

Flow-based Performance Enhancements of SAGE Visualcasting using OpenFlow Programmable Networking

Sungho Shin, Namgon Kim, Namgon Lucas Kim, and JongWon Kim

Networked Media Lab., School. of Information and Communications,

Gwangju Institute of Science and Technology (GIST), Gwangju, 500-712, Korea,

Email:{shshin, ngkim, namgon, jongwon}@nm.gist.ac.kr

Abstract—OpenFlow protocol allows researchers to use programmable networking techniques to flexibly assign networking resources and manage network traffic according to user demands. In this paper, by taking SAGE Visualcasting as an example user application, we propose that OpenFlow-based programmable networking can be interactively utilized together with other monitoring schemes to enhance the performance of SAGE Visualcasting. By visually identifying (i.e., separating) SAGE Visualcasting flows from all lists of flows (i.e., flowspace) and by monitoring current status of all involved nodes, we can interactively capture how flow-related parameters of SAGE Visualcasting performance vary over time. Then, by linking it with the visible performance (i.e., displayed visual quality) of SAGE Visualcasting, we can interactively apply flow-based enhancement (e.g., alternating the paths of flows in trouble) by taking advantage of OpenFlow-based programmable networking.

I. INTRODUCTION

A number of important innovations are creating a paradigm shift in networking that leads to higher levels of network programmability. These innovations include the separation between transmission hardware and control software, availability of open programmable network interfaces, accelerated virtualization of networking infrastructure, rapid creation and deployment of new network services and environments for resource partitioning, and coexistence of multiple distinct network architectures [1]. The basic idea of OpenFlow [2] is one of the programmable networking solutions to create a system that grants researchers the greatest possible amount of control over the packet flow in their routers and switches, while still being easy and cheap to integrate into commercial networking hardware. In order to achieve this, packet handling decisions are based on a common subset of information that different switches extract from a packet during its processing: the interface it was received on and the most basic contents of its Ethernet, IP and TCP/UDP headers (when they are applicable) [2]. OpenFlow-enabled devices store tuples of this data in their Flow Table and associate them with an action, e.g., dropping the packet or sending it out on a specific interface. An OpenFlow network is formed basically by one or more OpenFlow-switches that contains a flow table with the defined flow entries and the actions to be performed, one or more controllers that are the responsible of adding and dropping flow entries, a secure channel that interconnects the switch

with the controller and the OpenFlow protocol for signaling the whole architecture [3].

Recently OpenFlow has been extensively applied to improve network efficiency. OpenRoads experiments with seamless handover between the WiMAX and WiFi wireless nodes [4]. By dynamically re-directing how traffic flows through the network, OpenRoads can provide finer-grained control of mobility policies, e.g., make-before-break or break-before-make connections. Plug-N-Serve also shows the effect of dynamically adding and removing computing resources to the system, increasing the request arrival rate, altering the CPU or network load of each request, and changing load-balancing algorithms [5].

By exploiting advanced high-performance networking capability, SAGE(scalable adaptive graphics environment) [6], as a solution for multiparty visual sharing, generates quite a number of control and data flows among the participating sites. Especially, SAGE Visualcasting [7] provides bridging-based multi-party sharing of visualization. That is, via an entity called as SAGE Visualcasting bridge, it selectively combines the visual sources (from video camcorders or from rendered 3D graphics) of participating sites and displays the combined visual on the networked tiled display (NeTD) of each site. Since SAGE Visualcasting aims interactive collaboration among remote sites, the visual sources are bandwidth intensive and delay stringent. For example, quite often, uncompressed HD video (i.e., slightly over 1Gbps) or light-weight compressed version of it (i.e., DXT-compressed around 300Mbps) is adopted as the visual source. Thus, when the underlying network can not provide the required performance guarantee, it is easily subject to performance degradation and the participants are suffering from the slowdown and/or corrupted visual display. However, when the performance degradation happens, it is not easy to figure out why it is happening. We have to know what is the topology and current state of the underlying network. We also need to identify which nodes, along the flow paths of SAGE Visualcasting control/data, are either network congested or system bounded (i.e., CPU overflow). Only after checking out all this information, we may be able to find an appropriate reaction to solve the performance problem.

To address the above performance trouble-shooting challenge of SAGE Visualcasting, in this paper, we propose flow-

based performance enhancements by employing OpenFlow-based programmable networking. By visually identifying (i.e., separating) SAGE Visualcasting flows from all lists of flows (i.e., flowspace) and by monitoring current status of all involved nodes, we can interactively capture most flow-related parameters of SAGE Visualcasting performance. Then, by linking it with the visible performance (i.e., displayed visual quality) of SAGE Visualcasting, we can interactively apply flow-based enhancement (e.g., alternating the paths of flows in trouble) by taking advantage of OpenFlow-based programmable networking.

The rest of paper is organized as follows. Section 2 provides an overview of OpenFlow network and SAGE Visualcasting. Section 3 presents SAGE Visualcasting over the OpenFlow network and the explains how we can interactively identify and solve SAGE Visualcasting performance problems. Experiment results are provided and evaluated in Section 4. Finally we conclude this paper in Section 5.

II. BACKGROUND

A. OpenFlow-based Programmable Networking

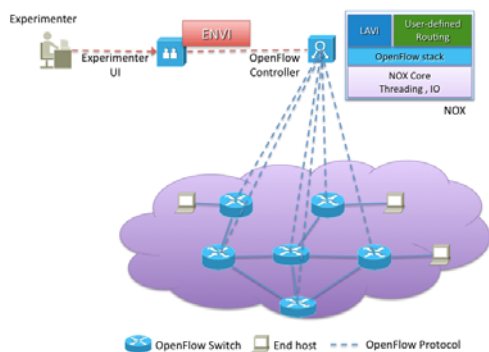


Fig. 1. An OpenFlow-based programmable network.

The OpenFlow protocol lets researchers to do experiments of testing new networking protocols in production networks without interfering existing traffic. It allows switches to segment traffic into flows by using rules configured in a particular switching table called OpenFlow flow table. As shown in Figure. 1, an OpenFlow network generally consists of OpenFlow switches, which are either PC-based OpenFlow switches with many network interfaces or commercial OpenFlow switches, and a OpenFlow controller responsible for commanding how to handle OpenFlow switches. An OpenFlow switch consists of at least three parts. First, a flow table, with an action associated with each flow entry, tells the switch how to process the flow. A secure channel connects the switch to a remote control process (called the OpenFlow controller), allowing commands and packets to be sent between an OpenFlow controller and the switch using the OpenFlow Protocol. Note that the OpenFlow protocol provides an open and standard way for a controller to communicate with a switch. By specifying a standard interface (i.e., the OpenFlow protocol) through which entries in the flow table can be defined externally, the OpenFlow switch

avoids the trouble of researchers to directly program the switch. The majority of OpenFlow controllers are currently using NOX (network operation system) [8]. NOX runs on commodity hardware and provides a software environment on top of which programs can control large networks. To provide a central programming model for an entire network, NOX exports a programmatic interface on top of which multiple networking programs (which we call applications) can run. These applications can hook into network events, gain access to traffic, control the switch forwarding decisions, and generate traffic.

OpenFlow-based programmable networking generally operates in the following steps. First, end and a connected source host sends packets. OpenFlow switch receives packets. All packets processed by this switch are compared against the flow table. If a matching entry is found, any actions for that entry are performed on the packet (e.g., the action might be to forward a packet out a specified port). If no match is found, the packet is forwarded to the controller over the secure channel. The OpenFlow controller, which is responsible for determining how to handle packets without valid flow entries, manages the switch flow table by adding and removing flow entries. The OpenFlow switch forwards the flow's packet to given port(s). End destination host finally receives packets.

To the network administrators, OpenFlow provides a set of elements that allows them to define flows and to define the path that they will follow without disturbing the existing traffic. It also provides methods to define policies to automatically find paths that accomplish certain characteristics, like having higher bandwidth, suffering less latency, reducing the number of hops and reducing the required energy that needs traffic to reach its destination. Therefore we can see OpenFlow as a tool that allows network administrators to perform traffic engineering in their systems. To assist this, ENVI(extensible framework for network visualization and control) was built with special consideration for visualizing an OpenFlow network [9]. The ENVI framework provides built-in functionality to take advantage of the OpenFlow protocol. For example, it uses the OpenFlow protocol to retrieve switch configurations, query link utilizations, or even modify a switch's flow table to alter routing.

B. SAGE Visualcasting and Performance Limitations

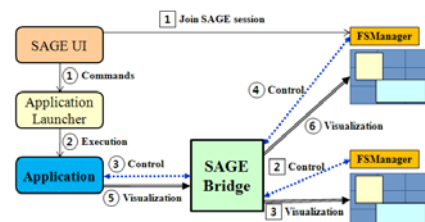


Fig. 2. Typical operation procedures of SAGE Visualcasting system.

Visual sharing means interactive and concurrent sharing of the visualization of multiple high-resolution media objects like

images, videos, graphics and so on. SAGE Visualcasting [7], as a visual sharing solution, relies on SAGE middleware for managing real-time media streams from remotely distributed rendering clusters to scalable display walls (i.e., NeTDs) over high-speed networks. This high-resolution visual sharing of each display-wall must be capable of multicasted visualization to all collaborating nodes.

As illustrated in Fig. 2, a SAGE Visualcasting system is composed of SAGE application nodes, a SAGE bridge, and SAGE NeTDs. The SAGE application nodes are divided into two types of devices: media servers and media adapters. The media servers play the role of media providers while the media adapters are specific devices to process media adaptations. The SAGE bridge duplicates visualization frames and adaptively transmits them to the SAGE NeTD of each end site. Each SAGE site also has a FSManager (FreeSpace Manager) to control the number of displays, IP addresses of displays and other running parameters.

This SAGE Visualcasting has several weaknesses as follows. The main bottleneck is at the media adapter and SAGE bridge. The media adapter is involved with heavy processing activities like video decoding and graphics of rendering. The SAGE bridge not only transports decoded media but also tiles the rendered data according to the display configuration of end sites. With UDP transport, the flow control of SAGE can not perfectly guarantee reliability. Network congestion reduces the overall available bandwidth and transport loss and delay occurs. If the notifying message to lower transport rate is lost, the SAGE bridge sends large amount of unnecessary data to NeTDs before it receives this control message. Delayed synchronization packet causes somehow irregular result. Especially the delay of control traffic can cause bigger problems than that of data traffic. If long delayed control traffic is sent from the FSManager, the SAGE bridge can not transmit the data traffic on time.

III. FLOW-BASED PERFORMANCE ENHANCEMENTS USING OPENFLOW PROGRAMMABLE NETWORKING

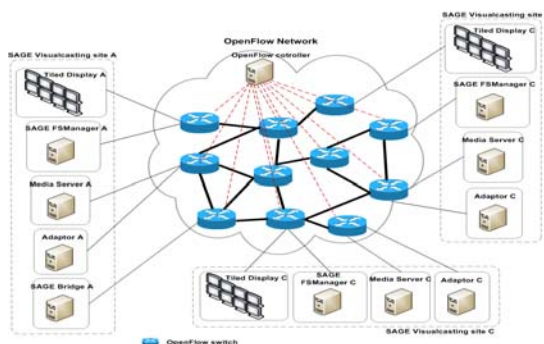


Fig. 3. SAGE Visualcasting over an OpenFlow network.

To overcome current limitations of SAGE Visualcasting, we experimentally integrate SAGE Visualcasting and OpenFlow programmable networking. As shown in Fig. 3, all SAGE Visualcasting traffic goes through a OpenFlow network. We

handle each traffic generated from SAGE Visualcasting by applying appropriate actions.

A. Design for Flow-based Performance Enhancements

We now present an effective solution as shown in Figure. 4. We first identify the traffic of SAGE Visualcasting. We then monitor the status of flows such as network bandwidth. Finally, we present flow-based enhancements to overcome SAGE Visualcasting limitations in the congested network.

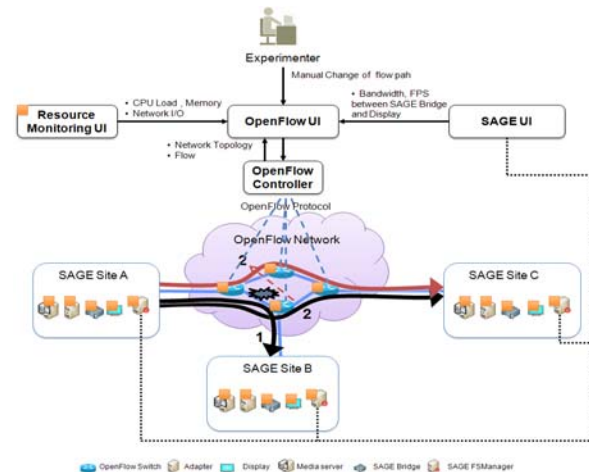


Fig. 4. Architecture of flow-based enhancement.

1) *Identify the flows of SAGE Visualcasting:* To identify the flows of SAGE Visualcasting, we need to clearly distinguish the control flows generated from the FSManager and the data flows among media servers, media adapters, SAGE bridge, and NeTDs. By extending ENVI, we provide an OpenFlow UI to the experimenters. The OpenFlow UI supports experimenters together with the OpenFlow controller as follows.

- The OpenFlow controller collects the IP address of SAGE Visualcasting nodes and then makes the database of them.
- The OpenFlow controller collects flow information through each OpenFlow switch and then makes the database of flows when the flow is generated by SAGE Visualcasting.
- The source and destination IP addresses of flows in the flow database are matched with the database of SAGE Visualcasting nodes. If a flow includes the IP address of FSManager, it belongs to the control flows. Otherwise, it mostly belongs to the data flows.
- Each distinguished flow is stored in the separate tables for the control and data flows respectively. Each flow has a "type" value that is used to differentiate between the control and data flows. This "type" information is important because other modules use it to differentiate flows. Depending on "type", the flow is drawn differently, i.e., in different shapes, colors and so on.

2) *Monitor the status of flows and computing/ networking resources:* We can monitor the status of flow bandwidth, delay and loss in each OpenFlow switch. The OpenFlow controller

collects these information from the OpenFlow switches and forwards it to the OpenFlow UI. The key is how to monitor the throughput of SAGE Visualcasting. We normally visualize monitored networking performance through the SAGE UI. However, users can not easily identify the flows at each SAGE site. To overcome this issue, we approach as follows:

- The FSManger collects the status of networking performance from each NeTD node and then stores it. Data includes source and destination IP addresses.
- Through the interface between OpenFlow and SAGE Visualcasting, the NOX-driven OpenFlow controller obtains these data.
- To allow interaction between the OpenFlow controller and OpenFlow UI, with flow identification, we match them with the identified flows based on IP address.

Flow Monitoring is not sufficient and we still need to know the status of computing resources (i.e., CPU usage, memory and so on) in each node. Separate Resource Monitoring UI monitors the status of resources of all OpenFlow switches and SAGE Visualcasting nodes. When the OpenFlow UI sends a status request using an interface addressed to a predefined IP address and port, and the response is directly returned by using the information in Resource Monitoring UI. The OpenFlow UI visualizes these status when there are problems in specific nodes. For example, when the SAGE bridge rapidly increases CPU usage or the OpenFlow switch uses up the network bandwidth, this node can be marked.

3) *Interactive flow-based enhancements using UIs:* Through the OpenFlow UI, SAGE UI, and Resource Monitoring UI, the experimenter monitors the status of SAGE Visualcasting over OpenFlow networking. We propose interactive flow-based enhancements using OpenFlow UI. For example, we can alternate the paths of flows in trouble. The OpenFlow UI requests the message of changing flow path to the OpenFlow controller by interacting with mouse or keyboard. The OpenFlow Controller finds the flow that will be changed from the flow database and then commands to the OpenFlow switches to modify the flow table. Detailed procedure of the flow-based enhancements is as follows:

- With provided UIs, the experimenter periodically monitors not only the performance status of SAGE Visualcasting but also the problem status of networking and computing resources.
- If the experimenter observes the performance degradation of SAGE Visualcasting, the experimenter first checks the performance bottleneck points.
- The Experimenter interactively changes the flow path to avoid the trouble points using the OpenFlow UI.

B. Implementation

Our preliminary implementation is summarized as follows:

- Flow identification for the flows of SAGE Visualcasting: All traffic in SAGE Visualcasting is represented in the OpenFlow UI with pre-arrayed network topology of the nodes and links. We develop a NOX application by

extending the ENVI framework [9]. In the OpenFlow networks, we provide an easy way to do network visualization by using LAVI and ENVI. LAVI is an extensible backend for network visualization, and has been developed in sync with the GUI fronted, ENVI. We modified LAVI to support communication between ENVI and visualization.

- Monitor the status of flows and computing and networking resources: We currently do not have flow status monitoring. But we do monitor the computing and networking resources of each node by using a Ganglia Monitoring tool [10]. For the Resource Monitoring UI, Ganglia can monitor the network I/O, CPU, memory and so on in each node of SAGE Visualcasting and OpenFlow. Networking performance of SAGE Visualcasting is also monitored by the SAGE UI.
- Interactive flow-based enhancement using UIs: Changing of the flow path is operated by mouse drag and drop in the OpenFlow UI.

IV. EXPERIMENTS SETUP AND RESULTS

In this section, we experiment SAGE Visualcasting over OpenFlow programmable networking with two collocated clusters.

A. Testbed

We create a visualization sharing environment by involving two sites connected by an OpenFlow network, as shown in Figure. 5. Cluster-A consists of media server A, media adapter A, 4x2 NeTD and FSMangerA. Cluster-B consists of media server B, media adapter B, 3x2 NeTD, FSMangerB and a SAGE bridge. Each display machine covers single tile of NeTD. The OpenFlow network consists of an OpenFlow controller and OpenFlow switches. All the machines are networked by using 1Gbps connections.

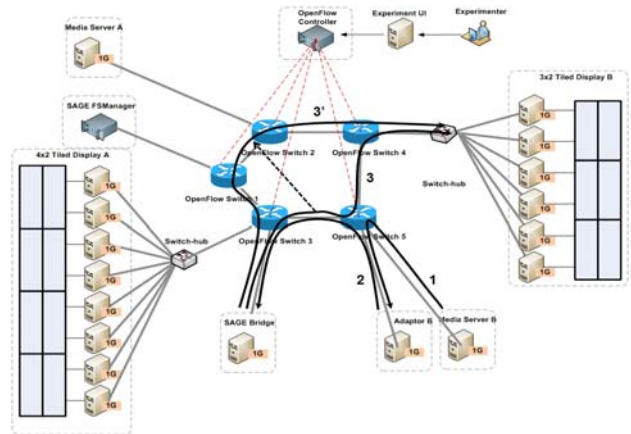


Fig. 5. Testbed for SAGE Visualcasting over OpenFlow.

B. Scenarios and Experiments

To check whether we can identify data traffic and control traffic of SAGE Visualcasting, we perform following experiment shown in Figure. 5. Real-time video stream captured at a

resolution of 352x576 from the media server B is sent out at a frame rate of 21 fps. The video is encoded in MPEG4 format. The sending rate for this video stream is around 1.3Mbps in the media server B, 78Mbps in the media adaptor B and 78Mbps in the SAGE bridge.

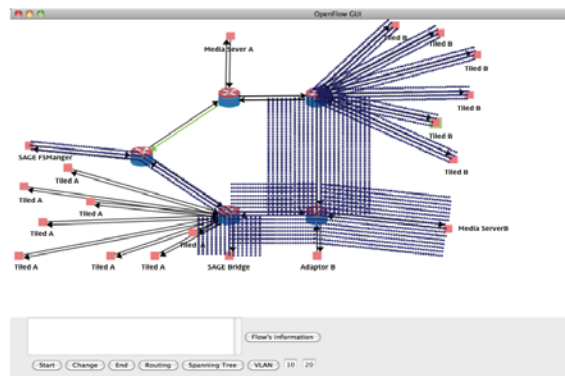


Fig. 6. Flow identification using OpenFlow UI.

1) *Identify flows and monitor computing/networking resources:* We can visualize all traffic in SAGE Visualcasting by using OpenFlow UI, as shown in Fig. 6. The red box is end hosts such as media server, media adapter, SAGE bridge, SAGE FSManager and NeTDs. Black line is the link connected by physical cables. A set of blue circles is traffic generated by SAGE Visualcasting.

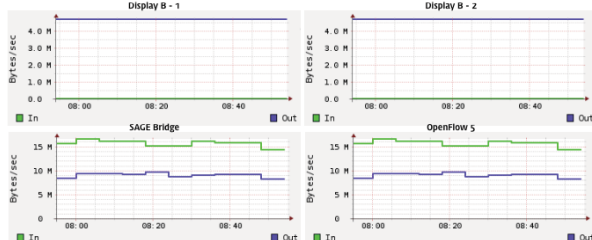


Fig. 7. Status of resource monitoring using Ganglia.

Fig. 7 shows the network I/O at the SAGE bridge, OpenFlow 5, Display B-1 and display B-2. The SAGE bridge sends around 10MByte/sec (80Mbps). Display B-1 and display B-2 each receive 5MByte/sec (40Mbps). OpenFlow 5 also sends around 10MByte/sec (80Mbps). We also measure bandwidth at SAGE UI. The sending rate is around 78Mbps and 20.5 fps at the SAGE bridge and is around 78Mbps at the SAGE NeTD.

2) *Interactive flow-based enhancement using UIs:* To validate our flow-based enhancements, we perform a similar experiment. We create a background task by using UDP iperf to generate a continuous traffic load between the OpenFlow switch 3 and 5. We increase the background traffic, and check the quality of video between SAGE bridge and SAGE display with the SAGE UI. Without changing of flow path, the frame rate decreases around 380Mbps background traffic. When background traffic goes over 550Mbps, SAGE bridge did not send video streams to the SAGE NeTD, as shown in

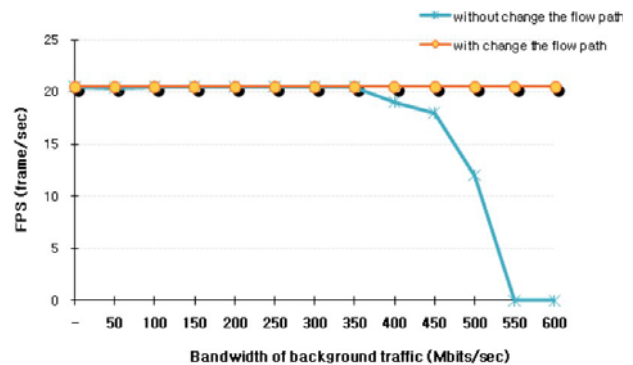


Fig. 8. Networking performance between SAGE bridge and SAGE NeTD with(out) changing the flow path.

Figure. 8. If we change the flow path from 3 to 3' at 350Mbps background traffic, the frame rate is maintained around 21 fps.

V. CONCLUSIONS

In this paper, we present flow-based enhancements by taking advantage of OpenFlow programmable networking. We provide knowledge on the flow status of network and computing node of SAGE Visualcasting. Based on these, we present alternating the paths of flows in trouble.

ACKNOWLEDGMENTS

This work has been supported by the project (2009-F-050-01), "Development of the core technology and virtualized programmable platform for Future Internet" sponsored by MKE and KCC.

REFERENCES

- [1] A. T. Campbell, H. G. D. Meer, M. E. Kounavis, K. Miki, J. B. Vicente, and D. Villela, "A survey of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 12, pp. 7–23, Apr. 1999.
- [2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, Mar. 2008.
- [3] OpenFlow Consortium. OpenFlow switch specification. <http://www.openflowswitch.org>.
- [4] K.-K. Yap, T.-Y. Huang, M. Kobayashi, M. Chan, R. Sherwood, G. Parulkar, and N. McKeown, "Lossless handover with n-casting between WiFi-WiMAX on OpenRoads," in *Proc. MobiCom Demo*, Sep. 2009.
- [5] N. Handigol, S. Seetharaman, M. Flajslik, N. McKeown, and R. Johari, "Plug-n-Serve: Load-balancing web traffic using openflow," in *Proc. ACM SIGCOMM Demo*, Mar. 2009.
- [6] B. Jeong, L. Renambot, R. Jagodic, R. Singh, J. Aguilera, A. Johnson, and J. Leigh, "High-performance dynamic graphics streaming for scalable adaptive graphics environment," in *Proc. of ACM/IEEE SuperComputing*, Nov. 2006.
- [7] L. Renambot, B. Jeong, H. Hur, A. Johnson, and J. Leigh, "Enabling high resolution collaborative visualization in display rich virtual organizations," *Elsevier Future Generation Computer Systems*, vol. 25, no. 2, pp. 161–168, Feb. 2009.
- [8] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: Towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, Jul. 2008.
- [9] ENVI. <http://www.openflowswitch.org/wp/guil/>.
- [10] M. L. Massie, B. N. Chun, D. E. Culler, M. L. Massie, B. N. Chun, and D. E. Culler, "The Ganglia distributed monitoring system: design, implementation, and experience," *Elsevier Parallel Computing*, vol. 30, no. 7, pp. 817–840, Jul. 2004.