Designing Experiments using GENI Cluster B

Ihsan Ayyub Qazi
GENI Project Office
BBN Technologies
Cambridge, MA 02138 USA
iqazi@bbn.com

Abstract

In this paper, we analyze Cluster B aggregates' technologies and design an experiment using them.

1. Introduction

Global Environment for Network Innovations (GENI) is a novel suite of infrastructure now being designed to support experimental research in network science and engineering. GENI is envisioned to support at-scale experimentation on shared, heterogeneous, highly instrumented infrastructure. In GENI, all components (computers, storage clusters, switches, and sensor networks, etc.,) will be deeply programmable and virtualizable. Deep programmability will allow researchers to program networked systems at multiple levels e.g., from implementing new modulation schemes to how CDNs are designed. Virtualization would provide for shared, cost-effective usage of GENI resources among researchers. The heterogenous infrastructures, part of GENI, will be federated and the ability to federate will allow new technologies to get incorporated in GENI as they become available [3].

Twenty-nine academic-industrial teams have been building prototypes since fall 2008 and integrating them with national backbones provided by Internet2 and National Lambda Rail (NLR). Most of these teams demonstrated early prototypes at GENI Engineering Conferences (GEC)-4 and GEC-5. These teams have been placed into five clusters (A-E). Each cluster has at least one project that is developing a control framework and a number of projects that provide different kinds of aggregates (e.g., backbone networks, regional optical networks, programmable WiMax base stations, etc.,). The five clusters are as follows (for details, see [4]):

- Cluster A – TIED control framework: This cluster is on the DETERlab testbed at USC/ISI and focusses on federation, trust, and security issues
- Cluster B – PlanetLab control framework: This cluster is based on the PlanetLab implementation at Princeton, emphasizing experimentation with distributed virtual machines over the Internet. Aggregates in this cluster include PlanetLab, Enterprise GENI, MAX, SPP, and GpENI
- Cluster C – ProtoGENI control framework: a cluster based on the Emulab implementation at the University of Utah, emphasizing network control and measurement. Aggregates in this cluster include ProtoGENI, CMULab, Programmable Edge Node, and Million Node GENI
- Cluster D – ORCA control framework: from Duke University and RENCI, a cluster emphasizing development of resources allocation strategies and integration of sensor networks ORCA/BEN, DOME, ViSE, and KanseiSensorNet
- Cluster E – ORBIT control framework: from Rutgers University, emphasizing wireless networks. The aggregates include the ORBIT testbed and WiMAX base station nodes

In this paper, we analyze cluster B aggregates in detail and come up with possible experiments using them [1].

2. Cluster B Aggregates and Technologies

In this section, we will analyze the technologies being used by each of the Cluster B aggregates. This analysis would provide insights into the strengths and limitations of these technologies in enabling networking experiments.

2.1 PlanetLab

The PlanetLab aggregate provides virtual machines on nodes distributed throughout the world. These nodes are able to virtualize a subset of OS resources and functions such as the CPU, Memory and Storage. While they provide some degree of network stack *isolation* they do not virtualize the network stack (i.e., they do not contextualize the variables in the network stack for each virtual node). As a result, different virtual nodes share a common kernel forwarding table and, thus, they cannot be used directly to allow each user to define a custom network topology or forwarding mechanisms. Virtual machines inside a node share a common IP address and the service provided by these nodes is best-effort in nature.

In isolation, this aggregate can be used for doing application layer research e.g., distributed file systems, Internet path measurements, CDNs, p2p protocols, etc., and for doing transport protocol performance measurements.

This aggregate contains more than 1000 nodes at over 480 sites around the world. This provides researchers with a rich blend of Internet paths. See [9] for more details on this.

2.1.1 Technology

The PlanetLab aggregate provides virtual nodes using the Linux VServer technology. VServer is a light-weight, container-based operating system which provides isolation of filesystem and network stack without having to run a (potentially heavy-weight) instance of a virtual machine for each experiment. The motivation for using VServer was to accommodate larger number of users than are possible with full-blown virtual machines.

2.2 VINI

Unlike PlanetLab, VINI provides network stack virtualization[2]. Therefore, virtual interfaces can be associated with each slice,

[1] It is important to note that aggregates in all clusters are expected to be integrated in the future and some are already being integrated

[2] Note that VINI is not an aggregate in Cluster B. However, nodes with VINI software have been deployed in some of the Cluster B

which can be assigned different IP addresses and VLAN ids. Slices can configure their own forwarding table entries, congestion control parameters, etc.,. In addition, VINI also provides L2 virtual links using Ethernet GRE tunnels. The virtual nodes and virtual links are connected by a bridge (similar to a Linux software bridge). Some possible experiments include, evaluating new routing protocols, deploying router-assisted congestion control protocols, etc.,. However, for router assisted congestion control protocols, the processing has to be done at the user-level because the current implementation doesn't support non-IP packet processing inside the kernel. This is likely to result in degraded performance and would limit the number of simultaneous experiments.

2.2.1 Technology

VINI uses the Trellis kernel. Trellis combines two container-based virtualization technologies VServer and NetNS. While VServer virtualizes CPU, memory, and storage, NetNS virtualizes the network stack[3].

2.3 Enterprise GENI

This aggregate provides programmable and virtualizable *OpenFlow* switches [8]. The deployments are now taking place across different University campuses. It is likely that these switches would also get deployed in regional and backbone networks.

An OpenFlow switch consists of a *flow table*, which performs packet lookup and forwarding, and a secure channel to an external controller (possibly located anywhere on the Internet) using the OpenFlow protocol. The flow table contains a set of flow entries (which are a set of header values to match the packet against), activity counters, and a set of zero or more actions to apply to matching packets. A controller is responsible for managing flow table entries. As of now, the possible header values to match against include ingress port, Ethernet source/destinations addresses, Ethernet type, VLAN id, VLAN priority, IP source/destination addresses, IP protocol, Transport source/destination ports, ICMP Type and ICMP Code. The counters are maintained per-table, per-flow, and per-port. The actions that are required to be supported by all OpenFlow switches include ALL (broadcast to all ports except the incoming port), CONTROLLER (send to controller), LOCAL (send to local networking stack), TABLE (perform actions in flow table), IN_PORT (send the packets out the input port)[4][5] [8].

OpenFlow switch can forward non-IP packets based on Ethernet addresses or VLAN ids but it doesn't allow *processing* of non-IP packets in a scalable way. Non-IP packets could be sent to the controller for processing, which might be useful for testing the functionality of a new protocol but it is unlikely to scale in a large network. To facilitate processing of non-IP packets, an OpenFlow switch can route these packets to programmable switches/routers such as NetFPGA-based programmable routers [6]. In this case, users would still benefit from the traffic isolation provided by OpenFlow switches [7].

This aggregate (5 Switches HP+NEC, 25 Wireless APs,) is currently located at Stanford University, CA. Other campuses are likely to join as GENI aggregates.

2.3.1 Technology

Uses OpenFlow switches which leverage the existing flow tables in switches and routers.

2.4 Mid-Atlantic Crossroads (MAX)

This aggregate provides a regional, multi-wavelength optical network DRAGON test-bed, a physical DWDM and Layer 2 network deployment of open-source GMPLS control plane software development and the deployment of that control plane software over other networks. The nodes use the PlanetLab node software which they have extended to include per-slice IP addresses and VLAN ids. Using this, a virtual circuit, with reserved bandwidth, may be created.

As of now, there are four PlanetLab nodes in this testbed that may be connected via virtual circuits with an arbitrary topology.

2.4.1 Technology

Extends the DRAGON's open-source GMPLS-based control plane implementation to include edge compute resources and support network virtualization (viewing DRAGON as an aggregate/component manager)

2.5 Supercharged PlanetLab Platform (SPP)

A SPP node is a high performance PlanetLab node which combines the performance of Network Processors (NP) with the programmability of conventional servers. It extends the PlanetLab node by providing a fastpath-slowpath application structure, where the fastpath uses NPs to process most data packets at high speed while slow path is used for handling control and exception packets. However, it requires application developers to structure their code to take advantage of the NP resources [11].

SPP nodes will be located at Internet2 backbone sites. As of now, 3 SPP nodes are expected to be deployed.

2.5.1 Technology

SPP nodes provide two kinds of processing engines; General-Purpose Processing Engines (GPE) which are conventional server blades and network processor blades. These are connected by a Chassis Ethernet switch. Slices are created on GPEs and optionally, they can instantiate a fathpath on the network processor blade, associate additional resources, and configure these resources as required by the application. Fast path resources include packet filters, packet buffers, queues, memory, and bandwidth. Slices can also configure different logical interfaces on the fast path [11].

2.6 Great Plains Environment for Network Innovation (GpENI)

The GpENI aggregate provides PlanetLab nodes, VINI nodes, and Optical switches (which use the DRAGON software). The GpENI network has equipment co-located at an Internet2 POP in Kansas City, MO.

2.6.1 Technology

GpENI uses the technologies described earlier.

3. Experiment: Evaluation of Router-Assisted Congestion Control Protocols

aggregates such as GpENI. Hence, we mention its strengths and limitations here.

[3]This is functionally equivalent to OpenVZ, which is another container based virtualization software

[4]Optionally, the switches may support processing of packets using the traditional forwarding path. Also, actions that require modifications in the packet fields are optional but it is suggested that at least VLAN modification actions be supported.

[5]It is worth noting that NEC switches support all modification actions. While modification of the destination address is done in hardware, all other are done in software. This may result in performance degradation

[6]a typical NetFPGA-based programmable router supports only 4 ports

Many congestion control protocols (e.g., XCP [5], RCP [2], BMCC [10]) require explicit feedback from the network for improved performance. The evaluation of such protocols is typically carried out using simulations or in small testbeds that provide homogenous technologies and offer limited flexility [2, 5, 10, 12]. Using GENI, we'll design an experiment that would allow us to evaluate these protocols across heterogenous technologies, under a variety of scenarios, and at an unprecedented scale.

We will use the Binary Marking Congestion Control (BMCC) protocol for experimentation. BMCC is compatible with the IPv4 header. It requires changes at the end-hosts and inside routers. With BMCC, routers periodically compute the load on each of their output links. Using the load information and the value in the IPID field, they set the ECN bits in the packets they forward.

3.1 Changes needed at the end-hosts

BMCC end-host functionality can be implemented as a loadable kernel module [7]. New congestion control protocols are often implemented as kernel modules to avoid the need to patch the kernel and the subsequent kernel recompilation. To expedite the end-host implementation, one can adapt existing implementations of protocols that employ a similar structure [6].

3.2 Changes needed inside the routers

The BMCC router functionality is implemented on NetFPGA cards for achieving high performance. NetFPGAs are programmed to periodically compute load on each of its links. The data plane operations performed by the NetFPGA-based router includes updating a byte counter, comparing the link load with the value in the IPID field, and based on this, setting of the ECN bits in the IP header. The control plane operations take place at a much larger timescale. Link utilization is computed every 200 ms and the average queue length is measured every 10 ms. Load (computed every 200 ms) is computed as a weighted average of these two values.

One can adapt existing implementations of router-assisted congestion control protocols on NetFGPAs for expediting this process [12].

3.3 Performance Evaluation on High Speed Networks

Our first scenario evaluates the performance of BMCC on a high speed network. There are two clients and one server. The server is connected to an OpenFlow switch which in turn connects to two NetFPGA cards, which implement the BMCC router functionality. One NetFPGA card connects to Client 2 directly whereas the other routes the packets to the Internet2/NLR POP. Internet2/NLR has a connection to the DRAGON testbed, which has a path to Client 1 (see Figure 1).

The client downloads a long file via Path 1 as shown in Figure 1. During this transaction, a number of measurements of interest are collected e.g., total file transfer time, time to attain full bottleneck utilization, packet loss rate, etc.,. In the next test, we start two flows with an inter-arrival time of t secs. Flow 1 starts at Client 1 following Path 1. After t secs, a new flow is started at Client 2 which follows Path 2. We measure the time it takes for flow 2 to achieve its fair share[8]. These tests can be repeated to achieve statistically significant results.

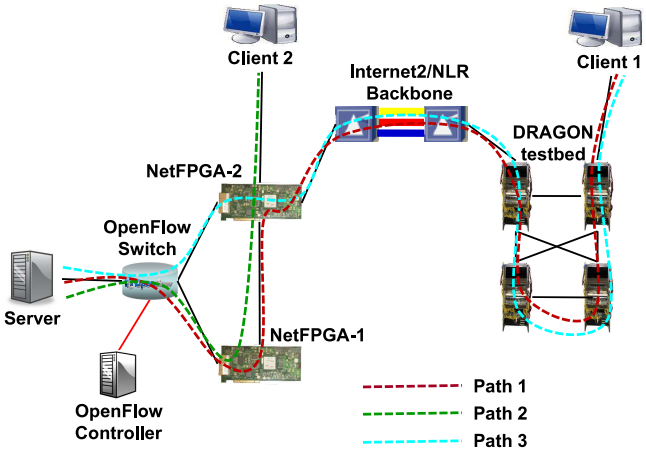

Figure 1: Experiment Scenarios

3.4 Performance under Topology Changes

BMCC uses bottleneck load information to adjust its rate. When the flow path changes, the bottleneck may also change, requiring the source to adjust to the new rate [9]. In this scenario, we are interested in understanding how quickly BMCC adapts to the new bottleneck. We start two long-lived BMCC flows as in the previous case. Using the OpenFlow switch controller, we change the Client 1 flow entry to follow Path 3, resulting in a change in the bottleneck.

3.5 Performance Evaluation on a Path with a combination of Wired and Wireless links (last hop)

In this experiment, we evaluate the performance of BMCC over a network where the last hop is a wireless link. To change link properties, we can use the CMU Wireless Emulator testbed (based on NetFPGAs).

4. Acknowledgements

I would like to thank Aaron Falk, Chip Elliot, Heidi Dempsey, Christopher Small, Harry Mussman, Henry Yeh, Chunhui Zhang, and Nidhi Tare for useful discussions and their feedback.

5. Conclusion

GENI provides deep programmability and a unique blend of technologies that enable realistic experimentation of new protocols and architectures. We analyzed the technologies being used by GENI Cluster B aggregates. We then designed an experiment, using GENI, that required changes inside the routers. The experiment used heterogenous technologies and could test the performance of the protocool under a variety of scenarios.

6. References

[1] BHATIA, S., MOTIWALA, M., MUHLBAUER, W., MUNDADA, Y., VALANCIUS, V., BAVIER, A., FEAMSTER, N., PETERSON, L.,

[7] For experimentation with protocols that require changes in the IP header format or require a new header, it is useful to implement this functionality on top of IP (perhaps below TCP) so that traditional routers allow such packets to pass through and only those routers, which are aware of the header, can process them

[8] Note that it is being assumed that NetFPGA-1 – NetFPGA-2 link is the bottleneck

[9] This may happen due to client/server mobility, load balancing, or when routers/links fail resulting in a change of path.

AND REXFORD, J. Trellis: a platform for building flexible, fast virtual networks on commodity hardware. In *3rd International Workshop on Real Overlays and Distributed Systems (ROADS)* (New York, NY, USA, 2008), ACM, pp. 1–6.

[2] DUKKIPATI, N., KOBAYASHI, M., ZHANG-SHEN, R., AND MCKEOWN, N. Procesor sharing flows in the internet. In *IWQoS* (Jun 2005).

[3] ELLIOTT, C., AND FALK, A. An update on the geni project. *SIGCOMM Comput. Commun. Rev. 39*, 3 (2009), 28–34.

[4] Global Environment for Network Innovations (GENI). http://www.geni.net/.

[5] KATABI, D., HANDLEY, M., AND ROHRS, C. Internet congestion control for high bandwidth-delay product networks. In *ACM SIGCOMM* (Aug 2002).

[6] LI, X., AND YOUSEFI'ZADEH, H. A linux implementation and experimetal study of the variable-structure congestion control protocol. In *IEEE MILCOM* (2007).

[7] MCKEOWN, N., ANDERSON, T., BALAKRISHNAN, H., PARULKAR, G., PETERSON, L., REXFORD, J., SHENKER, S., AND TURNER, J. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev. 38*, 2 (2008), 69–74.

[8] The OpenFlow Switch Specification. http://OpenFlowSwtich.org/.

[9] Planetlab. http://www.planet-lab.org/.

[10] QAZI, I. A., ANDREW, L. L. H., AND ZNATI, T. Congestion control using efficient exlicit feedback. In *IEEE INFOCOM* (Apr 2009).

[11] Internet Scale Overlay Hosting. http://wiki.arl.wustl.edu/index.php/Internet_Scale_Overlay_Hosting.

[12] TAI, C. H., ZHU, J., AND DUKKIPATI, N. Making large scale deployment of RCP practical for real networks. In *IEEE INFOCOM Mini-Symposium* (2008).

[13] Virtual Network Infrastructure (VINI). http://www.vini-veritas.net/.