

User-Centric Network Provisioning in Software Defined Data Center Environment

Taimur Bakhshi, Bogdan Ghita

Center for Security, Communications and Network Research

University of Plymouth

Plymouth, United Kingdom

{taimur.bakhshi, bogdan.ghita}@plymouth.ac.uk

Abstract— Present data center (DC) network provisioning schemes primarily utilize conventional load-balancing technologies, offering individual application performance improvement. Diversity in application usage however, makes isolated application prioritization a performance caveat for users with varying application trends. The present paper proposes a user profiling approach to capture application trends based on generic flow measurements (NetFlow) and employs the extracted profiles to create DC traffic forwarding policies. The scheme allows operators to define a global profile and application hierarchy based on extracted profiles to prioritize traffic for individual user classes. The proposed design was tested by extracting user profiles from a realistic enterprise network, and further simulated to dynamically manage DC traffic using the software defined networking paradigm. Compared to conventional traffic management schemes, the frame delivery ratio and effective throughput of our design was significantly higher for high priority north-south user traffic as well as the inter-server east-west application traffic.

Index Terms—Data center networking, software defined networking, user traffic profiling

I. INTRODUCTION

Swift coordination of changes for service provisioning in conventional data center (DC) networks is rather complex, involving the implementation of distributed protocols to facilitate multiple services for traffic routing, switching and guaranteeing application quality of service [1]. The recently emerged paradigm of software defined networking (SDN) proposes a fundamental shift in real-time traffic management through the separation of network control and traffic forwarding features. SDN offers automated on-demand resource allocation as well as affords faster state changes in data centers, a significant improvement over manually intensive conventional configuration techniques [2-4]. However, traffic measurements in the DC network show tremendous volatility in network workload when multiple applications are sharing the same physical or virtual fabric [5]. Both enterprise users as well as cloud subscribers may comprise several user classes representing varying application trends sharing the same data center infrastructure. Using stringent application bandwidth guarantees to manage traffic utilizing conventional technologies such as spanning tree

protocol (STP) or dynamic equal cost load balancing over multiple paths (ECMP) does not fully account for real-time application usage diversity among users, often resulting in service trampling and over use of one application affecting others [5-6]. Furthermore, chained service delivery architecture sometimes necessitates communication between multiple servers to enable user request fulfilment causing traffic fluctuations and rendering per-application bandwidth allocation impractical [7]. Per-application bandwidth guarantees and weighted bandwidth sharing models, therefore, fail to provide operators the desired granularity to fully optimize real-time network provisioning in view of actual user workloads. Seamless application delivery via SDNs, demands a more user-centric approach accounting for real-time user application trends and an accurate customization of prerequisite virtual and physical resources in data centers to meet user requirements. The primary contributions of this paper are as follows. 1) Improving workload characterization in data centers by deriving user traffic profiles based on application trends captured using generic Netflow measurements, and allowing operators to define profile and application hierarchy for computing routing paths between service endpoints. 2) A traffic management algorithm employing the profile hierarchy along with anticipated per-profile traffic in assigning north-south routes between users and front-end servers as well as east-west inter-server traffic routes in the DC. The rest of this paper is organized as follows. Section 2 presents related work in optimizing traffic in the DC. Section 3 gives the design overview while profile derivation scheme is detailed in section 4 and the network provisioning algorithms in section 5. Section 6 highlights the simulation environment and evaluates the proposed design along with a discussion of the results. Section 7 draws final conclusions.

II. RELATED WORK

Allocating resources in the DC is quite frequently based around individual application requirements. Application connectivity models were proposed as a possible solution in [8] and [9] to allocate per-application network bandwidth. However, application delivery constraints due to several applications simultaneously competing for resources are prevalent in data centers. To address DC bandwidth contention,

Kumar et. al [3] employed user space daemons running on application servers to anticipate traffic bursts and assign forwarding paths to individual applications. Fang et al. [4] proposed implementing host congestion controls to prevent excessive traffic influx into the network and multipath selection to achieve optimal network resource utilization. Jeyakumar et. al [10] viewed application bandwidth guarantees to be too stringent and proposed a weighted bandwidth sharing model among nested service endpoints allocating resources hierarchically at core fabric, rack, and individual machine level. The resulting operator defined per-application bandwidth sharing schemes in each of the above scenarios were, however, highly dependent on the stability of application demands for long enough periods to manage network traffic. Efforts to relieve bandwidth contention from a network design perspective have seen the deployment of Clos networks gain momentum which counter link oversubscription by using large number of smaller switches making failures localized and the architecture more cost-effective. Greenberg et. al [5] proposed VL2, a data center framework using 3-layer Clos topology and decentralized load balancing to spread traffic across all available paths but instead of using per-application bandwidth allocation heavily relied on TCP windowing to rate limit flows. The datacentres for the Facebook social networking website also reported using Clos topology to cope with the substantially high internal traffic [7]. Despite clustering servers from each application tier in close proximity, the resulting Clos topology and cabling was considered to be incredibly complex. High end network vendors propose solutions primarily recommending unification of services to improve performance [11]. However, the application differentiation available at system and network level to assign machine limits and create end-to-end network topology per application does not explicitly consider user application trends. Resource provisioning on a per-application basis, therefore, leads operators to pre-set network provisioning models dictating end-user experience regardless of real-time network use. A more user-centric approach utilizing traffic profiling to derive a resource abstraction model, could instead offer service providers the ability to fine tune per-user (profile) network provisioning in view of individual user and business requirements instead of isolated applications.

III. DESIGN OVERVIEW

The proposed design comprises of two components a) a traffic profiling scheme used to discriminate user classes based on application trends and b) a traffic management mechanism utilizing the derived profiles to define and implement network provisioning policies as illustrated in Fig. 1. The traffic profiling scheme includes the measurement of user-generated NetFlow records at the data center edge further subjected to unsupervised cluster analysis to segregate users into traffic classes (profiles). The extracted user profiles record application usage ratios, along with anticipated user bandwidth requirements which are used by the traffic manager (using the SDN controller) in computing forwarding paths per-profile according to operator defined user profile priority (table), keeping track of the real-time profile memberships. The

computed paths for both external and internal traffic are installed and updated in individual network elements (access, aggregate and edge switches) using OpenFlow protocol [17]. The specifics of user profiling scheme and traffic management approach are detailed in the following sections.

IV. USER PROFILING SCHEME

User profiling requires recording of user traffic traversing the data center network and the identification of per user flows. For this purpose, NetFlow records are collected at traffic admission points i.e. the edge (core) routers. Following flow collection, the application usage ratios are quantized based on the number of user-generated flows towards each subscribed data center application (front-end server) to derive user traffic composition statistics. To measure the total number of user flows per application, the number of flows α destined to each front-end server need to be accounted as shown in Fig. 2. For user flows α_s towards front-end application server (s), the total flows per application are given by $\Sigma \alpha_s$. The corresponding vector depicting the total application usage as a function of all user u generated flows having n application subscriptions is given in (1).

$$u = [\alpha_1 / \sum_{s=1}^n \alpha_s, \alpha_2 / \sum_{s=1}^n \alpha_s \dots \alpha_n / \sum_{s=1}^n \alpha_s] \quad (1)$$

Once traffic composition vectors have been derived, users are partitioned into groups based on their application usage profile through an un-supervised k-means clustering algorithm [12-13]. Compared to other methods, such as hierarchical agglomerative clustering, k-means is resource efficient and produces tighter clusters resulting in meaningful user profiles as evaluated in [14-15]. K-means minimizes a given number of vectors by choosing k random vectors as initial cluster centers and assigns each vector to a cluster as determined by distance metric comparison with the cluster center (a squared function) given in (2). Cluster centers are then recomputed as the average of the cluster members. This iteration continues repeatedly, ending either when the clusters converge or a specified number of iterations have passed.

$$P = \sum_{a=1}^k \cdot \sum_{b=1}^n || u_b^a - c_a ||^2 \quad (2)$$

In (2), c_a represents the cluster center, n equals the size of the sample space translating for the number of users and k is the chosen value for number of unique clusters (centroids). Hence, using k-means, n users, can be partitioned into k groups or profiles based on their per application usage given by (1). The value of k is of significant importance as it directly influences the number of traffic profiles and affects over-fitting of users into profiles. To find the optimal number of clusters (user profiles) reflecting user activities the profiling scheme uses the Everitt and Hothorn technique [16]. This technique aims at plotting 'within groups sum of squares distance' in each cluster against k , with the knee in the curve suggesting an appropriate number of profiles having reduced within-cluster variance. The derived profile clusters segregate users according to application trends and provide per-profile bandwidth requirements.

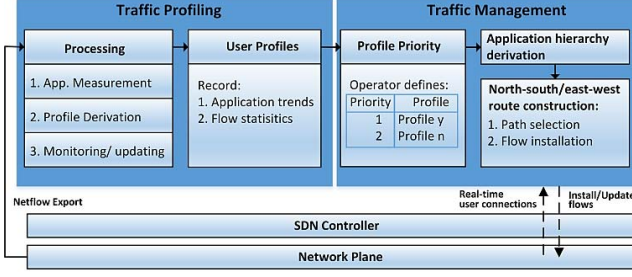


Fig. 1. Traffic profile derivation and network management

A. Flow statistics

The implementation of per-profile traffic policies requires determining the projected flow rates between a) *external* users and front-end application server traffic per profile and b) *internal* intra-server traffic generated per application in response to each user request. Given the per-profile transmitted and received data transfer rate z for each front-end application server collected over time t , we calculate the maximum probability for data rate z to take on a given value between the lowest and greatest values of the range (x, y) using the probability density function given by (3).

$$\Pr [x \leq z \leq y] = \int_x^y f_z(z) dz \quad (3)$$

We measure the maximum probability of z for both inbound and outbound (external) traffic per profile for each subscribed application independently. The respective data transfer rates for inbound and outbound (external) flows for application α_i belonging to profile P_k can therefore be given by $\Pr_{\max} [z_{\text{in/out}}(P_k, \alpha_i)]$ and the corresponding flow inter-arrival times by $\Pr_{\max} [\Delta t_{\text{in/out}}(P_k, \alpha_i)]$. While the external north-south traffic between users and front-end servers is relatively easy to record, the internal flows generated between the application servers per user connection are subject to significant variation and greatly depend on the application logic as well as the respective server connectivity. We therefore, define a maximum threshold value for the inter-flow arrival time $\Delta t_{\text{internal}}$ and the respective data rate z_{internal} between internal servers per-application in response to each user (profile) connection. Flow rates are generated up to a high set threshold to evaluate the effect of varying inter-server traffic on the proposed design, as detailed later in section 6. A summarization of flow parameters recorded and used for real-time traffic management are summarised in Table 1.

B. Profile stability and re-evaluation

The derivation of meaningful user traffic profiles depends on selecting suitable data (flow) usage window which dictates clustering efficacy and subsequent traffic management performance. Prior profile sensitivity analysis [15] indicates highly converged cluster (profile) derivation when flow records are concatenated using 24 hour time-bins to account for the time dependency of user activities, which may drastically differ with the time of the day. The present study therefore, used a similar flow record usage window to profile users in the

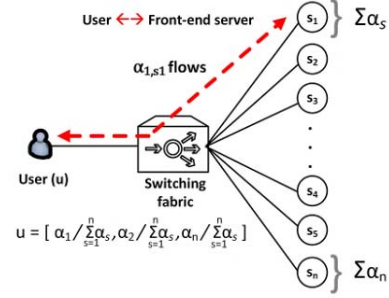


Fig. 2. User generated traffic flows per application

enterprise network segment, detailed later in section 6. Additionally, the frequency and time of user profile generation also has to be practical (sensitive enough) to accommodate any temporal traffic anomalies without affecting the efficacy of profile clusters. The present design proposes employing the north-south profile flow parameters summarized in Table 1 to monitor profile stability. The computed flow probabilities provide an intuitive means to monitor profile traffic trends in real-time, with any anomalous (out-of-profile) pattern serving as an advisory for operator to reevaluate profiles and policies.

V. TRAFFIC MANAGEMENT APPROACH

This section describes the proposed traffic management scheme, comprising of four procedures: profile priority and application hierarchy, north-south and east-west route construction, scheduling frequency of the respective flow computations, and reducing flow table sizes in individual network elements to expedite traffic forwarding.

A. Profile priority and application hierarchy

Conventional Ethernet uses best effort delivery of traffic which is prone to dropping of frames in face of network congestion in the DC. While technologies such as equal cost multipath (ECMP) promise higher throughput by leveraging multiple links over the legacy spanning tree protocol (STP), these alone are inadequate in policing traffic in view of the inherent mix of user classes present in the data center environment [4]. To account for the real-time traffic load and different priority user class flow forwarding, operator-defined profile priority tables shown in Fig. 3 are introduced. Based on the application usage weighting per-profile, a global application hierarchy table is also derived. The priority lists aim to reduce the effects of network congestion for different user categories (in order of hierarchy) by routing flows on different paths between the core, aggregate and access switches for external traffic. Additionally the application hierarchy table is used to prioritize route allocation for inter-server traffic between dispersed application servers connected to different access switches.

B. Route construction

In the context of modelling and evaluating the proposed design, Fig. 4 presents a three layer inter-connected network topology prevalent in modern datacentre architectures. Fig. 5 provides the associated network provisioning algorithm for

optimizing internal and external traffic flows. The algorithm computes the inbound and outbound forwarding links per profile for each front-end application server before moving to the next, in order of profile and application hierarchy. A link threshold l_T gives the maximum available $\beta_{\max}$ bandwidth per link. If l_T is greater than the required inbound or outbound flow rate for u profile users ($l_T > u.z$), the link is selected for forwarding. Otherwise users are split between alternate routes with flows equating l_T/z using the selected link and the rest ($u - l_T/z$) split over alternate links as computed during the subsequent iteration. The available bandwidth per link is correspondingly updated ($\beta_{(E,D)}(D,A) = \beta_{(E,D)}(D,A) - u.z$ inbound/outbound). The resulting flow configurations are pushed via OpenFlow *flow_mod* messages to switches using SDN controller [17]. The same process is repeated until all links are at full capacity ($l_T = 0$), triggering respective flow forwarding via the link used by preceding application/profile. The algorithm ensures that higher priority profiles continue to experience higher throughput in the event of network congestion by routing lower priority profile flows over links with relatively higher congestion. Similarly, once north-south flow construction for all profiles is complete, the east-west route construction uses the maximum available bandwidth $\beta_{\max}$ over each link (l_T) to optimize inter-server traffic according to the derived application hierarchy. If the pre-set internal flow rate threshold z_{internal} per application for u users is greater than the link threshold l_T , the route is selected for flow forwarding otherwise l_T/z_{internal} flows are forwarded through the link with remaining flows split over alternate links. The process continues for each application until only one route with $l_T > 0$ remains between the respective switches, otherwise the last installed route is used for forwarding traffic. Conventional multipath load balancing schemes (ECMP, DMBP, etc.) split traffic flows at frame level and each path having a different delay causes out-of-order frame delivery. This results in TCP interpreting these reordered frames as a sign of congestion which ultimately results in degraded performance [18-19]. Our proposed approach ensures that all frames per flow (for each user) are forwarded over the same path (set of links) to preserve ordered delivery and any splitting of user traffic only occurs at the flow level. The scheduling frequency of route construction to keep the management and computational overhead to a minimum is presented in the next sub-section.

C. Route updates scheduling

Each installed route, by default can accommodate additional users (per link) dictated by inter-arrival times Δt . The north-south flow construction algorithm installs flows per link utilizing $z_{\text{in|out}}$ flow rates, however, flow inter-arrival time of $\Delta t_{\text{in|out}}$ allows the selected link to accommodate users equalling $u_{(ED||DA)}$ given by (4). Similarly, for inter-server routes with flow inter-arrival time threshold $\Delta t_{\text{internal}}$, the selected link is capable of handling flows $\text{flow}_{(AD||DE)}$ given by (5). The high inter-arrival times therefore, translate into a higher tolerance of the installed flows to user connection updates requiring less frequent re-evaluation. The SDN controller performs real-time monitoring of users via edge/core switch port monitoring using an average flow threshold to track new/existing and stale user

TABLE I. PROFILE FLOW PARAMETERS

Parameter	Usage
North-south flow rate ($z_{\text{in out}}$)	- North-south route construction. - Flow updates scheduling.
Flow inter-arrival time ($\Delta t_{\text{in out}}$)	- Profile stability monitoring.
Inter-server flow rate (z_{internal})	- East-west route construction.
Inter-server flow arrival threshold ($\Delta t_{\text{internal}}$)	- Flow updates scheduling.

connections. The controller periodically re-computes external as well as internal forwarding over link(s) following additional user connections when $u_{(ED||DA)} > u_{\max}$ or $\text{flow}_{(AD||DE)} > \text{flow}_{\max}$, and immediately on disconnections.

$$u_{(ED||DA)} = u_{\max} / (1 - \Delta t_{\text{in|out}}) \quad (4)$$

$$\text{flow}_{(AD||DE)} = \text{flow}_{\max} / (1 - \Delta t_{\text{internal}}) \quad (5)$$

Furthermore, as depicted in Fig. 6, re-evaluation of installed routes only requires the forwarding routes of the profile with updated users and subsequent lower priority profile routes along with the inter-server links to be updated, leaving higher priority pre-installed flows in force.

D. Reducing flow table size

In view of the limited capacity of switch memory and the significant user numbers per profile, we use network address translation (NAT) to place users (per profile) into segregated

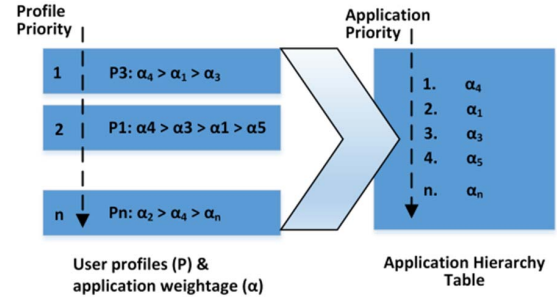


Fig. 3. Profile and application hierarchy

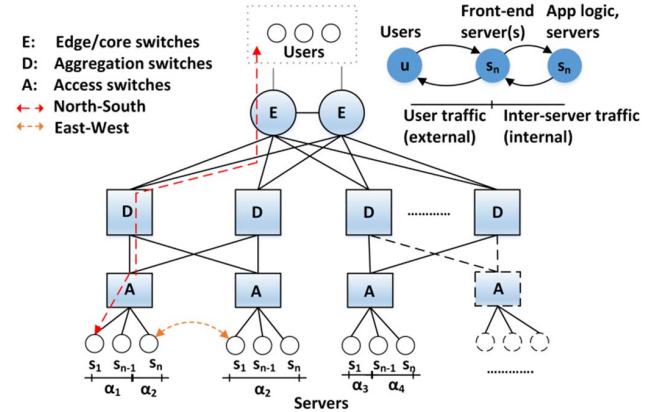


Fig. 4. Three layer data center topology

```

start function path_selection (external)
for (profile priority P=1, P<=n; P++)
for (application priority  $\alpha=1$ ;  $\alpha \leq n$ ;  $\alpha++$ )
#Path selection inbound & outbound traffic, *=last used link
do
for  $\forall l_{(E,D)} || (D,A)$ 
 $l_{(E,D)} || (D,A)^* = \beta l_{(E,D)} || (D,A)^*$ 
 $u_{\max(E,D)} || (D,A) \leftarrow l_{(E,D)} / Z_{inbound} || outbound$ 
if  $u_{\max(E,D)} || (D,A) \geq u$  &&  $l_{(E,D)} > 0$ 
flow_mod [ $l_{(E,D)}, l_{(D,A)}$ ];
 $\beta l_{(E,D)} || (D,A) = \beta l_{(E,D)} || (D,A) - u \cdot Z_{inbound} || outbound$ ;
end if
else if  $u_{\max(E,D)} || (D,A) < u$  &&  $l_{(E,D)} > 0$ 
flow_mod [ $l_{(E,D)}, l_{(D,A)}$ ]max_users;
 $\beta l_{(E,D)} || (D,A) = \beta l_{(E,D)} || (D,A) - u \cdot Z_{inbound} || outbound$ ;
 $u = u - u_{\max}$ ;
end else
else if  $l_{(E,D)} < 0$ 
flow_mod [ $l_{(E,D)}^* || l_{(D,A)}^*$ ];
end else
end for
while  $u > 1$ ;
end for
end function

```

```

start function path_selection (internal)
for (application priority  $\alpha=1$ ;  $\alpha \leq n$ ;  $\alpha++$ )
#Path selection internal server-server traffic, *=last used link
do
for  $\forall l_{(A,D)} || (D,E)$ 
 $l_{(A,D)} || (D,E)^* = \beta l_{(A,D)} || (D,E)^*$ 
 $flow_{\max} = \beta l_{(A,D)} || \beta l_{(D,E)}$ 
if  $l_{(A,D)} || (D,E) \geq Z_{internal} \cdot u$  &&  $l_{(E,D)} > 0$ 
flow_mod [ $l_{(A,D)}, l_{(D,E)}$ ];
 $\beta l_{(A,D)} || (D,E) = \beta l_{(A,D)} || (D,E) - Z_{internal} \cdot u$ ;
end if
else if  $l_{(A,D)} || (D,E) < Z_{internal} \cdot u$  &&  $l_{(E,D)} > 0$ 
 $flow_{\max} = \beta l_{(A,D)} || \beta l_{(D,E)} / Z_{internal}$ ;
flow_mod [ $l_{(A,D)}, l_{(D,E)}$ ];
 $u = u - flow_{\max} \cdot Z$ ;
 $\beta l_{(A,D)} || (D,E) = \beta l_{(A,D)} || (D,E) - Z_{internal} \cdot u$ ;
end else
else if  $l_{(E,D)} < 0$ 
flow_mod [ $l_{(E,D)}^* || l_{(D,E)}^*$ ];
end else
end for
while  $u > 1$ ;
end for
end function

```

Fig. 5. Route construction (north-south; east-west)

subnets, each identified with a VLAN ID. A schematic representing pipeline processing of the network traffic towards application server through individual switches is depicted in Fig. 7. OpenFlow table0 in S1 (edge switch) translates and sets NAT IP addresses to profile users and forwards processing to table1. Flow table1 assigns a VLAN ID per subnet and the output port for outbound flows. Identification of per-profile flow traffic towards the application server can in turn, therefore, be identified only by the destination address and the VLAN ID to select outgoing ports in each subsequent switch on the path. Internally within the DC, east-west server traffic uses the respective server MAC addresses for flow referral in switch tables to further reduce latency and achieve traffic transfer close to line rate.

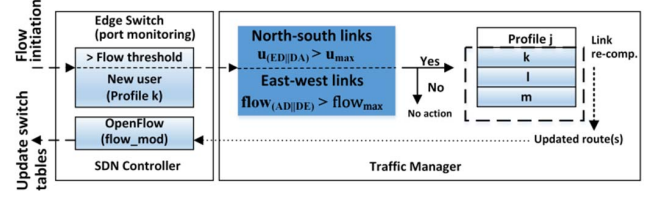


Fig. 6. Route updates scheduling (rs)

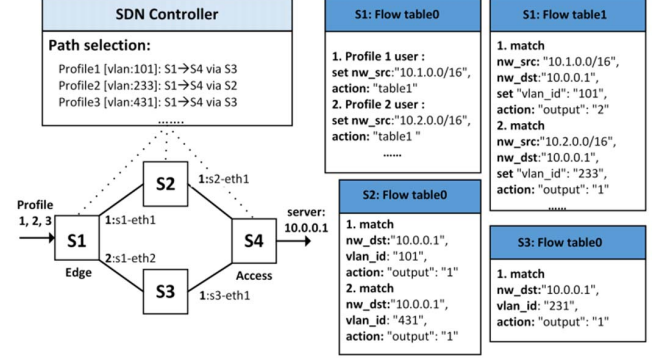


Fig. 7. OpenFlow pipeline processing –reducing flow entries in switches

VI. EVALUATION

To evaluate the effectiveness of the proposed traffic management method in an actual network scenario, user traffic profiling was carried out in a real enterprise network of an academic environment. The derived user traffic profiles were afterwards used in a simulated DC environment using Ryu SDN framework to optimize north-south traffic between users and the front-end servers as well as inter-server traffic within the data center. The derived profiles, DC simulation topology and traffic optimization results are presented as follows.

A. Traffic profile derivation

The profile derivation scheme used NetFlow records exported from the edge of an academic enterprise network segment transmitted between user devices and consolidated DC application servers over a span of ten weeks from 01/12/2015 to 15/02/2016. NetFlow records were collected (exported) at default 5 minute interval and a total of approximately 72.2 million flows were examined. To account for replication in nature of user activities, applications were further grouped into distinct categories depicted in Fig. 9 (a). User traffic was classified by matching user flows against destination IP addresses used by servers and the resulting flows were concatenated every 24 hours. The corresponding traffic composition vector depicting application usage distribution as percentage of generated flows for a user u_i as per (2), for one day (24 hours) of activity (01/12/2015) is given in (6).

$$u_{i[01/12/2015]} = [2.3 \ 8.4 \ 25.6 \ 23.1 \ 11.2 \ 10.3 \ 15.5 \ 2.2 \ 1.4] \quad (6)$$

The resulting traffic composition vectors for all users were subjected to k-means cluster analysis (2). To determine the optimal number of clusters (translating for traffic profiles)

appropriately reflecting user activities, the derived clusters were examined starting from $k=2$, using Everitt and Hothorn technique described earlier. The corresponding plot of ‘within groups sum of squares distance’ per observation in each cluster against k for present data is given in Fig. 8 where a significant drop can be seen up to a cluster size $k=6$, and minimal subsequent variations up to $k=20$, indicating an optimal value of 6 profiles for the entire subscriber base. The resulting traffic profiles ($k=6$), detailing the application trends among user traffic profiles as a percentage of user generated flows are given in Fig. 9 (b). Since general network service traffic such as DNS and multicast traffic is not a user-triggered application but a functional one, hence, it was excluded while clustering users and later separately calculated as a percentage of total flows generated per user profile. From a network management perspective the resulting profiles showed significant variation in user activity. For example, Profile 1 concentrated mainly on web browsing (53.2%) with relatively limited usage of other applications apart from the corporate information and content management services. Profile 2 focused on using office instant messenger and VoIP with limited use of content management applications and minimal use of others. Profile 3 heavily inclined towards corporate information service (65.2%) with a significant use of email service (10.3%) compared to other profiles. Profile 4 heavily tilted towards document and print content creation with fairly considerable use of storage facility (11.8%). Profile 5 mainly used centralized storage server with small use of content and corporate information server. Profile 6 was a mix of web browsing, emails, streaming and corporate applications. Hence, each enterprise user profile represented a significant discrimination towards a certain mix of services. The use of software distribution was however, significantly low

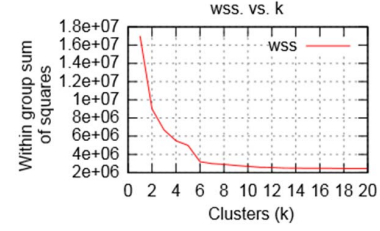


Fig. 8. Application clusters (k): wss vs. k graph

compared to other applications among all user profiles. The upload and download rate per profile derived using the probability density function given in (3) are represented in Fig. 10 (a). The corresponding inter-flow arrival times per profile are given in Fig. 10 (b). In view of the discriminative application trends depicted in the derived user profiles and varying flow rates, operators may want control over which users to prioritize in terms of network bandwidth allocation. A sample profile priority table and application hierarchy is employed in the following sub-section aimed at increasing the throughput of high priority profile users in the face of network congestion.

B. Simulation environment

The proposed design was evaluated using Mininet network emulation software [20] utilizing Ryu SDN framework [21] and the derived user traffic profiles in a DC topology comprising a total of eighty servers, one per application tier as shown in Fig. 11. The servers for each application were dispersed between disjoint pods to evaluate the traffic management design under a high inter-server traffic scenario. Ostinato traffic generation utility [22] was used for modelling north-south and east-west flow rates according to the derived

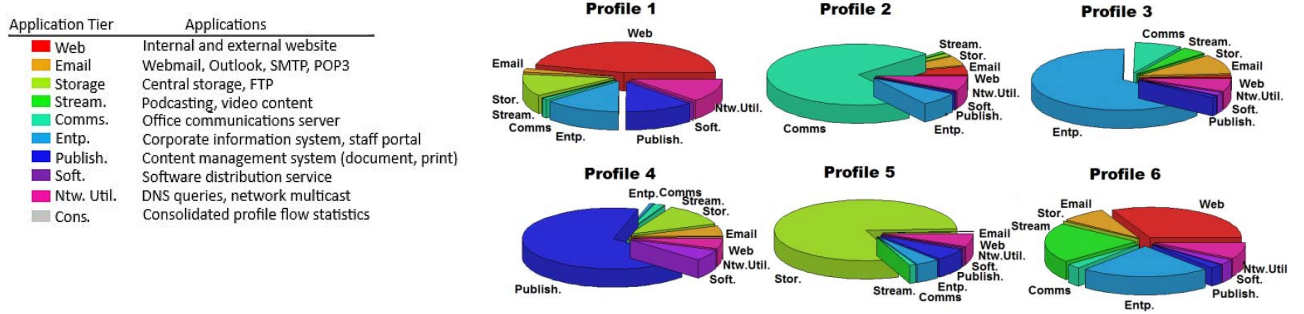


Fig. 9 (a) Application tier definition and (b) User traffic profiles

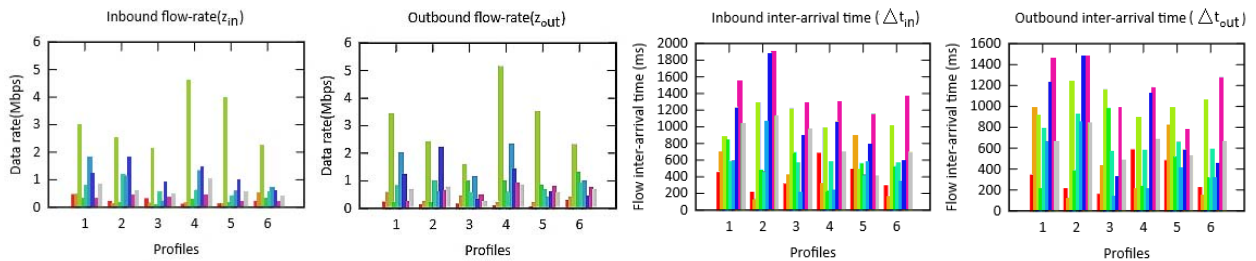


Fig. 10 (a) Flow rates per profile ($z_{in/out}$) and (b) Flow inter-arrival time per profile ($\Delta t_{in/out}$)

statistics as per Fig. 10 with an effective flow threshold set to one flow per user in the simulation to identify active users. Given the enormity of inter-server traffic within the DC compared to north-south traffic as discussed in [4], [7] and [9], the threshold for internal flow rates between each pair of application servers (Z_{internal}) was set to randomly transmit at up to four times the north-south outbound flows rates per-profile. Correspondingly the flow inter-arrival time ($\Delta t_{\text{internal}}$) for east-west traffic between servers per application tier depicted in Fig. 10 (b) was proportionally reduced to analyse the effects of network congestion on high and low priority profiles with increasing user connections. Furthermore, a sample profile priority table was used with the corresponding application hierarchy based on application usage weighting starting with highest profile.

C. Results and discussion

The simulation results measured the effectiveness of the proposed profiling based real-time traffic optimization against conventional traffic management schemes i.e. ECMP and STP. Traffic statistics including effective throughput between users and front-end servers (SW[1-2], SW[11-18]) as well as between individual application servers (SW[11-18]) on disjoint pods were recorded at varying user loads, by collecting respective switch port statistics via the SDN controller. In the first test we compared STP, ECMP and the profiling-based scheme frame delivery ratio and throughput (total received frames) performance. External inbound traffic for the top priority profile 3 and application (Enterprise) was recorded using the three traffic management approaches while increasing user loads across all profiles. As shown in the Fig. 12 (a) and (b) profile 3 users consistently experienced high frame delivery and throughput using profiling based path computation compared to the ECMP and STP scheme. STP uses a loop free environment limiting utilization of all available links while ECMP equally balances the traffic across all links without discriminating between flows from different user classes or applications. The absence of profiling-based forwarding, therefore, resulted in profile 3 experiencing significantly lower frame delivery ($\leq 21\%$) and throughput ($\leq 35\%$) with increasing user loads (>300 users per profile) despite being high priority users. Moving on to the lowest priority profile 5

and highest priority Enterprise application tier, profiling based traffic management again resulted in improved frame delivery (up to 39%) and throughput (up to 37%) of the respective flows in comparison with conventional approaches as given in Fig. 12 (c) and (d). This was due to the higher application priority accorded to enterprise tier despite sharing of links with other lower priority applications on adjacent pods (i.e. the publishing tier). As evident from the profile 5 routing paths in Fig.11, at very high user loads (≥ 600 users per profile), profiling based route construction ensured profile 5 traffic continued to use link SW1:SW6:SW14 localizing the effects of oversubscription on the remaining available link SW2:SW6:SW14, mainly carrying internal inter-server traffic. Next we consider a profile in the middle of the priority table such as profile 1 and a medium priority application tier such as streaming (traffic parameters in Fig. 11). We observe profiling-based optimization significantly outperforming ECMP and STP (≥ 300 users per profile) due to ECMP load-balancing traffic over all links while profiling scheme routing profile 1 traffic over the first link and forwarding lower priority traffic over the second (oversubscribed) link illustrated in Fig. 12 (e) and (f).

The second test evaluates inter-server traffic performance between servers belonging to the same application but residing on disjoint pods. To realize the effectiveness of the global application hierarchy in the proposed design, throughput was measured for the Email tier closer to the top of the hierarchy table and web-browsing from the bottom. The results are depicted in Fig.12 (g) and (h). For the Email tier, throughput was measured between switches SW12 and SW17 (pods 2 and 7) and for web browsing servers between SW11 and SW18 (pods 1 and 8). Even at maximum simulated user load of 700 per profile, overall throughput for Email tier using profiling scheme was significantly more than either ECMP or STP based route selection. This was due to Email traffic having higher priority among other applications traversing the same links i.e. web-browsing, software and storage tiers via the aggregate and core switches. At a load of approximately 600 users per profile, the north-south outbound traffic preceding inter-pod Email traffic forced Email traffic to split over two links SW17:SW9 the primary and SW17:SW10 the secondary with the later also carrying remaining application traffic from software tier.

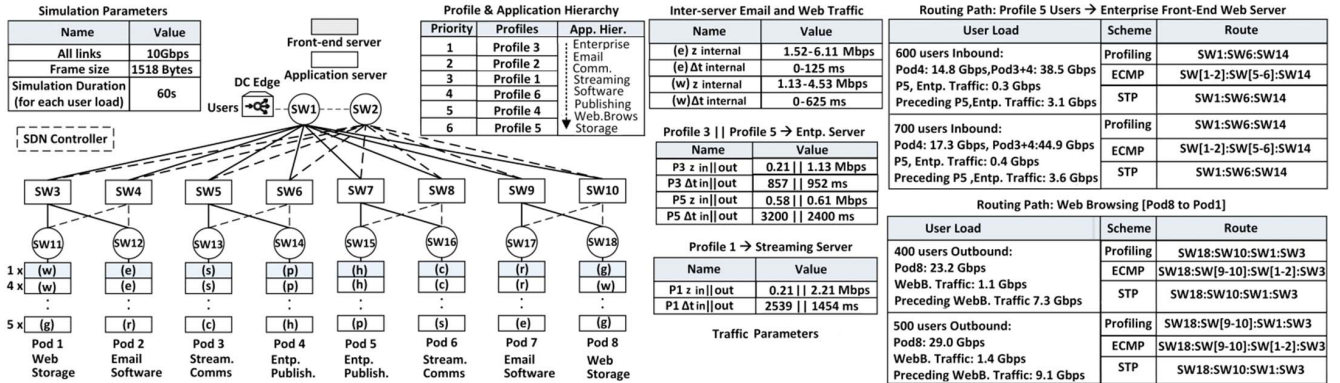


Fig. 11. Data center simulation topology and traffic parameters

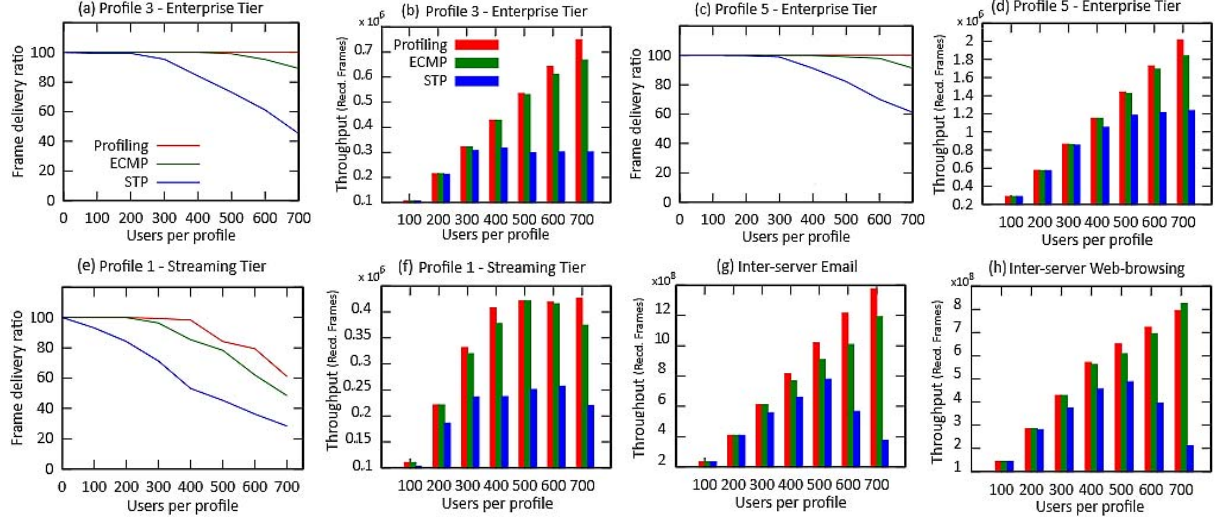


Fig. 12. Frame delivery ratio and throughput measurement

Similar to north-south traffic management, increase in priority traffic from higher tier applications sharing links with low priority applications force the later to endure greater cost of link oversubscription. For the lower priority web-browsing tier, however, both profiling and ECMP based traffic management performed almost equally until a user load of 400 users per profile with ECMP eventually overtaking the profiling link selection at 700 users per profile. As per the routing path given in Fig. 11 for pod8 to pod 1 inter-server web browsing traffic, ECMP traffic was being load balanced using all links SW18:SW[9-10] while profiling forced inter-server browsing traffic on the last oversubscribed link SW18:SW9 resulting in a performance penalty. From the above tests, it is evident that profiling based traffic management forwards priority traffic over links with higher available bandwidth forcing succeeding profiles and applications on tertiary links reducing the effects of network congestion on high priority traffic due to oversubscription.

We further evaluate the flow update scheduling frequency and corresponding latency at each simulated user load to benchmark the route scheduling (rs) proposal discussed in section 5. The route scheduling (rs) scheme aims to minimize real-time flow updating frequency employing per profile flow inter-arrival duration and flow rate computations given in (4) and (5). The percentage of flows updated at each user load simulation (duration 60s) was therefore, accounted at individual switch levels using *flow_stat* OpenFlow messages [17]. The cumulative distribution of the total flow updates to varying user load ratio using the route scheduling (rs) scheme is given in Fig. 13 (a). The flow update frequency at the core switch level was lowest. This was followed by the access and distribution switches. Minimum updating at the core level was mainly attributed to minor and less frequent changes required in the core switches despite increase in user load. As shown in Fig. 13 (a), the absence of route update scheduling would have resulted in additional route updates ranging between 18%-31% at each of the three switch levels. To evaluate the time complexity benefit of the route scheduling (rs) approach, the

average latency involved in computing routes and completing flow installations in individual switches was calibrated and is illustrated in Fig. 13 (b). The analysis considered the total duration of OpenFlow *flow_mod* message generation via the controller and respective flow activation in the switches. To measure the processing of *flow_mod* messages, an OpenFlow *barrier_request* was sent after sending all *flow_mod* messages per switch. The OpenFlow *barrier_request* message ensures that the switch completes processing of all sent messages before issuing a *barrier_reply*, indicating the flow-mod message has been fully processed at the switch [17]. The results show that reduced flow update frequency due to the route scheduling scheme (rs) translates into comparatively lower latency in creating flow constructs in switches. The highest average latency was observed in updating access switch routes approximating at 2248ms. Access switches serve both internal and external traffic and therefore, the average computational and processing latency attributed to the lowest level in the switch hierarchy was greater than the aggregate and core level. The average maximum latency recorded per core and distribution switch approximated at 2120ms and 1998ms respectively. The overall reduction in latency due to route scheduling ranged between 23%-41% (135ms-725ms) for the three switch levels at each simulated run. The route computation and installation duration comprised approximately 0.67%-3.6% of the total simulation duration (60s) per user load across the entire DC switching fabric. The route scheduling scheme therefore, allows a considerable measure of reduction in route construction latency following profile membership updates improving the real-time design scalability.

The traffic management and time complexity evaluation discussed above utilized a single sample profile priority table to provide an appreciation of the benefits of integrating user profiling controls in the DC over conventional load balancing schemes. However, the effectiveness of the proposed traffic forwarding design in essence relies on relieving adverse network performance due to link oversubscription at the expense of low priority users and (or) applications. Selection of

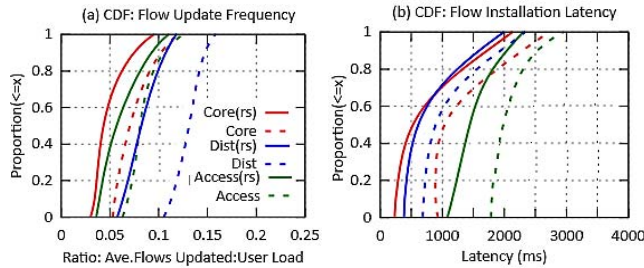


Fig. 13. (a) Flow update frequency and (b) Flow installation latency

alternate profile and application priority would therefore, also result in entities at the top of the hierarchy experiencing lesser congested routes with higher throughput and frame delivery performance than lower priority users and applications. In comparison, conventional schemes such as ECMP or STP fall short of delivering for high priority users, and at best only allow uniform network provisioning or cater to isolated application performance. Furthermore, the use of SDN technology allows real-time programming of DC switches by automatically tracking user loads (per profile) and dynamically re-evaluating existing forwarding constructs. Such automated network provisioning utilizing user traffic profiles would be relatively impossible to achieve with manually intensive traditional switch configuration methods. SDN also improves scheme scalability, addition of front-end servers per application tier given in Fig. 11 or increase in server pods, for example, would only mean the re-computation of forwarding paths on multiple front-end servers requiring no major design changes.

VII. CONCLUSION

Implementing isolated application performance in consolidated DC utilizing conventional load-balancing methods for the extracted user profiles leads to degraded performance for users requiring optimal forwarding. The benefits of profiling based network provisioning are particularly significant at higher user loads, where high priority profiles experience higher throughput and frame delivery ratio compared to the values observed while utilizing conventional load-balancing techniques. It is however, noticeable that due to greater link oversubscription at higher loads only a subset of profiles and applications may be allocated lesser congested DC paths, the selection depending on operator-assigned priority to the respective user profiles and subsequently derived application hierarchy. Furthermore, using SDN technology allows the dynamic implementation of constructed routes with changing user connections offering increased real-time design scalability, a significant improvement over present manually intensive network configuration techniques.

REFERENCES

- [1] Santana G., "Data Center Virtualization Fundamentals: Understanding Techniques and Designs for Highly Efficient Data Centers with Cisco Nexus, UCS, MDS, and Beyond", Cisco Press 2013. ISBN 1587143240
- [2] SDN Architecture, Issue 1, Open Networking Foundation. 2014. Website: <https://www.opennetworking.org/technical-reports/>
- [3] Alok Kumar et. al. 2015. BwE: Flexible, Hierarchical Bandwidth Allocation for WAN Distributed Computing. In Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication. ACM, NY, USA, 1-14.
- [4] Shuo Fang; Yang Yu; Chuan Heng Foh; Khin Mi Mi Aung, "A loss-free multipathing solution for data center network using software-defined networking approach," in APMRC, 2012 Digest, vol., no., pp.1-8, Oct. 31 2012-Nov. 2 2012.
- [5] Albert Greenberg et. al. 2011. VL2: a scalable and flexible data center network. Commun. ACM 54, 3 (March 2011), 95-104.
- [6] Bakhshi T. and Ghita B, "User traffic profiling in a software defined networking context", 6th International Conference on Int. Technologies & Applications (ITA15), Wales, Sep 2015.
- [7] Farrington, N.; Andreyev, A., "Facebook's data center network architecture," in Optical Interconnects Conference, 2013 IEEE, vol., no., pp.49-50, 5-8 May 2013.
- [8] Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Ant Rowstron. 2011. Towards predictable datacenter networks. In Proceedings of the ACM SIGCOMM 2011 conference (SIGCOMM '11). ACM, NY, USA, 242-253.
- [9] S. Kandula, D. Katabi, S. Sinha, and A. Berger, "Dynamic load balancing without packet reordering," ACM Computer Communication Review, vol. 37, no. 2, pp. 51-62, 2007.
- [10] V. Jeyakumar, A. Kabbani, J. C. Mogul, and A. Vahdat, "Flexible Network Bandwidth and Latency Provisioning in the Datacenter," in arXiv:1405.0631, 2014.
- [11] Cisco Virtualized Multiservice Data Center Framework, 2016. Available Online: <http://www.cisco.com/enterprise/vmdc.html>
- [12] Murtagh, F. and P. Legendre, "Ward's Hierarchical Agglomerative Clustering Method", Journal of Classification, 2014. 31(3): p. 274-295.
- [13] Hartigan, J. A. and Wong, M. A. (1979). "A K-means clustering algorithm". Applied Statistics 28, pp.100-108.
- [14] Hongbo J., Zihui G., Shudong J., Jia W., (2010), "Network prefix-level traffic profiling: Characterizing, modeling, and evaluation". Comput. Netw. 54, 18, pp.3327-3340.
- [15] Bakhshi T. and Ghita B, "Traffic profiling: Evaluating stability in multi-device user environments", 30th IEEE IANA Workshops, Crans-Montana, Switzerland, Mar 2016.
- [16] Brian S. Everitt and Torsten Hothorn (2006), "A Handbook of Statistical Analyses Using", Boca Raton, FL: Chapman & Hall/CRC, 2006
- [17] OpenFlow Cert., 2016. Website: <https://opennetworking.org/>
- [18] T.Chim, K. Yeung, and K. Lui, "Traffic distribution over equal-cost-multi-paths," Computer Networks, vol. 49, no. 4, pp. 465-475, 2005.
- [19] S. Kandula, D. Katabi, S. Sinha, and A. Berger, "Dynamic load balancing without packet reordering," ACM SIGCOMM Computer Comm. Review, vol. 37, no. 2, pp. 51-62, 2007.
- [20] Mininet: An Instant Virtual Network on your Laptop (or other PC), Website: <http://mininet.org>
- [21] Ryu: Component-based software defined networking framework Website: <http://ryu.readthedocs.org/en/latest/>
- [22] Ostinato: Packet Traffic Generator and Analyzer, Website: <http://ostinato.org/>