NANOG 80 Hackathon Recap

Michael Costello (just one of a cast of tens)

Hackathon stats

- 11th Hackathon (first virtual)
- Participants worked from 13:00 EDT Saturday to 17:00 EDT Sunday
- Participants from the US, Canada, and South Africa

Hackathon overview

- In support of NANOG's educational mission
- Organized by the Program Committee
- Weekend before the general conference
- Participants work individually or self-organize into groups
- Participants choose their own projects

Projects included

- A from-scratch BGP daemon
- Performance monitoring and notifications using home control systems
- Simulate traffic and visualize streaming telemetry data
- gnmi-gateway Kafka exporter and Netbox target loader
- A modern NNTP client library in Rust

Projects you'll see here today

- RIB compliance using Model Driven Telemetry
- A fully automated data center Proof of Concept
- A simple, automated, and easy-to-deploy SDN solution

Hackathon for NANOG 81

- We’re looking for participants, mentors, and sponsors
- We’ll be reviewing survey results ahead of planning
- See you in February

Lawrence Bird
https://github.com/TheBirdsNest/
https://www.linkedin.com/in/lawrenceabird/

Yordan Sutanto
https://github.com/yordangit12/
https://www.linkedin.com/in/yordan-sutanto-5ba4487/

BGP RIB COMPLIANCE USING MDT

https://github.com/petermoorey/NANOG-80-Hackathon/

NANOG-80 Hackathon

Peter Moorey
https://github.com/petermoorey/
https://www.linkedin.com/in/pmoorey/

Vladimir Yakovlev
https://github.com/VladimirGHC/
https://www.linkedin.com/in/vladimir-yakovlev-398ba5a0/

GOAL

- Evaluate various technology stacks
- Evaluate network telemetry capabilities
- Develop a network capable of showcasing various routing scenarios
- Provide real-time evaluation of BGP routes to detect:
  - Poorly configured route-maps
  - Incorrect provider policies
  - Route hijacking

TECHNOLOGY STACK

- CSR 1000v
- CML

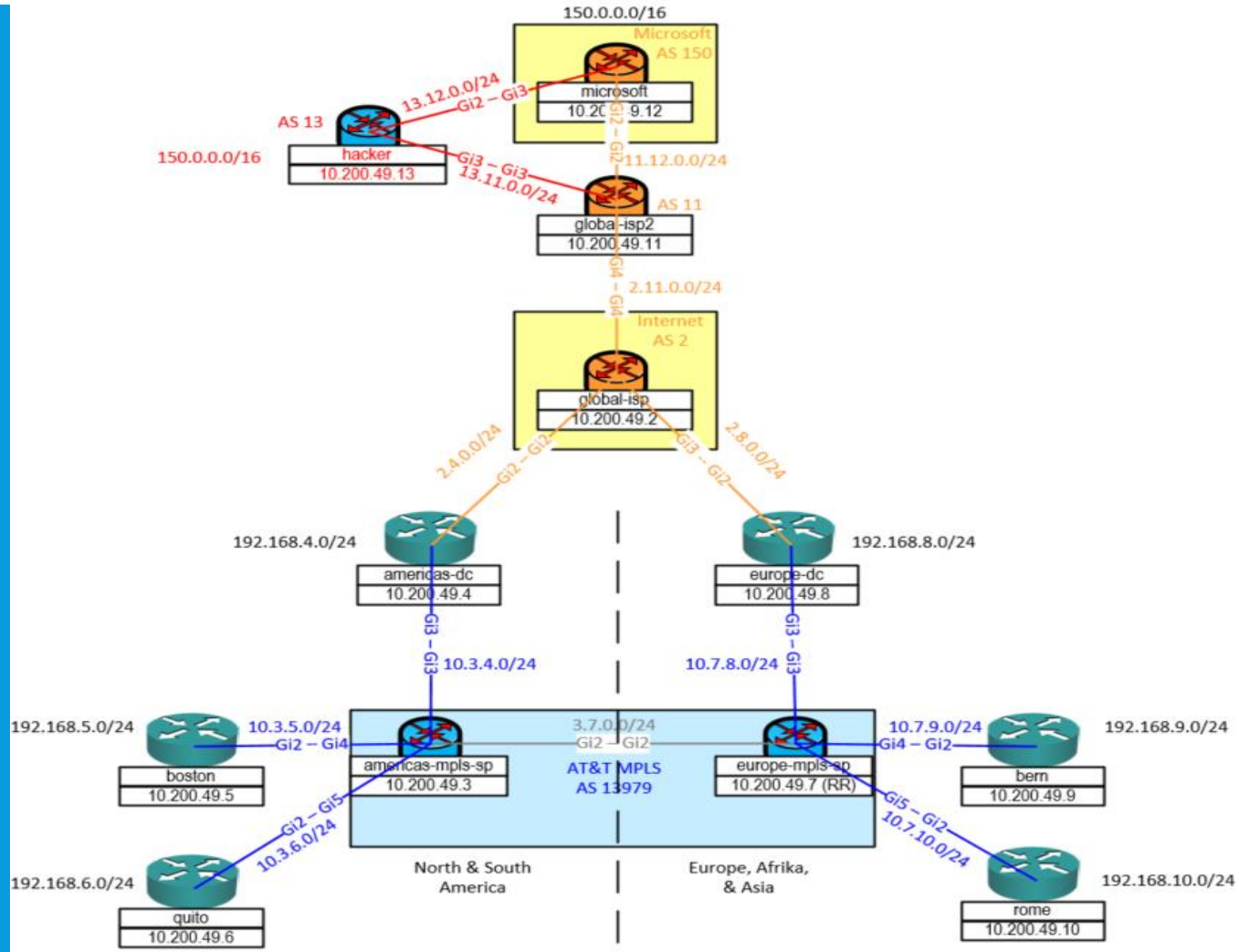
- Service Provisioning
- Service Chaining

- InfluxDB
- Telegraf
- Chronograph

- Policy Processing
- Notification Handling
- Websocket Manager

LAB SETUP

- Challenges:
  - Are the regional route preferences being honored for the default route?
  - Is traffic to/from our providers being hijacked?



DATA COLLECTION

templates > mdt_xml.jinja2

```
<config>
 <mdt-config-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-cfg">
{% for id, sub in data["subscriptions"].items() %}
  <mdt-subscription>
   <subscription-id>{{ id }}</subscription-id>
   <base>
    <stream>yang-push</stream>
    <source-vrf>{{ sub["sourcevrf"] }}</source-vrf>
    <encoding>encode-kvgpb</encoding>
{% if sub["period"] == "on_change" %}
    <no-synch-on-start>false</no-synch-on-start>
{% else %}
    <period>{{ sub["period"] }}</period>
{% endif %}
    <xpath>{{ sub["xpath"] }}</xpath>
   </base>
   <mdt-receivers>
    <address>{{ sub["rx"]["ip"] }}</address>
    <port>{{ sub["rx"]["tcp_port"] }}</port>
    <protocol>grpc-tcp</protocol>
   </mdt-receivers>
  </mdt-subscription>
{% endfor %}
 </mdt-config-data>
</config>
```

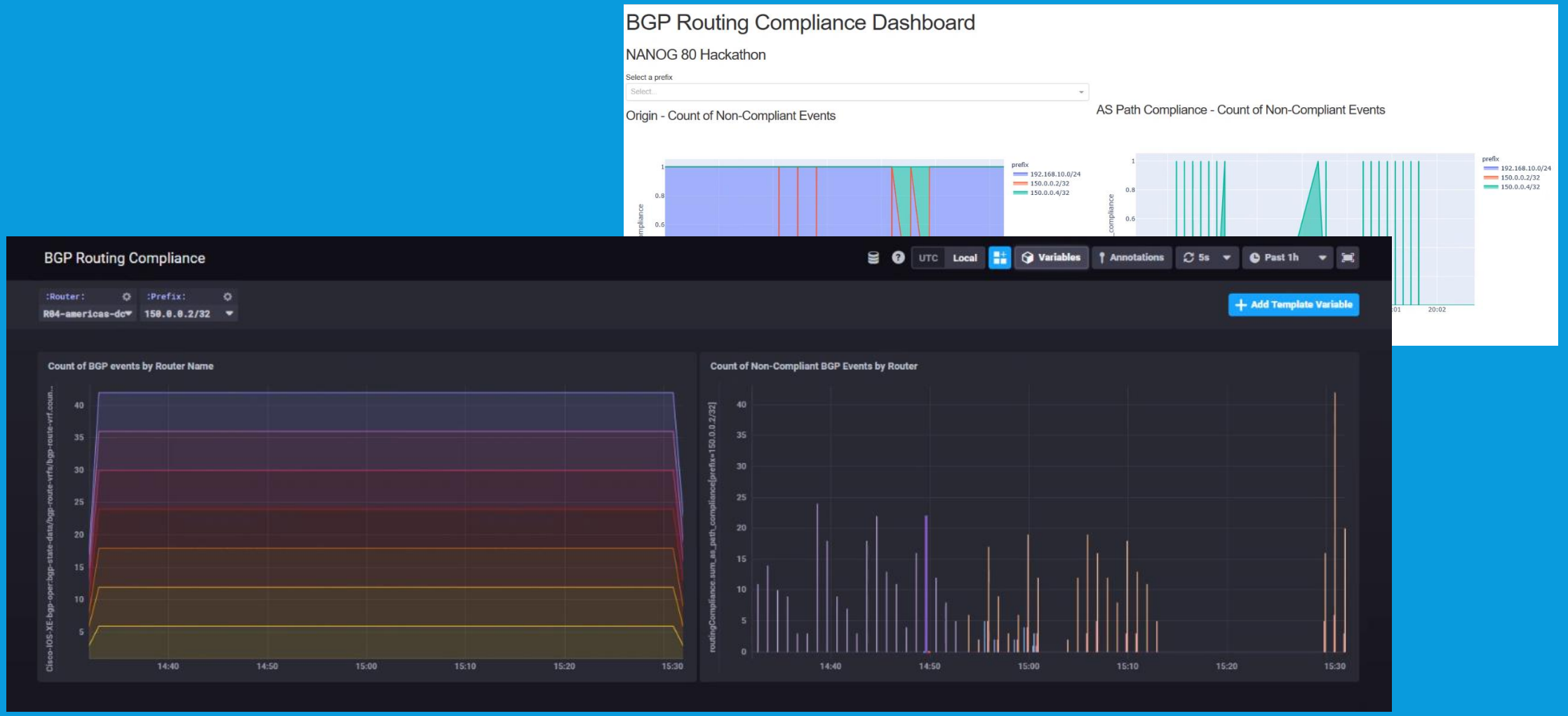
```python
def main(grpc_host):

    hosts = []

    with open("mdt_subscriptions.yml", "r") as f:
        mdt_sub = safe_load(f)

    with open(INVENTORY_FILE, "r") as f:
        inventory = safe_load(f)

    for router in inventory['routers']:
        if inventory['routers'][router]['mdt_bgp']:
            hosts.append(inventory['routers'][router]['ip'])

    for subscription in mdt_sub['subscriptions']:
        mdt_sub['subscriptions'][subscription]['rx']['ip'] = grpc_host
    j2_env = Environment(loader=FileSystemLoader("."), trim_blocks=True, autoescape=True)
    template = j2_env.get_template("templates/mdt_xml.jinja2")
    new_config = template.render(data=mdt_sub)

    for host in hosts:

        connect_params = {
            "host": host,
            "username": USER,
            "password": PWD,
            "hostkey_verify": False,
            "allow_agent": False,
            "look_for_keys": False,
            "device_params": {"name": "csr"},
        }

        with manager.connect(**connect_params) as conn:
            print(f"NETCONF session connected: {host}")

            # Perform the update, and if success, print a message
            config_resp = conn.edit_config(target="running", config=new_config)
            if config_resp.ok:
                print(f"Added ({len(mdt_sub['subscriptions'])}) subscriptions")

        print(f"NETCONF session disconnected: {host}")
```

POLICY EVALUATION

```yaml
# BGP RIB Compliance Policy file for demo lab

root_source: Cisco-IOS-XE-bgp-oper:bgp-state-data/bgp-route-vrfs/bgp-route-vrf
root_path: bgp_route_af/bgp_route_filter/bgp_route_entry/bgp_path_entry/
regions:
    emea:
        - 10.200.49.9       # Bern
        - 10.200.49.10      # Rome
    americas:
        - 10.200.49.5       # Boston
        - 10.200.49.6       # Quito
    external:
        - 10.200.49.4       # Americas DC
        - 10.200.49.8       # Europe DC
policy:
    0.0.0.0/0:                          # Match default route
        match: explicit                 # Match type is explicit. RIB prefix must match 0.0.0.0/0 exactly
        region: americas                # Region to monitor for RIB updates on
        attributes:                     # List of attributes to evaluate and expected values
            community: 100:1            # Community value expected in RIB update
    0.0.0.0/0:                          # Match default route
        match: explicit                 # Match type is explicit. RIB prefix must match 0.0.0.0/0 exactly
        region: emea                    # Region to monitor for RIB updates on
        attributes:                     # List of attributes to evaluate and expected values
            community: 100:2            # Community value expected in RIB update
    192.168.0.0/16:
        match: any                      # any match will include any network prefix that is within the subnet, or the subnet itself
        region: external
        attributes:
            origin: 100
            # community: 100
            as_path: 13979 13979
    150.0.0.0/16:
        match: any
        region: external
        attributes:
            origin: 150
```

```python
def is_in_policy(self, prefix: str) -> str:
    try:
        target_network = IPNetwork(prefix)

        for prefix, policy in self.policies.items():
            parent_network = IPNetwork(prefix)

            if policy['match'] == 'any':
                if target_network in parent_network:
                    print(f"{target_network} in {parent_network}")
                    return str(parent_network)
            elif policy['match'] == 'explicit':
                    if target_network == parent_network:
                        print(f"{target_network} is {parent_network}")
                        return prefix
            else:
                raise NotImplementedError
    except KeyError:
        print("Malformed Policy")

    return None

def evaluate(self) -> bool:
    query = f'SELECT * FROM "{self.source}" WHERE time > now() - 15s'
    print(query)
    results = self.db.query(query)
    for result in results.get_points(self.source):
        prefix_path = 'bgp_route_af/bgp_route_filter/bgp_route_entry/prefix'
        prefix = result[prefix_path]
        print(f"Prefix is {prefix} on source {result['source']}")

        if prefix:
            compliance = {}
            policy_prefix = self.is_in_policy(prefix)
            print(f"Policy prefix: {policy_prefix}")
            print(f"evaluating prefix {prefix} in policy")
            if policy_prefix:
                policy = self.policies[policy_prefix]

                for key, value in policy['attributes'].items():
                    for attr in result:
                        if 'bgp_neighbor' not in attr:
                            if re.search(f"/{key}$", attr):
                                if result[attr] == value:
                                    compliance[f"{key}_compliance"] = 0
```

VISUALIZATION



NANOG 80 HACKATHON

- Aakash Rawal
- Hast Patel
- Mukesh Jaiswal
- Nikhil Gadre
- Swati Niture

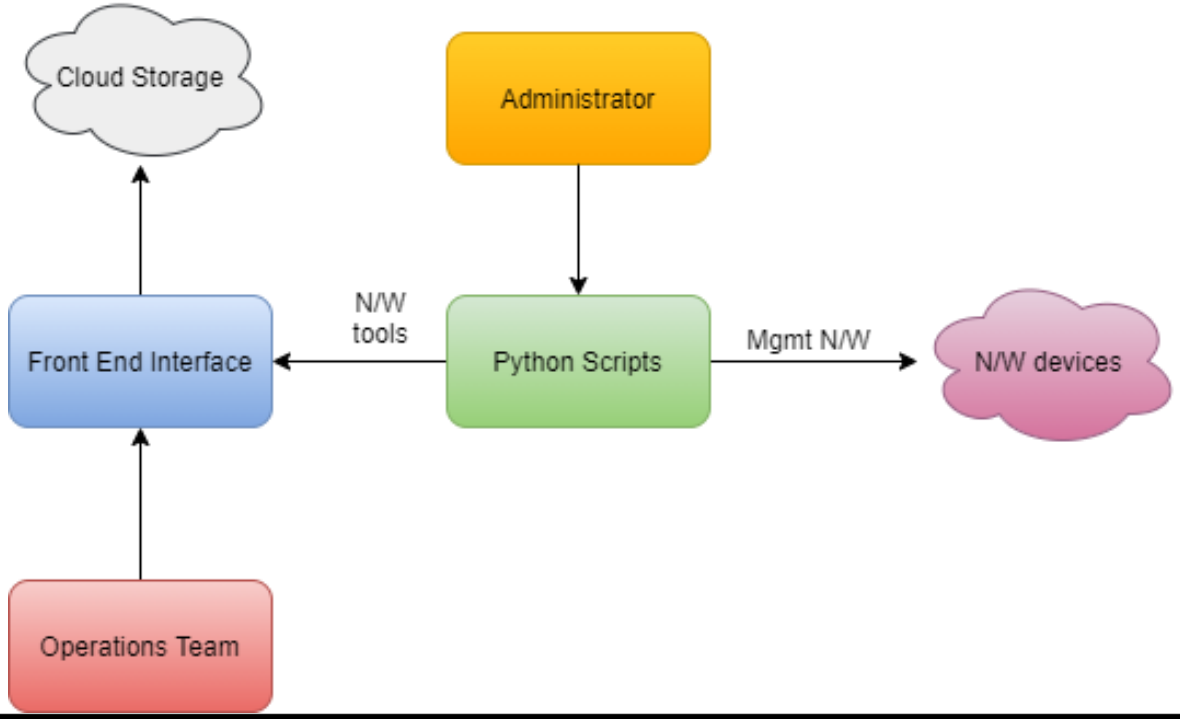
PROPOSAL

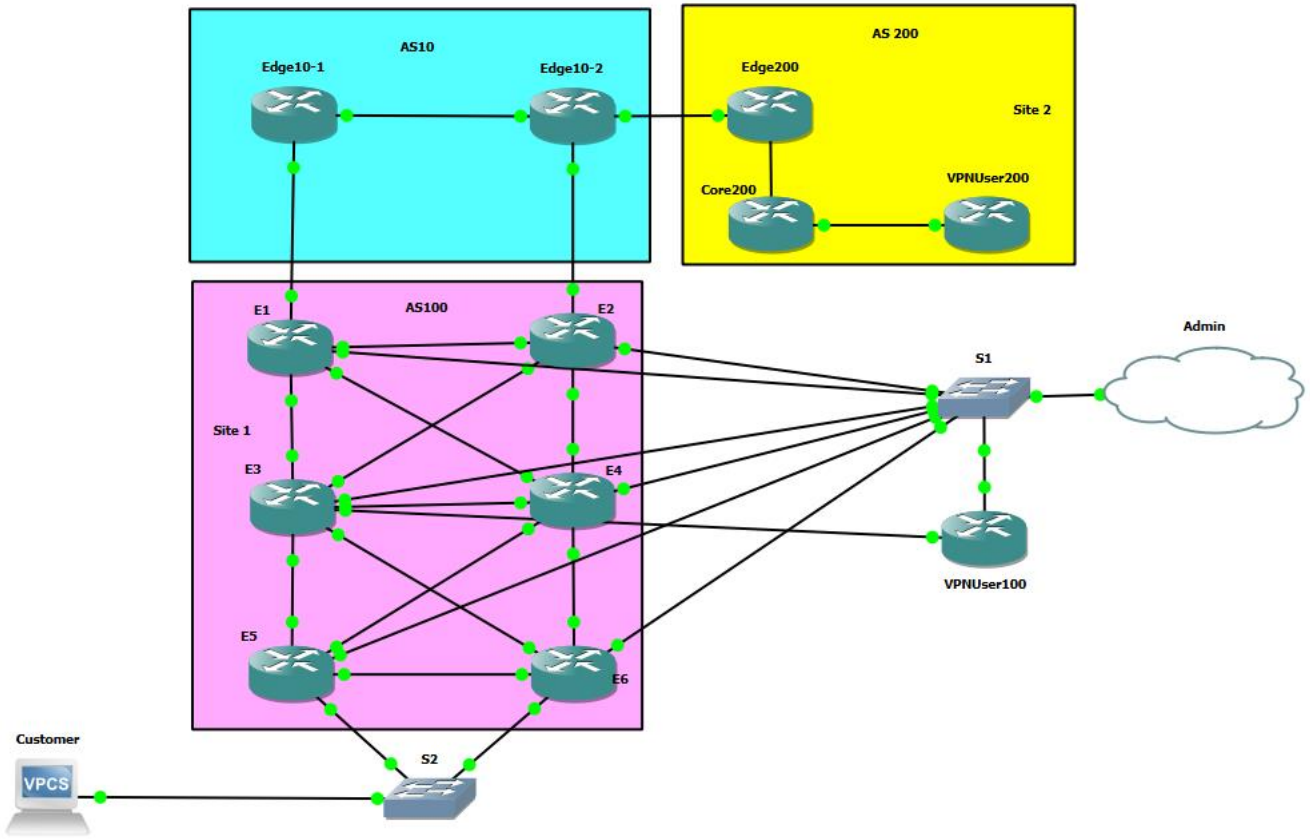
Proof of Concept for a fully automated Tier 2 ISP Data Center – Deployment and Monitoring

What did we achieve ?

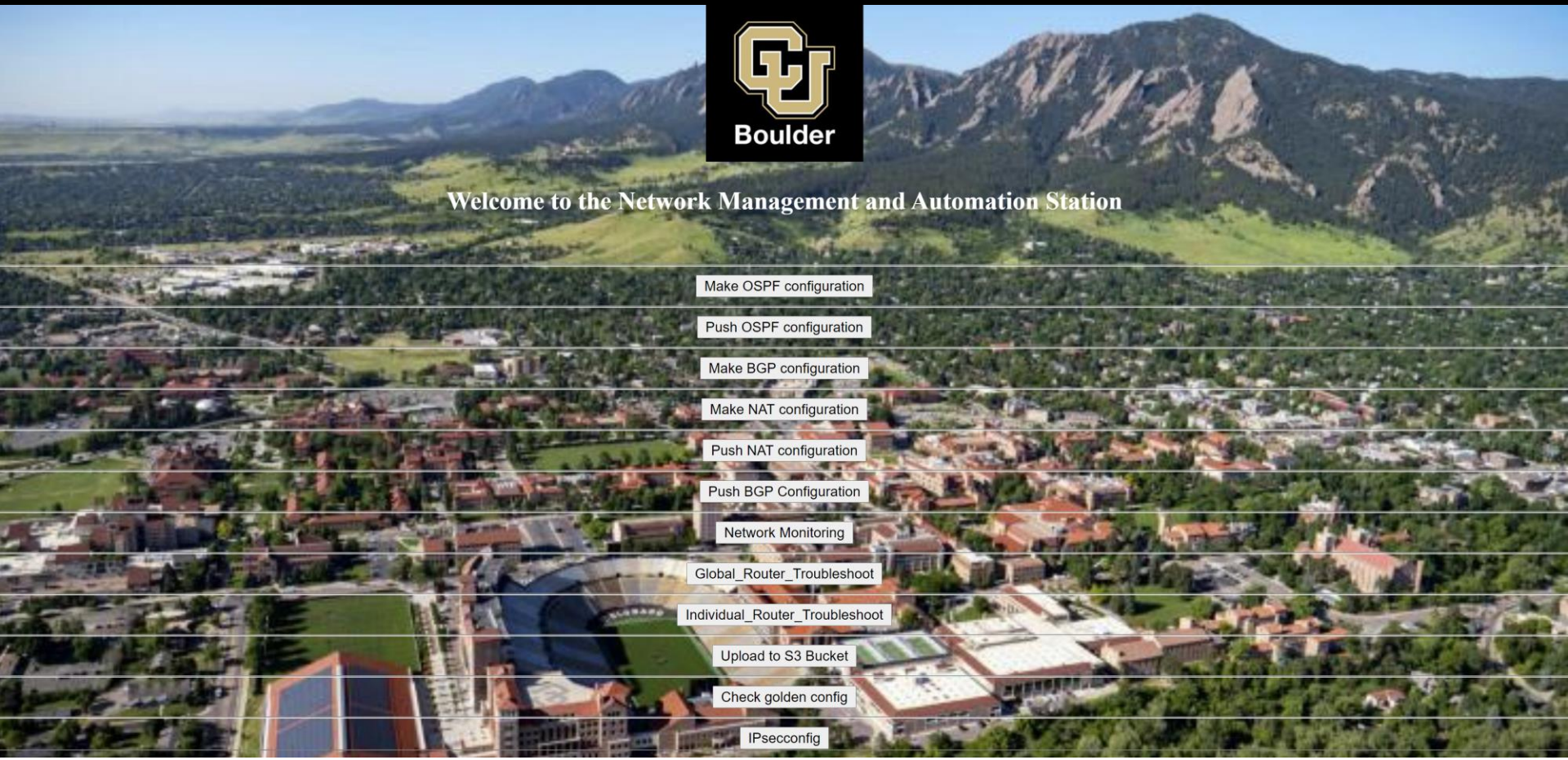
- Centralized control and monitoring of the network
- Cost-effectiveness (reduces OpEx)
- One-click configuration and deployment (saves time)
- Reduction in human errors

HIGH-LEVEL DIAGRAM




AS10
Edge10-1
Edge10-2
AS 200
Edge200
Site 2
Core200
VPNUser200
AS100
E1
E2
Site 1
E3
E4
E5
E6
S1
Admin
VPNUser100
Customer
VPCS
S2

University of Colorado Boulder



University of Colorado Boulder

NETWORK MANAGEMENT AND AUTOMATION SYSTEM (NMAS)

Web App

- Generate Configuration Files
- Push Configuration to Devices
- Monitor Individual Devices and Network
- Compare Running Config to Golden Config
- Backup Configuration to Cloud

GENERATING CONFIG FILES

- Python
- Jinja2 Templates
- Modules
  - Netmiko
  - NAPALM
  - Flask
- Multi-threading

University of Colorado Boulder

```
{% for k in natinfo["Interfaceout"] %}
interface {{ k }}
ip nat outside
{% endfor %}

{% for k in natinfo["Interfacein"] %}
int {{ k }}
ip nat inside
{% endfor %}

{% for k in natinfo["accesslist1"] %}
access-list {{ k }}
{% endfor %}

{% set count=namespace(value=1) %}
{% if natinfo["accesslist1"] | length == 1 %}
ip nat inside source list 1 interface {{natinfo["Interfaceout"]
{% else %}
    {% for k in natinfo["Interfaceout"] %}
ip nat inside source list {{count.value}} interface {{k}} overl
    {% set count.value=count.value +1 %}
    {% endfor %}
{% endif %}
```

```
{% for k in ospfinfo["int"] %}
int {{ k }}
ip address {{ ospfinfo["int"][k] }}
no shut
exit
{% endfor %}
{% for j in range((ospfinfo["network"])|length) %}
router ospf {{ ospfinfo["pid"] }}
network {{ ospfinfo["network"][j] }} area {{ ospfinfo["area"] }
exit
{% endfor %}
```

University of Colorado Boulder

nat_info.txt - Notepad

File Edit Format View Help

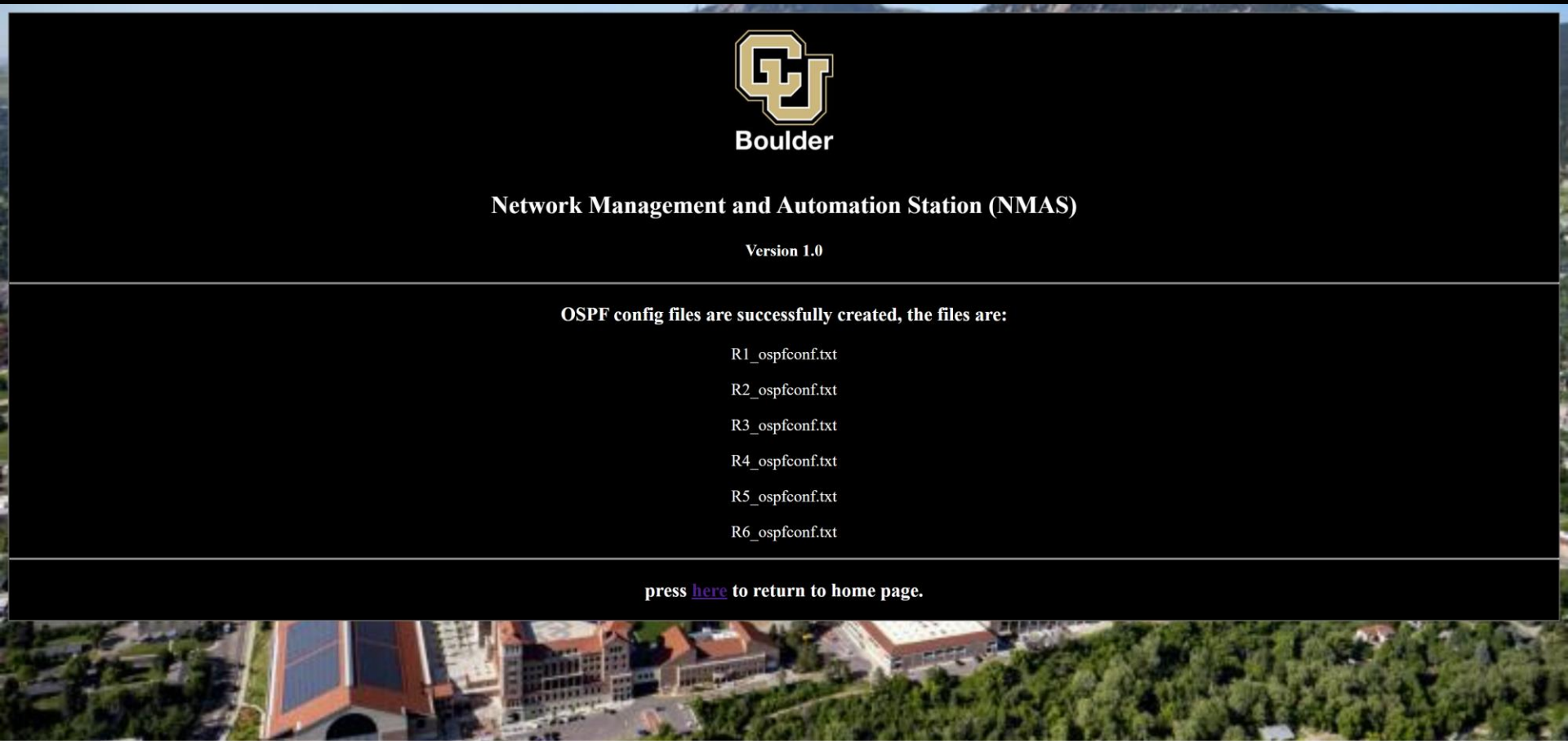
```
{

"R1":
{"Interfaceout":["fa4/1"], "accesslist1": ["1 permit 198.51.0.0
0.0.255.255"], "Interfacein":["fa0/1","fa1/0","f0/0"]},

"R2":
{"Interfaceout":["fa4/1"], "accesslist1": ["1 permit 198.51.0.0
0.0.255.255"], "Interfacein":["fa0/1","fa1/0","f0/0"]},

"R5":
{"Interfaceout":["fa1/1","fa2/0","fa0/0"], "accesslist1": ["1 permit
205.0.0.0 0.0.255.255","2 permit 205.0.0.0 0.0.255.255","3 permit
205.0.0.0 0.0.255.255"], "Interfacein":["fa4/1"]},

"R6":
{"Interfaceout":["fa1/1","fa2/0","fa0/0"], "accesslist1": ["1 permit
205.0.0.0 0.0.255.255","2 permit 205.0.0.0 0.0.255.255","3 permit
205.0.0.0 0.0.255.255"], "Interfacein":["fa4/1"]}

}
```

R6_natconf.txt - Notepad

File Edit Format View Help

```
interface fa1/1
ip nat outside
interface fa2/0
ip nat outside
interface fa0/0
ip nat outside

int fa4/1
ip nat inside

access-list 1 permit 205.0.0.0 0.0.255.255
access-list 2 permit 205.0.0.0 0.0.255.255
access-list 3 permit 205.0.0.0 0.0.255.255

ip nat inside source list 1 interface fa1/1 overload
ip nat inside source list 2 interface fa2/0 overload
ip nat inside source list 3 interface fa0/0 overload
```

University of Colorado Boulder



Boulder

Network Management and Automation Station (NMAS)

Version 1.0

OSPF config files are successfully created, the files are:

R1_ospfconf.txt

R2_ospfconf.txt

R3_ospfconf.txt

R4_ospfconf.txt

R5_ospfconf.txt

R6_ospfconf.txt

press here to return to home page.

PUSH CONFIGS TO DEVICES

Netmiko, Napalm

- Interface Configs
- OSPF
- NAT
- BGP
- IPSec
- 6to4 Tunneling

University of Colorado Boulder

Boulder

Network Management and Automation Station (NMAS)

Version 1.0

OSPF files are successfully loaded into routers, press here to return to home page.

University of Colorado Boulder

Boulder

Local tunnel ep	Remote tunnel ep	No. of encrypted pkts	No. of decrypted pkts	Tunnel status
70.70.70.1	60.0.0.2	8	8	ACTIVE

Type escape sequence to abort. Sending 5, 100-byte ICMP Echos to 2001::3, timeout is 2 seconds: Packet sent with a source address of 2001::1 !!!!! Success rate is 100 percent (5/5), round-trip min/avg/max = 200/300/504 ms

MONITOR INDIVIDUAL DEVICES

- Device Interfaces
  - IP / Status
- OSPF Neighborship
- BGP Neighborship
- View Running Config
- Show Commands

University of Colorado Boulder

Boulder

Network Management and Automation Station (NMAS)

Version 1.0

Interfaces IPs

BGP data

OSPF data

Running configs

press here to return to home page.

University of Colorado Boulder

Boulder

Network Management and Automation Station (NMAS)

Version 1.0

Router	Interface	IP
R1	FastEthernet0/0	198.51.101.1
R1	FastEthernet0/1	198.51.103.1
R1	FastEthernet1/0	198.51.102.1
R1	FastEthernet1/1	unassigned
R1	FastEthernet4/0	192.168.100.1
R1	FastEthernet4/1	200.0.0.1
R2	FastEthernet0/0	198.51.101.2
R2	FastEthernet0/1	198.51.105.2
R2	FastEthernet1/0	198.51.104.2

University of Colorado Boulder

Boulder

Network Management and Automation Station (NMAS)

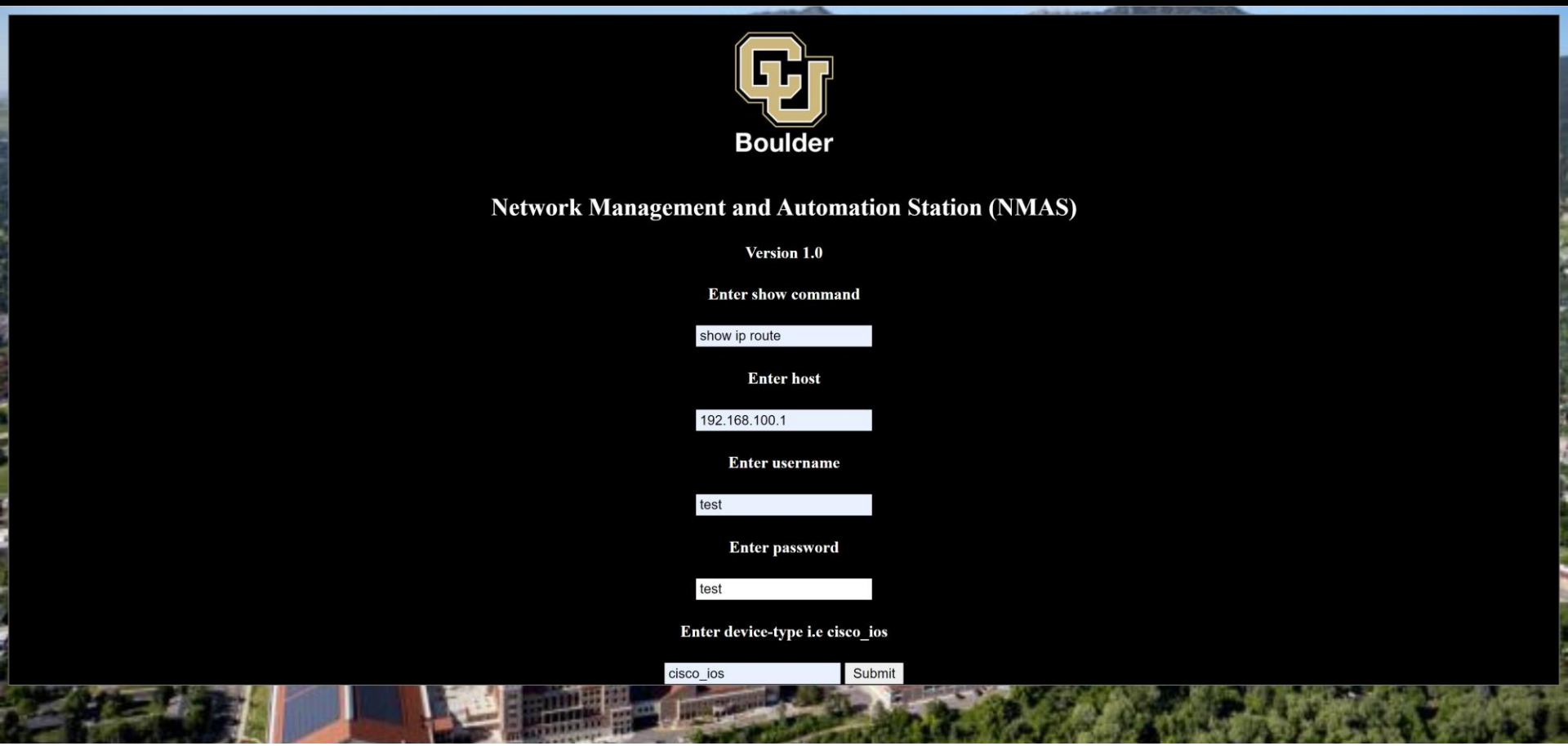
Version 1.0

Enter show command

show ip bgp | Submit

- show running-config
- debug ip bgp
- show ip interface brief
- show ip bgp
- show ip route
- show running-config

University of Colorado Boulder



COMPARE RUNNING CONFIG TO GOLDEN CONFIG

- Backup Running Config
- Compare
  - Latest Backed Up Running Config
  - Golden Config
- Show Difference

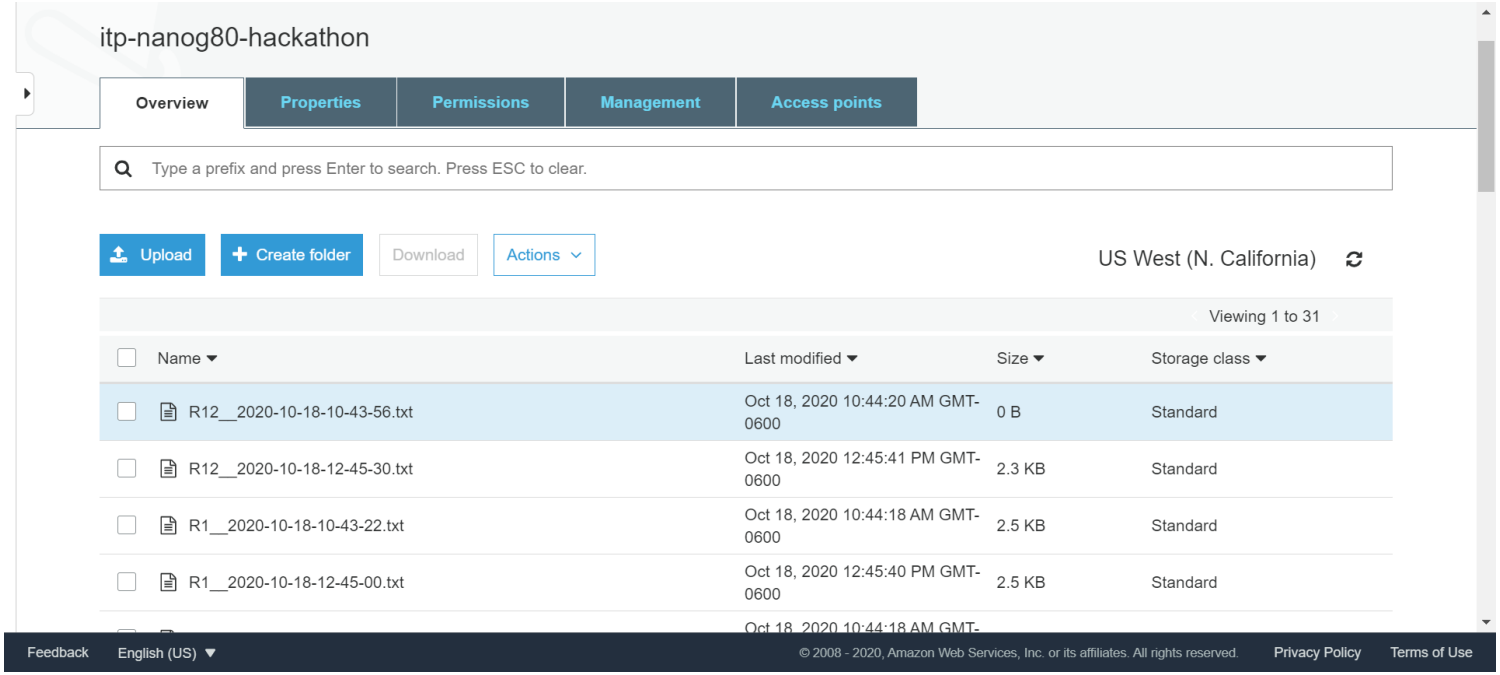
University of Colorado Boulder

Boulder

['Difference found for R1', 'Difference found for R2', 'Difference found for R3', 'Difference not found for R4', 'Difference found for R5', 'Difference found for R6', 'Difference not found for R12']

University of Colorado Boulder

BACKUP CONFIG FILES TO CLOUD



FUTURE ENHANCEMENTS

- Zero Touch Provisioning (ZTP) – Management Network
- Automated Device Backups – Cron Job
- Security
  - Firewall
  - Access-lists
- Redundancy
  - For Default Gateway

THANK YOU

A Simple, Automated and Easy to Deploy SDN Solution

Roni Mukherjee

MS, Telecommunications 2019-21, CU Boulder

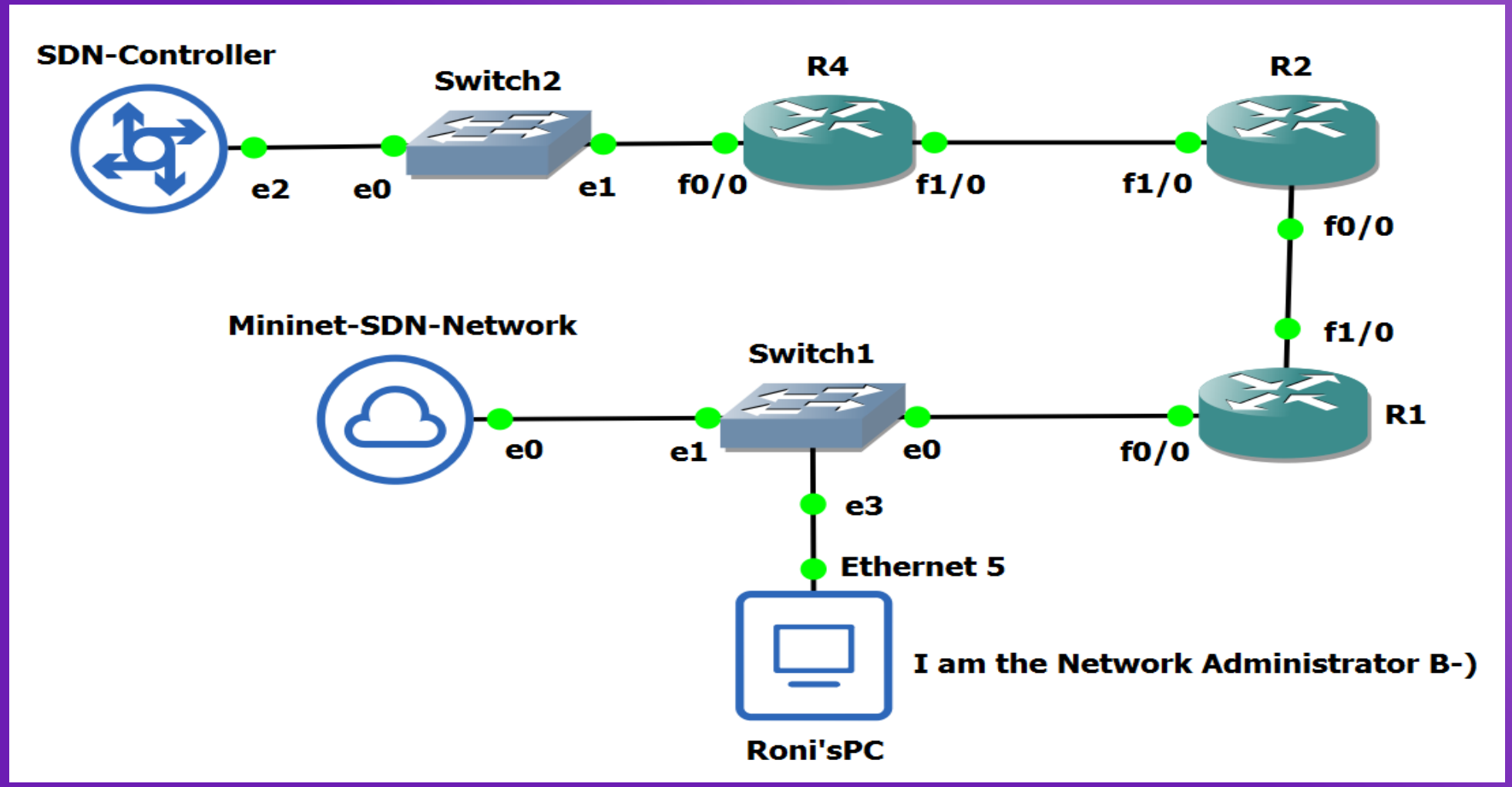
Why SDN?

- Reduced CapEx: Simple device – move control away from devices
- Reduced OpEx: Simplified network monitoring – automation
- Build your solution
- Faster troubleshooting

Hybrid Network

- Combination of SDN and Traditional Routing and Switching Networks
- No need to change traditional network
- SDN network is spun up to join the network
- Based on the implementation, SDN Controller can be anywhere in the network
  - Only, IP connectivity is required to establish a TCP/TLS connection between the SDN switch (s) and the controller (s)

Hybrid Network – Proof of Concept



Automated SDN Deployment

- Mininet boots up, gets DHCP address, routing enabled, controller configured – Automated

```
PS C:\MyCanvas\NANOG80_hackathon\submission> python .\sdn_network_spinup.py
********* The IP leased to Mininet VM is: 192.168.100.2


********* Routing enabled on: ios_r1
********* Routing enabled on: ios_r2
********* Routing enabled on: ios_r4


******** OpenFlow version changed to --> 1.3

******** Controller configured!
******** Controller connected!
PS C:\MyCanvas\NANOG80_hackathon\submission>
```

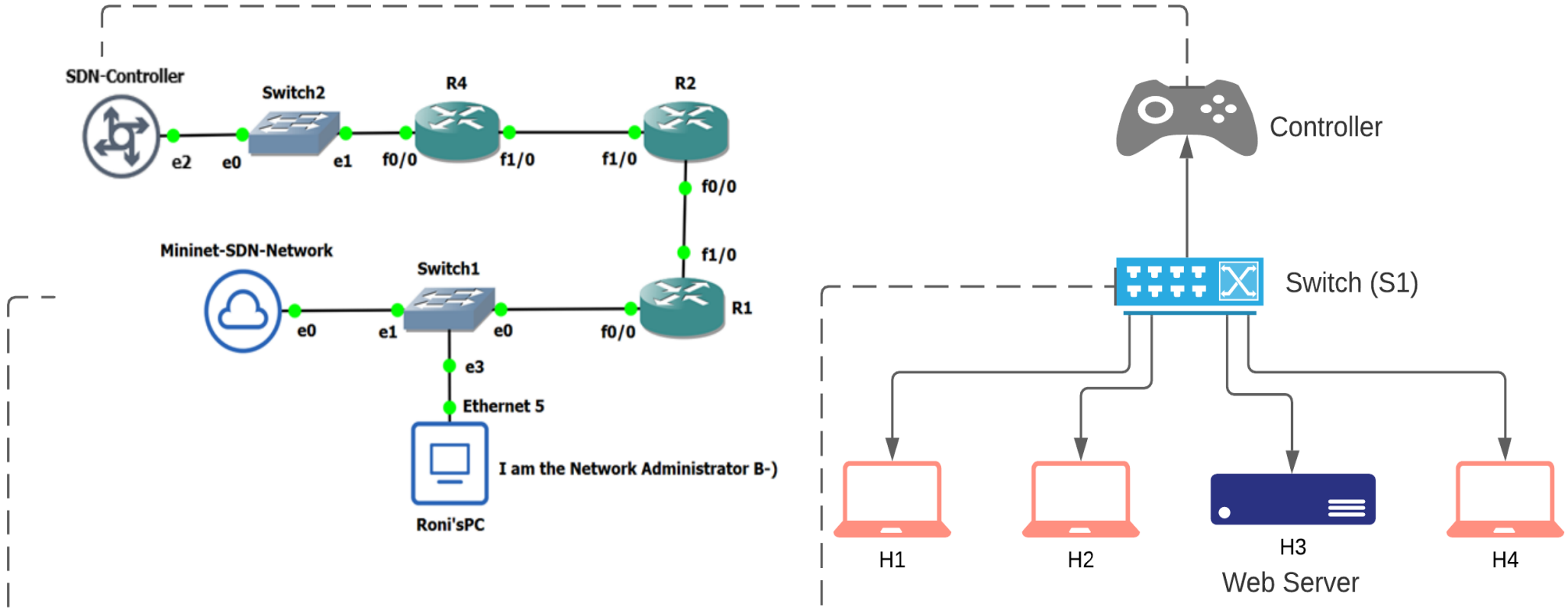
Advantages of the proposed solution

- Easy to deploy – Automated script (boot up/routing/monitoring)
- SDN – Centralized control, global view, reduced CapEx and OpEx
- API – Use of APIs make it faster and human readable
- No need to change existing network
- Reuse SDN multipurpose simple devices
- GUI: Greatly increased troubleshooting efficiency

SDN GUI – Reusable/Extendable Global View

- Purpose-built for the proposed network
- Web Application Console logs
- Web GUI
- Real-time graphical plots of the network (auto-refresh every 5 seconds)
- Greatly reduced troubleshooting steps – match on 40 fields and create own flow

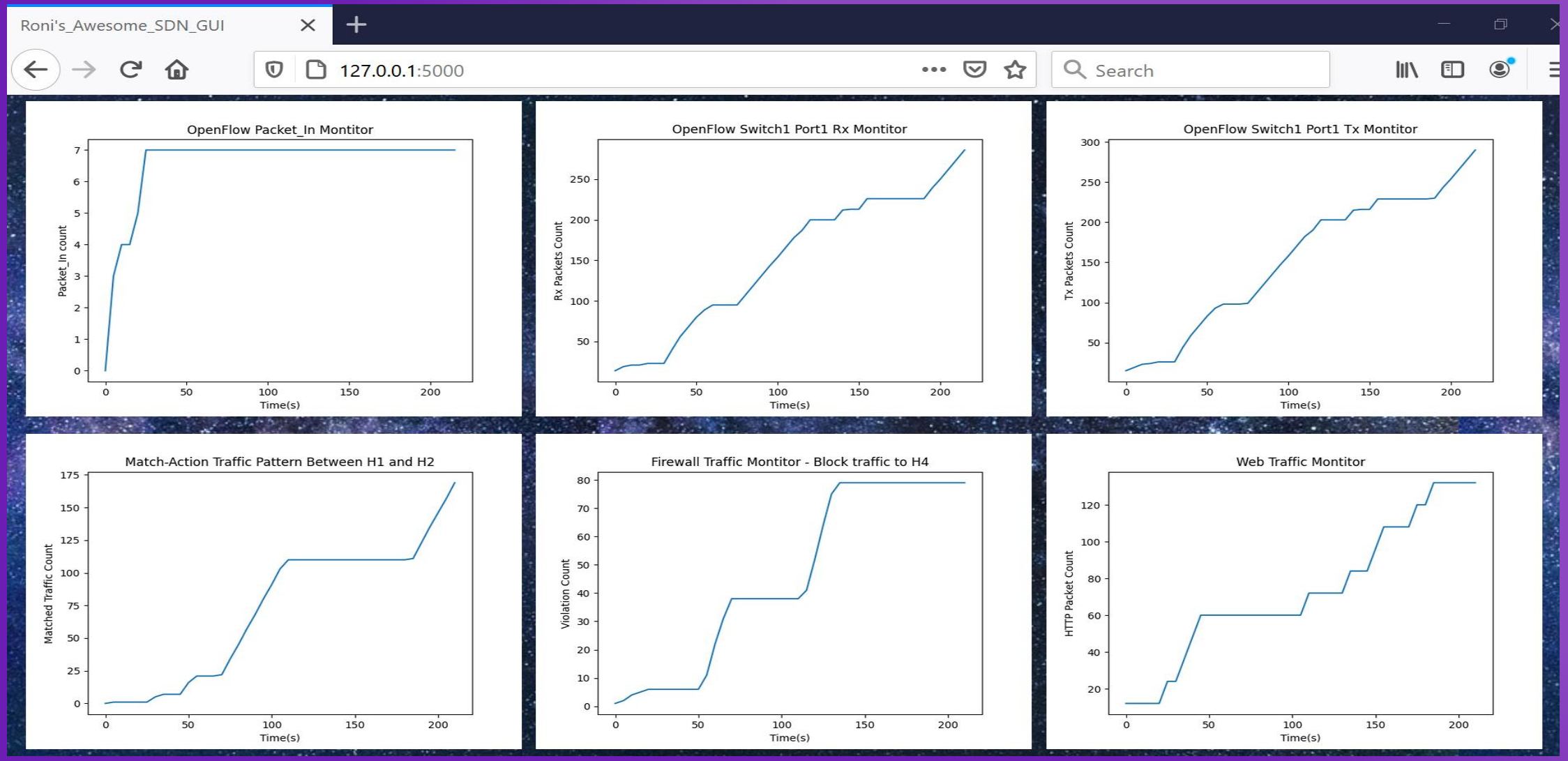
SDN Topology



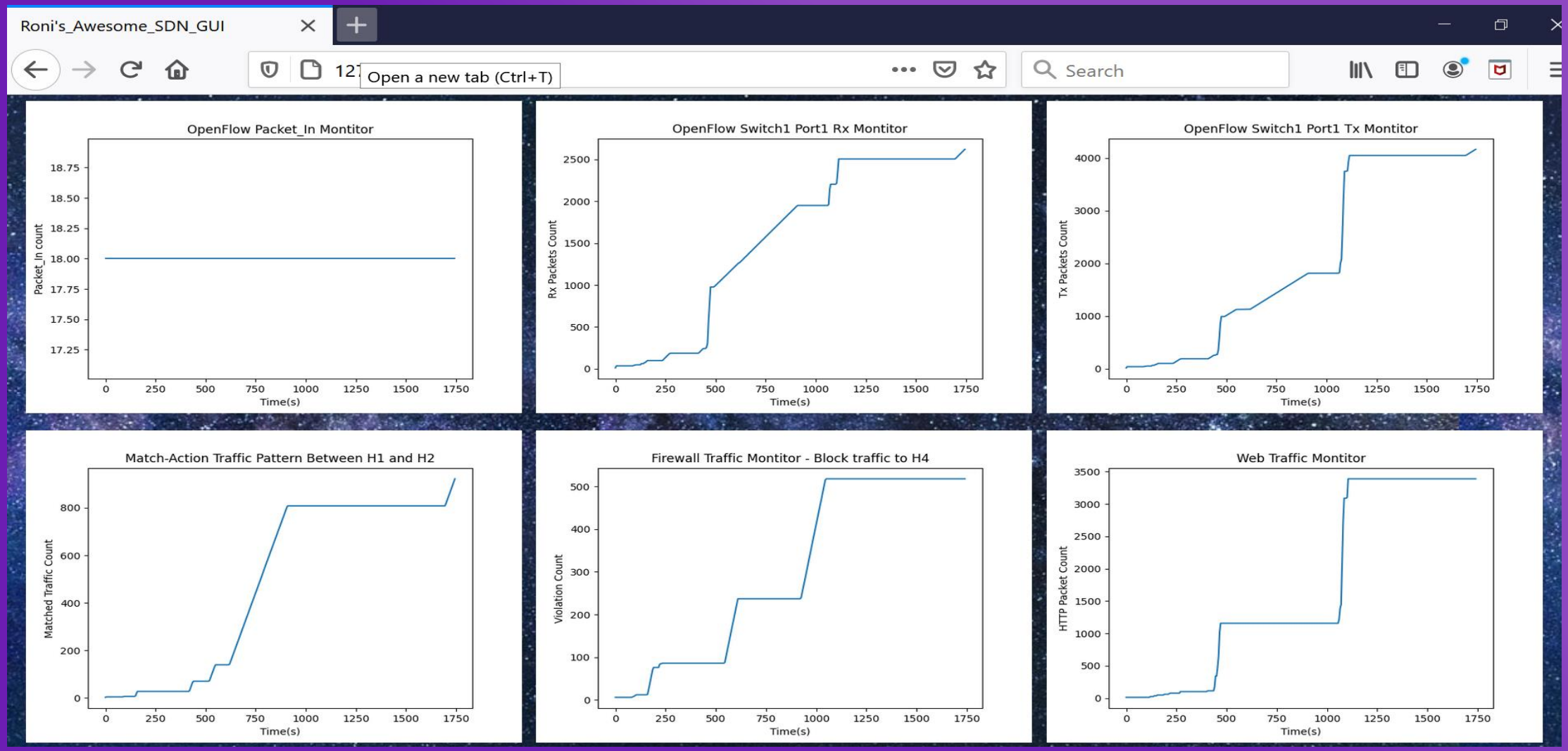
SDN GUI – Console Logs

```
PS C:\MyCanvas\NANOG80_hackathon\submission> python .\sdn_gui.py
******** SDN GUI running
 * Serving Flask app "sdn_gui" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
******** SDN GUI running
 * Debugger is active!
 * Debugger PIN: 123-884-149
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
Number of Packet_INs to controller in this iteration is: 3
Number of Rx packets of port1 of switch1 in this iteration is: 31
Number of Tx packets of port1 of switch1 in this iteration is: 31
Number of matched traffic pattern sourced from H1 and destined to H2 in this iteration is: 0
Number of violation count for the block traffic flow rule to H4 in this iteration is: 1
Number of HTTP packets at switch1 in this iteration is: 24
127.0.0.1 - - [17/Oct/2020 23:15:10] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [17/Oct/2020 23:15:10] "GET /static/img/1.jpg HTTP/1.1" 304 -
Number of Packet_INs to controller in this iteration is: 4
Number of Rx packets of port1 of switch1 in this iteration is: 33
Number of Tx packets of port1 of switch1 in this iteration is: 35
Number of matched traffic pattern sourced from H1 and destined to H2 in this iteration is: 1
Number of violation count for the block traffic flow rule to H4 in this iteration is: 3
Number of HTTP packets at switch1 in this iteration is: 24
127.0.0.1 - - [17/Oct/2020 23:15:22] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [17/Oct/2020 23:15:22] "GET /static/img/1.jpg HTTP/1.1" 304 -
```

SDN GUI – Web GUI

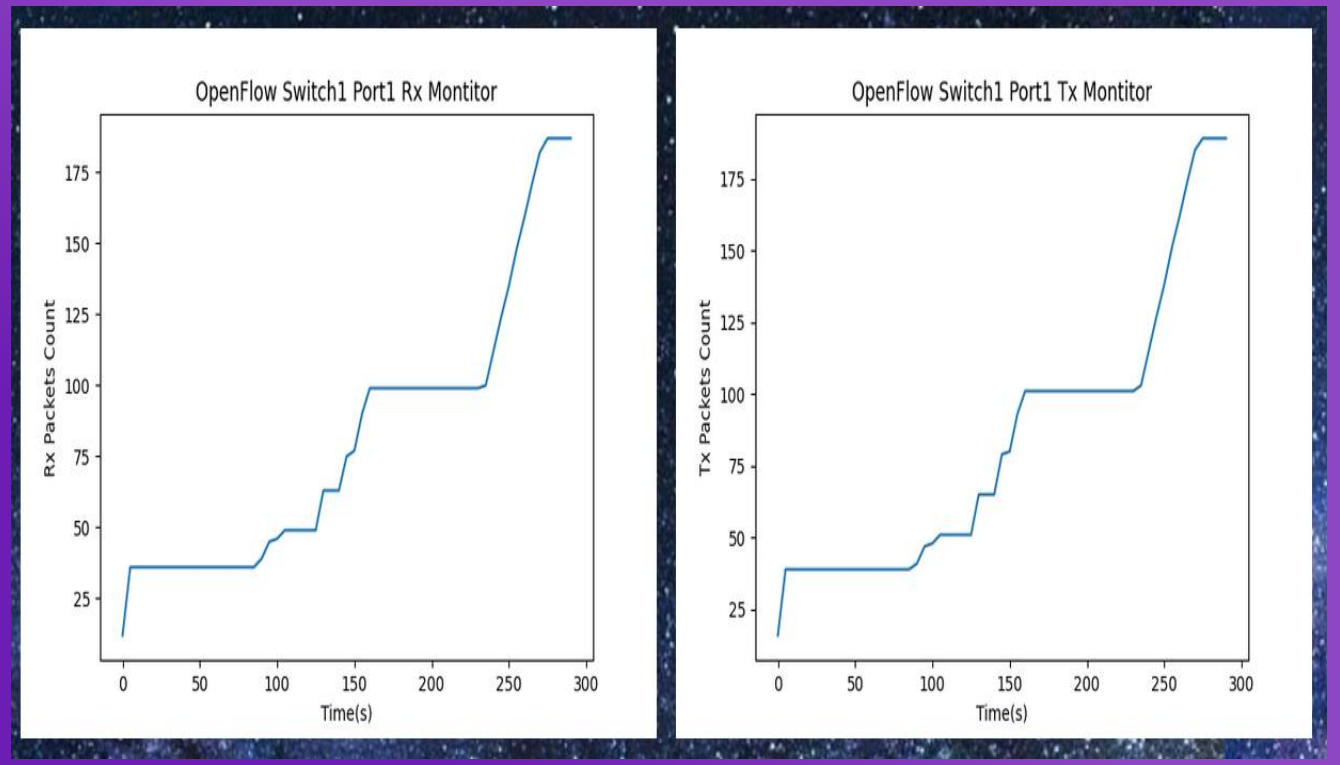


SDN GUI – Web GUI



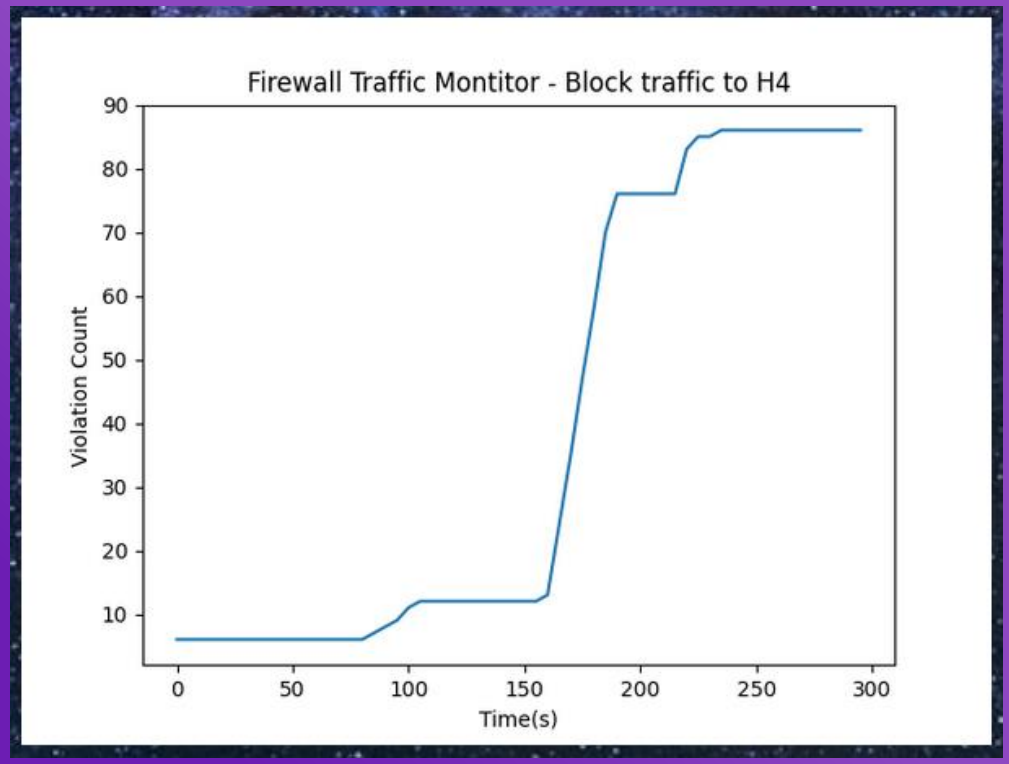
SDN Web GUI – Switch Tx/Rx Monitor

- Tx/Rx traffic has a similar pattern.
- If there is a significant difference between Tx/Rx traffic, captured on graph and an email is sent to the administrator for immediate action.
- Graph is plotted against real-time data and refreshed every 5 seconds.



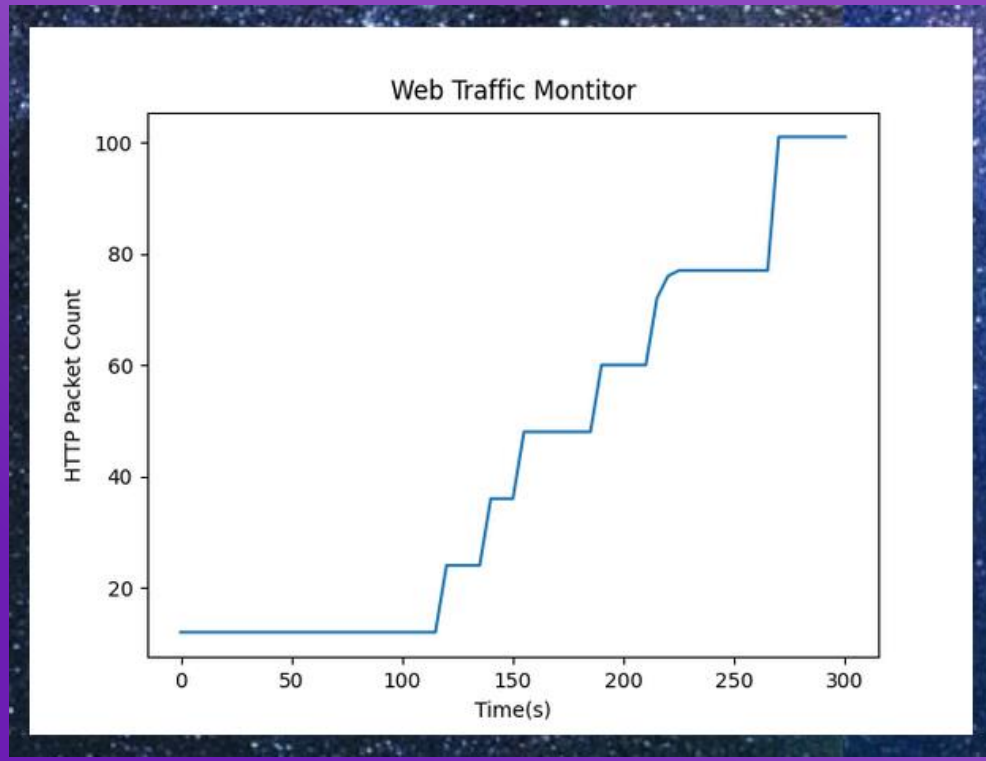
SDN Web GUI – Firewall Monitor

- Custom firewall rule is created.
- If there is a significant increase in violation count, captured on graph and email is sent to administrator for immediate action.
- Graph is plotted against real-time data and refreshed every 5 seconds.



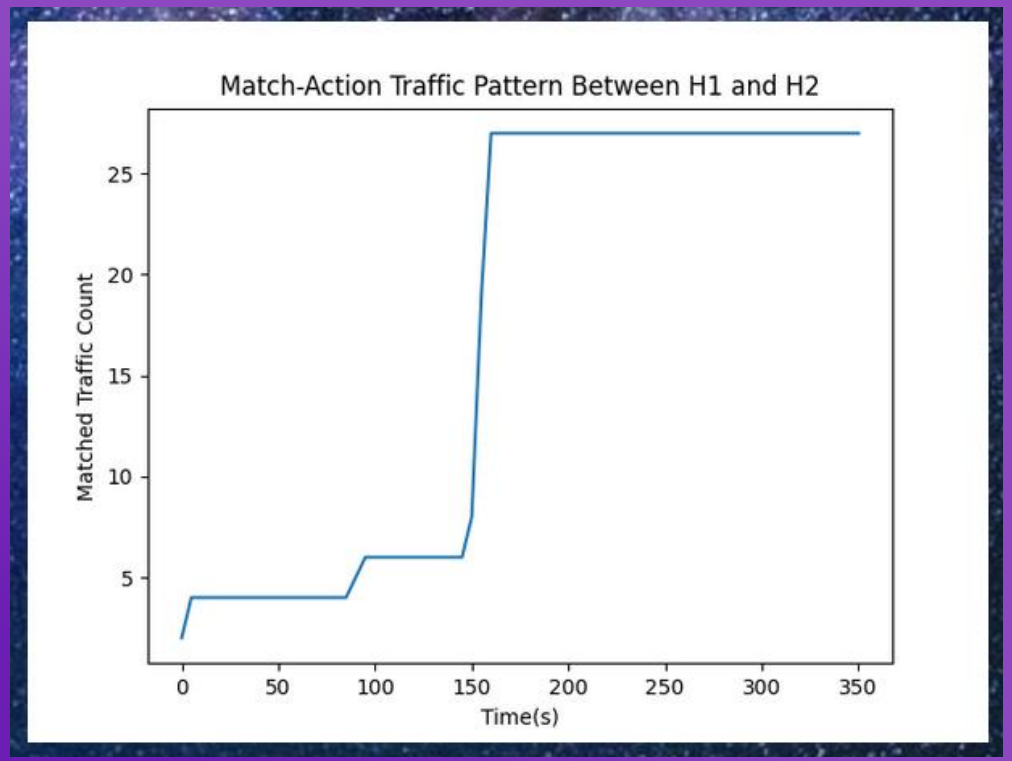
SDN Web GUI – Web Traffic Monitor

- One of the host is web server and serving HTTP requests.
- This is a critical service.
- If there is a significant change in traffic pattern, captured on graph and an email is sent to the administrator for immediate action.
- Graph is plotted against real-time data and refreshed every 5 seconds.



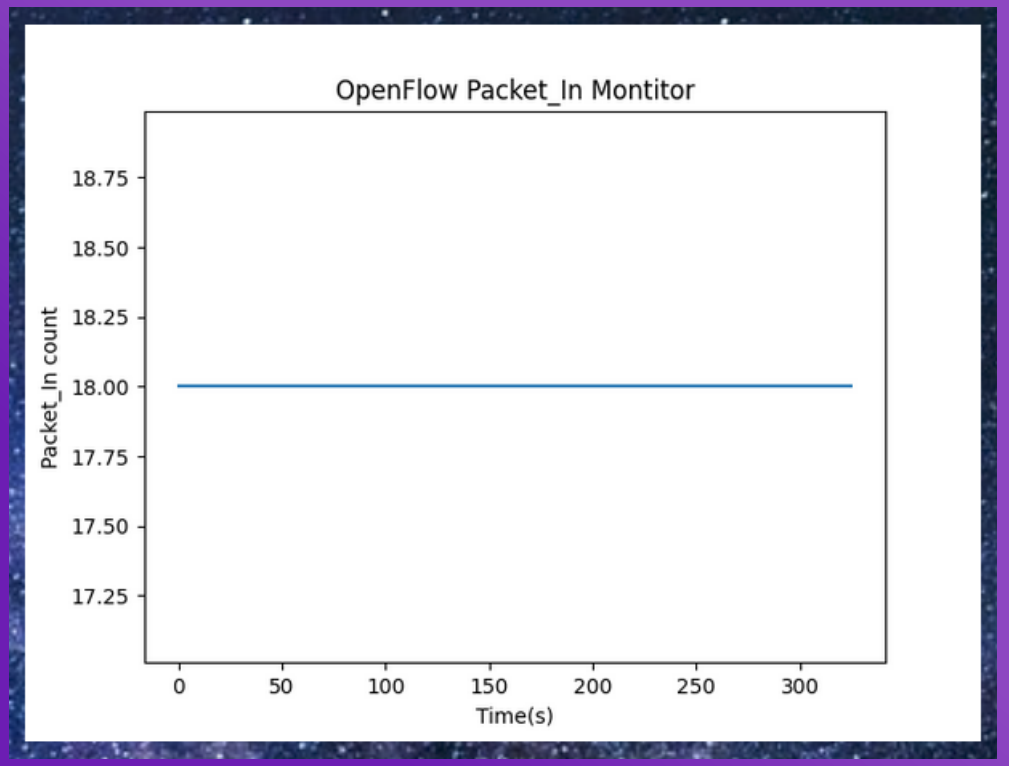
SDN Web GUI – Pattern Traffic Monitor

- Custom match is created for traffic pattern, e.g. traffic sourced from MAC address of H1 and destined to MAC address of H2.
- This is considered as critical service.
- If there is a significant change in traffic pattern, captured on graph and an email is sent to the administrator for immediate action.
- Graph is plotted against real-time data and refreshed every 5 seconds.

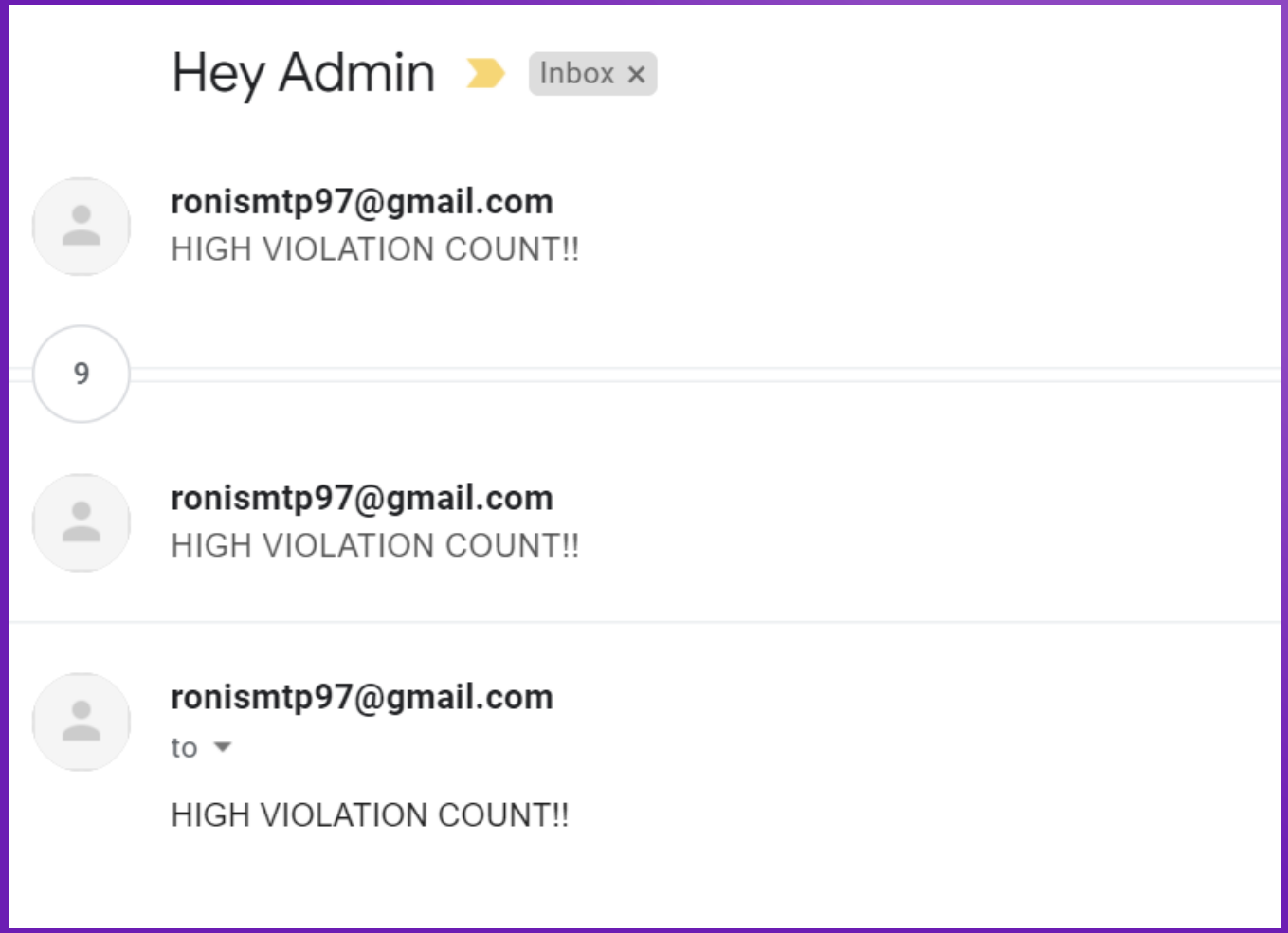


SDN Web GUI – OpenFlow Packet_In Monitor

- Traffic from switch to controller.
- This is the traffic that matches table-miss.
- Graph is plotted against real-time data and refreshed every 5 seconds.



SDN Web GUI – Violation email



Tools used

- GNS3
- Cisco IOS routers
- Mininet
- Ryu SDN controller
- Ubuntu VMs
- VirtualBox
- Flask
- HTML/CSS
- Netmiko
- NETCONF
- Python
- Wireshark

Conclusion

- Custom flow rules using any match criteria (from 40 different fields)
- Graphical representation – significantly increased troubleshooting efficiency
- Hybrid Network
  - Reduced CapEx: Simple device, reusability
  - Reduced OpEx: Significantly improved network monitoring and troubleshooting through automation
- Automated deployment of the Hybrid-SDN solution

GitHub

- https://github.com/rm-viable/NANOG80-hackathon

Questions