ADON: Application-Driven Overlay Network-as-a-Service for Data-Intensive Science

Sripriya Seetharam, Prasad Calyam, Tsegereda Beyene
University of Missouri-Columbia, USA; Cisco Systems, USA
Email: \{ssn68, calyamp\}@missouri.edu, tbeyene@cisco.com

Abstract—Campuses are increasingly adopting hybrid cloud architectures for supporting data-intensive science applications that require “on-demand” resources, which are not always available locally on-site. Policies at the campus edge for handling multiple such applications competing for remote resources can cause bottlenecks across applications. These bottlenecks can be proactively avoided with pertinent profiling, monitoring and control of application flows using software-defined networking principles. In this paper, we present an “Application-driven Overlay Network-as-a-Service” (ADON) that can manage the hybrid cloud requirements of multiple applications in a scalable and extensible manner using features such as: programmable “custom templates” and a “virtual tenant handler”. Our solution approach involves scheduling transit selection and traffic engineering at the campus-edge based on real-time policy control that ensures predictable application performance delivery for multi-tenant traffic profiles. We validate our ADON approach with an implementation on a wide-area overlay network tested across two campuses, and present a workflow that eases the orchestration of network programmability for campus network providers and data-intensive application users. Lastly, we present an emulation study of the ADON effectiveness in handling temporal behavior of multi-tenant traffic burst arrivals using profiles from a diverse set of actual data-intensive applications.

I. INTRODUCTION

Data-intensive applications in research fields such as bioinformatics, climate modeling, particle physics and genomics generate vast amounts of data that need to be processed with real-time analysis. The general data processing facilities and specialized compute resources do not always reside at the data generation sites on campus, and data is frequently transferred in real-time to geographically distributed sites (e.g., remote instrumentation site, federated data repository, public cloud) over wide-area networks. Moreover, researchers share workflows of their data-intensive applications with remote collaborators for multi-disciplinary initiatives on multi-domain physical networks [1].

Current campus network infrastructures place stringent security policies at the edge router/switch and install firewalls to defend the campus local-area network (LAN) from potential cyber attacks. Such defense mechanisms significantly impact research traffic especially in the case of data-intensive science applications whose flows traverse wide-area network (WAN) paths. This has prompted campuses to build Science DMZs (de-militarized zones) [1] with high-speed (1 - 100 Gbps) programmable networks to provide dedicated network infrastructures for research traffic flows that need to be handled in parallel to the regular enterprise traffic.

The advanced network infrastructure components in Science DMZs that help with high-performance networking to remote sites and public clouds include: (i) software-defined networking (SDN) based on programmable OpenFlow switches [2], (ii) RDMA over Converged Ethernet (RoCE) implemented between zero-copy data transfer nodes [3] for data transport acceleration, (iii) multi-domain network performance monitoring using perSONAR active measurement points [4], and (iv) federated identity and access management using Shibboleth-based entitlements [5].

However, if multiple applications accessing hybrid cloud resources compete for the exclusive and limited Science DMZ resources, the policy handling of research traffic can cause a major bottleneck at the campus edge router and impact the performance across applications. Figure 1 illustrates an actual problem scenario we faced when we initiated a file transfer as part of a research application (Advanced Data Transfer Service) using RoCE protocol on the high-speed overlay between the University of Missouri (MU) and The Ohio State University (OSU) [6]. As shown, the achieved transfer time was substantially low compared to the expected theoretical transfer time. Upon investigation, we discovered that our application’s traffic was being rate limited to 2 Gbps at OSU edge router even though the link’s capacity was capable of 10 Gbps speeds. Since RoCE protocol assumed a 10 Gbps link availability and is highly sensitive to packet loss, our application performance suffered severely.

As evident from the above scenario, there is a need to provide dynamic Quality of Service (QoS) control of Science DMZ network resources versus setting a static rate limit affecting all applications. The dynamic control should have awareness of research application flows with urgent or other high-priority computing needs, while also efficiently virtualizing the infrastructure for handling multiple diverse application traffic flows. The virtualization obviously should not affect the QoS of any of the provisioned applications, and also advanced services should be easy-to-use for data-intensive application users, who should not be worrying about configuring underlying infrastructure resources.
Our work in this paper aims to solve the network virtualization problem at campus-edge networks using dynamic queue policy management for individual flows, while making network programmability related issues a non-factor for data-intensive application users. More specifically, we present a new “Application-Driven Overlay Network-as-a-Service” (ADON) architecture to intelligently provision on-demand network resources by performing a direct binding of applications to infrastructure and provide fine-grained automated QoS control.

The novelty and contributions of our work are as follows: we detail how ‘network personalization’ can be performed using a concept of “custom templates” to catalog and handle unique profiles of application workflows. We also detail a multi-tenant architecture for real-time policy control of an ‘Overlay Network-as-a-Service’ through a “Virtual Tenant Handler” (VTH). The VTH leverages awareness of the overall ‘load state’ at the campus edge, and the individual application ‘flow state’ using software-defined performance monitoring integration within the overlay network paths. Using the custom templates and VTH concepts, ADON can manage the hybrid cloud requirements of multiple applications in a scalable and extensible manner. It ensures predictable application performance delivery by scheduling transit selection (choosing between regular Internet or high-performance Science DMZ paths) and traffic engineering (e.g., rate limit queue mapping based on application-driven requirements) at the campus-edge.

We validate our ADON implementation on a wide-area overlay network testbed across OSU and MU campuses connected with Internet2 Advanced Layer2 Service (AL2S), and present a workflow that we adopted to ease the orchestration of network programmability for both campus network providers and data-intensive application users. Lastly, we present a detailed emulation study of the ADON effectiveness in handling temporal behavior of multi-tenant traffic burst arrivals using profiles from a diverse set of actual data-intensive applications used in research and education use cases.

The remainder paper organization is as follows: Section II presents related work. Section III details custom templates for exemplar application workflows. Section IV describes ADON’s modular architecture with VTH components. Section V describes ADON implementation on an actual testbed, and in an emulation study. Section VI concludes the paper.

II. RELATED WORK

Existing works such as [7], [8] and [9] recognize similar application-driven network virtualization issues at network-edges, and have proposed new architectures based on SDN principles. In [7], application level requirements are programmed using an inter-domain controller implemented using OpenFlow, and a custom-built extensible session protocol is used to provide end-to-end virtual circuits across campus DMZs. In [8] that is closely related to our work, QoS parameters are programmed dynamically based on high-level requirements of different kinds of application traffic. The authors argue (similar to our argument in context of Figure 1) that there is a need for dynamic QoS configuration based on application needs, and current practice of manual configuration by network administrators hinders application performance. In [9], the authors propose a new controller design for QoS based routing of multimedia traffic flows.

In contrast, our work in this paper is focused on the requirements and challenges at the campus-edge. We handle Science DMZ resources and personalize network configurations (e.g., rate limit queue mappings) that need to be controlled in real-time for meeting hybrid cloud computing needs of data-intensive application flows.

III. NETWORK PERSONALIZATION FOR APPLICATIONS WITHIN ADON

Figure 2 shows how data-intensive applications can co-exist on top of a shared wide-area physical infrastructure topology, with each application demanding local/remote network or compute resources with unique end-to-end QoS requirements. In order to effectively handle the diverse QoS requirements at the campus edge router, several challenges need to be addressed. One of the important challenges is the need to have ‘application performance visibility and control’ within the provisioned resources for individual applications. This requires maintaining a catalog of application profiles in terms of Resource Specifications (RSpecs) and Quality Specifications (QSpecs). In addition, policies should be determined for the extent to which programmable capabilities at the campus edge can be used to ‘personalize’ the network overlay setup based on: (a) the individual application RSpecs and QSpecs, and (b) the temporal behavior of multi-tenant traffic burst arrivals.

In the following subsections, we first describe the concept of custom templates that can be used within ADON to develop a catalog of application profiles. Following this, we apply the custom template concept for exemplar data-intensive application workflows with diverse QoS requirements.
A. Custom Templates

Our concept of custom templates within ADON is similar to the best practices such as Amazon Web Services (AWS) Machine Image (AMI) [10] and RSpecs in the NSF-supported Global Environment for Network Innovations (GENI) [11]. Works such as [12] also suggest the value of using templates that can allow for composing and executing workflow pipelines for data-intensive applications in a reusable manner.

Figure 3 shows how custom templates can be used as part of the sequential steps of ADON auto-orchestration during on-demand resource provisioning for a data-intensive application flow that needs an overlay network path. The details of the steps in ADON orchestration are as follows: First, a researcher of a data-intensive application can securely request the ADON by authenticating with a Federated Identity and Access Management (Federated IAM) system that uses Shibboleth-based entitlements [5]. Such Federated IAM systems are necessary to handle multi-institutional policy specifications pertaining to cases such as: (i) How can a data-intensive application user at Campus A be authenticated and authorized to reserve HPC resources or other scientific instruments at a remote Campus B? or (ii) How can a OpenFlow controller at one campus be authorized to provision flow spaces within a backbone network in an on-demand manner? (iii) Who can subscribe to the performance measurements related to a data-intensive application to monitor workflow status and track/troubleshoot any bottleneck anomaly events?

Subsequently, the researcher provides his/her data intensive application handling specifications through a simple and intuitive application dashboard mashup. The specifications can include details such as destination host (i.e., remote collaborator or remote instrument site) and application type (e.g., remote interactive volume visualization, video streaming, file transfer or compute resource reservation). Next, the application specifications are subsequently matched to a custom template with RSpecs and QSpecs that closely match the application type for discovery/reservation of the necessary compute and network resources.

The custom template can be pre-configured by a Performance Engineer to apply specific resource descriptions and associated campus policies that can be interpreted by e.g., a network flowvisor (i.e., proxy for OpenDaylight or POX [14]) to instantiate flows on intermediate OpenFlow switches, and by a compute hypervisor to instantiate virtual machines within a data center. We refer to Performance Engineer as the one who serves as the primary ‘keeper’ and ‘helpdesk’ of the Science DMZ equipment, and the success of this role is in the technician’s ability to augment traditional System/Network Engineer roles on campuses and serve high-throughput computing needs of researchers.

In addition to helping the Performance Engineer with the resource configuration, custom templates also help in configuring real-time network performance monitoring within the overlay network path to provide the application performance visibility that can be used to define triggers for dynamic resource adaptation. Moreover, performance bottlenecks such as those observed in Figure 1 can be avoided through use of custom templates, and in exception cases where end-to-end QoS configuration is not possible, bottlenecks can be relatively more easily discovered and overcome.

B. Applications Workflows

1) Neuroblastoma Data Cutter Application: As shown in Figure 4(a), the workflow of the Neuroblastoma application [6] consists of a high-resolution microscopic instrument on a local campus site (at MU) generating data-intensive images that need to be processed in real-time to identify and diagnose Neuroblastoma (a type of cancer) infected cells. The processing software and HPC resources required for processing these images are available remotely at OSU, and hence images from MU need to be transferred in real-time to the remote OSU campus. To handle the highly large scale data transfers, the application relies on advanced file transfer protocols such as RoCE and GridFTP technologies that support parallel TCP flows between the two campuses. A corresponding Neuroblastoma application template can be given as: (i) RSpec - parallel TCP flows with high bandwidth bursts, and (ii) QSpec - high flow throughput with no packet loss and high flow priority to provide fast-enough application response time for a histopathological evaluator.

2) Remote Interactive Volume Visualization Application (RIVVIR): As shown in Figure 4(b), the RIVVIR application [13] at OSU deals with real-time remote volume visualization of 3D models of small animal imaging generated by MRI scanners. When such an application needs to be accessed for remote steering and visualization by thin-clients, the network path between the two sites should have as much available bandwidth as possible. A corresponding RIVVIR
application template can be: (i) RSpec - Layer 2 network steering (over Internet2) of application traffic, and (ii) QSpec - Low latency/jitter flow with high bandwidth and medium flow priority to help with interactive analysis with a thin-client.

3) GENI Classroom Lab Experiments: As shown in Figure 4(c), a class of 30 students conducting lab experiments at MU in a Cloud Computing course [14] require resources across multiple GENI racks. As part of the lab exercises, multiple VMs need to be reserved and instantiated on remotely located GENI racks. There can be a sudden burst of application traffic flows at campus edge router, especially the evening before the lab assignment submission deadline. A corresponding GENI Classroom application template can be: (i) RSpec - Multiple layer 2 flows to remote sites with low bandwidth requirements, and (ii) QSpec - Low packet loss and medium flow priority to allow students to finish the lab exercises.

4) ElderCare-as-a-Service Application: As shown in Figure 4(d), the ElderCare-as-a-Service application [14] consists of an interactive video streaming session between a therapist on MU campus and a remotely residing elderly patient at a Kansas City residence for performing physiotherapy exercises as part of telehealth interventions. During a session, the quality of application experience for both users is a critical factor (especially in skeletal images from Kinect sensors), and the application demands strict end-to-end QoS requirements to be usable. A corresponding ElderCare-as-a-Service application template can be: (i) RSpec - Deterministic flow path with high-resilience for network link failures on Layer 3 (regular Internet with Google Fiber last-mile), and (ii) QSpec - Consistent high available bandwidth with very low or no jitter and high flow priority to allow an elderly patient to closely follow the postures being exercised in the session.

IV. ADON ARCHITECTURE

In this section, we describe the policy implementation architecture of ADON that leverages the custom templates for fine-grained QoS control and customization of programmable compute and network resources. Figure 5 shows the ADON architecture, which consists of the application, middleware and the infrastructure layers within which individual components interact with each other to provision resources for incoming application requests.

The high-level application requirements along with RSpecs, QSpecs and application priority are captured in the application layer. Depending upon the resources being requested in the infrastructure layer, and the campus policy rules (maintained by the Performance Engineer), the routing and queue policy assignments are applied in the middleware layer for each application being provisioned. Real-time performance monitoring of individual flows can be used to configure adaptation triggers within already provisioned flows, or even to reject a new application flow if the required QoS levels cannot be met given the load of already provisioned flows. Such middleware layer functions can be implemented with: (i) Control Module, (ii) Network Flowvisor, and (iii) Compute Hypervisor. In this paper, we mainly focus on the Control Module’s ‘Custom Template Catalog’ and Network Flowvisor module’s ‘Virtual Tenant Handler’ (highlighted in red in Figure 5) that are necessary to implement ADON, and the Compute Hypervisor issues are beyond this paper scope.

A. Custom Template Catalog

The Control Module consists of the Template Generator component which exposes RESTful APIs for configuring application type, application priority and routing specifications that can be programmed within the application layer. The Template Generator module also allows the Performance Engineer to save a successfully configured template in a custom template catalog database, which allows re-use for future flow provisioning instances. The QoS and application priority parameters are then fed into the Network Flowvisor module by programming the required REST APIs such as: queue policies and bandwidth requirements. The Federated IAM component within the Control Module features an ‘entitlement service’ module for all campuses that federate their Science DMZ infrastructures using third-party frameworks such as the Internet2 Incommon federation [15]. It also allows for centrally managing entitlements based on mutual protection of privacy policies between institutions to authorize access to different multi-domain infrastructure components.

B. Virtual Tenant Handler

Virtual Tenant Handler (VTH) is responsible for dynamically handling incoming application flows and providing the intelligence for adaptive network resource allocation. As shown in Figure 6, the Policy Engine interacts with the Template Generator component on the top layer for accepting the RSpec and QSpec parameters along with application priority within the custom templates. The Policy Engine then interacts with the Flow Scheduler to check the priority of the new application
with the existing applications. The Routing Controller is responsible for deciding on which port number of the OpenFlow switch should the application flow be provisioned for traffic steering such as Layer 2 (Science DMZ path over Internet2 AL2S) or Layer 3 (regular Internet).

If Science DMZ path is selected for the application flow, the Dynamic Queue Manager is responsible for provisioning the right QoS policies. To accomplish such right provisioning, we utilize the minimum rate and maximum rate properties of queue configuration on OpenFlow switches as provided by the OpenFlow 1.3 specification [16]. The configured queues on the OpenFlow switch ports are then mapped to incoming application flows using the set-queue action set of the OpenFlow specification. As shown in Figure 7, the queue slots are mapped based on the application priority as specified in the high-level application requirements. A higher priority application is mapped to a queue with the maximum available bandwidth.

In case the desired queue is not available, it is mapped down the queue priority level to be provisioned in the next best available priority queue. The mapping is done until a queue slot that can provide an acceptable QoS is selected. If found, the flow is provisioned on the selected queue, or else the flow is pushed to the Flow Scheduler component. However, for a medium priority application, if Layer 2 slot is not available and the QoS parameters are not constrained, the flow is directed to a Layer 3 default queue. If none of the slots are available, the flow can be pushed again to the Flow Scheduler to be retrieved later once the appropriate queue is available. The Dynamic Queue Manager then interacts with the Resource Aggregator to update the available resources once a given flow is provisioned on its overlay network.

The VTH also monitors the load states at each queue and flow states of each application using ‘Flow State Monitor’ and ‘Load State Monitor’ components. These components receive inputs from the ‘Network Performance Measurement’ module which provides network health status notifications such as flow statistics and other such QoS impacting factors. The module provides options to reconfigure existing paths based on the custom template directed resource reservations. In case the load state in terms of number of applications contending for Layer 2 service creates a link saturation, VTH can steer the traffic on Layer 3 if acceptable QoS parameters can allow a push of the new flows into the Flow Scheduler.

The VTH further interacts with the underlying OpenFlow controller for installing the required flows on the OpenFlow switches. The ‘Network Topology Manager’ (part of the OpenFlow controller) implements shortest available path algorithms with weighted and non-weighted deterministic routes to remote WAN-accessible sites and provides graphical abstractions to the VTH module with topology link states. The algorithm used in the VTH is formally discussed in the following section.

C. ADON Resource Provisioning Algorithm

As shown in Algorithm 1, a new application $a_{new}$ enters the VTH module. If the net resource available $R_N$ where $N$ is the total number of applications within VTH, is greater than the threshold resource value $T_r$ of the system, the application can be pushed in Step-1 to a weighted fair-queue FIFO Flow scheduler. The threshold value $T_r$ can be a parameter programmed by the Performance Engineer in the Application Layer. In case there are available resources, the priority of the new application $P_{a_{new}}$ is compared with the already scheduled application $a_s$ priority $P_{a_s}$. Whose ever priority is higher, the slice $s_a$ is instantiated using the createSlice function within the Policy Engine. Once the slice is created in Step-2, the optimal resource $r_a$ for the application is computed in Step-3 using the function computeResource based on application template parameters such as destination point, $RSpec(a_s)$ and $QSpec(a_s)$. Note that $r_a$ is a vector containing the information: (i) Which queue has to be allocated for the application? (ii) Which port such as Layer 2 or Layer 3 service has to be assigned for the application? (iii) What is the required bandwidth for the application? In Step-4, the computed resources are then subtracted from the net resource $R_N$ to indicate the new resource status. Suppose, after a certain timeout, the flow statistics provided by a certain flow implies under or no utilization of resources by the application; then in Step-5, the slice is deleted and resources are released into the available resource pool for other scheduled application flows in the Flow Scheduler component.

Algorithm 1 ADON Resource Provisioning Algorithm

1: **Input**: Application flow $a_{new}$
2: **Output**: Resource allocation $r_a$ for application flow $a_n$
3: begin procedure
4: /*Step-1: Resource allocation $r_a$ for application flow $a_n$*/
5: if net resource $R_N <$ threshold value $T_r$ then
6: if scheduler not null && $P_{a_s} > P_{a_{new}}$ then
7: $a_{new} = a_s$
8: else
9: $a_{new} = a_{new}$
10: end if
11: /*Step-2: Application slice creation*/
12: $s_a = createSlice(a_{new})$
13: /*Step-3: Resource usage calculation*/
14: $r_a = computeResource(s_a, a_r, a_q)$
15: /*Step-4: Update available resources*/
16: $R_N = R - r_a$
17: else
18: Push application flow to resource scheduler
19: end if
20: /*Step-5: Application termination or timeout*/
21: $R_N = R + r_a$
22: Pop existing application flow from scheduler queue and repeat the above steps

The above logic ensures high priority flows receive bandwidth provisioning that is adequate to satisfy performance requirements, and allows per-flow isolation when there are multiple lower priority application flows competing for network resources. In addition, it ensures low priority flows do not experience high rejection rates, while simultaneously maximizing the network resource utilization.
In this section, we first describe a case study featuring validation experiments of our ADON implementation on a wide-area overlay network testbed across OSU and MU campuses connected with Internet2 AL2S. Next, we present a detailed emulation study we conducted with application workflows to analyze the VTH algorithm results while handling temporal behavior of multi-tenant traffic burst arrivals.

A. Implementation Case Study

The testbed setup as shown in Figure 8 consists of OSU and MU campuses connected through an extended VLAN overlay that involves an Internet2 AL2S connection by way of local regional networks of OARnet in Ohio, and GPi/MoreNet in Missouri, respectively. Each Science DMZ has a matching DTN equipped with dual Intel E5-2660, 128GB of memory, 300GB PCI-Express solid state drive, and dual Mellanox 10 Gbps network cards with RoCE support. Each Science DMZ has perfSONAR measurement points for continuous monitoring at 1 - 10 Gbps network speeds. A common Dell R610 node in the OSU Science DMZ is used to run the VTH module along with OpenDaylight OpenFlow controller that controls both the OSU and MU Science DMZ OpenFlow switches.

We ran an experiment by providing the application requirements such as the source site, and destination site - along with application type, which is a real-time image transfer for processing of medical images within Neuroblastoma application - by following the steps illustrated previously in Figure 3. The application requirements and policy specifications were configured as high bandwidth, Layer 2 routing with priority individual priorities on the queues. At time $t_1$, the Dynamic Queue Manager within the VTH instantiated the queues on the OpenFlow switch to the requested bandwidth provided performance isolation to each of the concurrent flows. The total jitter observed when both flows coexisted on the link are captured with and without VTH dynamic queue management in Figure 10(a). We can see that the jitter is significantly reduced (improved performance) for both the applications, especially for the video flow using the VTH module with application-specific queue policies. This is due to the fact that the VTH reduced the external fragmentation of available bandwidth caused by static policy management on the switch ports. The dynamic queue mapping of the UDP flows to the requested bandwidth provided performance isolation to each of the flows, and hence reduced their jitter QoS metric values.

At time $t_3$, GENI Classroom workflow (see Figure 4(c)) was started as a TCP flow with burst traffic pattern (burst rate - 1 Mbps and buffer size- 10 KB similar to web traffic) parallel to the RIVVIR workflow. Both flows co-existed without affecting each other’s QoS policies as both flows were assigned their individual priorities on the queues. At time $t_4$, Neuroblastoma workflow (see Figure 4(a)) was started as a parallel TCP flow Owens cluster in 5.13 seconds. The time taken to process the image was about 4 seconds (i.e., compute time) on the cluster and the transfer of processed image back to MU took around 5.2 seconds (i.e., communication time) aggregating to a total of 14.4 seconds of “compute plus communication” time within the ADON provisioned path.

B. Emulation Study

We used Mininet network emulator for VTH experiments that involved synthetic traffic flows using the “tc” and “iperf” network utility tools. Figure 9 shows the timeline, from time $t_1$ to $t_7$ as seen by an edge OpenFlow router/switch handling application workflows, as they enter and exit the VTH module. RIVVIR application workflow (see Figure 4(b)) was initiated as a UDP flow at time $t_1$ with a guaranteed bandwidth of 10 Mbps (typical requirements of remote desktop access with raw encoding) and latency of 50 ms (RTT between OSU and MU). At time $t_2$, ElderCare-as-a-Service application workflow (see Figure 4(d)) was started as a new UDP flow with a guaranteed bandwidth of 100 Mbps (typical requirement of a Kinect video stream) and latency of 30 ms (RTT between MU and Kansas City).

At this moment, the Dynamic Queue Manager within the VTH instantiated the queues on the OpenFlow switch to provide the required QoS guarantees for each of the concurrent flows. The total jitter observed when both flows coexisted on the link are captured with and without VTH dynamic queue management in Figure 10(a). We can see that the jitter is significantly reduced (improved performance) for both the applications, especially for the video flow using the VTH module with application-specific queue policies. This is due to the fact that the VTH reduced the external fragmentation of available bandwidth caused by static policy management on the switch ports. The dynamic queue mapping of the UDP flows to the requested bandwidth provided performance isolation to each of the flows, and hence reduced their jitter QoS metric values.

At time $t_3$, GENI Classroom workflow (see Figure 4(c)) was started as a TCP flow with burst traffic pattern (burst rate - 1 Mbps and buffer size- 10 KB similar to web traffic) parallel to the RIVVIR workflow. Both flows co-existed without affecting each other’s QoS policies as both flows were assigned their individual priorities on the queues. At time $t_4$, Neuroblastoma workflow (see Figure 4(a)) was started as a parallel TCP flow...
with 5 parallel streams to simulate a GridFTP application for image file transfer. VTH then triggered the dynamic queue configuration for assigning a prioritized bandwidth of 600 Mbps for Neuroblastoma and 360 Mbps bandwidth for GENI Classroom experiments (on a 1 Gbps link).

Figure 10(b) shows throughput of the two workflows achieved with and without VTH. Bandwidth is equally split when there is no dynamic queue management for both flows. However with VTH, internal fragmentation of bandwidth is reduced. The total bandwidth is sliced between the two flows as per their individual priorities ensuring each flow is only utilizing the requested bandwidth as provided in the application QoS templates. This slicing happens until time \( t_2 \) when the GENI Classroom flow exits and resources are released. However, when a new RIVVIR workflow starts again at time \( t_0 \) while the Neuroblastoma application is currently provisioned, the new flow is rejected and pushed to the Flow Scheduler. This is because the new flow’s QoS requirements cannot be guaranteed and mapped in the Dynamic Queue Manager. This scenario occurs due to the resource unavailability of the link which is fully utilized by the prioritized data-intensive Neuroblastoma application flow.

Thus, we can conclude from the above experiments that the VTH is effective in scheduling transit selection and traffic engineering at the campus-edge based on real-time policy control that ensures predictable performance for multimedia delivery, when handling temporal behavior of multi-tenant traffic burst arrivals corresponding to a diverse set of data-intensive applications.

VI. CONCLUSION

In this paper, we presented a novel ADON architecture with an application-driven overlay network-as-a-service approach to support multi-tenant data-intensive application flows with hybrid cloud resource needs. With the pent-up resource requirements of data-intensive application flows, traditional network infrastructures are not scalable or flexible for effectively handling such flows, especially in cases with urgent or real-time computing requirements. Using our ADON architecture, we showed that the application-specific policies can be effectively controlled at the campus edge based on individual application flow requirements, and the ‘friction’ imposed due to firewalls for enterprise traffic flows can be overridden for data-intensive science applications.

The novelty of our work is in our approach for ‘network personalization’ that can be performed using a concept of “custom templates” that helps a Performance Engineer to catalog and handle unique application workflows in an automated and repeatable manner. We also presented design details and validation experiments of a multi-tenant architecture featuring a “Virtual Tenant Handler” (VTH) for real-time policy control of an ‘Overlay Network-as-a-Service’ within a campus Science DMZ environment with high-performance networking capabilities such as OpenFlow switches and RoCE-based data transfer nodes. Further, we demonstrated how our ADON architecture and implementation were capable of providing predictable performance for data-intensive applications, without any changes to existing campus network infrastructure designed for regular enterprise traffic.

Our future work includes integrating multiple geographically-distributed campuses to the ADON architecture approach as a community model, and conducting additional wide-area overlay network and emulation experiments.

REFERENCES