iOS Application to Configure and Launch a Data Center IP Fabric: User Manual

Sanjay Murthy and Lara Orlandic
Systems Engineering Interns
Juniper Networks

About the Documentation

This document will enable users to successfully download, deploy, and manipulate the iOS application to launch user-defined configurations of a virtual switch architecture. The application's installation and usage instructions are described in detail.

Application Overview

The purpose of this iOS application is to allow users to dynamically configure and launch a Data Center (DC) IP Fabric consisting of a L3 Clos Spine-Leaf architecture of Juniper Networks' virtual QFX switches. The application will enable field Systems Engineers at Juniper to showcase the versatile applications of our Data Center Switching solutions to customers. Through this Zero-Touch-Provisioning user interface, even clients who do not have much networking knowledge can automatically configure a DC IP Fabric.

With the application, users are able to:

- Design an L3 Clos architecture by selecting the desired number of spine and leaf switches
- Automatically configure the aforementioned number of spine and leaf vQFX switches on VirtualBox Linux machines
- Launch vQFXs at the click of a button
- Connect the vQXFs using an eBGP underlay
- Enable an eVPN overlay
- Configure a VTEP tunnel between leaf switches
- Receive feedback as to which virtual machines were successfully configured in real-time
- Delete the DC POD by automatically closing all machines

Downloading the Application

Dependencies

In order for the application to be run successfully, the host computer needs to contain the following software:

- Python (version 2.7 or later)
- Xcode
- Ansible
- Vagrant (version 1.7 or later)
- VirtualBox (version 5.0.10 or later)

Additionally, the user must own an Apple mobile device, such as an iPhone or iPad, on which to run the iOS application. Loading this application onto the device requires one-time use of an Apple computer, since Xcode only runs on MacOS.

Downloading Necessary Files

All of the files required to run the project can be acquired by cloning the following GitHub repository onto the user's computer: https://github.com/lara-juniper/iOS-Python-TCP-Interface

This can be done using the following command:

git clone https://github.com/lara-juniper/iOS-Python-TCP-Interface

Downloading Dependencies

1. Download and install Vagrant
    a. Download: https://www.vagrantup.com/downloads.html
2. Download and install VirtualBox
    a. Download: https://www.virtualbox.org/wiki/Downloads

   b. Note: If you are using an old version of Linux (Ubuntu 14.04, 14.10, etc.) the version installed with apt-get will most likely not work. In this case, you need to update the version manually
      i. Installation for Ubuntu :

wget https://releases.hashicorp.com/vagrant/1.8.1/vagrant_1.8.1_x86_64.deb

sudo dpkg -i vagrant_1.8.1_x86_64.deb

3. Download and Install the vQFX Box file locally
   a. Download: http://www.juniper.net/support/downloads/?p=vqfxeval#sw
   b. Note: Because vqfx10k boxes are not available on Vagrant Cloud yet, you will have to install them manually first. Use the following commands depending on the package you wish to use:
      i. Installation commands (full package):

vagrant box add juniper/vqfx10k-refull /{path to box file}/vqfx10k-re-virtualbox.box

vagrant box add juniper/vqfx10k-pfefull /{path to box file}/vqfx10k-pfe1-virtualbox.box

         1. Note: Both vqfx10k-re and vqfx10k-pfe need to be installed for full package. For an explanation of light mode versus full mode, see "Using Light Mode vs. Full Mode."
      ii. Installation commands (light package):

vagrant box add juniper/vqfx10k-re /{path to box file}/vqfx10k-re-virtualbox.box

   c. Note: Since Boxes are not yet publicly available, contact your SE administrator for access
4. Install Ansible
   a. Installation commands for MacOS:

      sudo easy_install pip

sudo pip install ansible

sudo ansible-galaxy install Juniper.junos

sudo pip install junos-eznc

   b. Installation commands for Ubuntu/Linux:

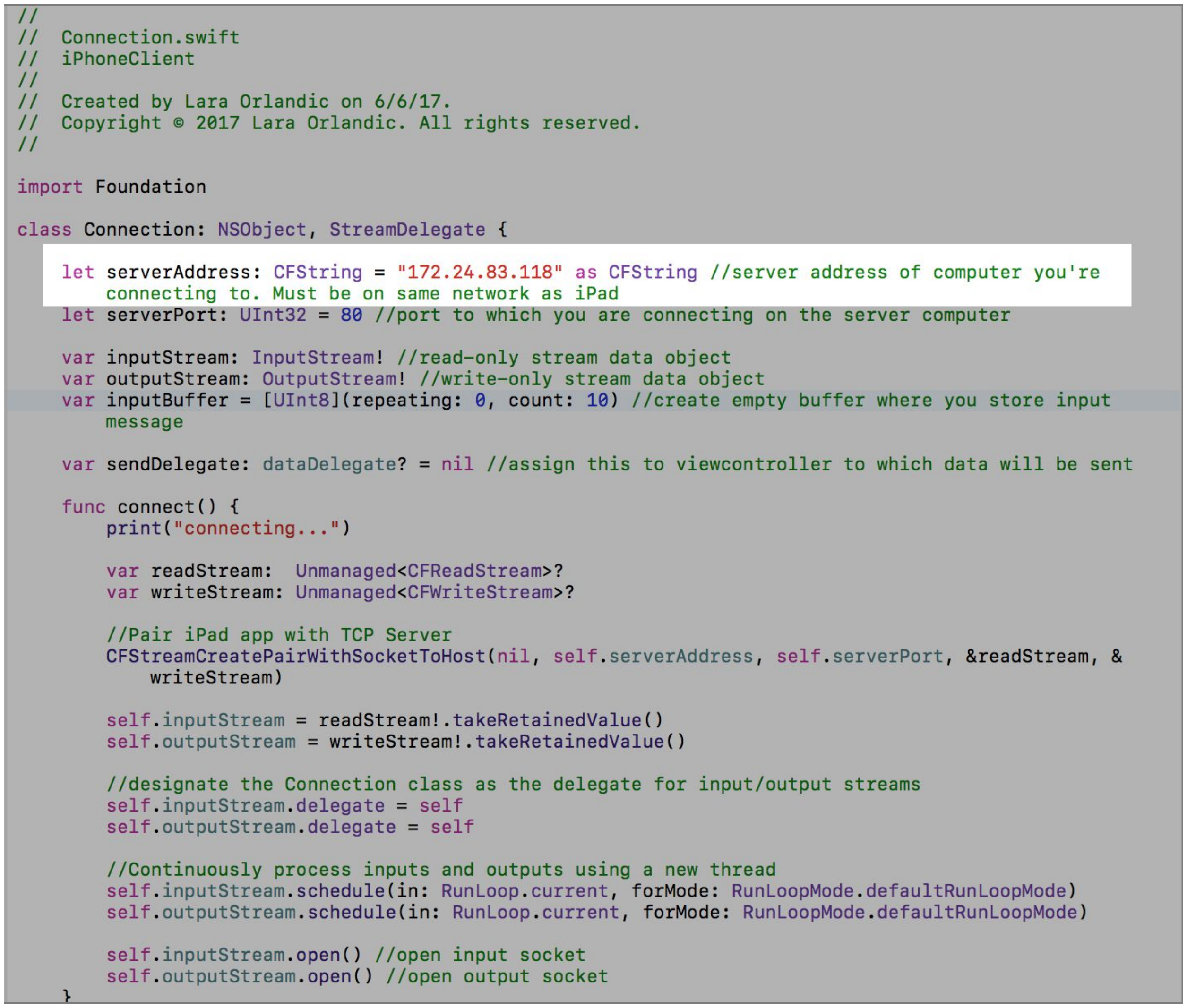
   apt-get install ansible

   ansible-galaxy install Juniper.junos

<u>One-Time Application Installation on an iOS Device</u>

The user has two platforms on which they can run the iOS application: on a physical iOS device or on an Xcode simulator. Follow these steps to load the Xcode project onto a host device, such as an iPhone or iPad, or on the simulator:

1. Open the iPhoneClient.xcodeproj file on Xcode.
2. Open the Connection.swift file on the left menu of files.
3. Locate the IP Address of your computer on the network. On a Mac, this can be done by going to System Preferences and selecting “Network.”
4. Change the serverAddress string to the IP address of your computer on the network
   a. <u>Note</u>: If you wish to use an iOS device simulator instead of a physical device, use the loopback address 127.0.0.1.

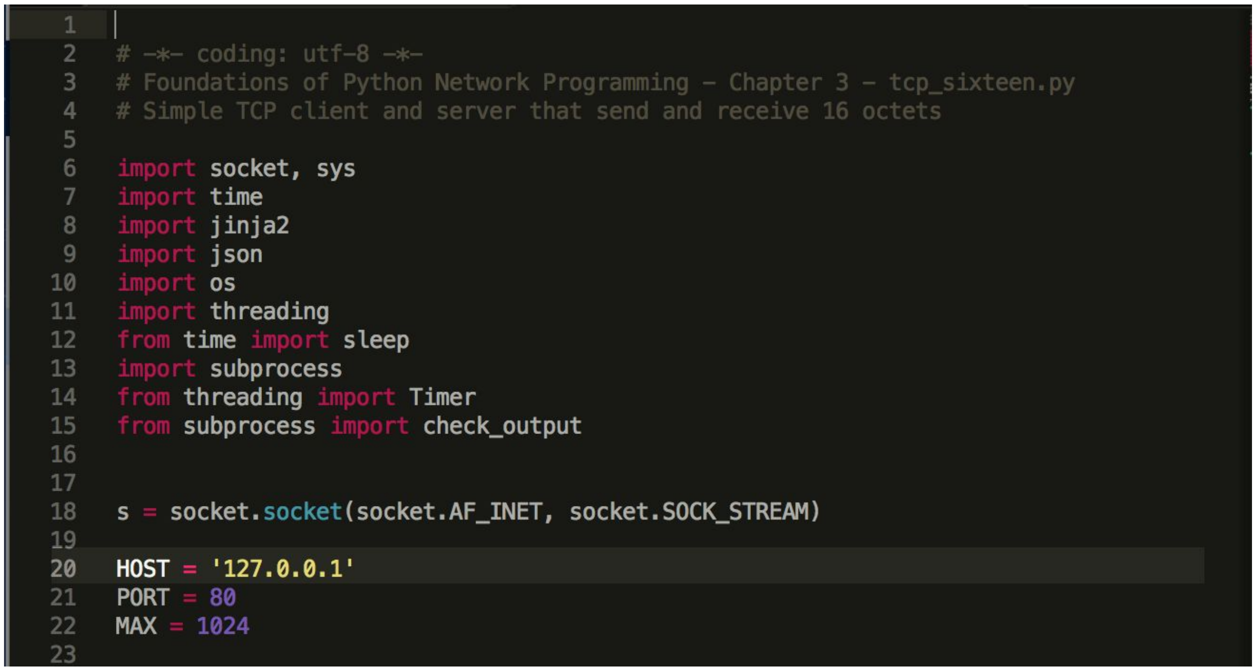


```
//
//  Connection.swift
//  iPhoneClient
//
//  Created by Lara Orlandic on 6/6/17.
//  Copyright © 2017 Lara Orlandic. All rights reserved.
//

import Foundation

class Connection: NSObject, StreamDelegate {

    let serverAddress: CFString = "172.24.83.118" as CFString //server address of computer you're
        connecting to. Must be on same network as iPad
    let serverPort: UInt32 = 80 //port to which you are connecting on the server computer

    var inputStream: InputStream! //read-only stream data object
    var outputStream: OutputStream! //write-only stream data object
    var inputBuffer = [UInt8](repeating: 0, count: 10) //create empty buffer where you store input
        message

    var sendDelegate: dataDelegate? = nil //assign this to viewcontroller to which data will be sent

    func connect() {
        print("connecting...")

        var readStream:  Unmanaged<CFReadStream>?
        var writeStream: Unmanaged<CFWriteStream>?

        //Pair iPad app with TCP Server
        CFStreamCreatePairWithSocketToHost(nil, self.serverAddress, self.serverPort, &readStream, &
            writeStream)

        self.inputStream = readStream!.takeRetainedValue()
        self.outputStream = writeStream!.takeRetainedValue()

        //designate the Connection class as the delegate for input/output streams
        self.inputStream.delegate = self
        self.outputStream.delegate = self

        //Continuously process inputs and outputs using a new thread
        self.inputStream.schedule(in: RunLoop.current, forMode: RunLoopMode.defaultRunLoopMode)
        self.outputStream.schedule(in: RunLoop.current, forMode: RunLoopMode.defaultRunLoopMode)

        self.inputStream.open() //open input socket
        self.outputStream.open() //open output socket
    }
```

Figure 1. Change the serverAddress constant to the IP address of the host computer.

5. Open the server.py file in either the light-1qfx or full-1qfx folders

6. Change the “HOST” variable to the IP address of the host computer.

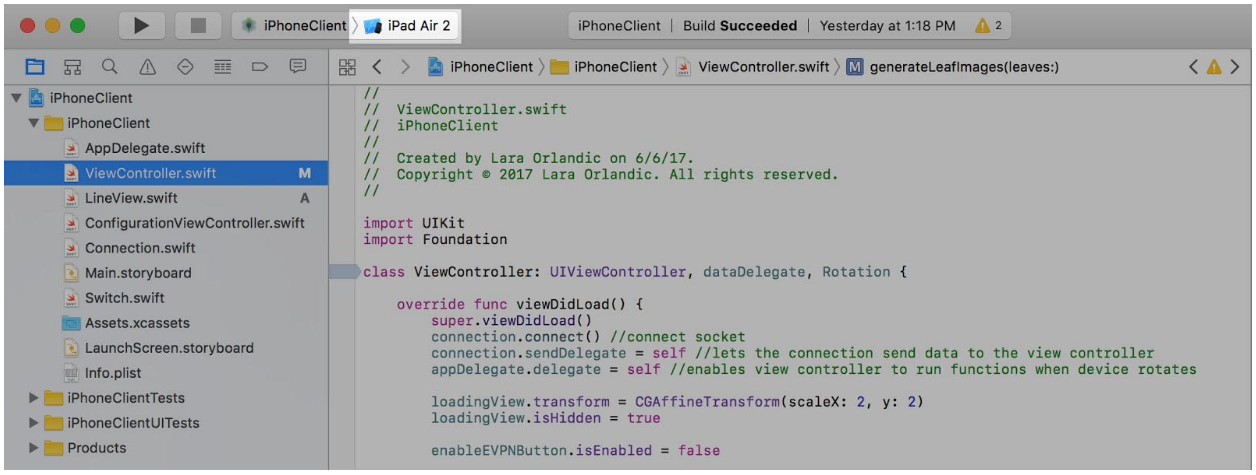


```python

# -*- coding: utf-8 -*-
# Foundations of Python Network Programming - Chapter 3 - tcp_sixteen.py
# Simple TCP client and server that send and receive 16 octets

import socket, sys
import time
import jinja2
import json
import os
import threading
from time import sleep
import subprocess
from threading import Timer
from subprocess import check_output


s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

HOST = '127.0.0.1'
PORT = 80
MAX = 1024

```

Figure 2. Change the HOST variable to the IP address of the host computer.

7. Open the iPhoneClient.xcodeproj file on Xcode.

8. Select the device listed in the top left corner next to the iPhoneClient icon. By default, it is set to a simulator. If you wish to use a simulator, select “iPad Air 2” and skip to Step 11.

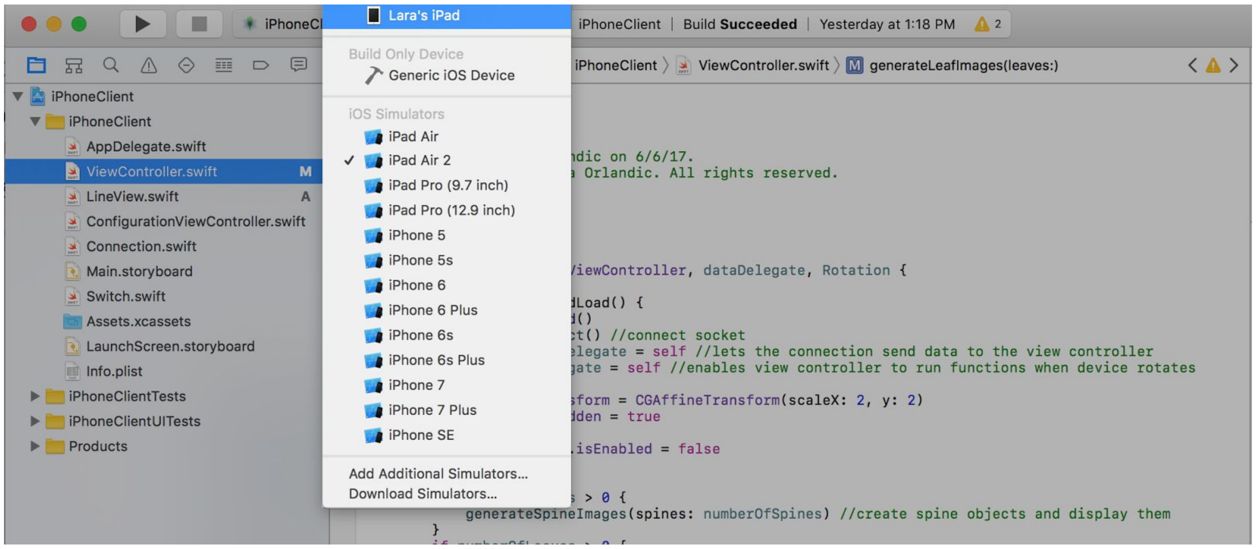


```swift
//
//  ViewController.swift
//  iPhoneClient
//
//  Created by Lara Orlandic on 6/6/17.
//  Copyright © 2017 Lara Orlandic. All rights reserved.
//

import UIKit
import Foundation

class ViewController: UIViewController, dataDelegate, Rotation {

    override func viewDidLoad() {
        super.viewDidLoad()
        connection.connect() //connect socket
        connection.sendDelegate = self //lets the connection send data to the view controller
        appDelegate.delegate = self //enables view controller to run functions when device rotates

        loadingView.transform = CGAffineTransform(scaleX: 2, y: 2)
        loadingView.isHidden = true

        enableEVPNButton.isEnabled = false
```

Figure 3. Screenshot of XCode depicting where the device selection button is located.

9. Connect a physical iOS device to your computer.

10. Select a physical device from the drop-down menu



```
ndic on 6/6/17.
a Orlandic. All rights reserved.



/iewController, dataDelegate, Rotation {

dLoad() {
d()
ct() //connect socket
elegate = self //lets the connection send data to the view controller
gate = self //enables view controller to run functions when device rotates

sform = CGAffineTransform(scaleX: 2, y: 2)
dden = true

.isEnabled = false


s > 0 {
    generateSpineImages(spines: numberOfSpines) //create spine objects and display them
}
```

Figure 4. Screenshot of XCode depicting selection of a physical device: an iPad called Lara's iPad.

11. Press the "Start" triangle button in the top left corner of the screen to build the XCode project on the iOS device. The application can now be launched from an icon named "iPhoneClient" on the iOS device.
12. If you use a physical device to run the application, you simply need to click on the app's icon on your screen every time you wish to run it. If you use a simulator, you will need to open Xcode and press the "Start" triangle in the upper left corner to launch the app each time.

Using Light Mode vs. Full Mode Application

There are two branches of the Git repository: "master" and "full-package." These correspond to the two different types of vQFX Boxes: light mode and full mode. Light mode boxes only launch VirtualBox instances of the Routing Engine (RE) for each vQFX, while full mode boxes launch two VirtualBox instances per vQFX: the RE and the Packet Forwarding Engine (PFE). Therefore, full mode images require much more processing power on the computer.

From an application standpoint, the full package supports all of the networking features of the application, such as enabling VTEP. However, full mode should not be used to configure more than four vQFXs total, as each machine requires two VirtualBox images, and having too many would crash the host computer.

The light mode of the application, on the other hand, supports all capabilities except for enabling VTEP. It can be used to spin up a maximum of ten vQFXs, since light mode does not use as much of the computer's memory as full mode.

To use the light mode of the application, navigate to the directory at which you cloned the Git repository and enter the command:

```
git checkout master
```

To use the full mode of the application, navigate to the same directory and run the command:

```
git checkout full-package
```

The user may switch between application modes as they please. To check the current branch they are working on, run the command:

```
git branch
```

Application Modes Summary

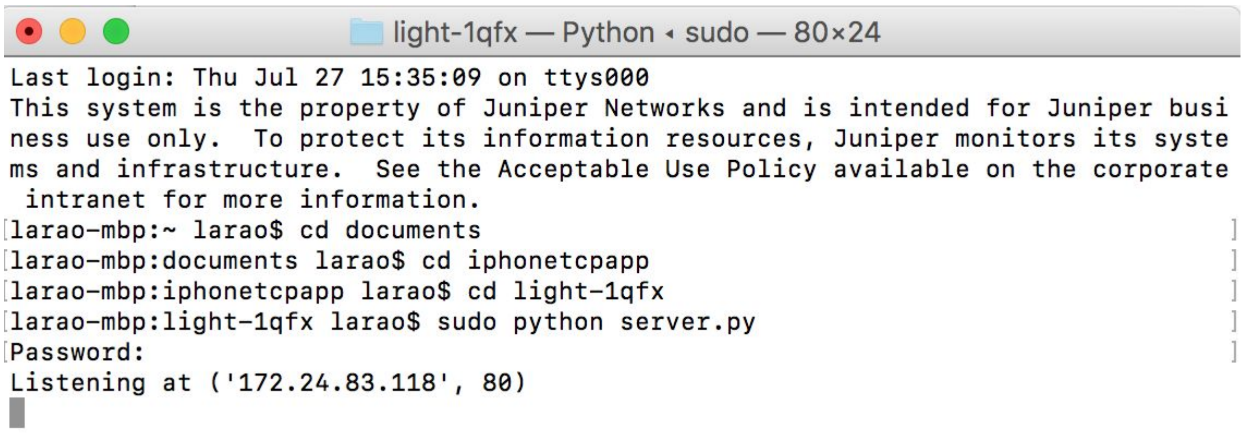
Light Mode	Full Mode
● Only spins up RE for each vQFX ● Supports 10 devices maximum(laptop) ● Does not support VTEP	● Spins up RE and PFE for each vQFX ● Supports 4-5 devices maximum(laptop) ● Supports VTEP

Launching the Application

Launching the Back-End Python Server

Before the application can be launched on the iPad, the back-end Python script must be running on the computer. **The computer and iOS device must be on the same network in order for communication between the two to be possible**. Follow these steps to launch the Python server code:

1. Navigate to the light-1qfx or full-1qfx directory (depending on which mode of vQFX image you wish to generate)
2. Launch the server.py Python file. This file must be run with sudo privileges. The Python server is now running and waiting for a socket connection from the iOS device.



```
Last login: Thu Jul 27 15:35:09 on ttys000
This system is the property of Juniper Networks and is intended for Juniper busi
ness use only.  To protect its information resources, Juniper monitors its syste
ms and infrastructure.  See the Acceptable Use Policy available on the corporate
 intranet for more information.
larao-mbp:~ larao$ cd documents
larao-mbp:documents larao$ cd iphonetcpapp
larao-mbp:iphonetcpapp larao$ cd light-1qfx
larao-mbp:light-1qfx larao$ sudo python server.py
Password:
Listening at ('172.24.83.118', 80)
```

Figure 5. Locating and launching the server.py Python script on a UNIX machine.

3. Optionally, open VirtualBox to see the virtual machines launching in real-time.

Once the Python script is running, click on the “iPhoneClient” application on the iOS Device. This should launch the application, and it will be ready for use at this point.

Application Usage

Launch Screen

When the application is first launched, the user is prompted to enter a desired number of spines and leaves, respectively, that make up the L3 Clos architecture of vQFX switches. Once the desired numbers are selected using the picker, click “Configure.”

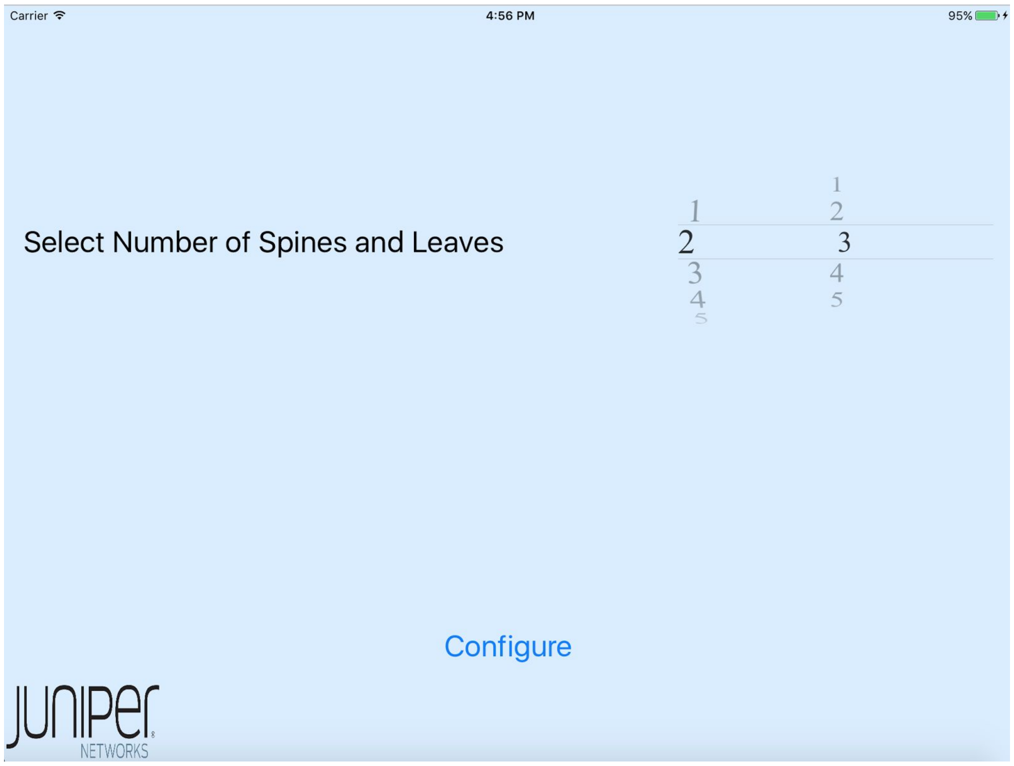


Figure 6. The first screen of the application, where users can customize the spine-leaf topology.

Main Screen

The main screen of the application creates a pictorial representation of the L3 Clos architecture that the user defined, with the top row of switch images representing the spines switches and the bottom row representing the leaf switches. The IP addresses of each device are displayed, as well as lines between the devices signifying the underlying eBGP connection. If the user wishes to go back and create a new configuration, they should click the “Back” button to return to the previous screen.

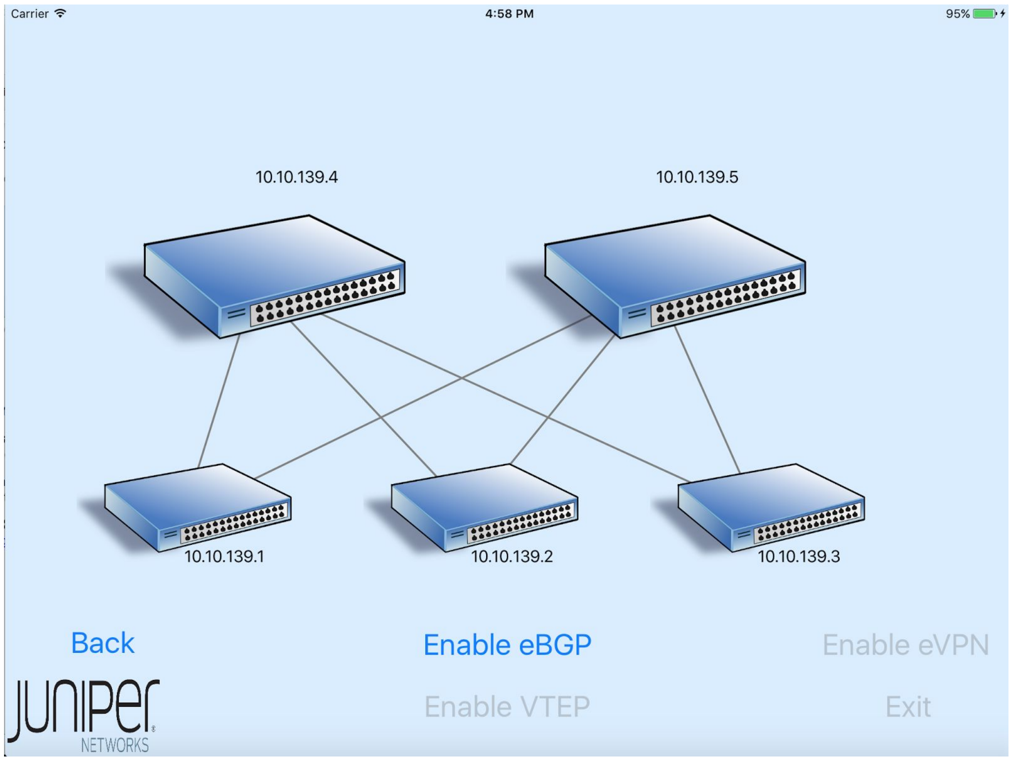


Figure 7. The main screen of the application, depicting the L3 Clos architecture and eBGP underlay.

When the main screen loads, the iOS device connects to the Python server through a TCP socket, but the DC IP Fabric has not been built yet. To launch the vQFX switches on VirtualBox and connect them through an eBGP underlay, click the “Enable eBGP” button. The buttons will become unclickable and a loading symbol will appear as the computer begins launching vQFXs. When a vQFX is done launching, it will turn green on the iPad screen.

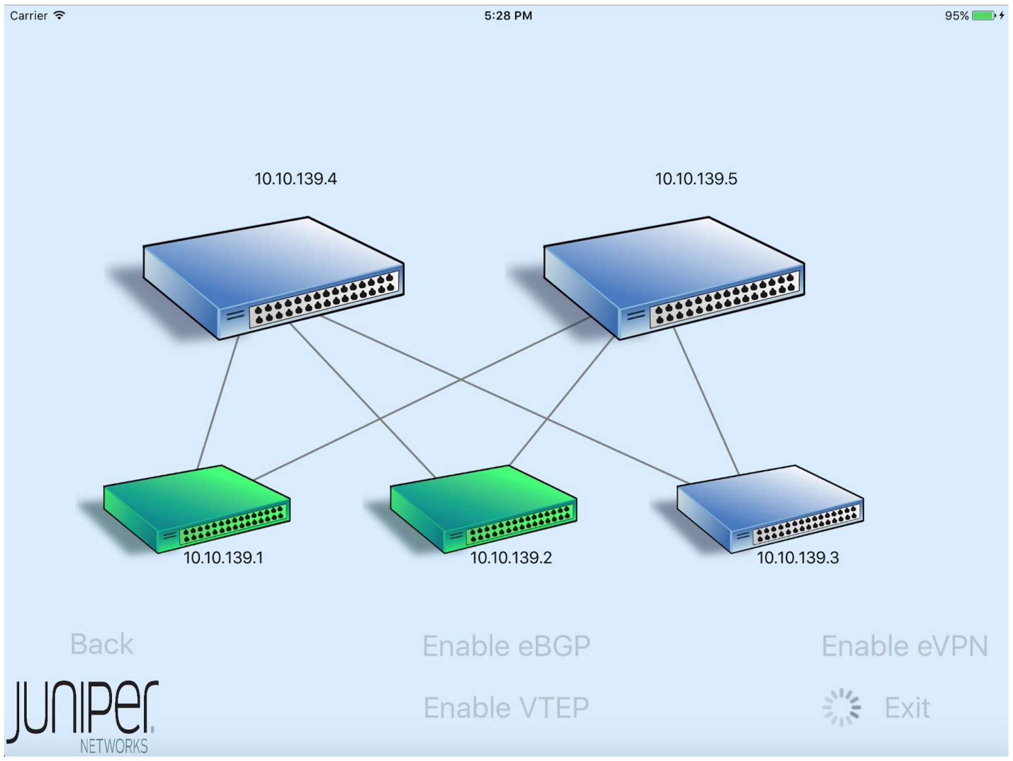


Figure 8. Switch images turn green in real-time as the vQFXs are successfully launched on the computer.

In the event that a switch does not launch successfully, it will turn pink. At this point, the user can click the “Exit” button to destroy all of the vQFX images running on VirtualBox and recreate the configuration. Otherwise, the user may click “Enable eVPN” to generate the eVPN overlay in the network. Once again, the loading symbol will appear and buttons will become unclickable as the eVPN overlay is created on the back-end.

Next, the user may either destroy the vQFXs, or enable VTEP, which will create a tunnel between two of the leaf switches.

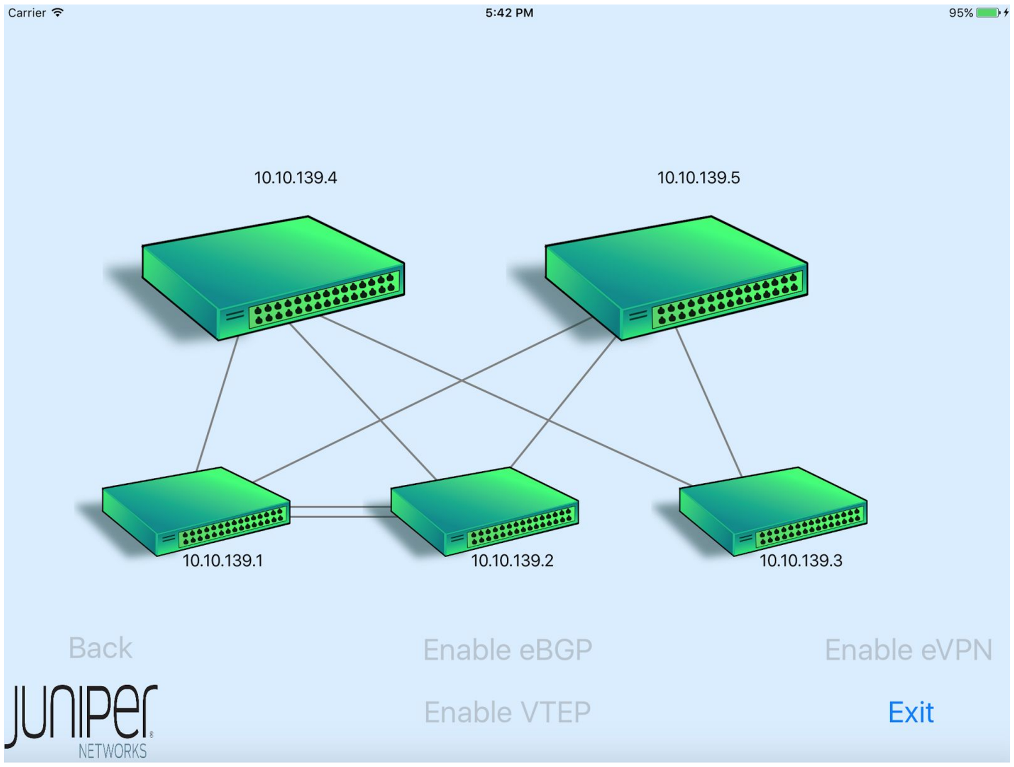


Figure 9. Completed DC IP Fabric with a VTEP tunnel between two leaf switches.

Finally, the user may press “Exit” to destroy the virtual machines, go back to design a new L3 Clos architecture, and start over. As each switch is destroyed on the back-end, its corresponding image on the iOS app will turn pink.

Alternative: Running Only Backend Code

If the user for some reason wishes to bypass the iOS application entirely and launch the virtual machines directly from a Python script, they may run the file light-1qfx/backendlight.py (on light mode) or the full-1qfx/backend.py (on full mode) and hardcode the numbers of leaf and spine switches they wish to spin up.