Programming Assignments for Graduate Students using GENI

Please direct comments regarding this document to `fahmy@cs.purdue.edu`.

1 Overview

This document summarizes ideas for possible programming assignments (exercises) for use by instructors of a **graduate** networking class (*e.g.*, CS 536 in the Computer Science Department at Purdue University).

All assignments require executing experiments on the GENI infrastructure, and understanding the performance that various GENI nodes and links provide. One of the planned assignments includes non-IP based networking.

The planned courseware will have two components: (1) Student assignments and (2) Instructor material that includes solutions to the assignments.

2 Prerequisites

The assignments assume knowledge of the C programming language, basic operating systems concepts including multi-threading, and basic networking concepts. The graduate networking course that will use the planned assignments covers the following material:

- Network services and applications: DNS, HTTP, SMTP, peer-to-peer systems
- Network transport architectures, TCP, UDP, TCP congestion control
- Routing and forwarding, intra-domain and inter-domain routing algorithms
- Link layers and local area networks, especially Ethernet and WiFi
- As time permits:
  - Multimedia communications and quality of service
  - Network measurement, inference, and management
  - Network experimentation and performance analysis
  - Network security
  - Protocol verification

The assignments can also be used in a more advanced graduate networking course.

3 Terminology

The following definitions give the normal meaning of terms used in this documentation. In places where specific usage differs from this section, the differences will be explained.

Slice A GENI slice created via a Slice Authority.

Sliver A GENI sliver create via a Component Manager or Slice Authority.

Interface A network port (actual or emulated) on a GENI node which has been assigned an IP address at sliver creation.

Manifest An XML document provided by the Component Manager or Aggregate Manager at the time of sliver creation, which describes the resources allocated to a sliver.

Node, GENI Node A physical or virtual machine allocated to a GENI Sliver.

4 Objectives

The assignments developed in this project aim to teach the basics of network programming, client/server architectures, how a reliable transport protocol runs on top of an unreliable delivery mechanism, how routing protocols operate, and how to differentiate among flows.

A key goal of using GENI (as opposed to other platforms such as PlanetLab or DETER) is exploring the greater variety available in the GENI infrastructure, and leveraging new GENI projects. Students will use a number of tools from the experimenter portal (`http://groups.geni.net/geni/wiki/ExperimenterPortal`), including tools that simplify resource allocation like Gush, Omni, Raven, and Flack. These simplify the allocation of GENI nodes and slices, and the parsing of manifests.

5 Planned Assignments

In this section, we describe a number of potential ideas for lab assignments that use GENI. The ideas can be classified into three broad categories:

1. Partial implementation of resource allocation mechanisms,

2. Partial implementation of reliability/congestion control mechanisms, and
3. Partial implementation of packet forwarding, filtering, and quality of service mechanisms.

These correspond to application-layer/middleware, transport-layer, and network-layer functions in the current TCP/IP protocol stack.

5.1 Measuring Application Performance and Resource Allocation Mechanism Implementation

5.1.1 Goals

This project introduces students to GENI, virtual machines, and alternative resource allocation mechanisms that increase application performance.

5.1.2 Tools Used

This assignment can leverage the Virtual Desktop Cloud project from the Ohio State University.

5.1.3 Description

The Virtual Desktop Cloud is a system with a variety of virtual machines and virtual machine containers (called VMNet). The system runs desktop machines in this cloud service, and explores ways to increase performance and minimize overhead. The system includes a desktop client benchmark suite that is capable of automating normal desktop activities by mimicking user behavior, and then timing the response of applications and reporting on the overall performance. OnTimeMeasure can be used to monitor the network utilization and performance.

The students will use this infrastructure to compare strategies for allocating CPU, memory, and bandwidth to virtual machines. They will study the ways in which virtual machine resource allocation affects user-perceived performance.

Designing and implementing a resource allocator will teach the students important concepts. The Virtual Desktop Cloud project provides all the tools to experiment with the effect of network performance on the usability of virtual desktops. The students can allocate cloud machines, allocate thin clients, then place a virtual network between the two, and have VDBench report on the usability characteristics of the thin client.

The system is currently independent of the remote desktop protocol. A project that couples specific network characteristics (long delay, fading, etc.) with specific protocol mitigations (aggressive pipelining, forward error correction (FEC), etc.), and evaluates the thin client performance can be a nice extension, possibly for a more advanced course or for extra credit.

5.2 Measuring Infrastructure Performance and Congestion Control/Reliability Implementation

5.2.1 Goals

This project aims at introducing students to the concepts of reliability and congestion control, and increasing their understanding of the performance of different types of GENI infrastructure. The students can implement parts of the functions that are typically included at the transport layer in today's TCP/IP stack. The students will compare how the goodput depends on GENI infrastructure performance.

5.2.2 Tools Used

This project can leverage some experimenter command line tools such as Gush (GENI User Shell), and can explore diverse GENI infrastructure such as the wireless testbed ORBIT and ProtoGENI nodes and links.

5.2.3 Description

The project will require students to allocate a small number of GENI nodes and links, and experiment with the impact of link delay and loss characteristics, and link bandwidth, on the performance (*e.g.*, goodput) of reliability and congestion control. The students can implement part of a transport protocol (specifically partial congestion control and reliability functionality) and study the impact of parameters such as window sizes, selective acknowledgments, lossy (wireless) links and variable traffic load.

The students can also create a small topology with hosts on both sides of a *bottleneck* link, and use a web or other traffic generator to saturate the link in one direction. They can then vary the link bandwidth and delay (and choose different link loss characteristics), and test their reliability and congestion control algorithm. They can also experiment with the size of the drop-tail queue on the bottleneck link (and the use of Random Early Detection), and observe the effects on goodput.

A possible extra credit extension can have the students implement a packet-pair or packet-train available bandwidth estimator tool using UDP packets. They can use either a tool from the literature (based on its published description) or propose their own. The students can evaluate their implementation's accuracy on network topologies of varying bandwidth, delay, and loss. They can also evaluate the impact of both constant bit rate and TCP cross traffic (*e.g.,* from a web traffic generator), including the case of saturated links (that is, no available bandwidth).

5.3 Using OpenFlow for Quality of Service/Filtering Mechanism Implementation

5.3.1 Goals

This project can introduce students to OpenFlow, and have them implement and evaluate different packet filtering and quality of service (QoS) operations. For example, they can design and implement network-layer QoS mechanisms such as Weighted Fair Queuing (WFQ) scheduling or Random Early Detection (RED) with multiple queues or thresholds. Label or tag switching can also be used to mark, forward, and switch flows.

5.3.2 Tools Used

This project can leverage one of the GENI resource allocation tools to simplify requesting resources. The project will also leverage OpenFlow-capable switches and networks such as the GPO Lab OpenFlow network.

5.3.3 Description

In this project, the students can explore forwarding functionality that OpenFlow-capable switches allow and quantify their performance on multiple aggregates.

For example, the students can develop an OpenFlow controller that implements a stateful firewall capable of processing established connections without contacting the controller.

Given a network such as:

```
Server ---                  --- Remote Server
           \               /
HostA  ---  +--- Firewall +-- Internet
           /
HostB  ---
```

The students can develop functionality to implement policies such as:

- Internet hosts may connect to TCP port 80 on “Server,” but no other port.
- “Server” may connect to “Remote Server” on port 8080, but no other hosts on the right side of the firewall.

- "Remote Server" may connect to TCP ports 80 or 8080 on "Server", but no other ports.
- "HostA" and "HostB" may connect to any host on the right side of the firewall on any port, but no host on the right side of the firewall may connect to "HostA" or "HostB."

Once a connection has been established, TCP segments related to the connection should pass through the OpenFlow switch without involving the controller. The controller should examine only packets from the three-way handshake, closing handshake, or reset (RST) packets.

The project can be extended to ask students to implement different tag/label switching mechanisms, and differentiated Quality of Service (QoS) functionality. This would be especially appropriate for more advanced graduate networking courses. For scheduling, weighted fair queuing, or a simple variant like Deficit Round Robin (DRR) can be implemented and evaluated. For buffer allocation/drop mechanisms, multiple queues implementing RED can be used with different thresholds, similar to RED with IN/OUT (RIO) for differentiated services.