CSCI 4950

Directed Study: SDN Notes

Spring 2018

Kathy Kwon, Tyler Yox, Neelima Pulagam

DEVELOPMENT ENVIRONMENT

In order to set up the development environment, we want to to run the SDN Controller and emulate the network. For the controller we are using OpenDaylight Controller created by the Linux Project, and we are using Mininet to emulate the Network. Both of these programs are linux based and work best in Ubuntu. To run them in a Windows or Mac environment, we need to use Virtual Machines. Here is a link to aid in this setup. However, since this article is outdated, you will want to get the newest STABLE versions of each component. For our study we used Windows 10 Home Edition, the Nitrogen release of OpenDaylight, and Mininet version 2.2.2 for 32-bit 14.04 Ubuntu, and Java 1.8.1. Instead of using the apt-get package manager to get all the components, I found it easier to download the prebuilt .zip files and .exes from the websites and extract/install them in the linux environments. Note that each VM has its own IP address that we will need to connect together later.

The features that you need to enable every time you run OpenDaylight are:

- odl-dlux-core - enables the ODL dlux GUI
- odl-dluxapps-nodes - enables the nodes GUI
- odl-l2switch-switch - allows ODL to recognize mininet switches
- odl-restconf - Allows access to RESTCONF API

To start the OpenDaylight Shell, run the karaf executable located in the bin folder. To enable the features type "$ feature:enable odl-dlux-core odl-dluxapps-nodes odl-l2switch-switch odl-restconf". We now have a SDN controller equipped with a GUI, and configuration tools using a REST API for our network. While these tools are powerful, at this time we currently just use it for the topology. Another software available is Wireshark which is included in the Mininet VM. Mininet also supports standard linux network performance measurement tools such as iperf.

Mininet works by emulating a network-layer topology and a network's implementation can be abstract for simple simulations, or concrete if you want to test a hardware topology. Mostly, mininet is used for the first task so that we can simulate and evaluate a network before investing in its construction. We chose to emulate the Internet2 Layer 3 Network because there is public data available on the geographic location of each router and each link's respective bandwidths. Additionally, it is of high interest due to it's ground-breaking primary bandwidth of 100Gbps. To create a custom network, you can use the Mininet Python API to define and run your topology. After building the network, you can run performance tests to measure RTT to different nodes, UDP send Jitter, and throughput with the ping and iperf tools. However, it would be best to generate more realistic, steady traffic to create a better simulation. Mininet also has a Command Line Interface (CLI) for interacting with the current running network. To get started just type "sudo -mn" in your Mininet VM! We can see the current running network interfaces with the bwm-ng tool.

UNDERSTANDING THE CODE

```
#!/usr/bin/python
from mininet.net import Mininet
from mininet.node import RemoteController, OVSSwitch, CPULimitedHost
from mininet.topo import LinearTopo
from mininet.link import TCULink
from mininet.cli import CLI
from mininet.log import setLogLevel
import time
import csv
```

Imports for the Mininet Python API

```
def placeController():

      net = Mininet(topo=None, link=TCULink, build=False)

      print "*** Adding Controller ***"
      c0 = RemoteController('c0',ip='192.168.56.102', port=6633);
      net.addController(c0);

      #Linear Topology - Each host gets their own switch
      print "*** Adding Switches ***"
      print "*** Adding Hosts ***"
      citylist = dict()
      with open('nodes.csv', 'r') as csvfile:
            rreader = csv.reader(csvfile)
            next(rreader)
            i = 0
            for row in rreader:
                  if row[2] not in citylist:
                        citylist[row[2]] = 's%d' % (i)
                        switch = net.addSwitch('s%d' % (i), cls=OVSSwitch)
                        host = net.addHost(row[2],type=CPULimitedHost, cpu=.5/30)
                        net.addLink(switch, host,bw=100, delay='5ms')
                        i = i + 1
```

Function that creates and tests the network. First we create the network object. The topo is none since we are defining our own topology. We set c0 as the remote IP to the OpenDaylight VM because that is where the controller is running. I defined the list of geographic locations in a csv file (comma separated values) so that nodes could be changed if needed. Repeated geographic locations are ignored in our simulation as they won't create much propagation delay and we are unsure how two PoPs (Points of Presence) are connected. We are using a linear switch topology where each node has its own corresponding switch. Because there are many nodes that we are simulating, the program is CPU intensive, so we can assign the amount of CPU time each host gets with CPULimitedHosts. The last part links each host with a switch.

```
print '\n*** INTERNET2 ***'
        #Create Links between switches
        with open('links.csv', 'r') as csvfile:
                rreader = csv.reader(csvfile)
                next(rreader)
                for row in rreader:

net.addLink(citylist[row[1]],citylist[row[2]],bw=int(row[3]),delay='2ms')

        net.build()
        net.start()
        i = 0
        net.pingAll()
```

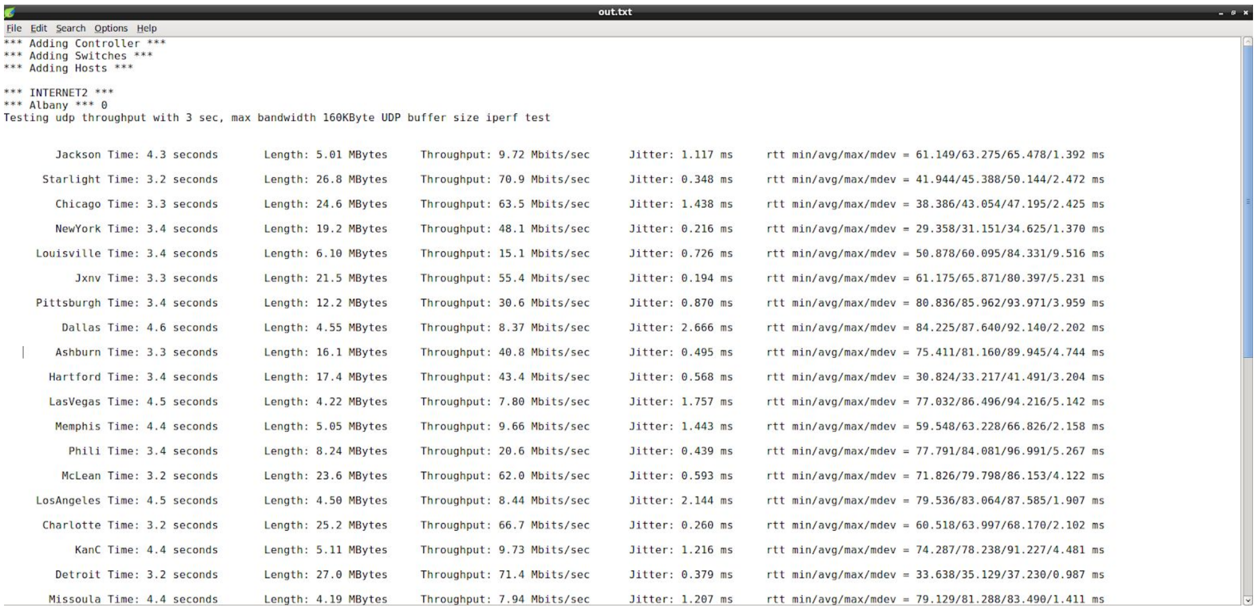
Now we are reading the geographic links that have been defined by the Internet2 architecture. The structuring of these links is a design choice that the engineers used when creating the network. We link together the specified switches, run the build command, then start the network. Lastly we do a pingAll to verify that each node can see all other nodes.

```
for city in citylist:
                avg_throughput = 0.0
                avg_jitter = 0.0
                avg_rtt = 0.0
                print ('*** '+city+' *** %d' % i)
                print ('Testing udp throughput with 3 sec, max bandwidth 160KByte UDP
buffer size iperf test ')
                net.get(city).popen('iperf -us')
                host = net.get(city)
                for city2 in citylist:
                        print ''
                        s = ''
                        if city != city2:
                                s = net.get(city2).cmd('iperf -uc ' + host.IP() + ' -b
100M -t 3')
                                s = s.splitlines()
                                s = s[10].split(' ')
```

Now we can begin performance measurement. We wanted to measure RTT to each node from a starting node, UDP jitter and throughput. We also take averages so that we can compare this node to other nodes more easily. The goal is to be able to find the best nodes to place SDN controllers. To measure throughput and jitter, we run an iperf udp server on one node, and connect to it with an iperf udp client on every other node. To measure RTT we ping the primary node from all other nodes 10 times each.

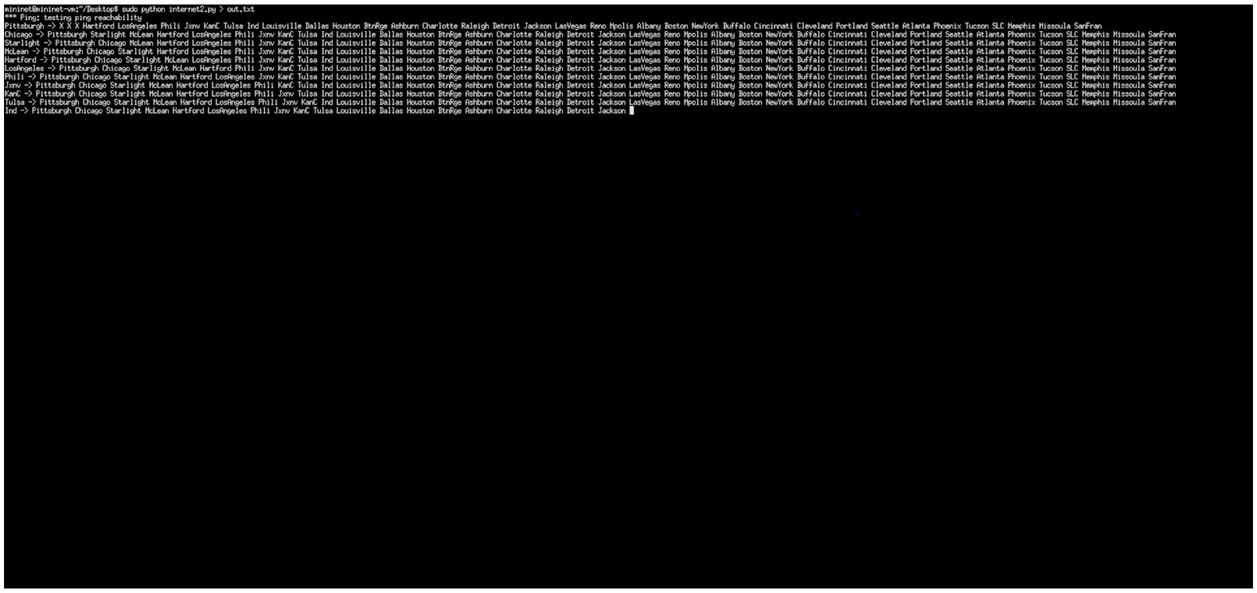
```
# use ping to get delays
                          t = net.get(city2).cmd('ping -c 10 -i .25 ' + host.IP())
                          t = t.splitlines()
                          t = t[-1]

                          print '%15s' % city2 + '\tTime: ' + s[5] + '
seconds\tLength: ' + s[8] + ' ' + s[9] +'\tThroughput: ' + s[11] + ' ' + s[12] +
'\tJitter: ' + s[15] + ' ' + s[16] + '      ' + t
                          avg_throughput = avg_throughput + float(s[11]) /
(len(citylist) - 1)
                          avg_jitter = avg_jitter + float(s[15]) / (len(citylist) -
1)
                          avg_rtt = avg_rtt + float(t[30:35]) / (len(citylist) - 1)
             print ('Average Throughput: ' + str(avg_throughput) + ' Mbits/sec')
             print ('Average Jitter: ' + str(avg_jitter) +  ' ms')
             print ('Average RTT: ' + str(avg_rtt) + ' ms\n\n')
             i = i+1
if __name__ == "__main__":
      placeController()
```
Most of this is is just formatting in order to print out our desired data. Iperf clients and servers generate a lot of boilerplate that we don't necessarily want so we can just parse for the hard data. We keep a running total of the Average Throughput, Jitter and RTT for each node that we are testing, this way so that we can compare the results to

SCREENSHOTS



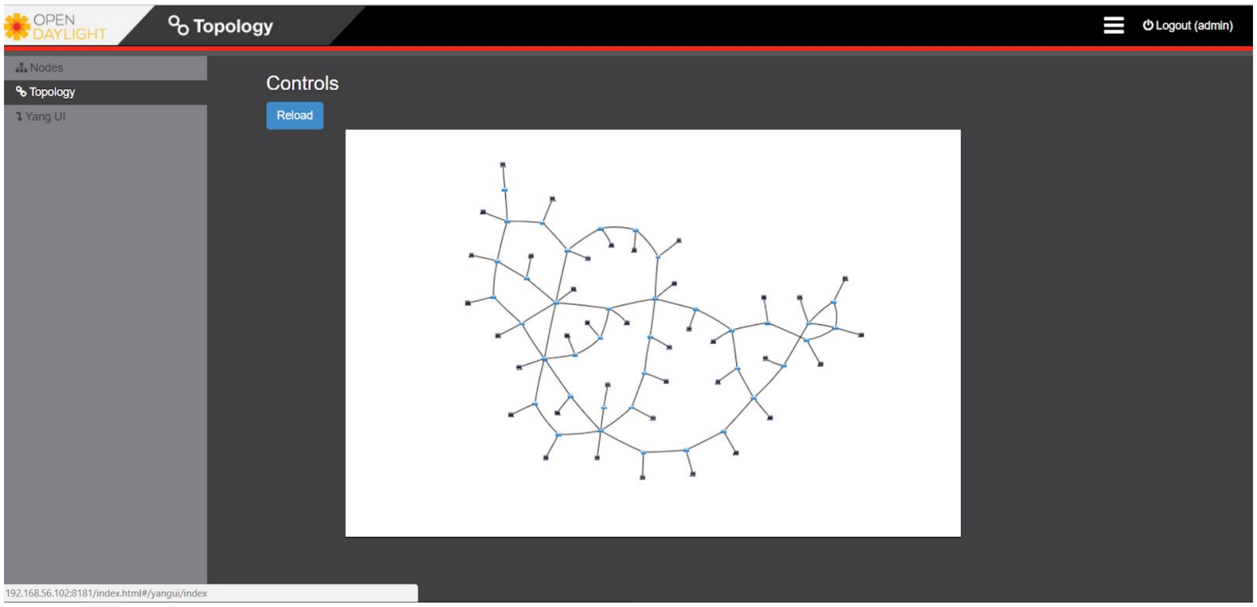
```
*** Adding Controller ***
*** Adding Switches ***
*** Adding Hosts ***

*** INTERNET2 ***
*** Albany *** 0
Testing udp throughput with 3 sec, max bandwidth 160KByte UDP buffer size iperf test


        Jackson Time: 4.3 seconds       Length: 5.01 MBytes     Throughput: 9.72 Mbits/sec      Jitter: 1.117 ms     rtt min/avg/max/mdev = 61.149/63.275/65.478/1.392 ms

      Starlight Time: 3.2 seconds       Length: 26.8 MBytes     Throughput: 70.9 Mbits/sec      Jitter: 0.348 ms     rtt min/avg/max/mdev = 41.944/45.388/50.144/2.472 ms

        Chicago Time: 3.3 seconds       Length: 24.6 MBytes     Throughput: 63.5 Mbits/sec      Jitter: 1.438 ms     rtt min/avg/max/mdev = 38.386/43.054/47.195/2.425 ms

        NewYork Time: 3.4 seconds       Length: 19.2 MBytes     Throughput: 48.1 Mbits/sec      Jitter: 0.216 ms     rtt min/avg/max/mdev = 29.358/31.151/34.625/1.370 ms

     Louisville Time: 3.4 seconds       Length: 6.10 MBytes     Throughput: 15.1 Mbits/sec      Jitter: 0.726 ms     rtt min/avg/max/mdev = 50.878/60.095/84.331/9.516 ms

           Jxnv Time: 3.3 seconds       Length: 21.5 MBytes     Throughput: 55.4 Mbits/sec      Jitter: 0.194 ms     rtt min/avg/max/mdev = 61.175/65.871/80.397/5.231 ms

     Pittsburgh Time: 3.4 seconds       Length: 12.2 MBytes     Throughput: 30.6 Mbits/sec      Jitter: 0.870 ms     rtt min/avg/max/mdev = 80.836/85.962/93.971/3.959 ms

         Dallas Time: 4.6 seconds       Length: 4.55 MBytes     Throughput: 8.37 Mbits/sec      Jitter: 2.666 ms     rtt min/avg/max/mdev = 84.225/87.640/92.140/2.202 ms

   |    Ashburn Time: 3.3 seconds       Length: 16.1 MBytes     Throughput: 40.8 Mbits/sec      Jitter: 0.495 ms     rtt min/avg/max/mdev = 75.411/81.160/89.945/4.744 ms

       Hartford Time: 3.4 seconds       Length: 17.4 MBytes     Throughput: 43.4 Mbits/sec      Jitter: 0.568 ms     rtt min/avg/max/mdev = 30.824/33.217/41.491/3.204 ms

       LasVegas Time: 4.5 seconds       Length: 4.22 MBytes     Throughput: 7.80 Mbits/sec      Jitter: 1.757 ms     rtt min/avg/max/mdev = 77.032/86.496/94.216/5.142 ms

        Memphis Time: 4.4 seconds       Length: 5.05 MBytes     Throughput: 9.66 Mbits/sec      Jitter: 1.443 ms     rtt min/avg/max/mdev = 59.548/63.228/66.826/2.158 ms

          Phili Time: 3.4 seconds       Length: 8.24 MBytes     Throughput: 20.6 Mbits/sec      Jitter: 0.439 ms     rtt min/avg/max/mdev = 77.791/84.081/96.991/5.267 ms

         McLean Time: 3.2 seconds       Length: 23.6 MBytes     Throughput: 62.0 Mbits/sec      Jitter: 0.593 ms     rtt min/avg/max/mdev = 71.826/79.798/86.153/4.122 ms

     LosAngeles Time: 4.5 seconds       Length: 4.50 MBytes     Throughput: 8.44 Mbits/sec      Jitter: 2.144 ms     rtt min/avg/max/mdev = 79.536/83.064/87.585/1.907 ms

      Charlotte Time: 3.2 seconds       Length: 25.2 MBytes     Throughput: 66.7 Mbits/sec      Jitter: 0.260 ms     rtt min/avg/max/mdev = 60.518/63.997/68.170/2.102 ms

           KanC Time: 4.4 seconds       Length: 5.11 MBytes     Throughput: 9.73 Mbits/sec      Jitter: 1.216 ms     rtt min/avg/max/mdev = 74.287/78.238/91.227/4.481 ms

        Detroit Time: 3.2 seconds       Length: 27.0 MBytes     Throughput: 71.4 Mbits/sec      Jitter: 0.379 ms     rtt min/avg/max/mdev = 33.638/35.129/37.230/0.987 ms

       Missoula Time: 4.4 seconds       Length: 4.19 MBytes     Throughput: 7.94 Mbits/sec      Jitter: 1.207 ms     rtt min/avg/max/mdev = 79.129/81.288/83.490/1.411 ms
```

Sample output from internet2.py



```
mininet@mininet-vm:~/Desktop$ sudo python internet2.py > out.txt
*** Ping: testing ping reachability
Pittsburgh -> X X X Hartford LosAngeles Phili Jxnv KanC Tulsa Ind Louisville Dallas Houston BtnRge Ashburn Charlotte Raleigh Detroit Jackson LasVegas Reno Mpolis Albany Boston NewYork Buffalo Cincinnati Cleveland Portland Seattle Atlanta Phoenix Tucson SLC Memphis Missoula SanFran
Chicago -> Pittsburgh Starlight McLean Hartford LosAngeles Phili Jxnv KanC Tulsa Ind Louisville Dallas Houston BtnRge Ashburn Charlotte Raleigh Detroit Jackson LasVegas Reno Mpolis Albany Boston NewYork Buffalo Cincinnati Cleveland Portland Seattle Atlanta Phoenix Tucson SLC Memphis Missoula SanFran
Starlight -> Pittsburgh Chicago McLean Hartford LosAngeles Phili Jxnv KanC Tulsa Ind Louisville Dallas Houston BtnRge Ashburn Charlotte Raleigh Detroit Jackson LasVegas Reno Mpolis Albany Boston NewYork Buffalo Cincinnati Cleveland Portland Seattle Atlanta Phoenix Tucson SLC Memphis Missoula SanFran
McLean -> Pittsburgh Chicago Starlight Hartford LosAngeles Phili Jxnv KanC Tulsa Ind Louisville Dallas Houston BtnRge Ashburn Charlotte Raleigh Detroit Jackson LasVegas Reno Mpolis Albany Boston NewYork Buffalo Cincinnati Cleveland Portland Seattle Atlanta Phoenix Tucson SLC Memphis Missoula SanFran
Hartford -> Pittsburgh Chicago Starlight McLean LosAngeles Phili Jxnv KanC Tulsa Ind Louisville Dallas Houston BtnRge Ashburn Charlotte Raleigh Detroit Jackson LasVegas Reno Mpolis Albany Boston NewYork Buffalo Cincinnati Cleveland Portland Seattle Atlanta Phoenix Tucson SLC Memphis Missoula SanFran
LosAngeles -> Pittsburgh Chicago Starlight McLean Hartford Phili Jxnv KanC Tulsa Ind Louisville Dallas Houston BtnRge Ashburn Charlotte Raleigh Detroit Jackson LasVegas Reno Mpolis Albany Boston NewYork Buffalo Cincinnati Cleveland Portland Seattle Atlanta Phoenix Tucson SLC Memphis Missoula SanFran
Phili -> Pittsburgh Chicago Starlight McLean Hartford LosAngeles Jxnv KanC Tulsa Ind Louisville Dallas Houston BtnRge Ashburn Charlotte Raleigh Detroit Jackson LasVegas Reno Mpolis Albany Boston NewYork Buffalo Cincinnati Cleveland Portland Seattle Atlanta Phoenix Tucson SLC Memphis Missoula SanFran
Jxnv -> Pittsburgh Chicago Starlight McLean Hartford LosAngeles Phili KanC Tulsa Ind Louisville Dallas Houston BtnRge Ashburn Charlotte Raleigh Detroit Jackson LasVegas Reno Mpolis Albany Boston NewYork Buffalo Cincinnati Cleveland Portland Seattle Atlanta Phoenix Tucson SLC Memphis Missoula SanFran
KanC -> Pittsburgh Chicago Starlight McLean Hartford LosAngeles Phili Jxnv Tulsa Ind Louisville Dallas Houston BtnRge Ashburn Charlotte Raleigh Detroit Jackson LasVegas Reno Mpolis Albany Boston NewYork Buffalo Cincinnati Cleveland Portland Seattle Atlanta Phoenix Tucson SLC Memphis Missoula SanFran
Tulsa -> Pittsburgh Chicago Starlight McLean Hartford LosAngeles Phili Jxnv KanC Ind Louisville Dallas Houston BtnRge Ashburn Charlotte Raleigh Detroit Jackson LasVegas Reno Mpolis Albany Boston NewYork Buffalo Cincinnati Cleveland Portland Seattle Atlanta Phoenix Tucson SLC Memphis Missoula SanFran
Ind -> Pittsburgh Chicago Starlight McLean Hartford LosAngeles Phili Jxnv KanC Tulsa Louisville Dallas Houston BtnRge Ashburn Charlotte Raleigh Detroit Jackson
```

Pingall command runs before performance measurements so switches can learn paths to other nodes in the network



Visualization of the internet2 topology in OpenDaylight, using our linear switch topology

SOFTWARE DEFINED NETWORKING (SDN) FOR BEGINNERS

What is SDN? [3]

Software defined networking (SDN) is an approach to using open protocols, such as OpenFlow, to apply globally aware software control at the edges of the network to access network switches and routers that typically would use closed and proprietary firmware. SDN strives to optimize network resources and adapt to traffic and flow. The use of SDN controllers allows for better management of networks.

What is the purpose of SDN?

The aim of SDN is to make the network more agile and flexible. This will allow network administration more flexible and allow the network administrator to shape traffic from a centralized control console without having to touch individual switches. The goal of SDN is to allow network engineers to respond quickly to changing business requirements.

Separation of the Control Plane

The control plane makes decisions about how packets should flow through the network from the data plane to the network

Data Plane

The data plane moves packets from place to place. It carries user traffic and enables data transfer to and from clients. Data plane traffic travels through routers rather than to or from them.

Controllers and Switches

Controller sends the switch rules about packet handling. Packet arrives at switch →Rules in switch's firmware tell switch where to forward the packet →switch sends packet to destination. Controllers and switches communicate through a controller's south bound interface, usually through openflow.The south bound interface allows a component to communicate with a lower level component. Biggest objective for networks architects currently is to create one mechanism to manage a network from the top to bottom.

OpenFlow [6]

The goal of OpenFlow is to convert route forwarding behavior into specific forwarding changes. Simply put it is a protocol that allows a server to tell network switches where to send packets.

Mininet

Mininet basically creates a realistic virtual network with a single command. It is useful for development, teaching and research as it is a great way to understand OpenFlow and SDN.

Heuristic Approaches to the Controller Placement Problem in Large Scale SDN Networks

- Software defined networking - shift toward externalized and logically centralized network control plane
    - Logically centralized architecture separates control and data plane
- Requirements - latency constraints, failure tolerance, load balancing
- POCO: Pareto-based Optimal COntroller

    - MatLab based framework
    - Performs an exhaustive evaluation of all possible placements
    - Good for small and medium sized networks, not for large
    - Toolset extended by heuristic approach
        - Less accurate
        - Faster computation
- OpenFlow - most popular SDN-enabling communications protocol b/w control and data plane, allowing controller-to-agent communication over southbound API. Separated into a wire protocol and a configuration and management protocol.
- Flow table - Uses more than just the IP/MAC address to decide the next route to take. Can use any information in the packet. MPLS is not required and virtual switching can be applied.
- In OpenFlow architecture, logically centralized controller manages switches by providing rules that dictate their packet handling behavior
- HyperFlow - allows partitioning of OpenFlow networks into multiple domains that are each handled by individual controllers
- Key issues in architectures w/ externalized control plane
    - 1. Number of SDN controllers required for reliable and resilient network operation
    - 2. Position of each controller in the network affects competing objectives, such as inter-controller latency, switch-controller latency, resilience

    - 3. To cope w/ large scale network and dynamically changing network conditions, such as traffic patterns and bandwidth demands
- Optimizing latency from nodes to their assigned controller - NP-hard
- Guarantee to find optima with respect to latency
- Eliminates necessity to impose constraints on objective values w/ priori, which might result in excluding feasible alternatives
- Multiobjective approach allows for clearer illustration of trade-offs b/w competing criteria
- Faster calculation = faster a switch to another placement triggered
- MOCO - multiobjective combinatorial optimization
    - PSA - Pareto Simulated Annealing
    - Allows POCO to handle huge network instances w/o sacrificing much accuracy
- Improvements to POCO include
    - extended filtering mechanism
    - possibility to visualize up to 4 dimensions of solution space - allowing analysis of interrelations b/w up to 2 diff objects
- SDN key principle - separation of data plane and control plane
    - Single, centralized controller
    - Multiple equally important controllers responsible for partitions of network
    - Set of distributed controllers arranged in hierarchy
- Purely software-based or hardware solutions for control plane allow trade-offs b/w flexibility, performance, cost

- Connections involving controllers realized inband or outband (sharing same physical link as data plane traffic or utilizing dedicated lines)

Textbook Notes

- Distributed vs. Centralized Control Planes: The current network model is a distributed, eventual consensus model. The characteristics of current databases didn't exist when this approach was used, so the more modern SDN uses a Centralized Control Plane. The advantage of this is the view of the network provided and the simplification of programmatic control.
- When we say centralized control plane, we are speaking in a logical sense. In a literal sense we cannot support a large-scale, high availability, close-by network in one geographic location. We use a logically centralized but physically distributed control plane.

Videos/Other

Helpful Videos: (slightly outdated, but still very useful)

- https://www.youtube.com/watch?v=2BJyIIIYU8E
- https://www.youtube.com/watch?v=DiChnu_PAzA

Video Notes

- SDN fundamental idea is that we are trying to move away from hardware into software
- Have a control panel that controls all routers instead of managing each separate, individual router

- QoS - quality of service
    - Priority of certain services over others
        - Ex) VoIP is more important b/c real time communication, opposed to FTP
- SDN - can dynamically model and shape traffic depending on what we need to do; separating out different components of network infrastructure so we can deal with them separately -> more efficient to manage
    - Data plane - all switches, routers
    - Control plane - management servers that communicate w/ all different networking equipment on data plane
        - Right at a certain time, how should data move on the data plane?
    - Service plane - firewall
    - Management plane - controls and makes sure servers on control plane are functioning the way they are supposed to
- What's the point of shaping traffic in real time?
    - Bandwidth management
    - Nice to have an infrastructure that we can prioritize traffic in real time
        - Ex) use an entire pipe (1 GBps) and depending on current needs, different services running on that network will have different priorities
        - Ex) shut off all protocols besides one, so that it can have priority
    - In modern networks, it's currently set so that if each service has a certain amount of allocation of bandwidth

Internet2

- 100Gbps network connecting US research communities. Mostly provides service to academia and researchers.
- Points of Presence (POPs) are located in major cities across US
- Massive bandwidth is ideal for more headroom, thus higher network performance and almost no congestion and dropped packets
- Connected to other research and education networks around the world

APPLICATIONS OF SDN [12]

The goal of software defined networking is to build a dynamic software-based network infrastructure that is able to control individual components. This allows for better allocation of bandwidth and across all networking layers, agility, and management. SDN applications can replace and expand upon functions that were originally implemented through hardware and firmware of a conventional network. The benefits of using SDN include managing data centers and cloud environments.

[5] The top five applications of SDN include: security services, network intelligence and monitoring, compliance and regulation-bound applications, high performance applications, distributed application control and cloud integration.

Companies such as Google, AT&T, Cisco, etc. have all implemented SDN into their network architecture, which is an excellent business investment in the long run. Controllers and switches are much less expensive than buying numerous expensive routers. On top of that, they have the benefits of SDN's flexible architecture.

If SDN were to become the standard control-plane approach that is use world-wide, similar to current-day traditional routing algorithm, then every big corporation would also have to adapt to the SDN architecture. This is arguably the biggest drawback right now to SDN, since most large corporations are not willing to make such a huge and expensive switch. It's also difficult to get every company to agree on this switch. Therefore, the complete switch to an SDN architecture continues to be delayed.

<u>Largest Applications of SDN</u>

SDMN

SDMN stands for Software Defined Mobile Networks and embracing SDN to design mobile networks could solve a lot of difficult problems in the cellular and wireless networks including managing heterogeneity,complexity and consistency in the network and catalyze fundamental changes in the mobile world. We know that the big idea of SDN is decoupling the data and control planes to give us a more centralized control of networks. Using this, scientists intend to create 5G networks, networks who are mainly software oriented. Essentially, we would be extending SDN approaches to include mobile networks.

SD-WAN [7]

SD-WAN stands for Software Defined Wide Area Network and it is a very specific SDN software applied to WAN connections. Enterprises and corporations are seeking a more flexible and cloud based WAN technology. This is much better than installing WAN technology at each office that can often be expensive, hard to maintain and require a lot of work to install at every branch. Software defined WAN technology is mainly uses internet broadband connections to replace other costly alternatives and virtualization also allows more security and VPN technologies to broadband internet connections.

Companies seek to switch to SD-WAN to focus on cost, reliability and security. It’s end goal is to provide simple cloud-enabled WAN connections with better application delivery control and VPN technologies. It is gaining more and more popularity each year as it helps connect company branch offices in a wide geographical network and brings heightened security with end-to-end encryption.

SD-LAN [15]

SD-LAN stands for Software Defined Local Area Network and SD-LANs are increasingly gaining popularity for their cloud management system and wireless connectivity without a physical controller. SD-LAN is similar to SD-WAN but approaches SDN design differently. A few main differences include topology "predictiveness", network security, loss, latency and jitter, flow of information.

SD-LAN has more of a predictive topology with layers of switches than SD-WAN and each layer aggregates bandwidth for each layer below it. This allows application optimization and SD-LAN changes the behaviour of the network based on the apps running giving it the ability to focus resources on where they are most needed. It adaptively heals, optimizes and organizes its access points and switches to enhance control by distributing the control plane to obtain better capacity and speed. Security for SD-LAN is primary served by authentication of end point devices and encryption of bits on the wire. It dynamically figures out what users and devices as well as when and where they can have access to SDN-LAN. Access can be micromanaged by controlling what time a set of users can have access or assigning a set amount of bandwidth. LAN also offers better quality of service because it prioritizes the flow of traffic. In concluding remarks, SDN-LAN has been at the forefront with respect to network segmentation and addressing more challenges we face with SDN everyday.

Security

With SDN environments, security is very important is almost always integrated into the architecture and it is delivered as a service to protect the privacy and availability of all the resources and information on a network. Because SDN allows a central view of the network, it has the capacity to reprogram the data plane at any time enhancing may network-related security applications. And because the SDN controller is centralized, this decision point needs to be very secure and tightly controlled because if the SDN controller goes down, so does the whole network.

Looking more at the application side of SDN and its contributions to security applications, we'll find DDoS detection and mitigation, bonet and worm propagation. By regularly collection network statistics using OpenFlow and then checking for any anomalies. If an anomaly is detected, the application instructs the controller to reprogram the data plane enabling security.

A growing research topic with SDN is MTD (moving target defense) algorithms which are security algorithms that make attacks on the network very difficult. By constantly changing the key properties of the network, it makes it harder for hackers to attack any part of the network. Flowvisor and FlowCheker are both applications that use SDN's centralized control plane as an advantage by separating, monitoring and configuring internet traffic.

Group Data Delivery

Data distribution applications are used for synchronization, fault resilience, load balancing and getting data close to users. To aid this, recent applications replicate data from data centers so that there is better fault tolerance and allows for easy recovery of data.SDN is very useful for Group data delivery because it very intelligently sets up forwarding tables for RGDD (Reliable Group Data Delivery) not to mention OpenFlow has supported supported Group Tables since its original version. So this has been a goal for SDN since the beginning, making the functions of networks very efficient and clean.

COMPANIES THAT USE/SELL SDN PRODUCTS (AS OF 2018) [9] [8]

- Google [10]
  - In 2017, Google introduced their architecture, Espresso, to the public with the purpose of bringing SDN to the public internet. Espresso runs BGP. Google claims that the SDN architecture reduces network latency between Compute Engine VMs by 40% with the newer Andromeda 2.0.
- AT&T [8]
  - AT&T models its future vision with an SDN architecture. Their goal is to virtualize and control over 75% of their network with the software-defined architecture by 2020. AT&T combines SDN and Network Functions Virtualization (NFV) to create a network service called AT&T FlexWare. It has the flexibility and cost-efficiency of SDN, while it simultaneously has the simplicity of NFV.
- Cisco [8]
  - Number 1 data center vendor. They create custom chips that collect application traffic flow data. Cisco is innovative in the SDN industry, since they integrate silicon into their hardware. Cisco sells SDN services to other service providers, as well as network virtualization and automation solutions.
- VMware [14]
  - Gave credibility to the SDN network virtualization overlay by commercializing Nicira. VMware is looking to abstracting the network and control plane. Their goal is the create a fully abstract network on top of a physical instructure.

VMWare itself also focuses on Network Virtualization, which is different from SDN in the sense that it truly separates itself from underlying hardware, unlike SDN.

- HP [13]
    - HP integrated SDN that supports OpenFlow, which back in 2012 when this was first announced was a new, innovative approach. It has since then expanded on its network virtualization with fully functional controllers and applications.
- Huawei [8]
    - Chinese technology company that has been up and rising for the past few years. Have been innovating SDN by creating a programmable switch, which has a programmable control plane in silicon. Huawei invests a lot of time and research into the SDN business.
- Hewlett Packard Enterprise [8]
    - Innovative because they launched the first SDN App Store and open networking, that uses bare metal switches. Hewlett Packard Enterprise has received thousands of downloads and the number of users continues to increase.
- Juniper Networks [11]
    - The products and services that Juniper Networks provides regarding SDN products is that it promises high performance, elasticity, and security. There are various different products that offer different services to the user, so that the user will be able to choose the service that best fits their needs.
- Cumulus Networks [8]

    - The first company to create a Linux operating system for Ethernet switches. They are very invested in DevOps. Although Cumulus Linux is similar to SDN, it is slightly different in the sense that if is not SDN itself, but rather it enables SDN solutions through being a highly automatic operating system.
- Broadcom [8]
    - Broadcom focuses on providing switching chips and developer tools for SDN. They are working with silicon to create newer and better chips. They also continue to work with developing tools and software to continue their innovations.
- Big Switch
    - They are working with large firms like Cisco but they are pioneers in making available software suites for SDN. They have provided open-source API's, and established themselves as huge proponents of SDN utilizing the three-tier architecture.
- Symantec
    - Symantec has emerged as one of the largest software companies becoming the global leader in Cyber security. Their main goals have been to utilize SDN to find better data center solutions, and build an environment that is both agile and secure making it well suited for the digital age.

NEXT STEPS: PROJECT CONTINUATION FOR FUTURE STUDENTS

Here is a list of additional features and directions that this project can take:

- Flexible defining of networks and topographies so that you don't need to hardcode every Point of Presence in a csv file
- Using other emulations or applications besides mininet. They do not allow the full bandwidth Internet 2 requires (caps at 1Gbps)
- Data analysis for deciding where to place the SDN controllers
- More integration with OpenDaylight and utilizing OpenFlow

Overall, this project was very eye-opening to us. We learned about SDN, which is less common since it is currently not the traditional routing system. We firmly believe that SDN will one day become the universal routing architecture that even everyday people use, not just large corporations.

WORKS CITED

1. http://www.datacenterknowledge.com/archives/2016/03/31/top-five-apps-and-services-that-can-benefit-from-sdn
2. https://www.webopedia.com/TERM/S/software_defined_networking.html
3. http://searchsdn.techtarget.com/definition/software-defined-networking-SDN ✓
4. http://searchnetworking.techtarget.com/definition/layer-2
5. http://searchtelecom.techtarget.com/tip/Unlocking-the-potential-of-SDN-controller-applications ✓
6. http://whatis.techtarget.com/definition/OpenFlow ✓
7. https://www.sdxcentral.com/sd-wan/definitions/software-defined-sdn-wan/ ✓
8. https://www.crn.com/slide-shows/networking/300079644/top-10-sdn-market-leaders-in-the-data-center-and-enterprise-in-2016.htm/ ✓
9. https://searchsdn.techtarget.com/news/2240212374/Thinking-about-SDN-Here-are-42-vendors-that-offer-SDN-products ✓
10. https://www.sdxcentral.com/articles/news/google-brings-sdn-public-internet/2017/04/ ✓
11. https://www.juniper.net/us/en/products-services/sdn/ ✓
12. https://www.juniper.net/us/en/solutions/sdn/what-is-sdn/ ✓
13. https://www.networkcomputing.com/networking/hp-details-sdn-strategy-announces-new-products/798094923 ✓
14. https://www.vmware.com/solutions/software-defined-datacenter/in-depth.html ✓
15. https://www.nojitter.com/post/240170988/the-same-but-different-sdlan-v-sdwan ✓

Checks represent the ones I’ve already referenced✓✓✓