Leveraging OpenFlow for Resource Placement of Virtual Desktop Cloud Applications

Prasad Calyam, Sudharsan Rajagopalan, Arunprasath Selvadhurai,
Saravanan Mohan, Aishwarya Venkataraman, Alex Berryman, Rajiv Ramnath
University of Missouri, Ohio Supercomputer Center/OARnet, The Ohio State University, USA;
Email: *calyamp@missouri.edu*; {*srajagopalan*, *aselvadhurai*, *smohan*, *venkatar*, *berryman*}*@oar.net*; *ramnath@cse.ohio-state.edu*

***Abstract*—Popular applications such as email, photo/video galleries, and file storage are increasingly being supported by cloud platforms in residential, academia and industry communities. The next frontier for these user communities will be to transition 'traditional desktops' that have dedicated hardware and software configurations into 'virtual desktop clouds' that are accessible via thin-clients. In this paper, we describe an intelligent resource placement framework for thin-client based virtual desktops. The framework leverages principles of software-defined networking and features a 'unified resource broker' that uses special 'marker packets' for: (a) "route setup" when handling non-IP traffic between thin-client sites and data centers, (b) "path selection" and "load balancing" of virtual desktop flows to improve performance of interactive applications and video playback, and to cope with faults such as link-failures or Denial-of-Service cyber-attacks. In addition, we detail our framework implementation within a virtual desktop cloud (VDC) setup in a multi-domain Global Environment for Network Innovations (GENI) Future Internet testbed spanning backbone and access networks. We present empirical results from our experimentation that leverages OpenFlow programmable networking, as well as perfSONAR instrumentation-and-measurement capabilities for validating our framework in GENI under realistic settings. Our results demonstrate the importance of scheduling regulated measurements that can be used for intelligent resource placement decisions. Our results also show the feasibility and benefits of using OpenFlow controller applications for path selection and load balancing between thin-client sites and data centers in VDCs.**

***Keywords*-Virtual desktop clouds, Optimal resource allocation, Software-defined networking, Cloud experiment in GENI**

I. Introduction

There has been a rapid adoption of "cloud" platforms for online applications such as email, photo/video galleries and file storage in academia and industry. The next frontier for these user communities will be to transition their "traditional distributed desktops" that have dedicated hardware and software installations into "virtual desktop clouds" (VDCs) that are accessible via thin-clients. Moreover, in the not so distant future, we can envisage home users signing-up for virtual desktops (VDs) with a VD Cloud Service Provider (CSP) providing Desktop-as-a-Service (DaaS) as a utility. With such a utility service, a thin-client i.e., a set-top-box can be shipped to a residential user to access his/her personalized VD, similar to the model we have today for other common computing and communication needs such as VoIP (e.g., Vonage), and IPTV (e.g., Roku). This box can be connected to television monitors, or computer monitors, and multiple residential users can have their own unique login through this box to their VDs.

This material is based upon work supported by the National Science Foundation under award numbers CNS-1050225, CNS-1205658, and VMware. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or VMware.

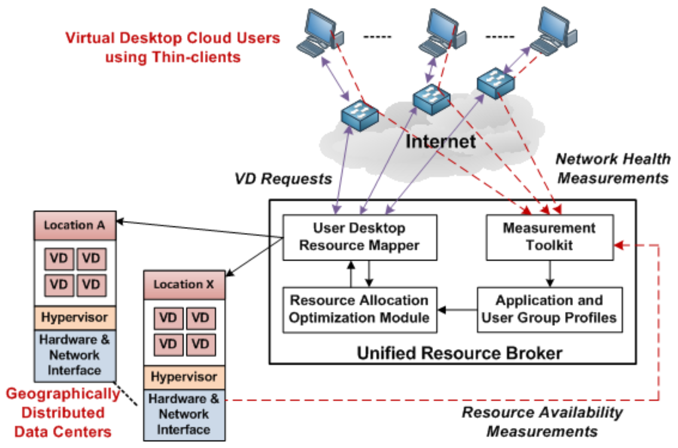
Virtual Desktop Cloud Users using Thin-clients
Internet
VD Requests
Network Health Measurements
Location A
VD
Hypervisor
Hardware & Network Interface
Location X
User Desktop Resource Mapper
Measurement Toolkit
Resource Allocation Optimization Module
Application and User Group Profiles
Unified Resource Broker
Geographically Distributed Data Centers
Resource Availability Measurements

Fig. 1. Components of Virtual Desktop Cloud Infrastructure

The drivers for these transitions of traditional desktops to VDCs are obvious in terms of user convenience, consistent user-perceivable peak performance, and cost-savings: (i) desktop support in terms of hardware, operating system, application and security upgrades will be easier to manage, (ii) the number of underutilized distributed desktops unnecessarily consuming power will be reduced, and (iii) mobile users will have wider access to their desktop applications and data.

Figure 1 shows the various system and network components in a typical VDC, comprising of two or more inter-connected data centers that handle thin-client connections from several user sites on the Internet. The "brain" of the VDC is a unified resource broker (URB) that receives the VD requests and handles them by making decisions regarding the "provisioning" (i.e., resource sizing) and "placement" (i.e., resource mapping) of resources. At each data center, a hypervisor framework (e.g., VMware ESXi, Xen) is used for computer virtualization i.e., to create VDs configured with popular applications (e.g., Excel, Internet Explorer, Media Player) as well as advanced applications (e.g., Matlab, Moldflow). VLAN extensions and software-defined networking [1] across multi-domain networks can be used for network virtualization in the infrastructure. At the thin-client side, users connect using remote display devices that use thin-client protocols such as Microsoft Remote Desktop Protocol (RDP) and Teradici PC over IP (PCoIP) [2].

In order to allocate and manage VDC resources in a cost-effective manner at Internet-scale user loads, CSPs are faced with unique provisioning and placement challenges. User workload profiles in VDCs are bursty (e.g., during daily desktop startup, when user switches between text and graphic intensive applications), and thin-client user Quality of Experience (QoE) is highly sensitive to network health variations in the Internet [3]. Unfortunately, existing works focus mainly on managing server-side resources based on utility functions of CPU and memory loads [4 - 7], and do not consider network health and thin-client user QoE. There is surprisingly sparse work [8 - 9] on resource adaptation coupled with measurement of network health and user QoE. Works such as [9] and [10] highlight the need to incorporate network health and user QoE factors into VDC resource allocation decisions.

It is self-evident that any cloud platform's capability to support large user workloads is a function of both the server-side desktop performance as well as the remote user-perceived QoE. In other words, *a CSP can provision adequate CPU and memory resources to a VD in the cloud, but if the thin-client protocol configuration or VD placement does not account for network health degradations and application context, the VD may become unusable for the user.* Hence, there is a need to couple "human-and-network awareness" within Internet-scale resource allocation frameworks for: (a) minimizing costly cloud resource over-provisioning, (b) avoiding guesswork in configuring thin-client protocols and VD placement, (c) adapting resource allocation and placement to changing fault-levels, and (d) ultimately delivering optimum user QoE of virtualized desktop applications.

Moreover, VD delivery using VDCs can be considered as a 'Future Internet' application due to its specialized infrastructure needs that the current Internet infrastructure does not inherently support. Thin-client based VD protocols require several tens of Mbps of network bandwidth for supporting few tens of users, and low network latency is critical for smooth user interactions. As shown from VD protocol characterization studies in [2], one of the most common thin-client protocol in today's enterprises namely, PCoIP will use as much bandwidth given to it, reacts sensitively to varying network health, and delivers corresponding user quality of experience (QoE). Better quality translates to smoother video, crisp keyboard/mouse control actions for user applications in VDs. It should be noted that thin-client encoding latency may be of greater importance in certain scenarios where user interactivity with VD is more critical than audiovisual quality perception of VD screen scrapes. Moreover, bandwidth consumption is dependent on the content characteristics, thin-client display resolution and tuning of encoding parameters.

In addition, resource requirements for Desktop-as-a-Service (DaaS) are much higher (e.g., every VD may need resources on the order of 2 GHz CPU, 2 GB RAM and 2Mbps end-to-end network bandwidth) and predicting resource allocation needs is challenging due to bursty nature of workload profiles. In comparison, current cloud infrastructures that support online applications (e.g., photo/video sharing web-sites, customer relationship management software) are transaction-oriented and their workloads are more predictable [6], and approaches such as overprovisioning are not cost-prohibitive, opposed to the DaaS case. Hence, DaaS platforms are not widely supported by Cloud Service Provider (e.g., Amazon, Microsoft, Dell) due to above challenges in delivering optimal user QoE with current cloud infrastructures on the Internet.

To develop and demonstrate validity of intelligent resource allocation methodologies for VDCs, researchers need to experiment on multi-data center VDC settings with realistic user, network and system loads, and with geographically distributed thin-client sites. We found that the Global Environment for Network Innovations (GENI) infrastructure [11], which has emerged as the federated cloud to be ideal for controlled as well as real-world VDC experiments. It provides state-of-the art system and networking resources, as well as a vibrant user community that is engaged in the creation, support and usage of Future Internet technologies. It also provides a 'sliceable' Internet infrastructure with wide-area software-defined networking using OpenFlow technologies that provide the ability to experiment with dynamic allocations and migrations in VDC experiment slices. The computational power, networking bandwidth, network programmability, user opt-in mechanisms, instrumentation-and-measurement services, and experiment workflow tools in GENI [12] - [16] are also appealing to run cloud-based research experiments in a repeatable, scalable and distributed manner.

In this paper, we present a novel, intelligent resource placement framework that uses human-and-network awareness (i.e., user group profiles and network health measurements) for thin-client based VD delivery in VDCs. The framework leverages principles of software-defined networking and features a URB component that uses special 'marker packets' for: (a) "route setup" when handling non-IP traffic within VLANs between thin-client sites and data centers, (b) "path selection" and "load-balancing" of virtual desktop flows to improve performance of interactive applications and video playback, and to cope with faults such as link-failures or Denial-of-Service cyber-attacks. URB orchestrates "control plane", "data plane" and "measurement plane" components to handle corresponding flows within VDCs in a Future Internet infrastructure.

We implement our framework in a multi-domain GENI testbed spanning backbone and access networks connecting data center as well as thin-client user sites. Our implementation in GENI parallels how similar infrastructure slices are setup within intrinsic federations of content providers (e.g., Hulu) and content-delivery network providers (e.g., Akamai). It is an established practice today for network providers to have commercial service agreements that provide access and control interfaces within their infrastructures to content providers. Consequently, content providers (i.e., VD content in our case) are able to service end-users with content at different scales and resolutions with appropriate resource placement techniques and technologies. Our emulation experimentation leverages OpenFlow programmable networking, as well as perfSONAR [17] instrumentation-and-measurement capabilities for validating our resource placement framework under realistic settings. Using empirical results from our framework implementation and related experiments in the GENI testbed, we demonstrate the importance of scheduling regulated measurements that can be used for intelligent resource placement decisions. We also show the feasibility and benefits of using OpenFlow controller applications for path selection and load balancing between thin-client sites and data centers in VDCs.

To the best of our knowledge, our work is the first to detail an intelligent resource placement framework for VDCs that uses human-and-network awareness. Also, our work is the first to use performance intelligence of user group profiles and network health factors and leverage it within an OpenFlow controller application to handle VDC resource placement decisions under realistic settings. We believe that our intelligent resource placement framework and implementation in GENI experiments lay the foundation to develop Future Internet DaaS offerings that can handle: (a) user behavior/cyber-attacks/cross-traffic, (b) actual user accessible VDC applications for enterprise and residential end-user mobile computing, and (c) virtual classroom labs in universities.

The reminder of the paper is organized as follows: Section II describes related work. Section III presents our intelligent resource placement framework and its implementation workflow in a VDC. Section IV details our VDC experiment slice setup in a multi-domain GENI testbed. Section V contains salient performance results from our VDC resource placement experiments. Section VI concludes the paper.

II. Related Work

The rapid development of cloud computing environments and recent advances in virtualization have created new requirements for traditional networks and distributed computing. There have been several recent works that have proposed novel network virtualization and resource adaptation techniques between geographically distributed data centers supporting cloud computing environments.

OpenFlow technology [1] that is built upon software-defined networking principles and supported in several vendor switches has emerged as a prominent solution for programmable control of routing and other services (e.g., virtual machine migration) for application flows. Authors in [18] show how programmable flows can be realized by matching packet flows based on the IP addresses, MAC addresses and the port numbers. In their packet forwarding scheme, packets are forwarded to a particular output port if a flow entry for a client is found. Else, the packet is forwarded to a controller application, which then makes forwarding decisions. The demonstrations done in [19] and [20] are based up on this approach for the controller application implementation. In our work, we also adopt a similar approach for installing flows at the time of a thin-client's request to the URB for a VD connection. Once the flows are setup, the traffic between the thin-client and corresponding VD do not pass through the controller application and thus no additional delays are introduced within the application flows.

Alternate technologies such as Overlay Transport Virtualization (OTV) [21], and more recently, Virtual Extensible LAN (VXLAN) [22] have been proposed within industry groups that are in contrast with the open-standards used in OpenFlow solutions. VXLAN has been adopted within several industry vendor products for scalable LAN segmentation and for automated provisioning of logical networks between data centers across Layer 3 networks. A 24-bit LAN segment identifier and MAC-in-UDP encapsulation are used within VXLANs for achieving segmentation between data centers and to realize Layer 2 elasticity and IP address localization to enable for e.g., multi-tenancy and migration of virtual machines across multiple Layer 2 domains. In our work that relies on OpenFlow technology, we use the concept of 'marker packets' within packet headers that are processed by the controller application to handle traffic (i.e., non-IP traffic) within VLAN extensions across multiple Layer 3 networks.

In the context of implementation of intelligent adaptation within cloud environments, authors in [23] recommend a unified broker architecture for resource allocations of client requests similar to our URB-centric approach to handle thin-client requests for VDs. They also describe how unified resource brokers can perform critical functions such as performance monitoring, application isolation, live migration and suspend/resume capabilities of virtual machines for appropriate provisioning and placement functions. The work in [24] also supports the concept of central and integrated control for such critical functions through a unified resource broker that leverages a controller application to improve application performance (Hadoop application was discussed in their case) with relatively small overhead for configuration.

The work in [26] leverages correlation information between desktop traces to decide how many desktops can be provisioned at any given data center location. In [25], a fair resource allocation scheme is proposed where client requests are statically placed in a round-robin manner across data centers based on resource dominance levels. Works such as [27] and [28] propose dynamic placement solutions that involve virtual machine migrations between data centers. Authors in [27] use a reinforcement-learning based approach to automate virtual machine reconfiguration in response to changing application demands or resource supply. Authors in [28] develop an online-reconfiguration scheme that features a genetic algorithm that dynamically reallocates virtual machines across data centers by predicting the future workloads of the application, and aims to conserve overall power consumption. Similarly, a genetic algorithm approach is used in [29] to make resource allocations based on network topology and application requirements for data-intensive workloads.

In comparison to these works, we use an utility-directed provisioning and placement model i.e., U-RAM [3] that is dynamic in nature, and considers utility-functions of user groups obtained from real-time performance intelligence feedback corresponding to system, network and user QoE measurements. U-RAM initially places VDs and later re-allocates (i.e., migrates) them amongst distributed data centers such that performance and scalability in the cloud infrastructure is maximized. We remark that our earlier GENI experiments and results that are mainly related to U-RAM based 'provisioning' considerations within VDCs can be found in [30]. Whereas in this work, our focus is on experiments and results that relate to 'placement' considerations within VDCs, particularly leveraging OpenFlow capabilities for route setup, dynamic path selection and load balancing of VD application flows. In addition, we introduce the novel concept of using a marker packet for installing flows in a manner that does not require all flow traffic to pass through the controller, which may impact the forwarding rate. Once a marker packet for an ingress flow is processed by the URB, there is direct flow communication between the thin-client site and the VD application within the data center chosen by U-RAM.

III. VDC Resource Placement Framework

A. Architecture

Figure 2 shows our proposed intelligent VDC resource allocation architecture that specifies how the various components interact with each other when handling VD requests and application flows. The URB module functions are orchestrated by the "measurement engine", "service engine" and "routing engine" sub-modules via the "control", and "measurement" planes. When a new VD request arrives and its security token is validated, a resource allocation decision is made by the service engine based on the performance intelligence provided by the measurement engine. Using a secure channel, the service engine provisions the system resources at the preferred data center that can provide the best utility to the VD. The service engine also instructs the routing engine to program the corresponding flow tables or group tables in the intermediate switches using the OpenFlow protocol on a secure channel. Once such a provisioning is in place, the thin-client and VD application communications involving RDP/PCoIP protocols occur via the "data" plane and do not involve the URB.

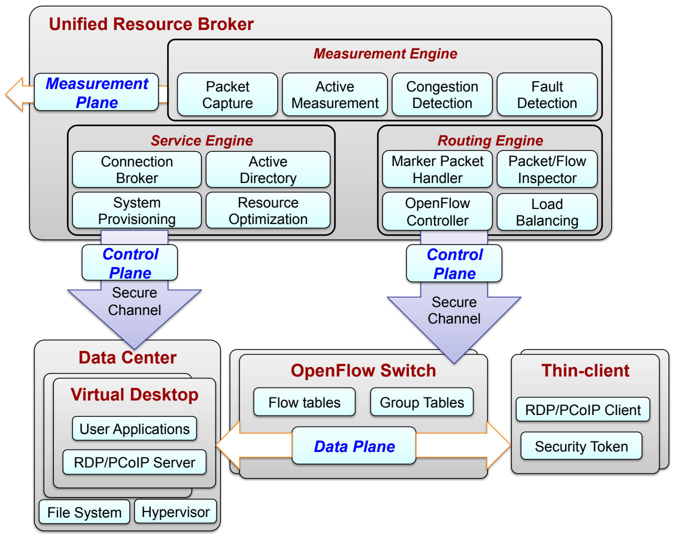

Fig. 2. Intelligent VDC Resource Allocation Architecture

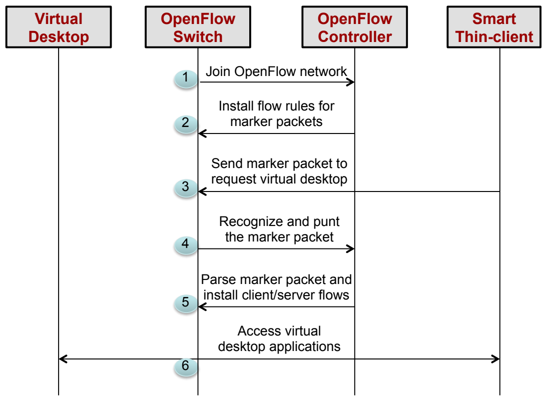
Virtual Desktop
OpenFlow Switch
OpenFlow Controller
Smart Thin-client
1
Join OpenFlow network
Install flow rules for marker packets
2
Send marker packet to request virtual desktop
3
Recognize and punt the marker packet
4
Parse marker packet and install client/server flows
5
Access virtual desktop applications
6

Fig. 3. Workflow Steps Illustration

Such a decoupling of the data plane achieved by leveraging OpenFlow technology for forwarding traffic avoids additional delays while handling VD application flows. It also allows any desired actions or adaptations to be taken in “real-time” at intermediate OpenFlow-capable network devices on packets that match these flows.

The measurement engine relies on instrumentation-and-measurement frameworks that are *open-source* (e.g., perfSONAR [17] for active measurements of network status using tools such as Ping and Iperf; Wireshark for passive measurements using TCPdump packet captures) as well as *proprietary* (e.g., VMware Power Tools or LiquidWare Labs Stratosphere UX that provide hypervisor related status measurements in terms of number of active VD connections, CPU and memory measurements). The collected measurements via the measurement plane are analyzed to identify device platform (e.g., tablet, smartphone, PC) and capabilities (e.g., resolution, identity attributes), detect congestion and faults in system and network devices, and such performance intelligence is passed to the service engine along with on-going monitoring information of the various status measurements.

The service engine authenticates user’s VD requests using standard mechanisms (e.g., Active Directory or LDAP), and allows them to access their entitled virtual desktops at a data center that provides the highest utility. The decision to provision VD resources at a data center for a VD request is provided by the resource optimization module in the service engine. The decision is based on the information available (through the measurement engine) regarding the resource slack levels obtained from hypervisor APIs, network health measurements, and based on the user group to which the VD request belongs to. For argument sake, we can consider two user groups: Engineering Site, and Distance Learning Site, each having a custom set of applications that use different amounts of CPU, memory and network bandwidth resources. We can generally assume that the Distance Learning Site uses more video streaming related applications, and hence the profile would indicate relatively higher amounts of network bandwidth provisioning. Similarly, the Engineering site can be generally assumed to use advanced applications (e.g., Matlab, Moldflow) that require relatively higher amounts of CPU and memory provisioning. The VD request is then assigned to a ‘desktop pool’ corresponding to the resource profile for the corresponding user group created by the system provisioning module at the chosen data center.

Algorithm 1 presents the sequential steps for the intelligent resource allocation to provision and place a new VD request. It uses U-RAM dynamic resource provisioning described in [3] as well as the path-selection and load-balancing that occurs by leveraging OpenFlow when maximizing Net-utility in the VDC. The input at the time of processing the new VD request includes the utility functions of the user groups, available slack levels in resources at data centers, and the possible path flow entries between the thin-client site of the new VD request and all of the data centers. If utility functions of user groups are not considered, fixed over provisioning schemes are selected that are cost-wise prohibitive compared to U-RAM. The decision of U-RAM placement i.e., the path selection for the thin-client to the data center that maximizes Net-utility is done by polling each data center’s change in utility if the new VD were to be placed at their site. The higher the change i.e., ΔU, the better the provisioning-placement combination at the data center due to resource slack availability, and thus better the Net-utility derived in the VDC. If sufficient resources to produce minimum satisfactory VD QoE do not exist at any data center, the VD request allocation is denied. The overall challenge is to deliver satisfactory VD QoE with changing load patterns and resource availability.

Based on the path selected i.e., the resource placement decision, the service engine delegates the routing engine to setup the routes between the thin-client site and the chosen data center. The service engine also ensures that any network disruptions do not affect the service (i.e., does not violate any Service Level Agreements (SLAs) pertaining to e.g., user-perceivable interaction response times in VD applications). A disruption will result in a VD session to be treated as a new VD request and the steps in Algorithm 1 will be invoked for subsequent re-placement. The routing engine maintains and controls the network links topology information. Hence, in the event there is a need to update the logic to switch paths (e.g., due to link failure or if a path that provides better performance is discovered by the measurement engine), or balance loads between the thin-client and data center sites (e.g., using backup routing instances with alternate routes), the routing engine can programmatically implement the adaptation rules as triggers that invoke steps of Algorithm 1 at intermediate OpenFlow-capable network switches in the order of milliseconds - without causing much loss in the original VD packet flow.

Algorithm 1 Intelligent Resource Allocation Algorithm

```
1:  Input: Utility functions $U_g(R)$ of user groups from offline
    profiling, list of available resources $\{R_1,\ R_2\ ...\ R_m\}$ at
    $l$ data centers, and list of paths $\{P_{i,1},\ P_{i,2}\ ...\ P_{i,k}\}$ from
    thin-client $i$ to every data center
2:  Output: Resource allocation $R_{i,l}$ and path flow entry $P_{i,k}$
    for new VD request $v_i$ that maximizes Net-utility $U$
3:  begin procedure
4:  for a VD request $vd_i$ belonging to group $g$ do
5:    for each data center $L_l$ do
6:      Update the current data center utility $U_l^{old}$
7:      Calculate $R_i^l$ using the U-RAM algorithm
8:      Calculate Set $\{P_{i,1},\ P_{i,2}\ ...\ P_{i,k}\}$
9:      for each path $P_{i,k}$ do
10:       $U_{l,P_{i,k}}^{new}$ = calculate the $U_l^{new}$ for path $P_{i,k}$
11:     end for
12:     $\Delta U_{l,P_{i,k}}$ = $max$ $\{$ $U_{l,P_{i,k}}^{new}$ $\}$ - $U_l^{old}$
13:   end for
14:   Place $vd_i$ at $L_l$ such that, $\Delta U$ = $max$ $\{$ $\Delta U_{l,P_{i,k}}$ $\}$
15:   Allocate $R_{i,l}$ to $vd_i$ with path $P_{i,k}$ only if $R_{i,l}$ produces
      $Q_{min}$ of usability, else deny the $vd_i$ allocation
16: end for
17: end procedure
```

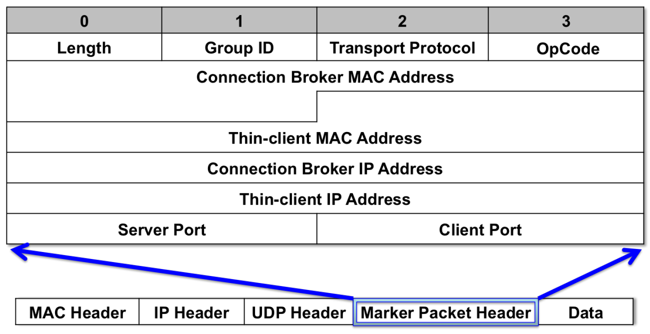

Fig. 4. Marker Packet and Header Format

B. OpenFlow Route Setup WorkFlow

To setup routes in the routing engine, a 'marker packet' handler module is used. The 'marker packet' concept is our novel contribution in this paper that is needed for orchestrating the workflow between the thin-client, URB and OpenFlow-capable switches, as shown in Figure 3.

The marker packet format as shown in Figure 4 is basically a UDP packet with the following header information:

- Length: Marker packet payload length
- Group ID: User group (e.g., Engineering Site, Distance Learning Site) that the VD request belongs to
- Transport Protocol: Remote desktop protocol used by the thin-client
- OpCode: Used to determine the services to be enabled on the packet flow
- Connection Broker IP/MAC Address: IP/MAC address of the connection broker
- Thin-client IP/MAC Address: IP/MAC address of the thin-client
- Server Port: Port number on the server to which the thin-client connects
- Client Port: Port number used at the thin-client

As shown in Figure 3, there is a six-step workflow to setup routes for thin-clients to connect to their provisioned VDs at the chosen data center. We can assume that the CSP has shipped a 'smart' thin-client that is pre-configured to connect to the CSP's VDC infrastructure. Steps 1 and 2 are completed a priori to any VD request arrival, and involves the OpenFlow switch joining the network controlled by the OpenFlow controller application running atop for e.g., NOX network operating system [19]. When a thin-client actually requests a VD in Step 3 by using the IP/MAC address of the connection broker pre-programmed by the CSP in the thin-client, it sends a marker packet that is recognized by the edge switch of the OpenFlow network, which then punts it in Step 4 to the URB of the VDC. The marker packet handler in the routing engine is invoked to parse the tuples in the header. Based on the parsing and the service engine instructions, the controller application in Step 5 sets up the client/server flows forwarding decisions from the thin-client location to the chosen data center. It also configures the intermediate network switches with any other service parameters such as actions upon link failure or events that trigger load balancing, flow traffic classification and QoS settings. The service engine then responds back to the thin-client with an updated marker packet that overwrites the IP/MAC address of the connection broker with the IP/MAC address of the entitled VD. In Step 6, the thin-client begins accessing the VD applications through the flows set up in the network by the controller application. Note that, in the event the URB migrates the VD as part of the resource optimization or system load balancing, it ensures that the controller application is consistent by updating the flow tables suitably.

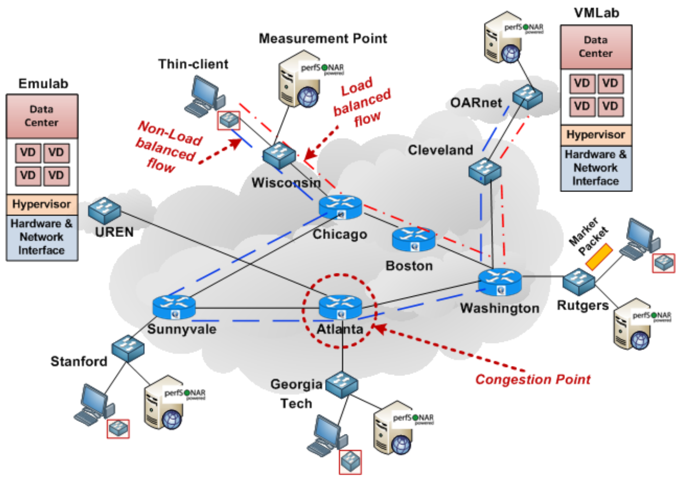

Fig. 5. VDCloud Experiment Slice Setup in GENI

IV. Slice Topology Setup and Configuration

We now describe our "VDCloud" experiment slice topology setup and configuration shown in Figure 5 that we developed to validate our intelligent VDC resource allocation architecture. The VDCloud experiment is part of a multi-domain GENI testbed spanning backbone networks (e.g., Ohio Regional Network - OARnet, Utah Regional Network - UEN, GENI meso-scale backbone - TangoGENI that is operated in part by Internet2/NLR) and access networks (campus networks of e.g., The Ohio State University, University of Utah, University of Wisconsin). Two parallel slices were configured on top of this common infrastructure testbed, one for the VDCloud experiment traffic, and another for the instrumentation and measurement traffic, which provides performance intelligence for resource adaptation decisions in the VDCloud experiment slice. Given that the system platforms are different for the instrumentation and measurement in comparison to the experiment, and to ensure better isolation and manageability, we chose to implement them in separate parallel slices.

There are four major components of our VDCloud experiment slice: (i) data centers, (ii) thin-client sites, (iii) programmable network, and (iv) unified resource broker. For the experiment results presented in this paper, we had a 2 data center configuration comprising of OSU VMLab and Utah Emulab resources with ESXi 4.0 hypervisor, and with extended VLAN connectivity (Layer 2) on the Internet. The "wide-area ProtoGENI" (WAPG) nodes located at meso-scale campuses (i.e., Stanford, Georgia Tech, Wisconsin, Rutgers) served as thin-client sites. One of the Emulab nodes was used to host our VDCloud experiment controller application. The traffic from the thin-client sites were made to flow within VLANs (Layer 2) through OpenFlow-enabled network segments in the TangoGENI meso-scale backbone [14] network before reaching the data centers. Each of the thin-client sites at the different universities also have a separate interface that allows IP routing (Layer 3) to the data centers. However, the Layer 3 paths comprise of additional router hops and multiple firewalls, and share network bandwidth resources at the edge with other campus core traffic. We configured the thin-clients with the RDP 6.0 protocol to access both interactive applications (e.g., Excel file handling with repeatable mouse clicks and keyboard strokes) and video playback (e.g., Windows Media Player file handling for a repeatable video file content) at the data center VDs.

The perfSONAR [17] capabilities for instrumentation-and-measurement were leveraged in the parallel slice that comprised of dedicated measurement points at various sites as shown in Figure 5. perfSONAR supports both active and passive measurements of various network health metrics (e.g., one-way delay, TCP throughput, UDP throughput, loss, jitter, router interface utilization) through support for tools such as OWAMP, Iperf/BWCTL, Cricket-SNMP and others. It supports measurement data collection, storage and publish/subscribe of data archives of current and historic measurements via standardized web services, and has been widely deployed by over 100 user communities in industry and academia, worldwide. We leverage perfSONAR active measurement tools for cross-traffic load generation, and to augment load generated by concurrent thin-client connections on common paths.

The slice topology has multiple network paths with long and short geographical distances between the thin-client sites and data centers. As a result, the testbed is suitable to investigate the use of OpenFlow for resource placement experiments (e.g., path selection, load balancing) as described in Algorithm 1, involving multiple thin-clients connecting to the data centers on diverse paths. We can see from Figure 5 how the testbed can be used to form a congestion point at Atlanta for Wisconsin thin-client flows, and how this congestion point can be avoided by invoking Algorithm 1 steps to load balance the flows through the Boston and Washington hops' path.

In addition, we can note the use of our 'marker packets' sent from the thin-client sites as part of non-IP traffic sessions in the VDCloud experiment slice. We ensured that our OpenFlow controller application implementation in conjunction with the marker packets handler processed the marker packets and related actions (as described in Section III) in the GENI infrastructure routers/switches at the hardware-level, as opposed to at the software-level, where performance can be very slow. Currently, most actions in OpenFlow controller application implementations that pertain to Layer 2 headers, such as those used in case of our marker packets, can be performed at hardware-level in the GENI infrastructure routers/switches. However, any controller application actions that involve manipulating IP headers are mostly done at the software-level. Further, we remark that - before actual deployment in the testbed, we recreated our slice topology in MiniNet [31] and validated our controller application functionality with simulated traffic.

V. VDCloud Performance Results in GENI

In this section, we first describe the instrumentation-and-measurement results to check the connectivity and setup-correctness across the multi-domain GENI testbed. Next, we present results to show how performance intelligence of Layer 2 OpenFlow path and Layer 3 Internet path can be used with OpenFlow technology in the URB to improve VD application performance. Lastly, we discuss the results implication and suggest any issues that need to be addressed.

A. Connectivity Measurements

We used Rutgers thin-client site as an origination point for the connectivity measurements using active measurement tools in the testbed. We used ARPing tool for Layer 2 path connectivity measurements, and IP Ping tool for Layer 3 path connectivity measurements, respectively. The TCP throughput measurements (or end-to-end available bandwidth measurements) on the Layer 2 and Layer 3 paths were obtained using the Iperf/BWCTL tool in perfSONAR. Tables I and II show the active measurement results averaged over 10 runs for the Internet and OpenFlow paths, respectively.

As expected, the latency on the paths increases as the geographical distance between the source and destination sites increase. For example, latency from Rutgers to VMLab (both source and destination sites are on the east coast of US) is lesser than that of Rutgers to Stanford or Emulab (destination sites are on the west coast of US). There are minor differences in the latency and throughput measurements between the Internet and OpenFlow paths, and the performance of the OpenFlow path is marginally better in all the cases; more packets can be sent and received on the OpenFlow path than on the Internet path.

As an example to differentiate the route topology in order to explain the performance differences between the Internet path from the OpenFlow path, we can use the Figures 6 (a) and (b) that show the intermediate hops on the two paths between Rutgers thin-client site and VMLab data center site. Although the network segments have common Internet Service Providers (i.e., NJEDge, MAGPI, Internet2 and OARnet), the routing in the OpenFlow path that we setup through the controller application is configured differently on Layer 2 devices and with lesser number of hops. Recall that the Rutgers edge for the Internet path has multiple firewalls, and the related network bandwidth resources are shared with other campus core traffic, whereas there is a separate direct connection to the NJEdge backbone at the Rutgers edge for the OpenFlow path. In the following sub-section i.e., Section V-B, we show that these differences in the OpenFlow and Internet paths notably impact the VD performance for interactive and video playback applications, and ultimately the VD user QoE.

To verify the performance isolation in the data plane spanning the two parallel slices of the testbed, we studied the impact of scheduling our BWCTL tool throughput measurements

TABLE I
ACTIVE MEASUREMENTS ON INTERNET PATH FROM RUTGERS SITE

Site Type	Site Name	Latency (ms)	Throughput (Mbps)
Data Center	VMLab	24	415
	Emulab	84	236
Thin-client	Wisconsin	36	346
	Stanford	86	223
	Georgia Tech	56	284

TABLE II
ACTIVE MEASUREMENTS ON OPENFLOW PATH FROM RUTGERS SITE

Site Type	Site Name	Latency (ms)	Throughput (Mbps)
Data Center	VMLab	23	434
	Emulab	84	247
Thin-client	Wisconsin	34	362
	Stanford	84	236
	Georgia Tech	55	305

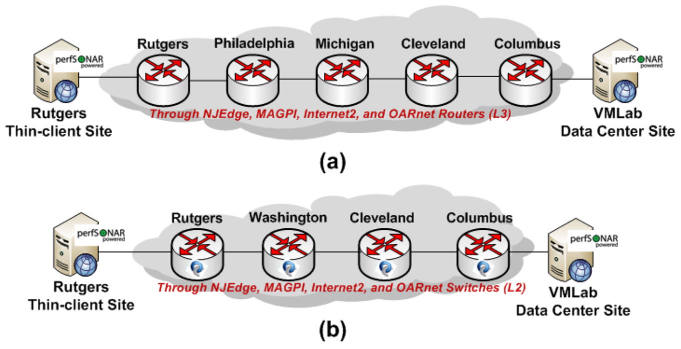

Fig. 6. (a) Internet Path Route; (b) OpenFlow Path Route

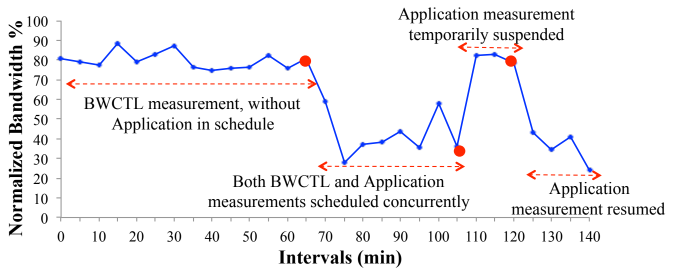

Fig. 7. Impact of Throughput Measurement and Application Traffic Scheduling

for gathering performance intelligence along the OpenFlow path in conjunction with the application traffic. Figure 7 shows how the end-to-end available bandwidth (normalized) along the OpenFlow path is impacted when both the throughput measurements and application traffic are scheduled concurrently. Given the lack of performance isolation in the OpenFlow path as seen from the impact, it is clear that the performance intelligence gathering must be carefully orchestrated and regulated so that it can be used for intelligent resource placement decisions, while not affecting the desired repeatability and predictability in the performance of application traffic flows in GENI experiments.

The advantage in using the controller application in the OpenFlow path is that the global network view is available for meta-scheduling of measurement and application traffic flows on the shared distributed resources so that there is isolation in the data plane. However, such meta-scheduling functionality needs to be explicitly programmed in the controller application because isolation of traffic within and between slices sharing infrastructure components is not inherently handled by the NOX network operating system [32]. Hence, if there are two independently-developed controller applications that share network segments, meta-scheduling should consider conflict-detection and resolution by mechanisms such as sharing flow-tables of common switches or using common switch ports for forwarding traffic. The conflict-detection can be specified to the controller application based on empirical results of co-scheduling traffic that results in critical impact, such as in our case shown in Figure 7 for measurement traffic and application traffic. For cases where performance is impact due to actual application loads, the controller application can be programmed to automatically react to event triggers of degraded performance for path switching and load balancing as shown in the following sub-section i.e., Section V-B.

B. Application Performance Measurements

Figure 8 shows the performance comparison between the Internet and OpenFlow paths as experienced by the VD applications at VMLab being accessed by thin-clients at the Rutgers site. As stated earlier, the VD applications we chose in our experiments correspond to interactive applications (e.g., Excel file handling with repeatable mouse clicks and keyboard strokes) and video playback (e.g., Windows Media Player file handling for a repeatable video file content). We remark that these two applications are representative of the broad 'interactive' and 'streaming' user QoE types, respectively.

We can observe that the interactive applications consume more bandwidth and in turn take higher task time in the Internet path, in comparison with the corresponding measurements on the OpenFlow path. This suggests higher cross-traffic congestion levels on the Internet path that cause more effort on the user side in terms of mouse clicks and keyboard strokes, and consequently higher number of packets being sent and received. In the case of video playback, we can observe that the video playback consumes more bandwidth, and in turn provides higher video quality in the OpenFlow path, in comparison with the corresponding measurements on the Internet path. This suggests that there is more end-to-end available bandwidth being observed at the media player client on the OpenFlow path that consequently leads to a higher resolution video encoding to be selected that has a higher

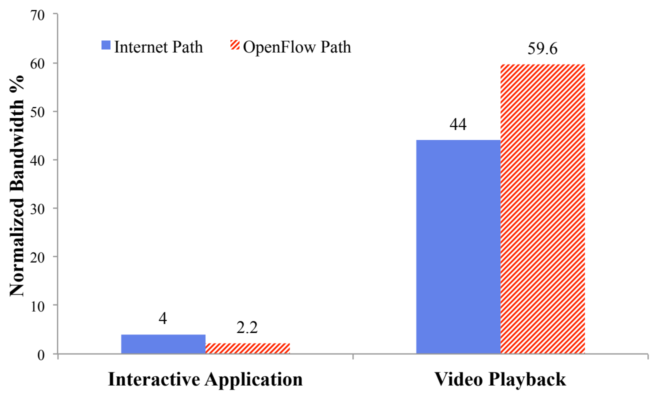

Fig. 8. Paths Performance Comparison

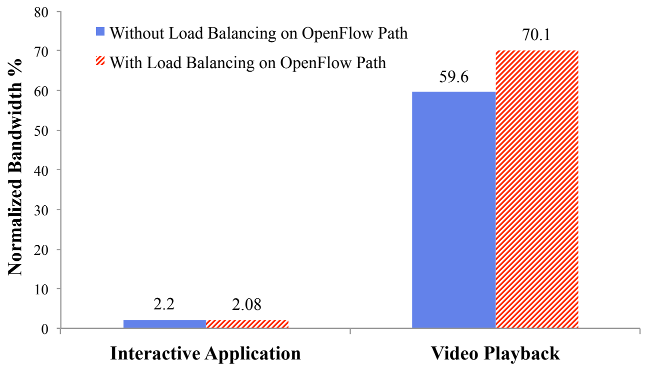

Fig. 9. Load Balancing Performance Comparison

packet rate. Such a performance intelligence information if available in real-time at the controller application during can be used for path selection that ensures improved VD application performance.

In addition, the VD application performance can be further improved if the global network view and performance intelligence of diverse OpenFlow paths can be leveraged at the controller application in real-time to - for e.g., route traffic around hot-spots or to react to faults such as link-failures or Denial-of-Service cyber-attacks. By using suitable triggers to recognize such human-and-network aware events, load balancing through Algorithm 1 steps can be invoked. Figure 9 shows how the interactive application and video playback performance measurements show further improvement at the Rutgers site by load balancing the thin-client flows from Wisconsin by sending them through the Boston and Washington hops' path.

C. Discussion

It should be evident that the performance improvements perceived in our limited GENI testbed scale of VD sessions can be extrapolated to obtain much greater benefit in more realistic VDC settings with VD requests on the order of several hundreds. The performance intelligence data related to human-and-network awareness, if collected effectively, can greatly help in defining triggers that invoke OpenFlow actions in intermediate switches that can cope with faults such as link-failures or Denial-of-Service cyber-attacks. Given that Future Internet infrastructures will rely on hypervisor and software-defined networking technologies, our framework can be generally implemented within slices of DaaS offerings serving different user groups. In addition, the path performance characteristics are likely to be more diverse, and thus our optimization will notably improve the scalability of VDCs, and will result in higher profit to CSPs or potentially cheaper price per VD to DaaS users.

VI. Conclusion

The emergence of OpenFlow technology that is built upon software-defined networking principles and supported in several vendor switches provides new capabilities for programmable control of routing and other services (e.g., virtual machine migration) for application flows. In this paper, we leveraged these capabilities in a novel and intelligent resource placement framework that uses human-and-network awareness (i.e., user group profiles and network health measurements) for thin-client based VD delivery in VDCs. We showed how special OpenFlow header 'marker packets' can be handled for implementing: (a) "route setup" when handling non-IP traffic within VLANs between thin-client sites and data centers, (b) "path selection" and "load-balancing" of virtual desktop flows to improve performance of interactive applications and video playback, and to cope with faults such as link-failures or Denial-of-Service cyber-attacks. From our framework implementation and empirical results in a multi-domain GENI testbed spanning backbone and access networks, we demonstrated the importance of scheduling regulated measurements that can be used for intelligent resource placement decisions. Further, we showed the feasibility and benefits of using OpenFlow controller applications for path selection and load balancing between thin-client sites and data centers in VDCs.

Our planned future work involves developing a meta-scheduler capability in OpenFlow controller applications to derive performance intelligence without affecting application traffic. We also are developing more comprehensive set of VD application performance metrics that can be used in OpenFlow-based load balancing triggers in VDCs.

References

[1] N. Mckeown, T. Anderson, H. Balakrishnan, et. al., "OpenFlow: Enabling Innovation in Campus Networks", *ACM SIGCOMM Computer Communication Review*, Vol. 38, No. 2, Pages 69-74, 2008.
[2] A. Berryman, P. Calyam, A. Lai, M. Honigford, "VDBench: A Benchmarking Toolkit for Thin-client based Virtual Desktop Environments", *IEEE Conference on Cloud Computing Technology and Science*, 2010.
[3] P. Calyam, R. Patali, A. Berryman, A. Lai, R. Ramnath, "Utility-directed Resource Allocation in Virtual Desktop Clouds", *Elsevier Computer Networks Journal*, 2011.
[4] D. Gmach, S. Krompass, A. Scholz, M. Wimmer, A. Kemper, "Adaptive Quality of Service Management for Enterprise Services", *ACM Transactions on the Web*, Vol. 2, No. 8, Pages 1-46, 2008.
[5] P. Padala, K. Shin, et. al., "Adaptive Control of Virtualized Resources in Utility Computing Environments", *Proc. of ACM SIGOPS/EuroSys*, 2007.
[6] B. Urgaonkar, P. Shenoy, et. al., "Agile Dynamic Provisioning of Multi-Tier Internet Applications", *ACM Transactions on Autonomous and Adaptive Systems*, Vol. 3, No. 1, Pages 1-39, 2008.
[7] H. Van, F. Tran, J. Menaud, "Autonomic Virtual Resource Management for Service Hosting Platforms", *Proc. of ICSE Workshop on Software Engineering Challenges of Cloud Computing*, 2009.
[8] K. Beaty, A. Kochut, H. Shaikh, "Desktop to Cloud Transformation Planning", *Proc. of IEEE IPDPS*, 2009.
[9] N. Zeldovich, R. Chandra, "Interactive Performance Measurement with VNCplay", *Proc. of USENIX Annual Technical Conference*, 2005.
[10] J. Rhee, A. Kochut, K. Beaty, "DeskBench: Flexible Virtual Desktop Benchmarking Toolkit", *Proc. of Integrated Management*, 2009.
[11] Global Environment for Network Innovations - http://www.geni.net
[12] ProtoGENI Aggregate in GENI - http://www.protogeni.net
[13] PlanetLab Aggregate in GENI - http://groups.geni.net/geni/wiki/PlanetLab
[14] TangoGENI Operations - http://groups.geni.net/geni/wiki/TangoGENI

[15] OnTimeMeasure Instrumentation and Measurement Service in GENI - http://groups.geni.net/geni/wiki/OnTimeMeasure
[16] Gush Experimenter Control Tool in GENI - http://groups.geni.net/geni/wiki/GushProto
[17] A. Hanemann, J. Boote, E. Boyd, et. al., "perfSONAR: A Service Oriented Architecture for Multi-Domain Network Monitoring", *Proc. of Service Oriented Computing*, Springer Verlag, LNCS 3826, Pages 241-254, 2005.
[18] N. Motoo, I. Atsushi, Y. Su-hun, W. Hiroyuki, I. Akio, K. Toshiyuki, "New Cloud Networking Enabled by ProgrammableFlow", *NEC Technical Journal*, Vol. 5, No. 2, 2010.
[19] A. Tavakoli, M. Casado, T. Koponen, S. Shenker, "Applying NOX to the Datacenter", *Proc. of ACM HotNets*, 2009.
[20] D. Erickson, G. Gibb, et. al., "A Demonstration of Virtual Machine Mobility in an OpenFlow Network", *ACM SIGCOMM Demo*, 2008.
[21] OTV IETF Submission - https://datatracker.ietf.org/doc/draft-hasmit-otv
[22] VXLAN IETF Submission - http://tools.ietf.org/html/draft-mahalingam-dutt-dcops-vxlan-00
[23] L. Grit, D. Irwin, A. Yumerefendi, J. Chase, "Virtual Machine Hosting for Networked Clusters: Building the Foundations for Autonomic Orchestration, *Proc. of Workshop on Virtualization Technology in Distributed Computing*, 2006.
[24] G. Wang, T. Ng, A. Shaikh, "Programming Your Network at Run-time for Big Data Applications", *Proc. of Workshop on Hot Topics in Software Defined Networking*, 2012.
[25] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, I. Stoica, "Dominant Resource Fairness: Fair Allocation of Heterogeneous Resources in Datacenters", *Proc. of NSDI*, 2010.
[26] A. Kochut, K. Beaty, H. Shaikh, D. Shea, "Desktop Workload Study with Implications for Desktop Cloud Resource Optimization", *Proc. of Workshop on Parallel and Distributed Processing*, 2010.
[27] J. Rao, X. Bu, G. Yin, "VCONF: A Reinforcement Learning Approach to Virtual Machines Auto-configuration", *Proc. of International Conference on Autonomic Computing*, 2009.
[28] H. Mi, H. Wang, G. Yin, Y. Zhou, D. Shi, L. Yuan, "Online Self-reconfiguration with Performance Guarantee for Energy-efficient Large-Scale Cloud Computing Data Centers", *Proc. of IEEE Conference on Services Computing*, 2010.
[29] G. Lee, N. Tolia, P. Ranganathan, R. Katz, "Topology-aware Resource Allocation for Data-intensive Workloads", *SIGCOMM Computer Communications Review*, Vol. 41, No.1, Pages 120-124, 2011.
[30] P. Calyam, M. Sridharan, Y. Xu, K. Zhu, A. Berryman, R. Patali, A.Venkataraman, "Enabling Performance Intelligence for Application Adaptation in the Future Internet", *Journal of Communications and Networks*, 2011.
[31] Mininet Software-defined Network Emulation Platform - http://yuba.stanford.edu/foswiki/bin/view/OpenFlow/Mininet
[32] C. Schlesinger, A. Story, S. Gutz, N. Foster, D. Walker, "Splendid Isolation: Language-Based Security for Software-Dened Networks", *Proc. of Workshop on Hot Topics in Software Defined Networking*, 2012.