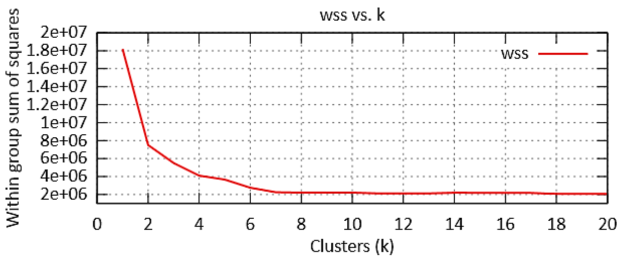

Fig. 5. Optimal cluster determination – wss. vs. k

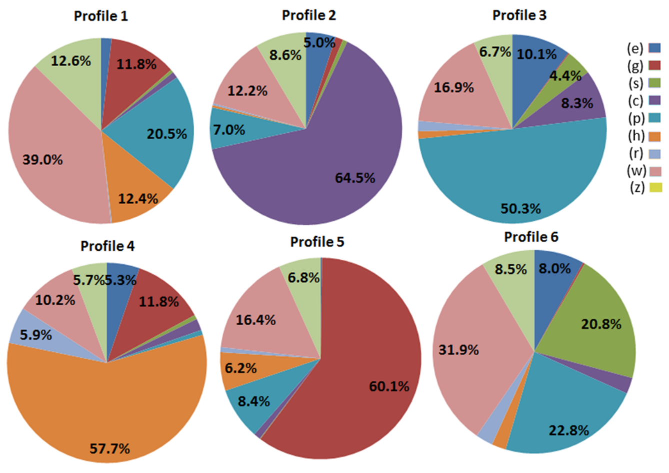

Fig. 6. User traffic profiles

network management perspective, the derived user profiles showed significant variation in activities among the derived traffic classes. For example, Profile 1 concentrated mainly on web browsing (39%) with relatively limited usage of other applications apart from the corporate information (20.5%) and content management services (12.4%). Profile 2 focused on using office instant messenger and VoIP largely (64.5%) with limited use of content management applications and web browsing (12.2%). In comparison with other profiles, Profile 3 users heavily interacted with corporate information services (50.3%) with a significant use of email service (10.1%). Profile 4 users concentrated on document and print content creation (57.7%) and using centralized storage facility (11.8%). Profile 5 mainly used central storage filer (60.1%) with small use of content and corporate information server. Profile 6 was a mix of web browsing, email usage, file storage and the staff portal along with streaming (20.8%). Each of the derived campus user profiles, therefore, represented a significant discrimination towards a certain mix of applications and services. The use of software distribution was, however, significantly low compared to other applications among all user profiles with profile 4 showing the highest proportion of software downloads (5.9%) from the campus software store .

To benchmark the traffic baseline for each profile, we calculated the maximum probability for the number of users, total traffic volume along with upstream and downstream flow rates and flow statistics given in Figs. 7-9. Profile 1 had the highest number of users (10-14 users) during office hours (09:00-17:00hrs) followed by profile 3 (6-8 users) while profile 6 membership increased during the evening. Despite having the lowest number of users (1-5), profile 6 accounted for the greatest traffic volume, primarily due to the greater usage of streaming application in this traffic class compared to other profiles. Minimum profile memberships (active users) were recorded between 03:00-06:00hrs. Fig. 8 represents the corresponding upload and download rate per user profile including per application and consolidated (x) data rates. In terms of individual application tiers, storage (g) had the highest upstream and downstream rates (4-5Mbps), followed by streaming (s) downloads (up to 5Mbps). Web browsing had the minimum data rate footprint for both upstream (0.2Mpbs) as well as downstream (0.6Mpbs) traffic. The minimum flow rate

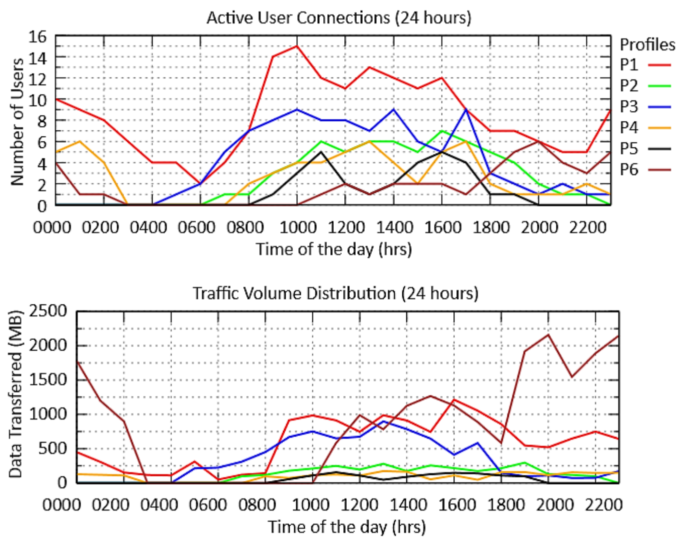

Fig. 7. Probability distribution: (a) profile memership and (b) traffic volume

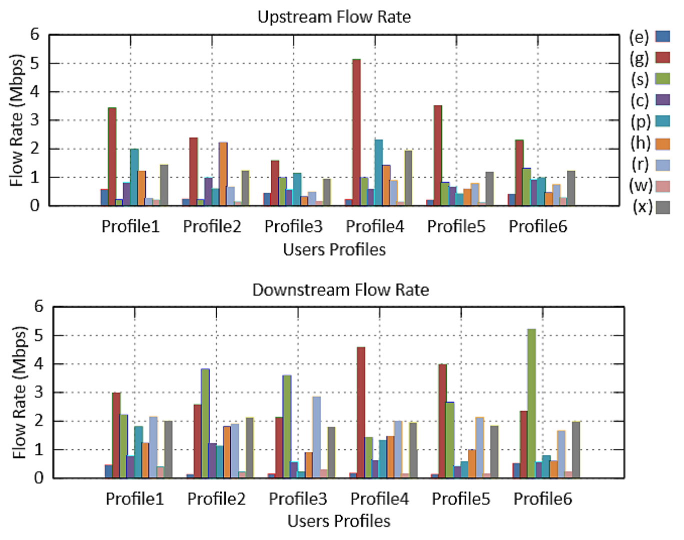

Fig. 8. Flow transfer rates: (a) upstream traffic and (b) downstream traffic

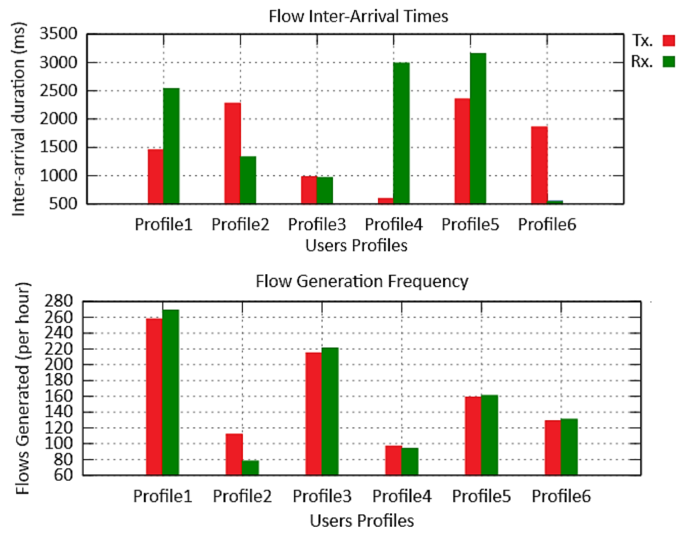

Fig. 9. Flow statistics: (a) flow inter-arrival time and (b) total flows (hourly)

was due to the relatively slower response of external web servers compared to local campus sources. The average data rates (x) remained consistent across all profiles ranging between 1-1.5Mbps for the upstream compared to 1.8-2Mbps on the downloads. The corresponding inter-flow arrival times per profile (for active users) on an hourly basis are given in Fig. 9. Profile 1 had the highest amount of flows generated and received (240-260 flow) per hour for active users, again due to the greater profile membership attributed to this user class. The lowest flow generation was for users in profile 2 (80-100 flows) mainly constituting communication service usage. Similarly profile 4 had the minimum inter-flow arrival duration (575ms) for transmitted flows, showing quick use of print services when active. Profile 6, concentrating on streaming had the minimum inter-flow arrivals for downstream traffic (275ms) mainly attributed to dynamic download of streaming content from multiple sources i.e. load-balanced video servers.

In view of the discriminative application trends, profile memberships and associated flow measurements for the evaluated campus network segment depicted in the derived user profiles, operators may want control over which users to prioritize in terms of bandwidth allocation as well as select optimal routes for resource intensive profiles. An overview of integrating profiling based controls in campus SDN is discussed further in section VI.

B. *Profile Stability*

Profile consistency highlights the significance of gaining a better insight to changes in user activity as well as benchmark the stability of the extracted profiles and re-profiling frequency. Therefore, to evaluate user profile retention, the average probability of profiles change for the same users for each subsequent day of study was computed and is presented in Table 3. Profile 5 users showed the highest consistency in retaining profiles at 99.1% followed by profile 4 at 99% while profile 6 showed the lowest at 96.1%. The reported profile retention of campus users was greater in comparison with a similar study aimed at evaluating profile stability for multi-device residential users reporting the lowest profile consistency at 81% [13]. Campus users hence showed a significantly greater degree of consistency in daily application usage in relation to residential users. The probability of a profile gaining or losing a device every 24 hours is also given in Table 3. Profile 1 had the highest probability of gaining users (93%) profile 6 had highest probability of losing users (80%). The average probabilities of inter-profile transitions every 24 hours are given in Table 4. Profile 6 users showed a tendency (up to 3%) to transition to profile 1, the primary difference between the two profiles being proportional changes in streaming and publishing tier respectively. Similarly, profile 1 users tilted towards profile 3 (1.1%) having greater corporate information system and staff portal usage. Profile 4 with heavy publishing inclined towards profile 1 (at 0.6%) having higher web access. It was therefore, noted that where users transitioned to a different profile, it was always to profiles having somewhat similar application usage ratios to their own. Inter-profile transitions were mainly due to proportional variation in the same kind of user activity rather than a complete change of user roles, increasing the applicability of derived profile baseline in campus network monitoring.

TABLE III. AVERAGE PROBABILITY OF PROFILE CHANGE (/24 HOURS)

User Profiles	**Probability of No Change**	**Probability of Change**		
		Change:	***Gain***	***Loss***
Profile 1	0.981	0.019	0.93	0.07
Profile 2	0.970	0.03	0.44	0.56
Profile 3	0.985	0.015	0.92	0.08
Profile 4	0.990	0.01	0.55	0.45
Profile 5	0.991	0.009	0.49	0.51
Profile 6	0.961	0.039	0.20	0.80

TABLE IV. INTER-PROFILE TRANSITION PROBABILITY (/24 HOURS)

User Profiles	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**
Profile 1	0.981	0.001	0.011	0.002	0.004	0.001
Profile 2	0.009	0.970	0.017	0.001	0.001	0.002
Profile 3	0.008	0.001	0.985	0.003	0.002	0.001
Profile 4	0.006	0.001	0.001	0.990	0.002	0.001
Profile 5	0.001	0.003	0.001	0.002	0.991	0.002
Profile 6	0.03	0.0091	0.001	0.0004	0.0003	0.961

C. *Profiling Computational Cost*

To evaluate the computational cost of the traffic profiling mechanism, memory and CPU utilization were recorded during the profiling workload completion. The purpose of this exercise was to appreciate the amount of computational resource needed in profiling user traffic from a practical campus network setting. The test machine (PC) used an Intel based i54310-M processor chipset with two CPUs, each at 2.70 GHz and 16GB of RAM. The operating system used a GNU/Linux kernel (3.14.4 x64) and it was verified that no other user processes (apart from the profiling engine) were consuming CPU cycles or any of the inherent operating system processes were CPU or I/O intensive. Fig. 10 illustrates the memory and CPU utilization vs. the number of records processed. The initial spike observed in CPU and memory utilization during the startup was further followed by

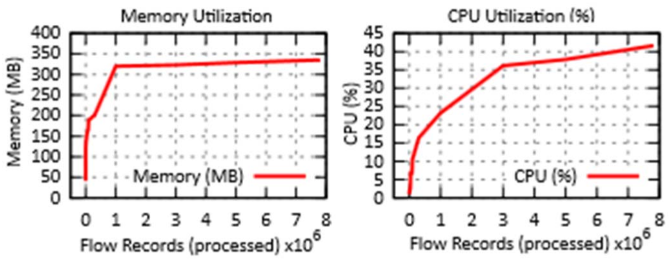

Fig. 10. Profiling overhead: (a) memory and (b) CPU utilization

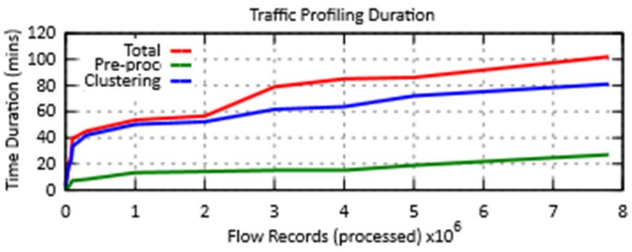

Fig. 11. Traffic profiling duration vs. traffic records processed

a breif linear curve for both resources in relation to the number of flow records processed. However, with continued increase in the number of records (≥1 million records) memory utilization reached a steady-state pattern having a maximum observed value of 335MB. CPU utilization on the other hand continued to increase with maximum value of 41.63% for the total 7.8 million records. Similarly, the time duration involved in processing the records is given in Fig. 11. As evident from the graph a substantial portion of the total time was spent in clustering compared to pre-processing (collating) per user statistics. The total duration for processing 7.8 million records was approximately 103 minutes. The observed CPU and memory footprint required in processing an order of x10^6 monitoring records highlight the viability of the proposed profiling mechanism in an online campus implementation. A dedicated server with relatively additional memory, particularly CPU power may be employed for profiling which may further expedite the clustering process and reduce the total profiling duration.

D. *Control Channel Overhead*

In addition to the profiling resource computation cost, it is important to consider the traffic workload added to the OpenFlow control channel as a result of the statistics collection required for traffic profiling. To evaluate the load attributed to the control channel due to the proposed customization i.e. profiling traffic from an edge switch, the experimentation workload was emulated in Mininet using Ostinato traffic generator utility [15]. Our analysis of the workload accounted (i) the monitoring information required for traffic profiling including the *packet_in*, *packet_out* and *flow_rem* messages, and (ii) the polling of flow tables via *flow_stat* and *table_stat* messages at regular 30s intervals. Fig. 12 presents the topology and the related traffic simulation parameters are given in Table 5. The topology comprised of 12 user machines, six representing each of the derived profile users and the remaining six sourcing campus server traffic. The traffic load was gradually increased starting from 10 users per profile up to a maximum of 100 users per profile to measure the control channel overhead generated by the edge switch. Employing the default OpenFlow behaviour, *packet_in* messages included the first complete packet of incoming flows as opposed to the alternate option of buffering the packet in the switch and sending *buffer_id* with routing request to the controller which requires considerably greater switch memory [2]. Using default OpenFlow option ensured the evaluation scaled to typical switch configuration. The resulting control channel traffic with varying workload is given in Fig. 13. The bulk of the traffic comprised of flow control *packet_in* and *packet_out* messages from the controller with minimum traffic attributed to statistics collection (*flow_stat*, *table_stat*, *flow_rem*) messages. This was primarily due to the relative size of *packet_in* and *packet_out* messages compared to counter polling messages. At the maximum user load of 600 users (100 users per profile), a total of 697 upload packets and control traffic rate of 326kbps was observed. On the downstream, the packet rate remained lower due to absence of switch-controller *flow_rem* messages, peaking at 676 packets. Downstream traffic rate was around 353kbps suggesting swifter processing on the controller side than upstream. The present scenario considered the messaging overhead of traffic

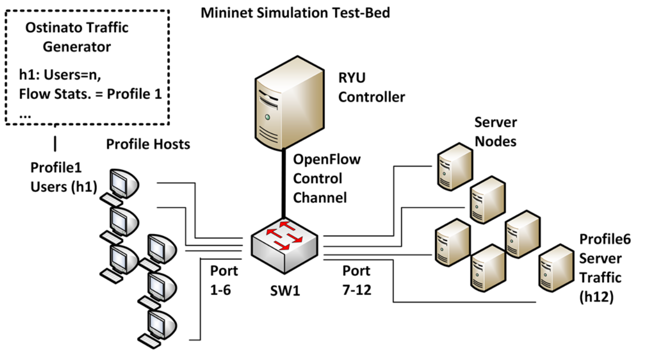
Ostinato Traffic Generator
h1: Users=n, Flow Stats. = Profile 1
...
Mininet Simulation Test-Bed
RYU Controller
Profile Hosts
Profile1 Users (h1)
OpenFlow Control Channel
Server Nodes
Port 1-6
SW1
Port 7-12
Profile6 Server Traffic (h12)

Fig. 12. Control channel overhead evaluation – Mininet topology

TABLE V. TRAFFIC CONFIGURATION PARAMETERS OF SIMULATION

Parameter	Value	Remarks
SDN related	OpenFlow: v1.4	Default behaviour; idle_timeout 60s; traffic monitor polling 30s
	Open vSwitch: v1.3.1	
	Ryu Controller: v3.3	
	Mininet: v2.21	
	Ostinato: v0.7.1	
Workload	10-100 users per profile	Min: 60, Max: 600 users
Runtime	15 minutes per workload	-
Flow duration	5.61-31.13s	Source: user profile flow statistics
Flow frequency	190-527 per hour	Source: user profile flow statistics
Packet size	64-1480 Bytes	Random variation by Ostinato generator

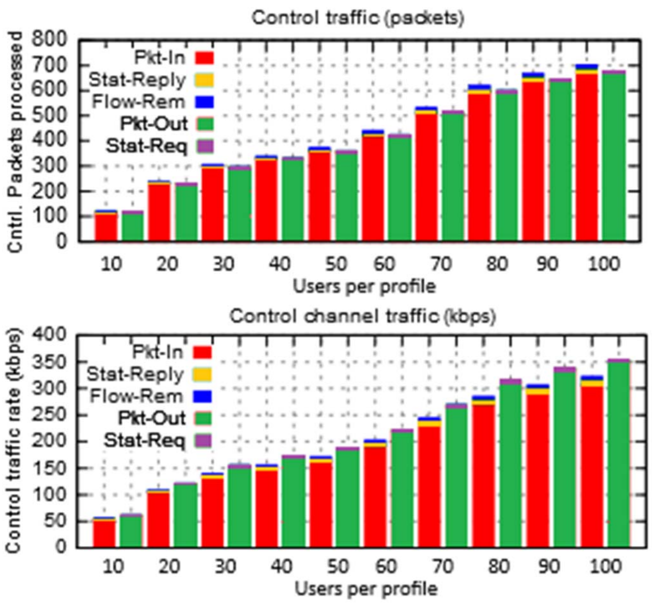

Fig. 13. Control channel (a) control packets (b) control traffic rate (kbps)

monitor in addition to the forwarding control messages, presenting majority of the edge switch bound control traffic. Any additional flow control traffic, i.e. *flow_mod* messages sent to intermediate campus switches, would be distributed depending on the underlying network topology. For an edge switch catering to approximately 600 users, the maximum bi-directional packet overhead (4.02%) and control traffic rate (4.96%) due to flow statistics collection alone poses no significant impact on existing OpenFlow channel traffic. Operators may therefore, utilize the existing network fabric (depending on capacity) to monitor edge switch user traffic from a central controller without requiring additional monitoring overlay. The next section highlights some of the applications of the proposed OpenFlow traffic monitoring solution.

V. Application: Campus Traffic Management

The extracted user traffic profiles from the campus network segment represent varying user application trends, giving network administrators an intuitive means to monitor an SDN based environment. User profiling gives administrators the ability to appreciate user tendencies and design user-centric solutions rather than focusing on individual applications as well as plan for future updates. We highlight three important avenues for integrating user profiling controls in the campus SDN framework in this regard.

1) Real-time network monitoring: User traffic profiles may provide a real-time visualization of user activity to monitor the campus network. The six extracted profiles in Fig. 6 showed considerable stability and consistency. Baseline of traffic profiles depicted in Fig. 7-9 including the time of the day profile memberships, traffic volume, the respective flow rates as well as flow generation frequency may aid the network administrator in monitoring the campus traffic in real-time via the proposed traffic monitor. Additionally, baseline statistics associated with each profile could serve as an input for timely anomaly detection, with any variation from anticipated trends triggering an alarm as well as serve as an indication for re-evaluation of the derived profiles.

2) Link management: The extracted profiles assist in the identification of resource heavy from lighter profiles. Implementation of a profile optimization scheme may allow operators to rate-limit as well as balance selected profile traffic on the available links between several departments. A similar profile prioritization and per profile traffic queueing approach tested in residential SDN to rate-limit user to service provider traffic, yielded greater bandwidth availability for high priority users under network congestion [16]. Furthermore, profile prioritization may also allow improved north-south campus-data center route selection to minimize server switch (ToR) oversubscription effects on priority users.

3) Energy conservation: A growing number of energy conservation techniques in SDN rely on switching off network components using customized controller-switch OpenFlow implementations. Determining which device subsets to dynamically switch off, as well as consolidating virtual machines to minimize active server instances, however, remains challenging [17]. Time of the day variation in profile membership, and flow statistics offers enhanced user traffic visualization which may aid operators in reducing energy consumption at the network and server level. Using profile statistics, operators could design optimal server placement algorithms according to real-time resource requirements and in tandem reduce the number of active devices (and ports), to conserve energy through efficient network provisioning.

VI. Conclusion

The present work derived six unique user traffic profiles from a campus network segment while solely using OpenFlow traffic monitoring attributes. The extracted profiles showed significant application usage discrimination among users. Furthermore, the six profiles remained largely consistent showing minimum user profile transitions over the two weeks of observation, making them viable for intuitive real-time monitoring and management of the campus SDN. Additionally, the low profiling computational cost and control channel overhead of the proposed design even at high user loads offers increased scalability for campus-wide deployment. The integration of OpenFlow-enabled user profiling controls may further allow operators to implement SDN specific user-centric traffic engineering solutions, maximize link and server utilization as well as derive energy efficient network provisioning models.

References

[1] SDN Architecture Overview. Website: https://www.opennetworking.org/sdn-resources/sdn-definition

[2] OpenFlow Cert., Website: https://www.opennetworking.org/openflow-conformance-certification

[3] T. Bakhshi and B. Ghita, "User traffic profiling in a software defined networking context," *Internet Technologies and Applications (ITA), 2015*, Wrexham, Wales, 2015, pp. 91-97.

[4] Open vSwitch. Website: http://openvswitch.org/support/

[5] Ryu Component-based SDN Website: http://ryu.readthedocs.org/en

[6] Mininet Network Emulator. Website: http://mininet.org

[7] P. H. Isolani, J. A. Wickboldt, C. B. Both, J. Rochol and L. Z. Granville, "Interactive monitoring, visualization, and configuration of OpenFlow-based SDN," *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Ottawa, ON, 2015, pp. 207-215.

[8] N. L. M. van Adrichem, C. Doerr and F. A. Kuipers, "OpenNetMon: Network monitoring in OpenFlow Software-Defined Networks," *2014 IEEE NOMS*, Krakow, 2014, pp. 1-8.

[9] S. Chowdhury, M. Bari, R. Ahmed, and R. Boutaba, "PayLess: A low cost network monitoring framework for Software Defined Networks," *in Proc. of the 14th IEEE/IFIP NOMS*, 2014, pp. 1–9.

[10] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. V. Madhyastha, "FlowSense: Monitoring Network Utilization with Zero Measurement Cost," *in Proc. of the 14th IPAM* , 2013, pp. 31– 41.

[11] Y. Zhang, "An Adaptive Flow Counting Method for Anomaly Detection in SDN," *in Proc. of the 9th ACM CoNEXT*, 2013, pp. 25–30.

[12] J. MacQueen, "Some methods for classification and analysis of multivariate observations", *the Proc. of the Fifth Berkeley Symposium on Mathematical Stats. and Prob,* 1967. vol. 1, pp. 281-297

[13] T. Bakhshi and B. Ghita, "Traffic Profiling: Evaluating Stability in Multi-device User Environments," *2016 30th IANA Workshops (WAINA)*, Crans-Montana, Switzerland, 2016, pp. 731-736.

[14] B. Everitt and T. Hothorn, "A Handbook of Statistical Analyses Using", Boca Raton, FL: Chapman & Hall/CRC, 2006.

[15] Ostinato: Packet Traffic Generator and Analyzer, Website: http://ostinato.org/

[16] T. Bakhshi and B. Ghita, "User-centric traffic optimization in residential software defined networks", *23rd International Telecommunications Conference*, Thessaloniki Gr., 2016, pp. 247-252.

[17] H. Shirayanagi, H. Yamada and K. Kono, "Honeyguide: A VM migration-aware network topology for saving energy consumption in data center networks," *Computers and Communications (ISCC), 2012 IEEE Symposium on*, Cappadocia, 2012, pp. 460-467.