Traffic Exceptions

Mat Wood

Network Automation Engineer @ Facebook

Traditional Routing

- Routing is prescriptive of pre-defined desired topology
  - Protocols and costs define desired traffic flow
  - BGP Policy expresses business logic as reachability
  - TE adds constraints to path selection
- Reactive scenarios focus around link failure
  - Solving: How to retain connectivity & capacity
  - IGP reconvergence of CSPF
  - LSP signaled over available capacity
  - Try to get back to desired topology

What if we could react to individual traffic flows?

Handling Traffic Exceptions

- Traffic Triggering
  - Monitor traffic flows and flag based on desired characteristics
- Network Config
  - Supports the desired outcome of triggered flows
  - E.g. Redirect traffic to desired network segments
- Traffic Influence
  - Mechanism to connect the triggering to the network data plane

Wait, this looks familiar…

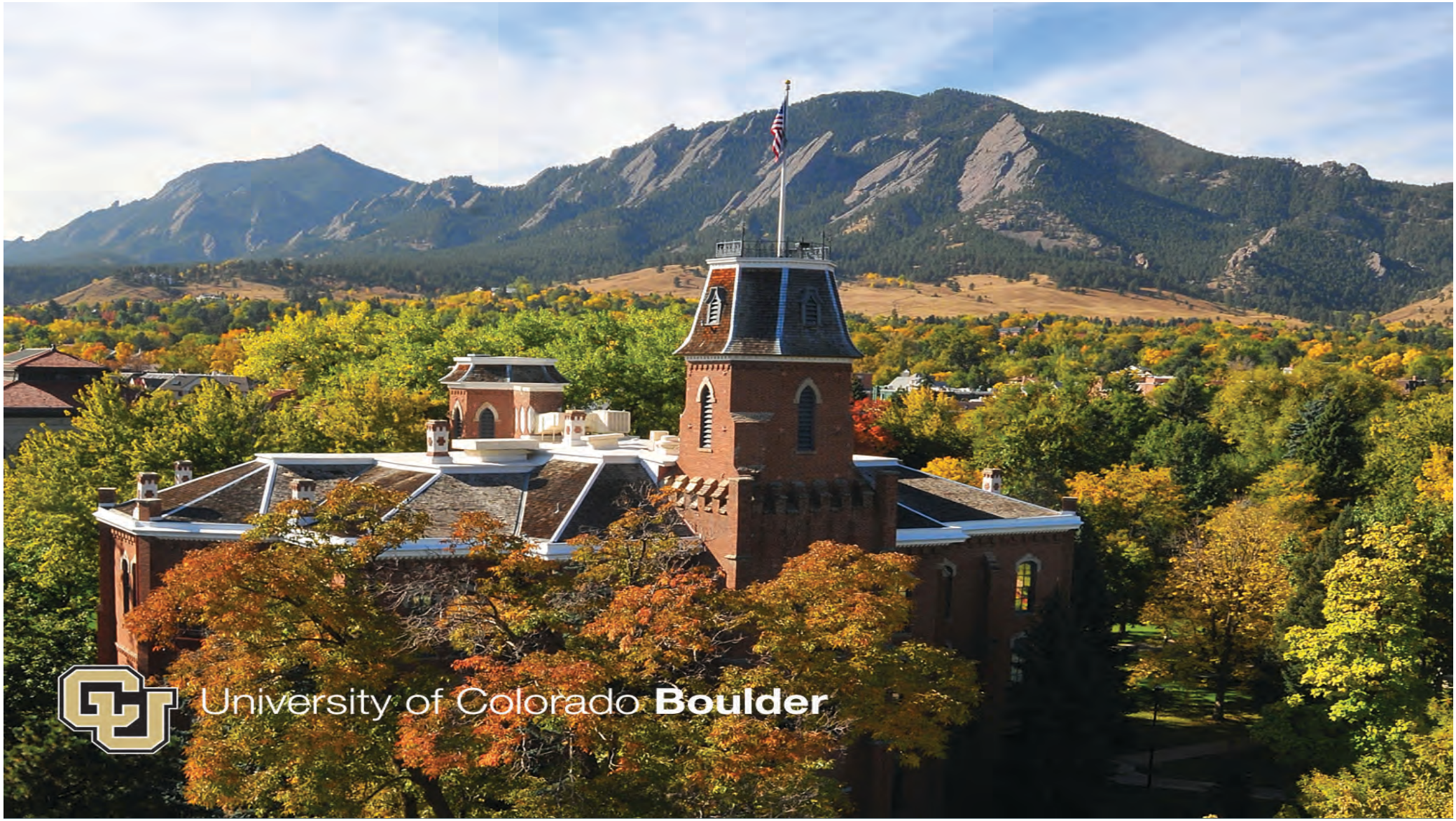
DDoS Mitigation

- Traffic Triggering
  - Detect attacks from rules/machine learning
  - Customer phone call
- Network Config
  - BGP with pre-defined policy & communities to drop traffic
- Traffic Influence
  - Remotely-Triggered Black Hole (RTBH)
  - BGP FlowSpec
    - Remote programming of Drop/Rate-limit for flows

We can do so much more!

Reactive Network

- Traffic Triggering
  - Malicious L7 API requests
  - TCP Retransmits, further analysis
  - TTL as source-defined priority
    - Higher TTL implies "scenic route" 😂
  - TCP options encoding of a BGP Community?
    - Intent Based Networking™
- Network Config
  - Network segment(s) attract traffic via BGP FlowSpec
- Traffic Influence
  - ExaBGP provides an API to advertise FlowSpec rules


University of Colorado Boulder

CUBUFFS

- Akshay Broota
- Parth Adroja
- Sadhvi Ravishankar
- Sahana Satyanarayana
- Tavleen Kaur

Key Points

Exception Traffic

- ExaBGP
- DSCP values
- Traffic shaping

Failover

- Route manipulation

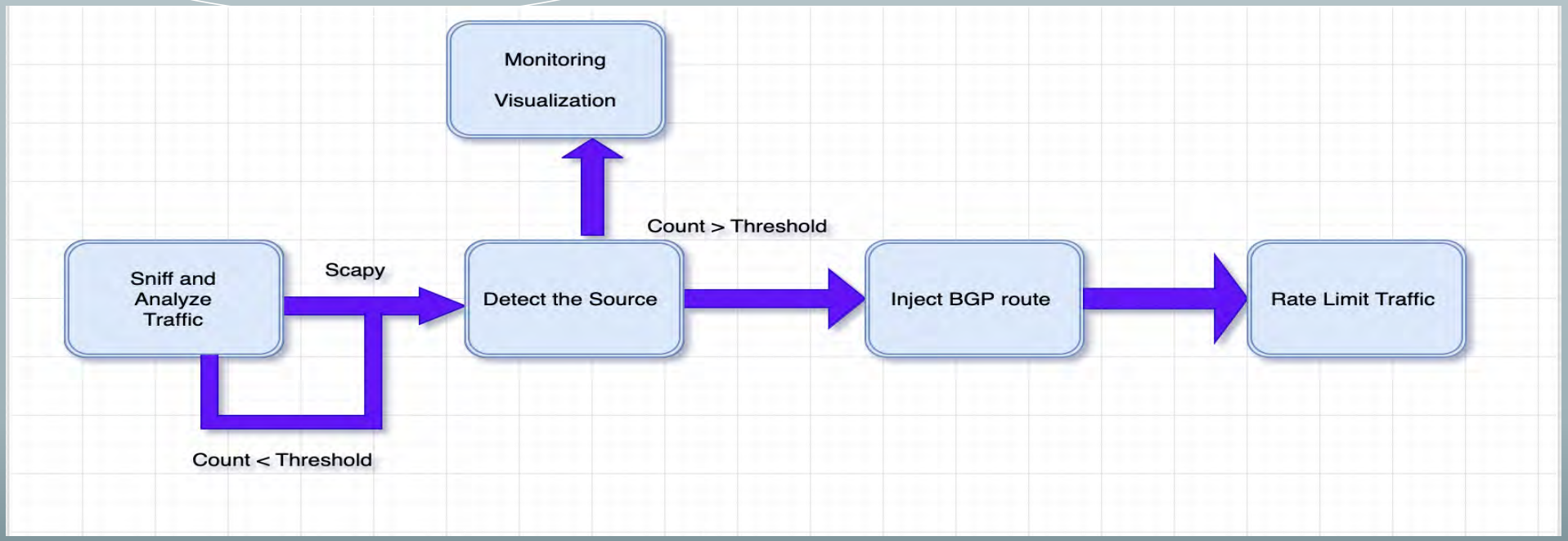
Monitoring

- Twilio API
- Freshdesk

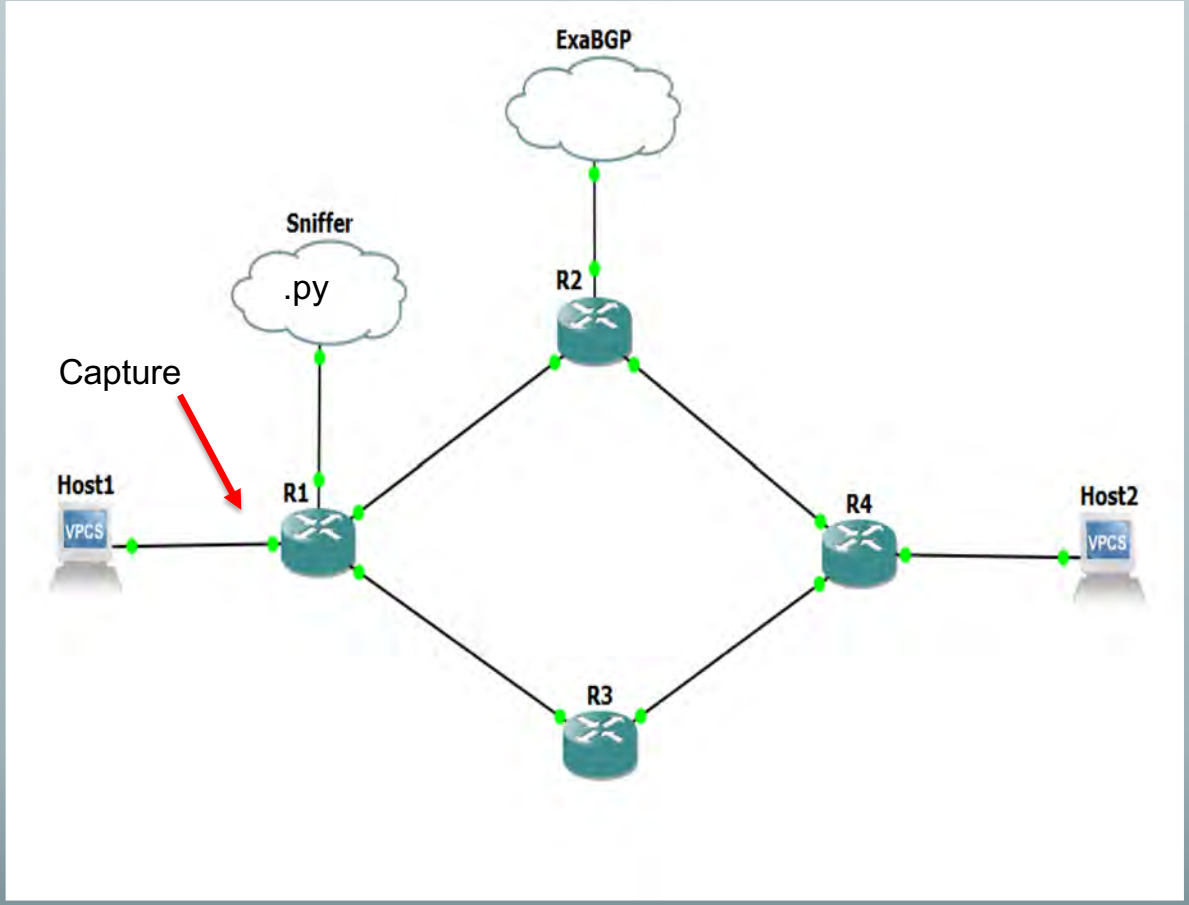
Visualization

- Grafana
- Matplotlib

WORKFLOW

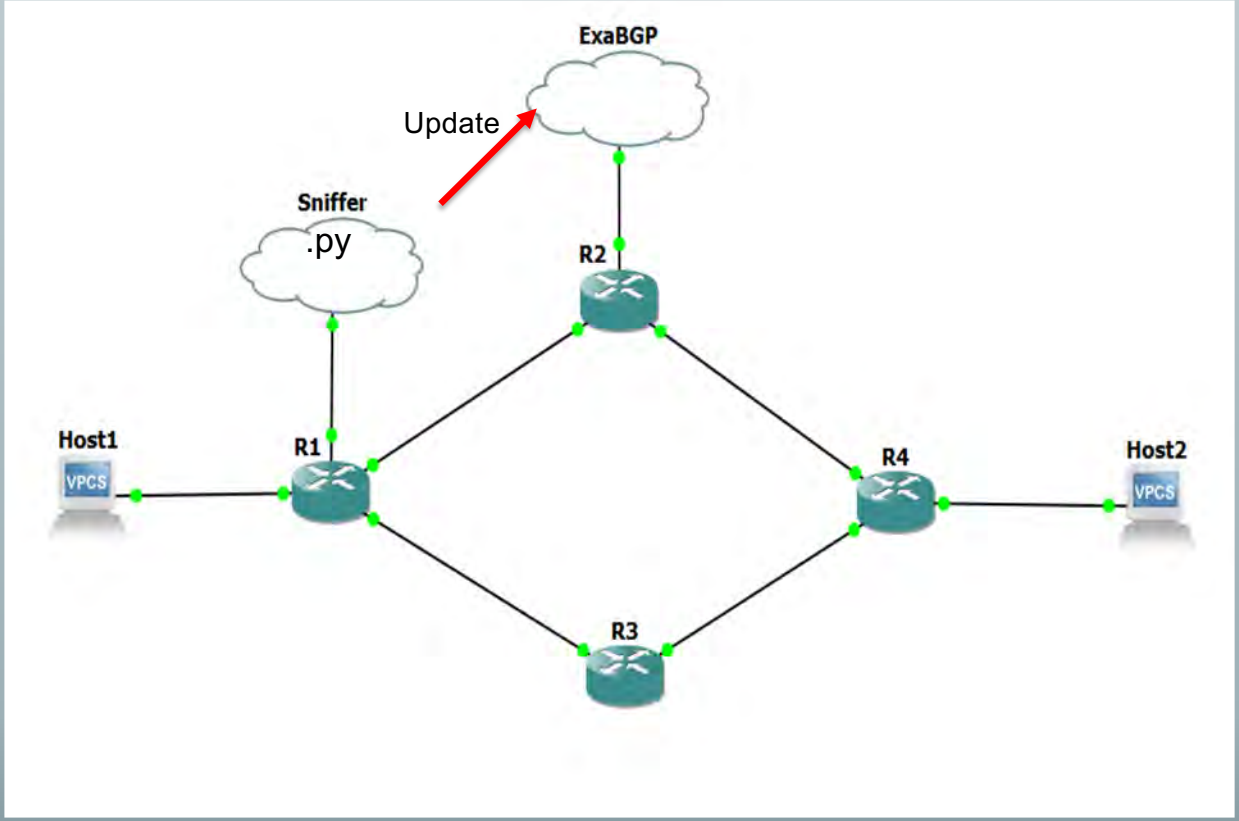

Monitoring
Visualization
Count > Threshold
Sniff and Analyze Traffic
Scapy
Detect the Source
Inject BGP route
Rate Limit Traffic
Count < Threshold

Topology




```
def feature_req_verify():
    packets = rdpcap("capture.pcapng")
    for packet in packets:
        packet.show()
        if packet.haslayer('TCP'):
            if packet['TCP'].dport == 80:
                if packet['IPv6'].src not in source_exception_time:
                    source_exception_time[packet['IPv6'].src] = {}
                time_in_1_min = float("%0.0f" %(packet.time/60))
                if time_in_1_min not in source_exception_time[packet['IPv6'].src]:
                    source_exception_time[packet['IPv6'].src][time_in_1_min] = 0
                source_exception_time[packet['IPv6'].src][time_in_1_min] += 1

    print source_exception_time
    for source in source_exception_time:
        for time_min in source_exception_time[source]:
            if source_exception_time[source][time_min] > thresh:
                print source,time_min
                push_config()
                twilio(source)
                freshdesk(source)
```

ExaBGP




```
process http-api {
    run /usr/bin/python3 /home/secure_rest.py;
    encoder json;
}

neighbor 3001:2:e10a::2 {
    router-id 10.10.10.10;
    local-address 3001:2:e10a::10;
    local-as 65010;
    peer-as 65000;

    announce {
        ipv6 {
            unicast 3001:99:a::/64 next-hop self;
        }
    }
     flow {
         route TEST {
             match {
                 source 3001:99:a::10/128;
                 destination 3001:99:b::10/128;
             }
             then {
                 redirect 6:302;
                 mark 10;
             }
         }
    }
}
```

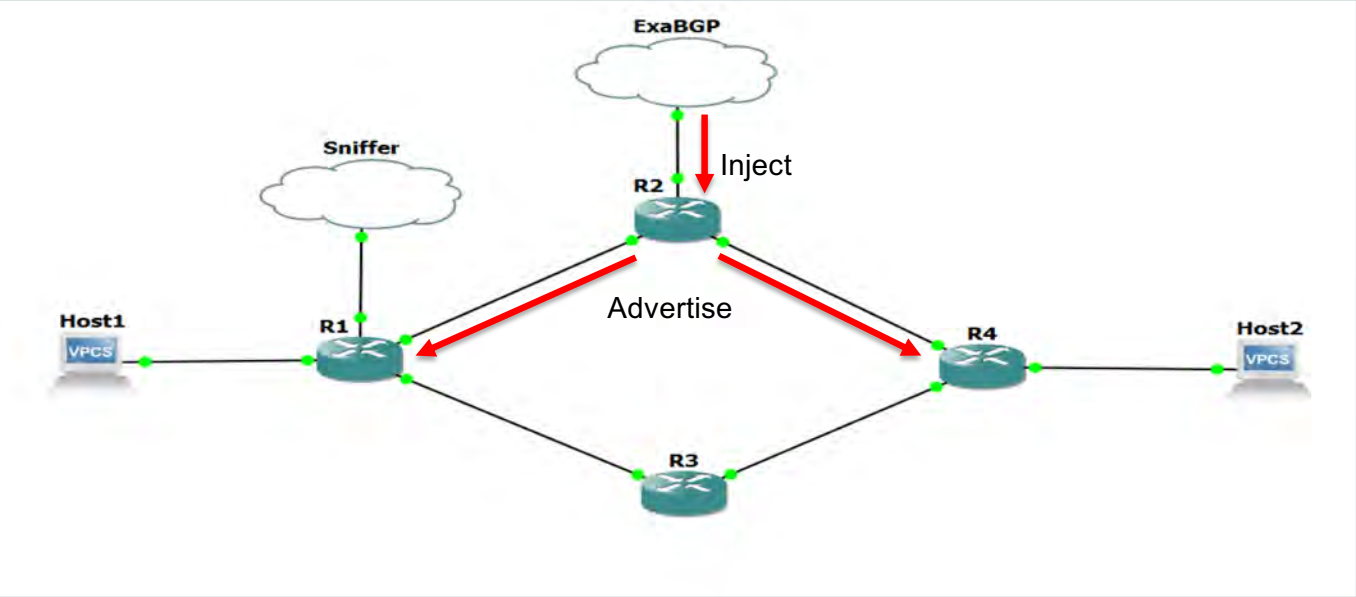
Secure API

```python
app = Flask(__name__)
api = Api(app, prefix="")
auth = HTTPBasicAuth()


users = {
    "parth": "secret@123"
}

@auth.verify_password
def verify(username, password):
    if not (username and password):
        return False
    return True


class PrivateResource(Resource):
    @auth.login_required
    def get(self):
        command = request.form["command"]
        stdout.write(f"{command}\n")
        stdout.flush()
        return f"{command}\n"


api.add_resource(PrivateResource,'/command')

if __name__ == '__main__':
    app.run(host="3001:2:e10a::10", port=5000)
```

Netmiko

```
ios_r1 = {
    'device_type': 'linux',
    'username': 'exa',
    'password': 'lab123',
    'ip': '192.168.1.2',
}

source_list = []
exception_time = {}
source_exception_time = {}
thresh = 10


def push_config(src_ip):
    net_connect = ConnectHandler(**ios_r1)
    net_connect.send_command('echo "process http-api {
    run /usr/bin/python3 /home/bitnet/http_api.py;
    encoder json;
}

neighbor 3001:2:e10a::2 {
    router-id 10.10.10.10;
    local-address 3001:2:e10a::10;
    local-as 65010;
    peer-as 65000;

    family {
       ipv4 unicast;
       ipv4 flow;
       ipv6 unicast;
       ipv6 flow;
    }
    flow {
       route TEST {
            match {
                source' + src_ip + ';
                destination 3001:99:b::10/128;
            }
            then {
                redirect 6:302;
                mark 10;
            }

        }
    }
}" >> exabgp-conf.ini')
```

Route Propagation with DSCP Marking

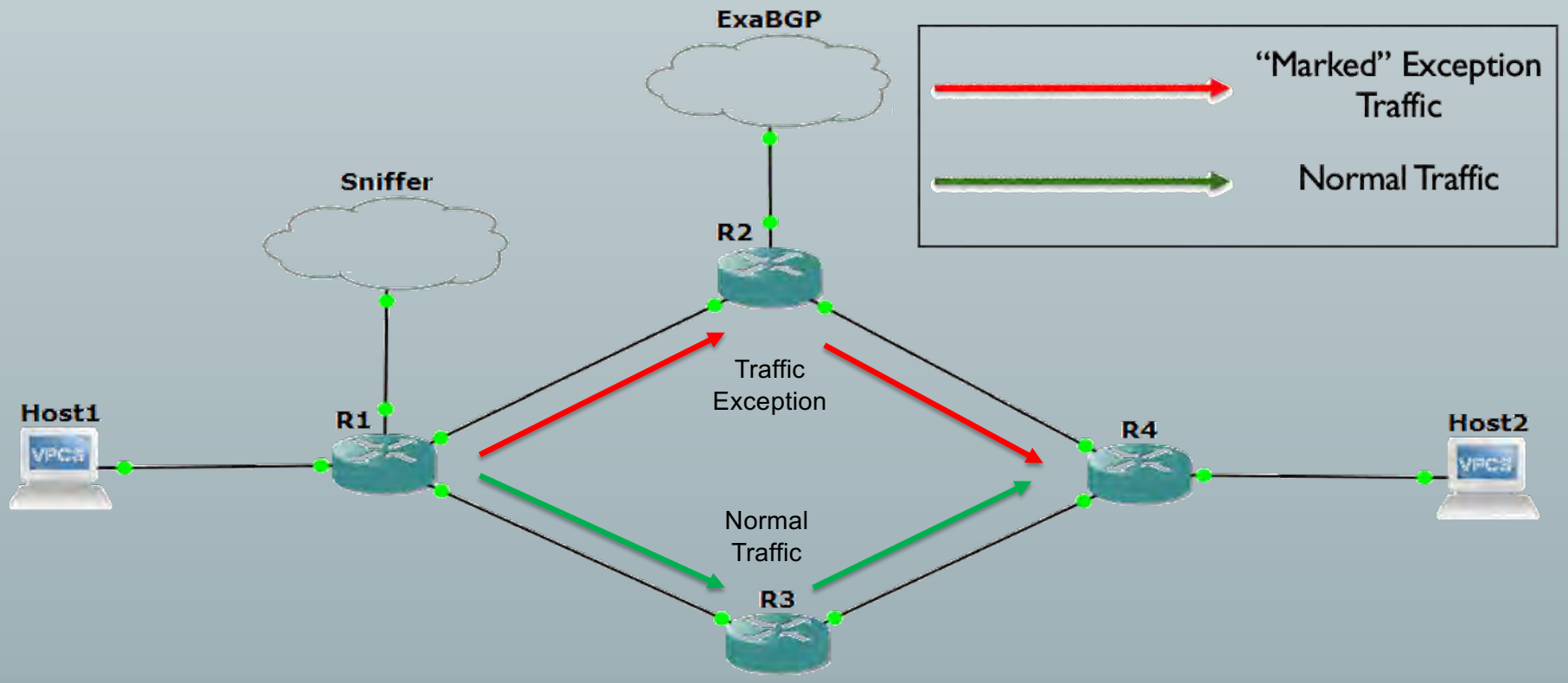


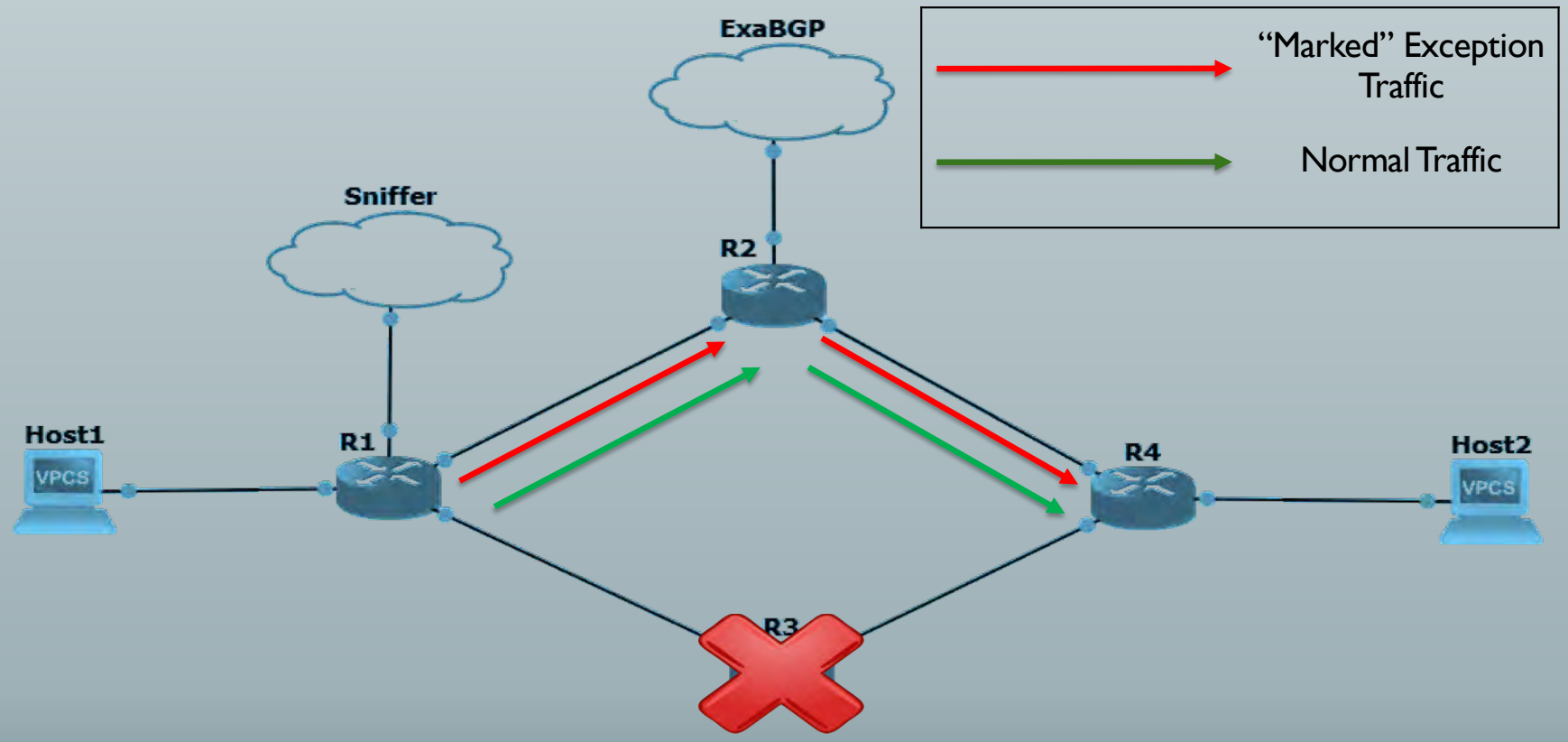
Router 1

```
inet.0: 18 destinations, 18 routes (18 active, 0 holddown, 0 hidden)

nanog.inet.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)

inet6.0: 23 destinations, 23 routes (23 active, 0 holddown, 0 hidden)

nanog.inet6.0: 7 destinations, 7 routes (7 active, 0 holddown, 0 hidden)

inet6flow.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
* 3001:99:b::10/128,3001:99:a::10/128/term:1 (1 entry, 1 announced)
     Accepted
     Flags: NoNexthop
     AS path: 65010 I
     Communities: redirect:6:302 traffic-marking:10
```

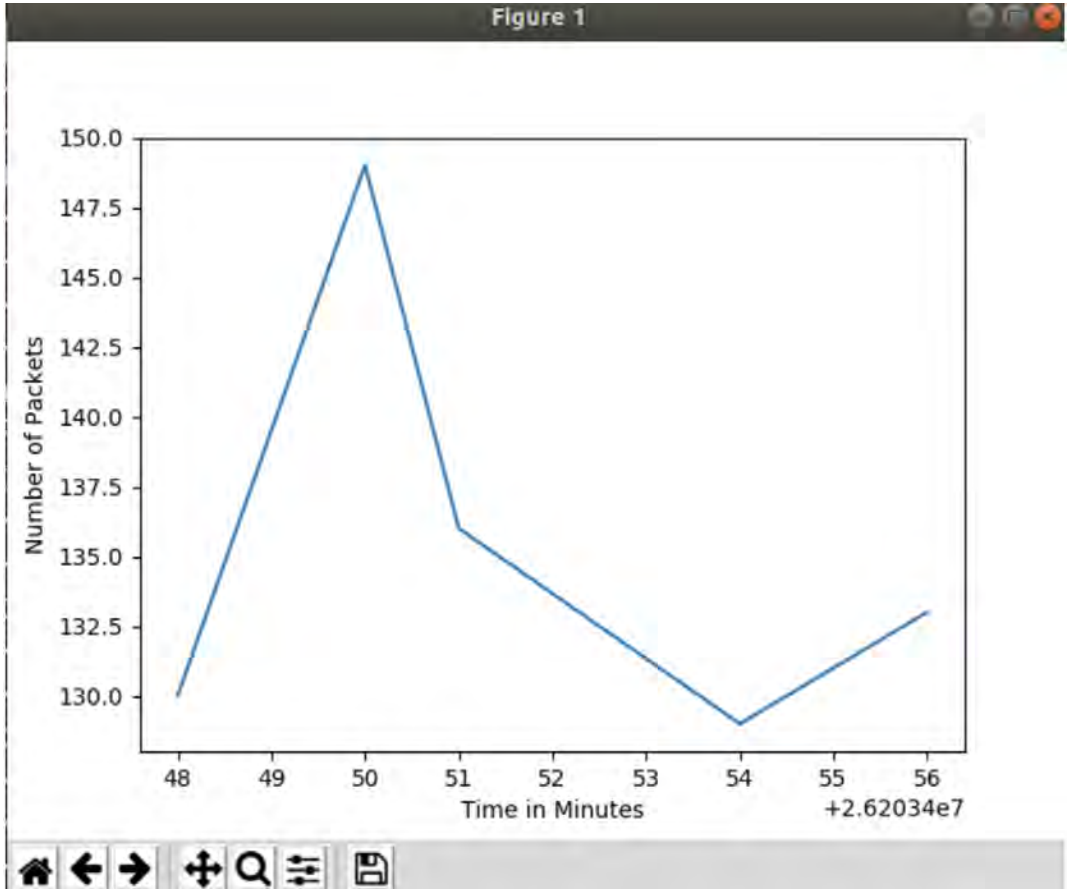
Router 2

```
inet6flow.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
* 3001:99:b::10/128,3001:99:a::10/128/term:1 (1 entry, 1 announced)
     Accepted
     Nexthop: 3001:2::2
     Localpref: 65000
     AS path: 65010 I
     Communities: redirect:6:302 traffic-marking:10

tesutocli@router1>
```

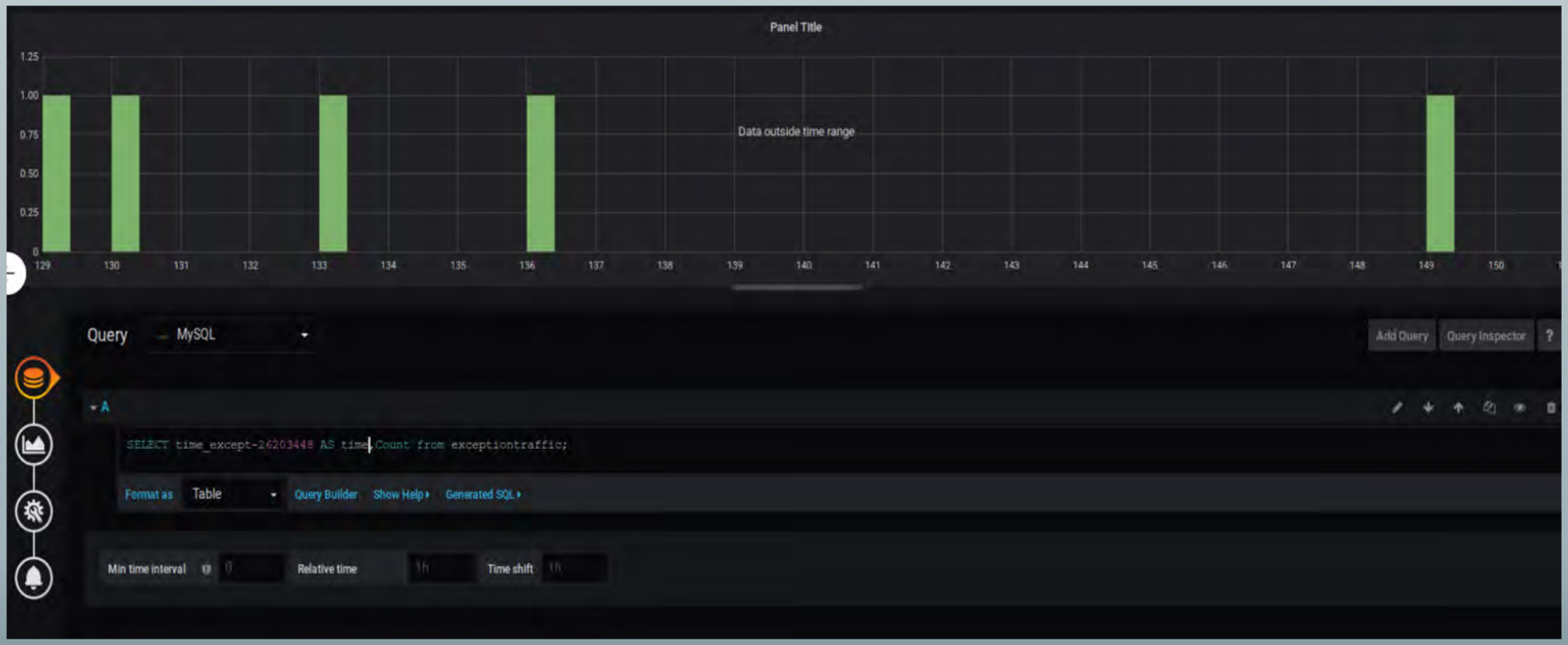

ExaBGP
"Marked" Exception Traffic
Normal Traffic
Sniffer
R2
Traffic Exception
Host1
R1
R4
Host2
Normal Traffic
R3


ExaBGP
"Marked" Exception Traffic
Normal Traffic
Sniffer
R2
Host1
VPCS
R1
R4
Host2
VPCS
R3

VISUALIZATION: MATPLOTLIB



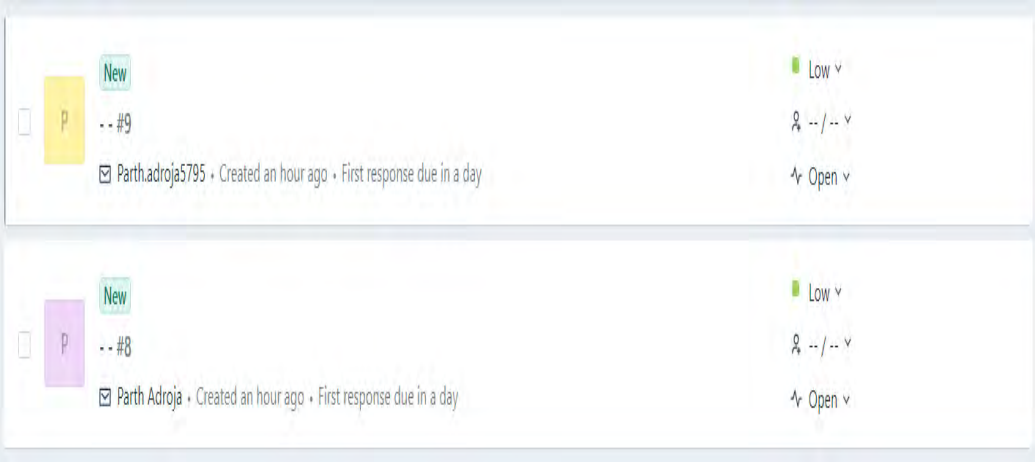
Visualization: Grafana



REPORT FAILURE:

FRESHDESK

```python
def freshdesk():

    fromaddr=#sender's email
    toaddrs="support@parthadroja.freshdesk.com"

    msg=""
    server=smtplib.SMTP('smtp.gmail.com:587')
    server.starttls()
    username=#sendser's email
    password=#password
    server.login(username,password)

    server.sendmail(fromaddr,toaddrs,msg)
    server.quit()
```



University of Colorado
Boulder

REPORT FAILURE:

TWILIO


```
def function_tw(source):
    #SID and Token for authentication on Twilio API
    account_sid = #twilio SID
    auth_token = #twilio authentication
    client = Client(account_sid, auth_token)


    message_info=("Detected too many HTTP traffic from host {}. Please check.".format(source))
    #Defining message body
    message = client.messages.create(
                                body=message_info,
                                from_= #twilio number
                                to= #network administrator number
                            )

    call = client.calls.create(
                                    url= #recorded voice message to be played on call
                                    from_= #twilio number
                                    to= #network administrator number
                                )
    print("Notifying Network Administrator")
```

University of Colorado
Boulder

Future Scope

- Scalability
  - Scale the solution to a higher number of nodes
  - Scale for multiple exceptions
- Applications
  - IDS, IPS, Blackholing

Takeaway

- Tools – tcpdump, Scapy, Python, ExaBGP, Netmiko, REST, Grafana, Twilio API, Freshdesk
- Power of Network Programmability
- Planning
- Team work
- Time Management
- Don't give up! Have Fun!!

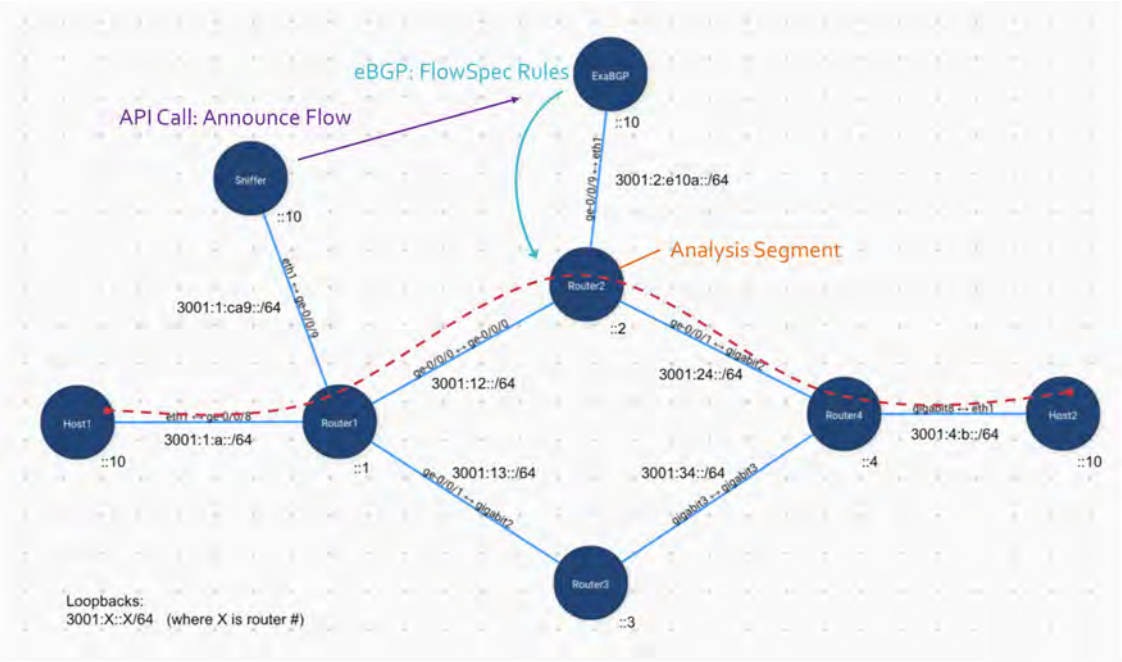
THANK YOU!

NANOG-77 Hackathon

POD-1

Building a FlowSpec Management System

Topology


eBGP: FlowSpec Rules
API Call: Announce Flow
ExaBGP
::10
ge-0/0/9 -- eth1
3001:2:e10a::/64
Sniffer
::10
Analysis Segment
Router2
::2
eth1 -- ge-0/0/9
3001:1:ca9::/64
ge-0/0/0 -- ge-0/0/0
3001:12::/64
ge-0/0/1 -- gigabit1
3001:24::/64
Host1
eth1 -- ge-0/0/8
3001:1:a::/64
::10
Router1
::1
Router4
::4
gigabit4 -- eth1
3001:4:b::/64
Host2
::10
ge-0/0/1 -- gigabit2
3001:13::/64
3001:34::/64
gigabit3 -- gigabit3
Router3
::3
Loopbacks:
3001:X::X/64 (where X is router #)

Overview

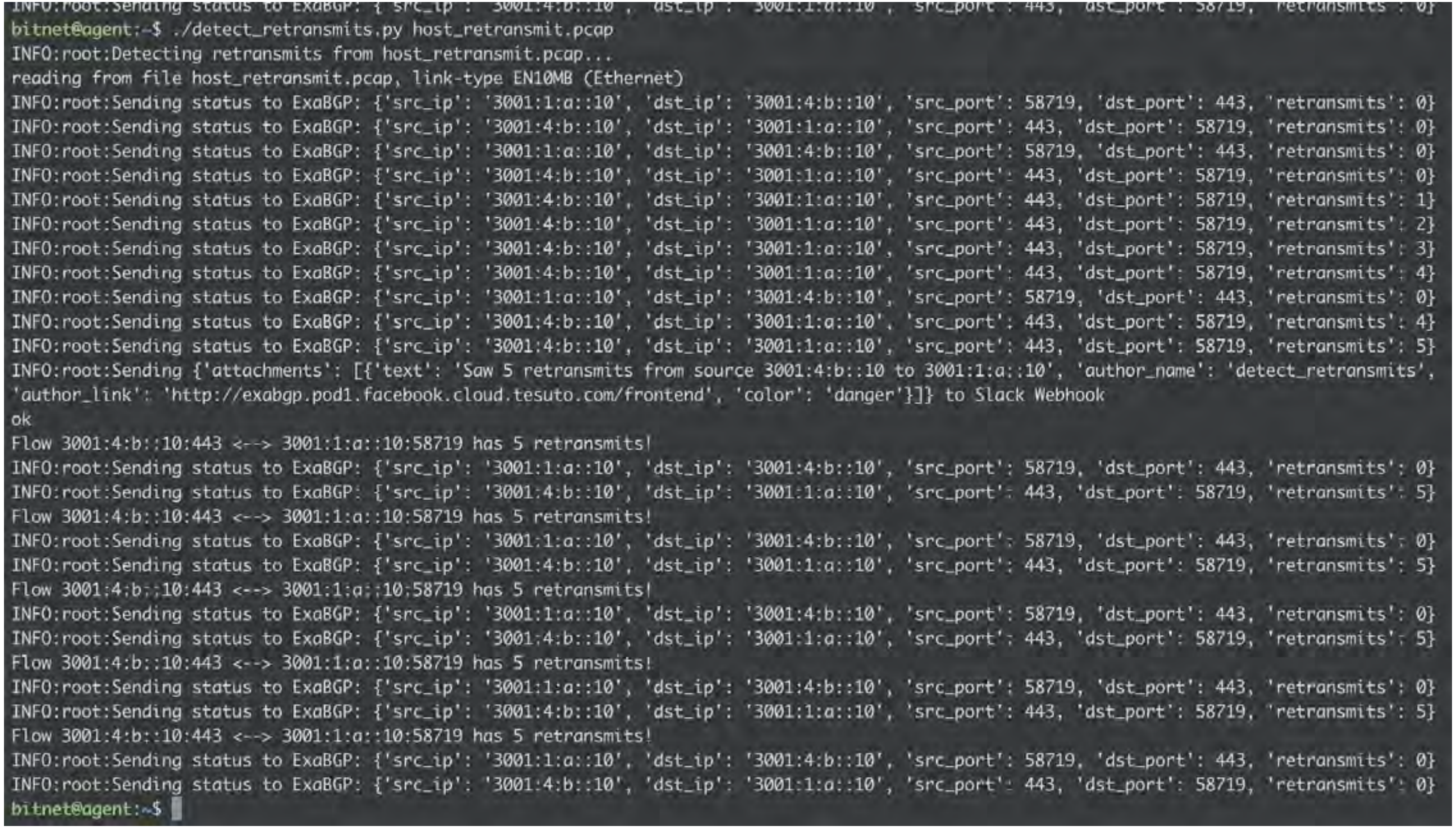
Expanded on the demo flowspec system with the following:

- Send retransmit data to an API
- Alert Slack if retransmits exceed the threshold
- Offer front-end for viewing status
- Offer means to manually announce and withdraw redirects

Demo: Initial Routing State

```
bitnet@router1> show route table inet6flow.0

bitnet@router1>

bitnet@router2> show route protocol bgp table inet6flow.0

bitnet@router2>
```

Demo: Run Sniffer



```
INFO:root:Sending status to ExaBGP: {'src_ip': '3001:4:b::10', 'dst_ip': '3001:1:a::10', 'src_port': 443, 'dst_port': 58719, 'retransmits': 0}
bitnet@agent:~$ ./detect_retransmits.py host_retransmit.pcap
INFO:root:Detecting retransmits from host_retransmit.pcap...
reading from file host_retransmit.pcap, link-type EN10MB (Ethernet)
INFO:root:Sending status to ExaBGP: {'src_ip': '3001:1:a::10', 'dst_ip': '3001:4:b::10', 'src_port': 58719, 'dst_port': 443, 'retransmits': 0}
INFO:root:Sending status to ExaBGP: {'src_ip': '3001:4:b::10', 'dst_ip': '3001:1:a::10', 'src_port': 443, 'dst_port': 58719, 'retransmits': 0}
INFO:root:Sending status to ExaBGP: {'src_ip': '3001:1:a::10', 'dst_ip': '3001:4:b::10', 'src_port': 58719, 'dst_port': 443, 'retransmits': 0}
INFO:root:Sending status to ExaBGP: {'src_ip': '3001:4:b::10', 'dst_ip': '3001:1:a::10', 'src_port': 443, 'dst_port': 58719, 'retransmits': 0}
INFO:root:Sending status to ExaBGP: {'src_ip': '3001:4:b::10', 'dst_ip': '3001:1:a::10', 'src_port': 443, 'dst_port': 58719, 'retransmits': 1}
INFO:root:Sending status to ExaBGP: {'src_ip': '3001:4:b::10', 'dst_ip': '3001:1:a::10', 'src_port': 443, 'dst_port': 58719, 'retransmits': 2}
INFO:root:Sending status to ExaBGP: {'src_ip': '3001:4:b::10', 'dst_ip': '3001:1:a::10', 'src_port': 443, 'dst_port': 58719, 'retransmits': 3}
INFO:root:Sending status to ExaBGP: {'src_ip': '3001:4:b::10', 'dst_ip': '3001:1:a::10', 'src_port': 443, 'dst_port': 58719, 'retransmits': 4}
INFO:root:Sending status to ExaBGP: {'src_ip': '3001:1:a::10', 'dst_ip': '3001:4:b::10', 'src_port': 58719, 'dst_port': 443, 'retransmits': 0}
INFO:root:Sending status to ExaBGP: {'src_ip': '3001:4:b::10', 'dst_ip': '3001:1:a::10', 'src_port': 443, 'dst_port': 58719, 'retransmits': 4}
INFO:root:Sending status to ExaBGP: {'src_ip': '3001:4:b::10', 'dst_ip': '3001:1:a::10', 'src_port': 443, 'dst_port': 58719, 'retransmits': 5}
INFO:root:Sending {'attachments': [{'text': 'Saw 5 retransmits from source 3001:4:b::10 to 3001:1:a::10', 'author_name': 'detect_retransmits',
'author_link': 'http://exabgp.pod1.facebook.cloud.tesuto.com/frontend', 'color': 'danger'}]} to Slack Webhook
ok
Flow 3001:4:b::10:443 <--> 3001:1:a::10:58719 has 5 retransmits!
INFO:root:Sending status to ExaBGP: {'src_ip': '3001:1:a::10', 'dst_ip': '3001:4:b::10', 'src_port': 58719, 'dst_port': 443, 'retransmits': 0}
INFO:root:Sending status to ExaBGP: {'src_ip': '3001:4:b::10', 'dst_ip': '3001:1:a::10', 'src_port': 443, 'dst_port': 58719, 'retransmits': 5}
Flow 3001:4:b::10:443 <--> 3001:1:a::10:58719 has 5 retransmits!
INFO:root:Sending status to ExaBGP: {'src_ip': '3001:1:a::10', 'dst_ip': '3001:4:b::10', 'src_port': 58719, 'dst_port': 443, 'retransmits': 0}
INFO:root:Sending status to ExaBGP: {'src_ip': '3001:4:b::10', 'dst_ip': '3001:1:a::10', 'src_port': 443, 'dst_port': 58719, 'retransmits': 5}
Flow 3001:4:b::10:443 <--> 3001:1:a::10:58719 has 5 retransmits!
INFO:root:Sending status to ExaBGP: {'src_ip': '3001:1:a::10', 'dst_ip': '3001:4:b::10', 'src_port': 58719, 'dst_port': 443, 'retransmits': 0}
INFO:root:Sending status to ExaBGP: {'src_ip': '3001:4:b::10', 'dst_ip': '3001:1:a::10', 'src_port': 443, 'dst_port': 58719, 'retransmits': 5}
Flow 3001:4:b::10:443 <--> 3001:1:a::10:58719 has 5 retransmits!
INFO:root:Sending status to ExaBGP: {'src_ip': '3001:1:a::10', 'dst_ip': '3001:4:b::10', 'src_port': 58719, 'dst_port': 443, 'retransmits': 0}
INFO:root:Sending status to ExaBGP: {'src_ip': '3001:4:b::10', 'dst_ip': '3001:1:a::10', 'src_port': 443, 'dst_port': 58719, 'retransmits': 5}
Flow 3001:4:b::10:443 <--> 3001:1:a::10:58719 has 5 retransmits!
INFO:root:Sending status to ExaBGP: {'src_ip': '3001:1:a::10', 'dst_ip': '3001:4:b::10', 'src_port': 58719, 'dst_port': 443, 'retransmits': 0}
INFO:root:Sending status to ExaBGP: {'src_ip': '3001:4:b::10', 'dst_ip': '3001:1:a::10', 'src_port': 443, 'dst_port': 58719, 'retransmits': 0}
bitnet@agent:~$
```

Demo: Slack Notifications

pod1team APP 9:59 AM

detect_retransmits
Saw 5 retransmits from source 3001:4:b::10 to 3001:1:a::10

new messages

pod1team APP 10:35 AM

detect_retransmits
Saw 5 retransmits from source 3001:4:b::10 to 3001:1:a::10

detect_retransmits
Saw 5 retransmits from source 3001:4:b::10 to 3001:1:a::10

detect_retransmits
Saw 5 retransmits from source 3001:4:b::10 to 3001:1:a::10

detect_retransmits
Saw 5 retransmits from source 3001:4:b::10 to 3001:1:a::10

Demo: Announce Redirect: UI

Not Secure | exabgp.pod1.facebook.cloud.tesuto.com/frontend

Flush State

Source IP	Source Port	Destination IP	Destination Port	Retransmits	Announce	Withdraw
3001:4:b::10	443	3001:1:a::10	58719	5	Announce Redirect	Withdraw Redirect
3001:1:a::10	58719	3001:4:b::10	443	0	Announce Redirect	Withdraw Redirect

Demo: Announce Redirect: Button Output

```
// 20191027141830
// http://exabgp.pod1.facebook.cloud.tesuto.com/redirect

{
  "commands_executed": [
    "announce flow route source 3001:4:b::10/128 destination 3001:1:a::10/128 redirect 6:302",
    "announce flow route source 3001:1:a::10/128 destination 3001:4:b::10/128 redirect 6:302"
  ]
}
```

Demo: Announce Redirect: ExaBGP Output

```
flow added to neighbor 3001:2:e10a::2 local-ip 3001:2:e10a::10 local-as 65010 peer-as
65000 router-id 10.10.10.10 family-allowed in-open : flow destination-ipv6
3001:1:a::10/128/0 source-ipv6 3001:4:b::10/128/0 extended-community redirect:6:302


flow added to neighbor 3001:2:e10a::2 local-ip 3001:2:e10a::10 local-as 65010 peer-as
65000 router-id 10.10.10.10 family-allowed in-open : flow destination-ipv6
3001:4:b::10/128/0 source-ipv6 3001:1:a::10/128/0 extended-community redirect:6:302
```

Demo: Announce Redirect: Route Update - Router1

```
bitnet@router1> show route table inet6flow.0

inet6flow.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

3001:1:a::10/128,3001:4:b::10/128/term:1
                   *[BGP/170] 00:00:01, localpref 65000
                      AS path: 65010 I, validation-state: unverified
                    >  to 3001:2::2
3001:4:b::10/128,3001:1:a::10/128/term:2
                   *[BGP/170] 00:00:01, localpref 65000
                      AS path: 65010 I, validation-state: unverified
                    >  to 3001:2::2
```

Demo: Announce Redirect: Route Update - Router2

```
bitnet@router2> show route protocol bgp table inet6flow.0

inet6flow.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

3001:1:a::10/128,3001:4:b::10/128/term:1
                   *[BGP/170] 00:24:35, localpref 65000, from 3001:2:e10a::10
                      AS path: 65010 I, validation-state: unverified
                       Receive
3001:4:b::10/128,3001:1:a::10/128/term:2
                   *[BGP/170] 00:24:35, localpref 65000, from 3001:2:e10a::10
                      AS path: 65010 I, validation-state: unverified
                       Receive
```

Demo: Withdraw Redirect: UI

Not Secure | exabgp.pod1.facebook.cloud.tesuto.com/frontend

Flush State

Source IP	Source Port	Destination IP	Destination Port	Retransmits	Announce	Withdraw
3001:4:b::10	443	3001:1:a::10	58719	5	Announce Redirect	Withdraw Redirect
3001:1:a::10	58719	3001:4:b::10	443	0	Announce Redirect	Withdraw Redirect

Demo: Withdraw Redirect: Button Output

```
// 20191027142052
// http://exabgp.pod1.facebook.cloud.tesuto.com/withdraw

{
  "commands_executed": [
    "withdraw flow route source 3001:4:b::10/128 destination 3001:1:a::10/128 redirect 6:302",
    "withdraw flow route source 3001:1:a::10/128 destination 3001:4:b::10/128 redirect 6:302"
  ]
}
```

Demo: Withdraw Redirect: ExaBGP Output

```
flow removed from neighbor 3001:2:e10a::2 local-ip 3001:2:e10a::10 local-as 65010
peer-as 65000 router-id 10.10.10.10 family-allowed in-open : flow destination-ipv6
3001:1:a::10/128/0 source-ipv6 3001:4:b::10/128/0 extended-community redirect:6:302

flow removed from neighbor 3001:2:e10a::2 local-ip 3001:2:e10a::10 local-as 65010
peer-as 65000 router-id 10.10.10.10 family-allowed in-open : flow destination-ipv6
3001:4:b::10/128/0 source-ipv6 3001:1:a::10/128/0 extended-community redirect:6:302
```

Demo: Withdraw Redirect: Route Update

```
bitnet@router1> show route table inet6flow.0

bitnet@router1>

bitnet@router2> show route protocol bgp table inet6flow.0

bitnet@router2>
```

How We Did It

- detect_retransmits.py
  - Swapped call to ExaBGP API with a POST to a Slack Webhook when the retransmit threshold is exceeded
  - On every packet process, send updated retransmit data for the applicable flow
- ExaBGP API
  - Added /status endpoint that stores retransmit data received from the Sniffer in Redis
  - Added /redirect and /withdraw endpoints to execute announcing and withdrawing the BGP redirect
- ExaBGP Frontend
  - Added a frontend for the API that exposes the status data received from the Sniffer and offers buttons for executing the redirect and withdraw endpoints using only HTML and CSS
- ExaBGP Process
  - Separated the API from the exabgp process wrapper and proxied it with uWSGI to make the API and frontend available via port 80
  - Used named pipes as the communication layer between the uWSGI process and the ExaBGP process

Future Enhancements

- JavaScript to implement sorting and filtering the status table
- Form input to specify the ExaBGP action instead of hardcoding to `redirect 6:302`
- Poll routing data from the routers and expose on the UI to view the effects of the ExaBGP commands live on the network
- Detect other traffic anomalies in addition to retransmits

Thanks!

Zoe Blevins

David Testa

Tony Franklin

Kyle Bean

Path Tracers

•••

Colin McIntosh
Kyle Birkeland
Soham Shah
Evan Alexandre
Nishit Bavishi

Understanding the problem

Current Tooling

Traceroute has issues when faced with lots of ECMP

Better Traffic Flow

Analyze the flow and detect link state

Take decisions based on IGP, BGP and physical link state

Re-analyze the need to drain if needed

Harness and deploy

- ★ Detect
- ★ Analyze
- ★ Configure
- ★ Influence

AND

Repeat !!

(Run it as a service)

Context

Building a better Traceroute

Client Implications:

- Ability to investigate routes that include ECMP

Influence Link State

React to interface metrics (flapping, errors)

Client Implications:

- Automatically respond to route traffic around problematic link

Product Overview

- NAPALM (Network Automation and Programmability Abstraction)
  - Widely adaptable and scalable
- Paris traceroute
  - Interesting take on path checks
- Facebook fbtracert

Our Traceroute Implementation

First Attempt

- We chose to use paris-traceroute with UDP for our method of analyzing different paths
- We initially began building an implementation of UDP traceroute in Golang with the intent to build the paris-traceroute feature on top
- Discovered https://github.com/facebook/fbtracert which provides an existing paris-traceroute implementation for TCP along with some helpful command-line utilities

Our Traceroute Implementation (cont)

Second Attempt

- After analyzing the existing fbtracert code we found a lot of similarities
  - Golang and the organization of goroutines
  - Logic to test various TTLs
- We also found things we didn't like
  - Lack of UDP or ICMP traceroute
  - Output was unclear and lacking useful data
  - Some bugs in the timing of traceroutes
  - Built as a command-line utility rather than a library
- With this we decided to expand on a fork of fbtracert to keep what we liked, add what we wanted, and planned to push the changes upstream.

Traceroute Implementation Results

- We were able to convert the fbtracert code from a command-line tool to a Golang library that is importable and usable by other code.
- We were able to improve the command-line output to be more clear and include additional metadata.
- We were able to build our implementation of UDP traceroute into fbtracert, **however** we were unable to finish integrating this into the analysis component of the script (good opportunity for a future hackathon!)

Results

Old:


```
{
        "Paths": {
                "32768": [
                        "?",
                        "10.253.1.1",
                        "?",
                        "?",
                        "97.77.2.2"
                ],
                "32769": [
                        "?",
                        "10.253.1.1",
                        "?",
                        "?",
                        "rrcs-24-73-240-134.se.biz.rr.com."
                ],
        },
        "Sent": {
                "32768": [
                        2,
                        2,
                        2,
                        2,
                        2
                ],
                "32769": [
                        2,
                        2,
                        2,
                        2,
                        2
                ],
        },
        "Rcvd": {
                "32768": [
                        0,
                        1,
                        0,
                        0,
                        1
                ],
                "32769": [
                        0,
                        1,
                        0,
                        0,
                        2
                ],
        }
}
```


New:


```
{
        "32768": [
                {
                        "SrcAddr": "?",
                        "SrcName": "?",
                        "Sent": 2,
                        "Received": 0,
                        "TTL": 1
                },
                {
                        "SrcAddr": "?",
                        "SrcName": "?",
                        "Sent": 2,
                        "Received": 0,
                        "TTL": 2
                },
                {
                        "SrcAddr": "?",
                        "SrcName": "?",
                        "Sent": 2,
                        "Received": 0,
                        "TTL": 3
                },
                {
                        "SrcAddr": "?",
                        "SrcName": "?",
                        "Sent": 2,
                        "Received": 0,
                        "TTL": 4
                },
                {
                        "SrcAddr": "97.77.2.2",
                        "SrcName": "97.77.2.2",
                        "Sent": 2,
                        "Received": 1,
                        "TTL": 5
                }
        ],
        "32769": [
                {
                        "SrcAddr": "?",
                        "SrcName": "?",
                        "Sent": 2,
                        "Received": 0,
                        "TTL": 1
                },
...
```

Traffic Influencer via Path Probing

Identify Traffic Paths

Combine health-check service with Sniffer service

Identify the available traffic paths using custom traceroute implementation

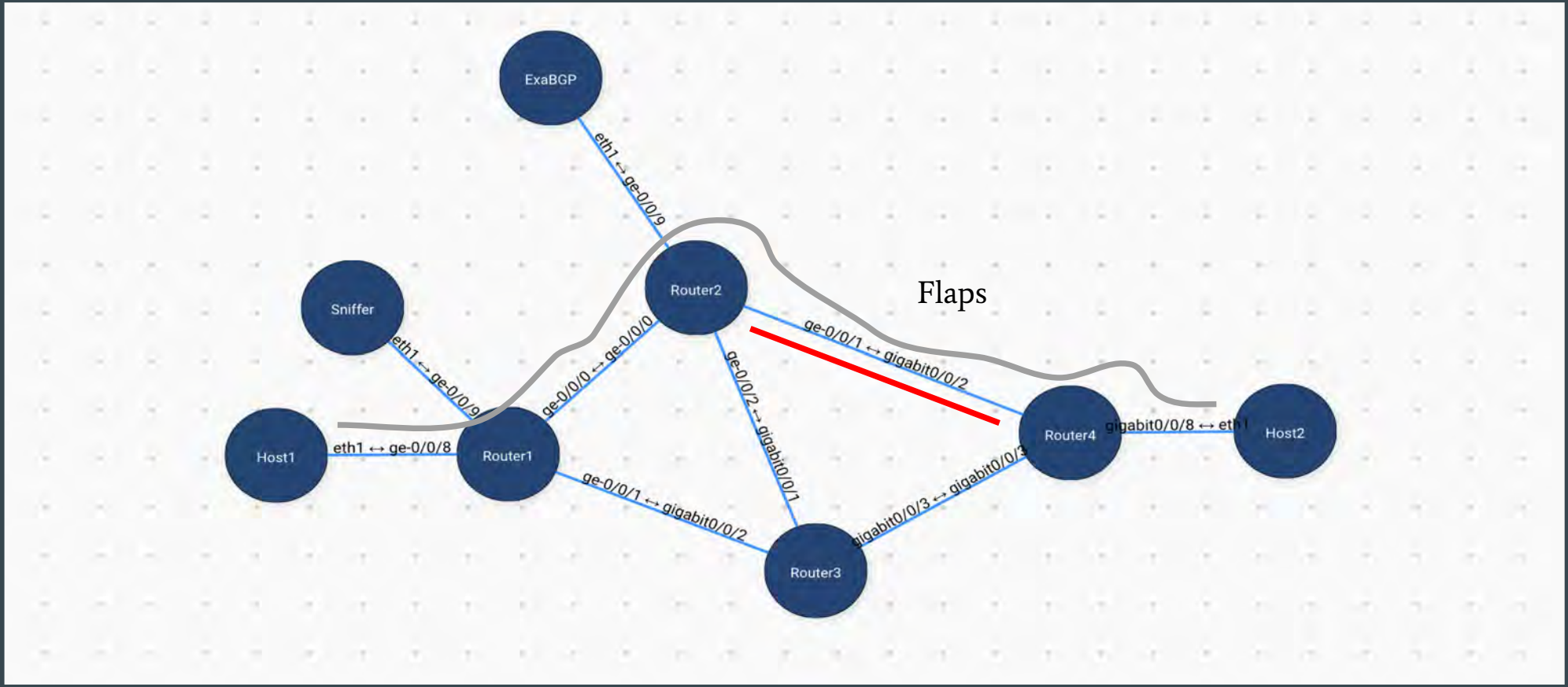
Influence and Act

Use a simplistic approach to further drain the link from production traffic via IGP metric cost

Identify rogue link

Run health-check service to identify link state (flaps/errors/utilization and more)

Traffic Flows



Results

```
bitnet@agent:~$ python3 detect_linkflaps.py
Opening ...
Interface:  ge-0/0/1
Last Flapped:  292.0
Logged into the router1
Increasing interface ge-0/0/1 ospf cost
Check over
```

Identifying the threshold for flapping, then reacting

Results

```
tesutocli@router1> show interfaces ge-0/0/0
Physical interface: ge-0/0/0, Enabled, Physical link is Up
  Interface index: 148, SNMP ifIndex: 526
  Description: Router2
  Link-level type: Ethernet, MTU: 1514, MRU: 1522, LAN-PHY mode, Speed: 1000mbps, BPDU Error: None, Loop Detect PDU Error: None, Ethernet-Switching Error: None,
  MAC-REWRITE Error: None, Loopback: Disabled, Source filtering: Disabled, Flow control: Enabled, Auto-negotiation: Enabled, Remote fault: Online
  Pad to minimum frame size: Disabled
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  Link flags     : None
  CoS queues     : 8 supported, 8 maximum usable queues
  Current address: 02:42:64:80:14:02, Hardware address: 02:42:64:80:14:02
  Last flapped   : 2019-10-27 05:24:36 UTC (00:00:08 ago)
  Input rate     : 664 bps (0 pps)
  Output rate    : 1152 bps (0 pps)
  Active alarms  : None
  Active defects : None
  PCS statistics                      Seconds
    Bit errors                             0
    Errored blocks                         0
  Ethernet FEC statistics              Errors
    FEC Corrected Errors                    0
    FEC Uncorrected Errors                  0
    FEC Corrected Errors Rate               0
    FEC Uncorrected Errors Rate             0
  Interface transmit statistics: Disabled
```

Results

```
tesutocli@router1> show configuration protocols ospf
area 0.0.0.0 {
    interface lo0.0;
    interface ge-0/0/0.0 {
        metric 65001;
    }
    interface ge-0/0/1.0;
    interface ge-0/0/8.0 {
        passive;
    }
    interface ge-0/0/9.0 {
        passive;
    }
}
```

Learnings

- We all learned what paris-traceroute is (except Kyle)
- Different flavours of traceroute
- Traffic Implementations and Flows

Challenges

- Socket programming for traceroute
- Time (as always a complaint)

Conclusion

- Fetching requisite data from the testbed to take decisions
- Identifying paths using custom traceroute and influence traffic decisions
- Implemented Basic Route Dampening in IGP

Code: https://github.com/kbirkeland/nanog77-hackathon/