Jonathan Campbell

Networks Project

CS 4390.001

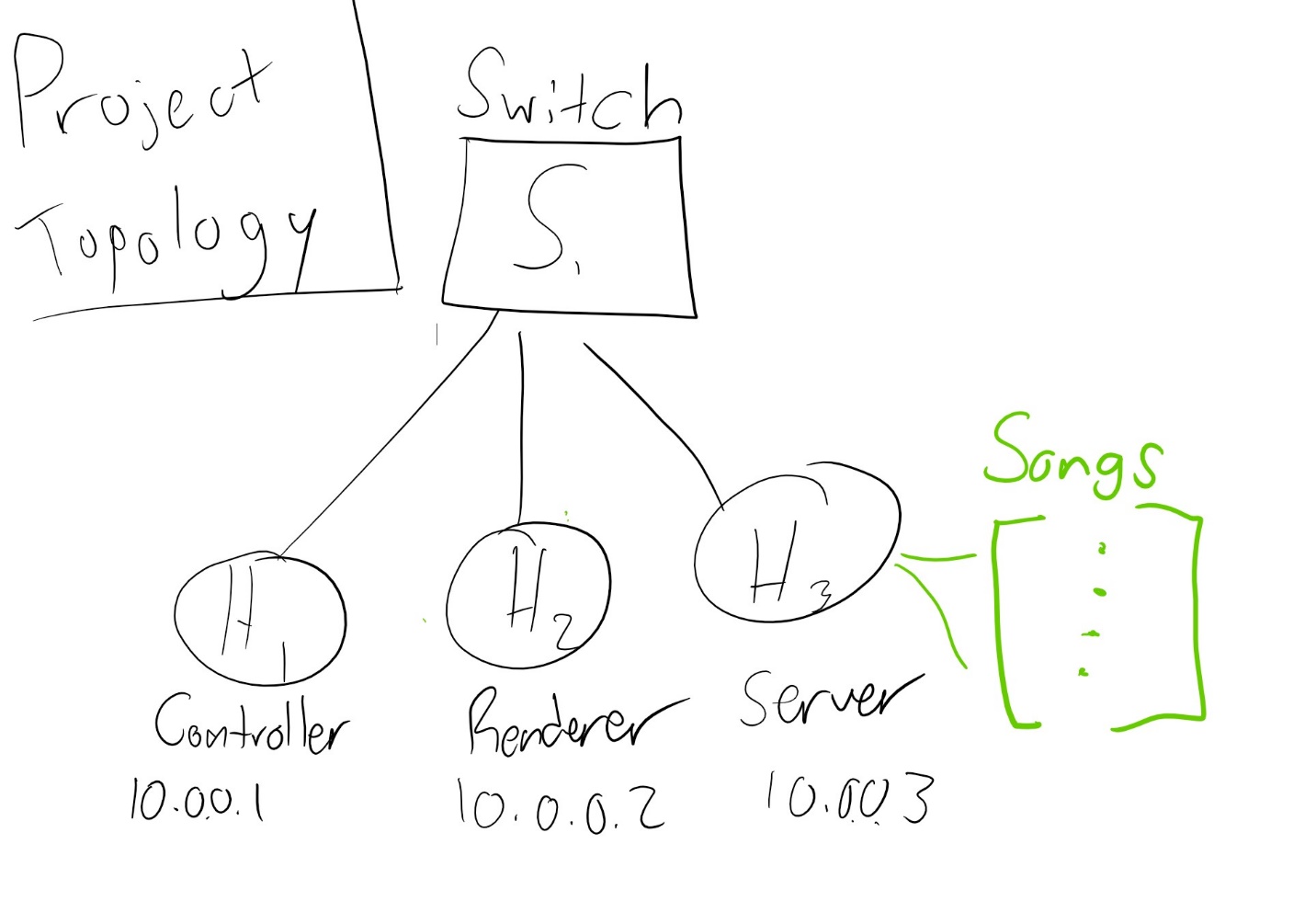
At the end of the semester, EACH student must submit a ZIP file containing

1. A project report (in Word format) covering

a) a description of the program

This program allows users to test the connection between hosts and a server in order to stream music between them. We used python code interfaced to Mininet to implement the solution to the requirements. The user has the ability to retrieve files in mp3 format across a server and is able to play and stop the files.

We will be using Python code, and Mininet to create an application that had the ability to stream data (text files/music files) to a client (host) over a test network.



***Figure 1***: Test Network Diagram

Figure 1 shows the layout diagram of the test network. It consists of three hosts and one switch. The switch is used to control flow and requests between hosts and servers. Each host represents a controller(H1), Renderer(H2), and Server(H3). The Server(H3) will store access to the files that the Controller(H1) can request. The Renderer(H2) will serve as the GUI between the user and the Server. Once the application begins, a request to the Server(H3) to show available accessible files. The list will be sent back to the Controller(H1) and will be displayed on the Renderer(H2).

To test, Python files for the Controller, Renderer(GUI), and Server will be written. Using VirtualBox software, we will use Mininet commands to create and topology and its links. Once these are initialized, the user will be able to select which file to access.

b) the challenges that you and your team had and how did you or the team overcome them,

A few challenges we faced in the project include:

* How to interface our python code in a fashion that Mininet would be able to understand the process.
  + This is important since this is the way our application would be run. Python can be used to run. Mininet has its own commands, and so does Python. In the project, we were able to use the Mininet CLI to run Python programs that were coded.
* How to design a protocol that allows the functionality required.
  + Initially, we had little idea of how to approach this problem. This is very open ended, however after observing exactly what the application would be doing, we were able to come up with how the network should communicate. We then were able to format this communication pattern in a similar fashion to some of the RFC standards found on the internet.
* How to layout specific pieces of the test network. How to specify and replicate desired test network.
  + This was a challenge since the requirements were very open ended. We did not know exactly how to implement this, but we were able to figure out a network that fit the requirements by noticing that the topology in the requirements can be translated into a Mininet default topology. We called the renderer, controller, and server

c) what you have learned by doing the project:

Some of the things I learned while doing the project include:

* How command line, Mininet, and python can be used to run simulation of networks.
  + When writing a program like this, pure programming is only the tip of the iceberg of software development. Being able to interface across many layers is key when designing a fully functional application.
* Python libraries and uses
  + Python has libraries that can be used as a music player as well as being interfaced with networking topologies using Mininet which mad it easier to code this networking application.
* How to run test network application
  + Mininet can be used to run code on separate terminals. This property can be exploited to observe how network topologies operate and network entities interact with each other.
* Protocol Design
  + When designing our protocol, we considered how the entities should work and communicate together. Once this was realized, we were able to format the communication pattern into RFC standard we found on the internet.

d) a discussion about algorithms and techniques used in the program,

Some of the techniques involved in the project was having the ability to use the CLI and the terminal created in the hosts to interface commands to run the application. We also used many tutorials and articles as aides and guides to complete the project. Algorithms used in the project mainly include establishing the topology. There are library calls that can be coded in Mininet that automatically create topologies. This was used as a guide for our project, but we had to make tweaks to it.

e) contributions of each team member,

Kevin – Protocol Specification

Shri – Code Implantation and Video

Enrique – Research and Code Implementation

Jonathan – Research and Documentation

2. A short video clip with audio narration demonstrating the network application in action. (If the video is too big you can post it on a website, e.g. youTube, and submit the link).

3. All codes that are needed to run your application.

4. A protocol specification document as mentioned above.

5. A design document describing your team's implementation of the application.

From the requirements given in the assignment, a controller, server, and renderer network were needed. From figure 1 (given above), three hosts, a server, and a switch were initialized as a topology in Mininet. In order to run Mininet, a virtual machine was used and interfaced with the Mininet command line and the virtual machine command line. Once Mininet was up and running, we used the commands in Mininet to initialize a topology comprised of the switch, a server, and two hosts (renderer and controller). For each entity in the network, a specific code was written (Server.py and Renderer.py) using python code. Once this was setup correctly, the terminals of the hosts was created using the python -xterm h1 h2 commend in the Mininet command line. Once these were created, Each Python script was run on different hosts which emulated the effect of the controller terminal (e.g. host 1) communicating with the renderer terminal ( host 2 ), in order to request and retrieve the list of files hosted on the server. In the Renderer.py script, the ability to play, pause, and stop was created for the user.

Obviously items 2. 3, 4 and 5. are the same for all team members but item 1. must be prepared by each student individually (but some sections of the report, e.g. a) and d), can be shared).

Let me or the TA know if there is any questions or comments.

Teams should get together as soon as possible to discuss the requirements and decide on how to break down the task to assign to each team member.