Validating Network High Availability with Batfish

Rick Donato

Snr Automation Consultant @ NetworkToCode

About Me

- Rick Donato
  - Senior Network Automation Engineer at NetworkToCode.
  - Previously - NFV/SDN Solution Architect, Principal Network Security Engineer.
- Twitter - @rickjdon
- Blog
  - https://networktocode.com/blog
  - https://packetflow.co.uk
- Github
  - https://github.com/rickdonato

About this Talk/You

About this Talk

- How to validate high availability using flow-based verification via Batfish
- Example topology
- Batfish features (snapshot forking, differential reachability)
- Code walk through
- Demo

About You

- Good knowledge of Python
- Experience with Batfish
- Network Engineer/Network Automation Engineers

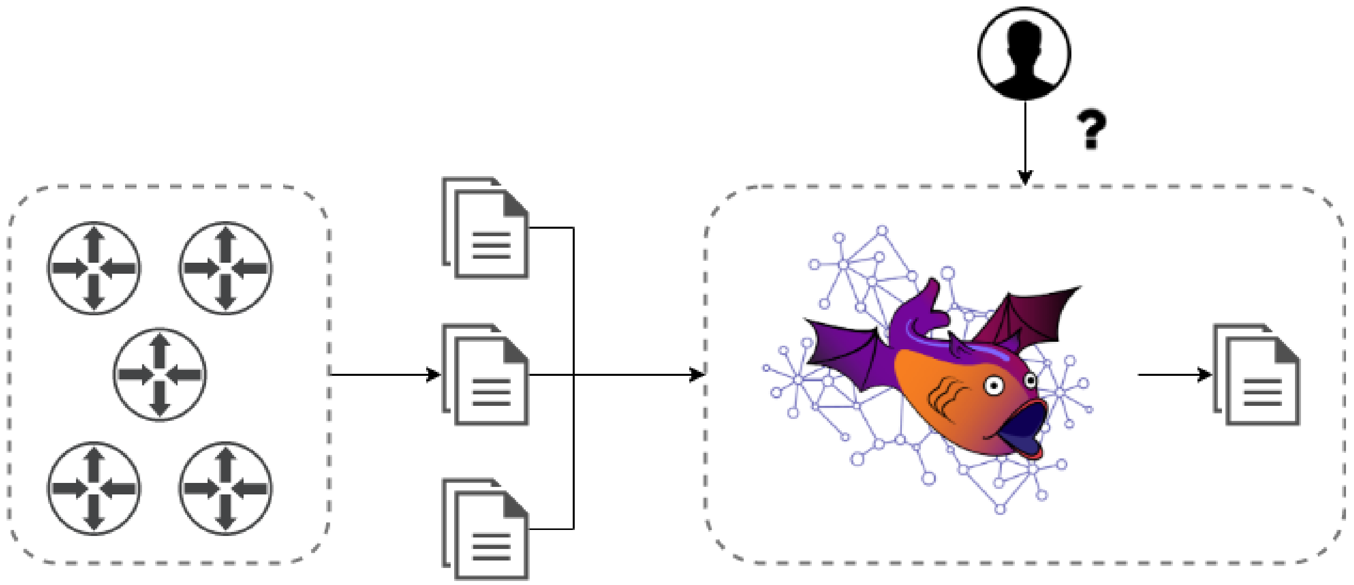
The Why

- HA Verification
  - Ensure network is:
    - Designed to handle node/link failures
    - Built to handle node/link failures
  - Key Goal: Business continuity/service is unaffected
- Configuration vs Flow Verification
  - Configuration based testing – Unit Tests
    - Great but can be hard to predefine all cases
    - Extremely difficult to check behavior
  - Flow based verification – Integration Tests
    - Compliments unit/configuration tests
    - Catch edge cases not covered unit/configuration tests

Batfish Overview

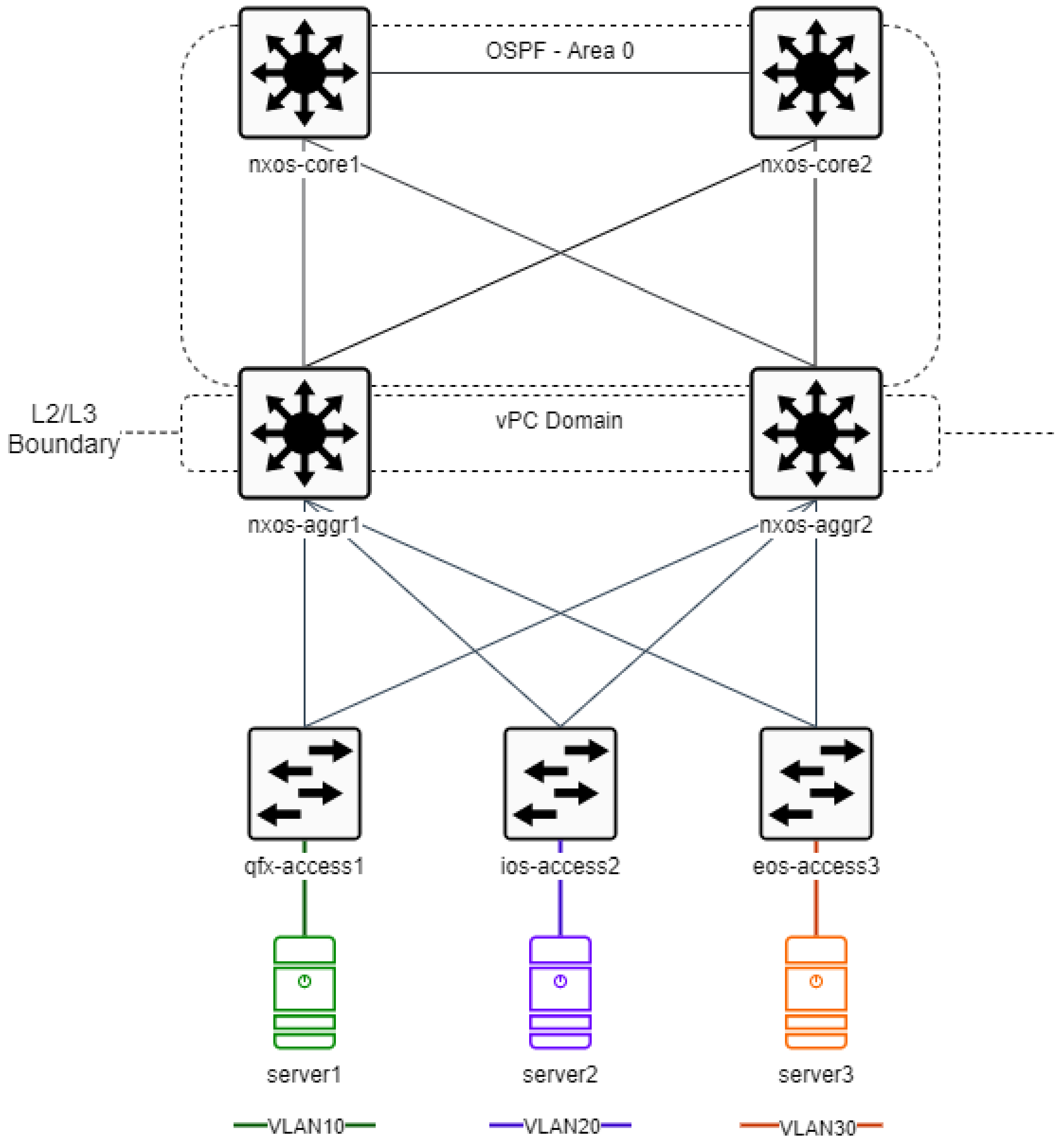
```
>>>
bfq.interfaceProperties(properties="MTU,Primary_Address,Switchport_Mode").answer().frame()
                         Interface   MTU Primary_Address Switchport_Mode
0     nxos-aggr2[Ethernet1/89]  1500            None          ACCESS
1  nxos-aggr2[port-channel20]  1500            None           TRUNK
2    nxos-aggr2[Ethernet1/124]  1500            None          ACCESS
3     nxos-aggr2[Ethernet1/20]  1500            None          ACCESS
4        nxos-aggr2[loopback0]  1500  192.168.1.4/32            NONE
```

- Opensource
- Containerized service
- Snapshots
- Offline model
- Models
  - Vendor agnostic
  - Configuration
  - Control plane
- Questions
  - REST API
  - pybatfish
  - Ansible Role
- Capabilities: https://github.com/batfish/pybatfish/tree/master/jupyter_notebooks
- Website: http://batfish.org/

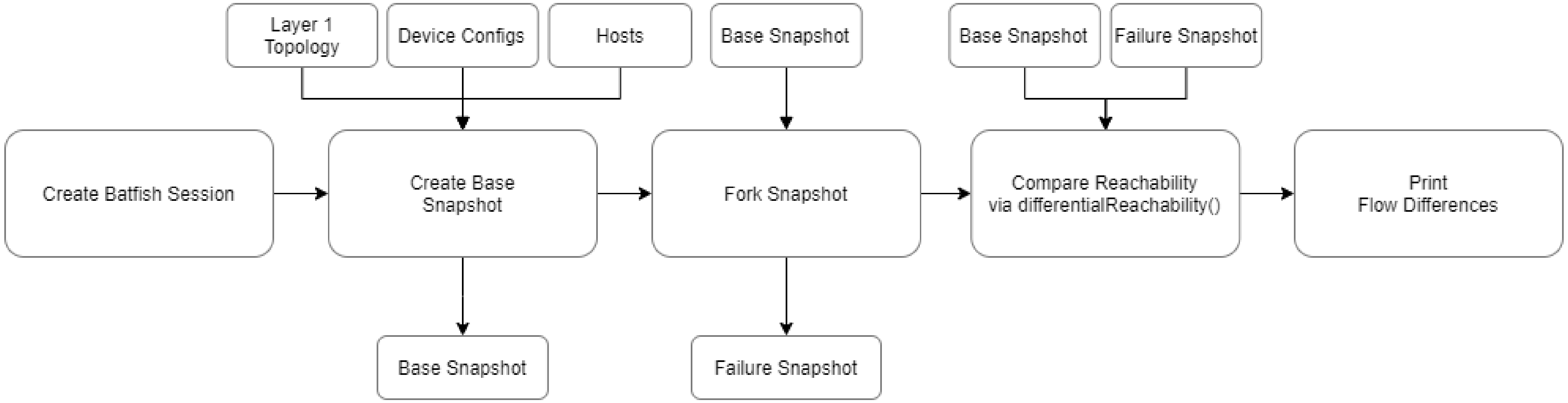


Topology

- Traditional 3 tier topology (access, aggregation, core)
- Core – OSPF backbone area.
- Aggregation – vPC, L2/L3 boundary.
- Access – DOT1Q, Port channels to aggregation



The Process


Layer 1 Topology
Device Configs
Hosts
Base Snapshot
Base Snapshot
Failure Snapshot
Create Batfish Session
Create Base Snapshot
Fork Snapshot
Compare Reachability via differentialReachability()
Print Flow Differences
Base Snapshot
Failure Snapshot

Snapshot – Layer 1 Topology

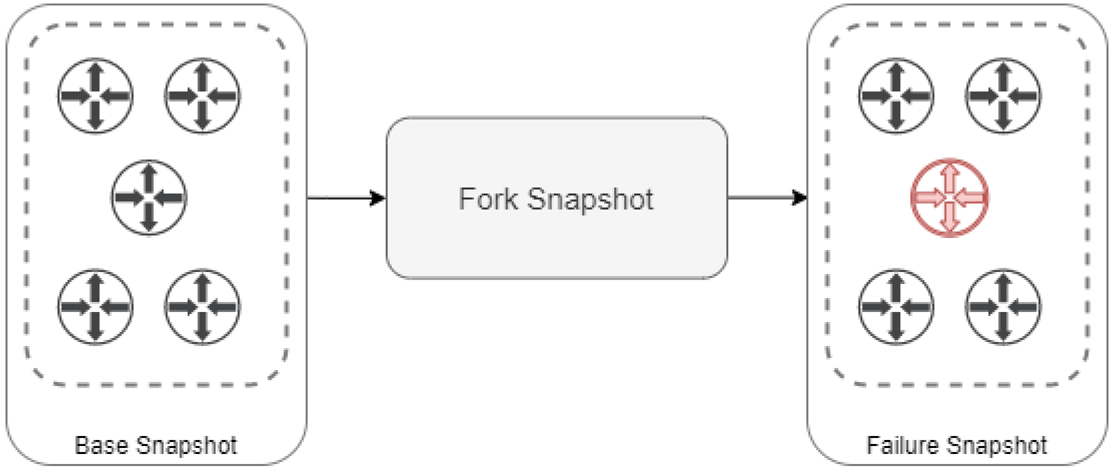
- Batfish constructs L3 topologies based on addresses
- Problematic cases:
  - IP addresses reused across multiple segments.
  - Layer 2 segments.
- Layer 1 topology used to improve topology understanding:
  - Added to snapshot
  - Filename: layer1_topology.json
  - Contains list of edge records
- Our use case – L2 access layer.

```
{
  "edges": [
    {
      "node1": {
        "hostname": "server1",
        "interfaceName": "eth0"
      },
      "node2": {
        "hostname": "qfx-access1",
        "interfaceName": "xe-0/0/0.0"
      }
    },
    {
      "node1": {
        "hostname": "server2",
        "interfaceName": "eth0"
      },
      "node2": {
        "hostname": "ios-access2",
        "interfaceName": "GigabitEthernet0/2"
      }
    },
...
```

Snapshot Forking

- Copies a snapshot.
- Allows you to alter parameters in the copy:
  - Interfaces – deactivate/reactivate
  - Nodes - deactivate/reactivate
  - Files – nodes, iptables etc.
- Useful for differential based questions:
  - Compare Filters
  - Differential Reachability

```
bf_fork_snapshot(
    base_name=BF_SNAPSHOT_BASE,
    name=BF_SNAPSHOT_FAIL,
    deactivate nodes=[a,b,c],
    deactivate_interfaces=[x,y,z],
    overwrite=True,
)
```



Differential Reachability

- Takes 2 snapshots.
- Returns flows that are successful in one snapshot but not in another.

```
bfq.differentialReachability(
    headers=HeaderConstraints(dstIps=DST_IP_REACHABILITY)
    )
    .answer(snapshot=BF_SNAPSHOT_FAIL, reference_snapshot=BF_SNAPSHOT_BASE)
    .frame()
```

Flow	Reference_Traces	Reference_TraceCount	Snapshot_Traces	Snapshot_TraceCount

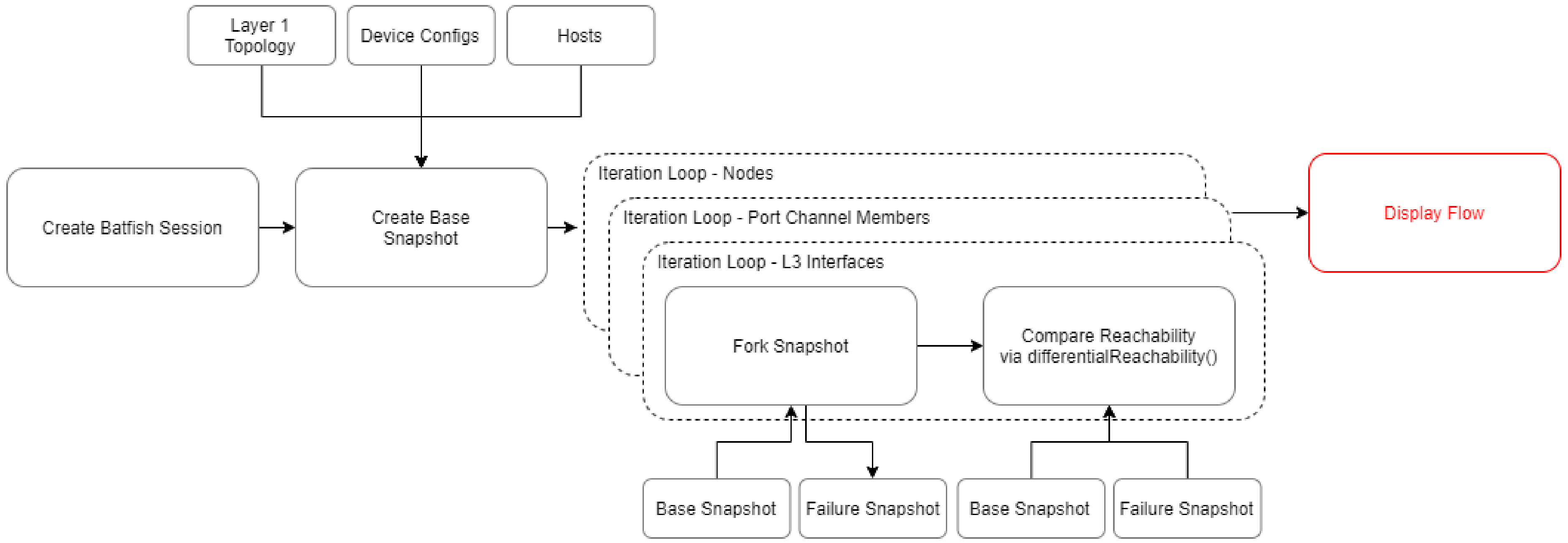
start=nxos-core1 interface=Ethernet1/2 [10.1.2.3->10.2.20.1 ICMP length=512]	ACCEPTED
	1. node: nxos-core1
	RECEIVED(Ethernet1/2)
	FORWARDED(ARP IP: 10.1.3.2, Output Interface: Ethernet1/3, Routes:)
	TRANSMITTED(Ethernet1/3)
	2. node: nxos-core2
	RECEIVED(Ethernet1/3)
	FORWARDED(ARP IP: 10.2.1.2, Output Interface: Ethernet1/1, Routes:)
	TRANSMITTED(Ethernet1/1)
	3. node: nxos-aggr2
	RECEIVED(Ethernet1/4)
	FORWARDED(ARP IP: AUTO/NONE(-1l), Output Interface: Vlan20, Routes:)
	TRANSMITTED(Vlan20)
	4. node: server2
	RECEIVED(eth0)
	ACCEPTED(eth0)
		1	NO_ROUTE
			1. node: nxos-core1
			RECEIVED(Ethernet1/2)
			NO_ROUTE
				1

Differential Reachability

- Inputs

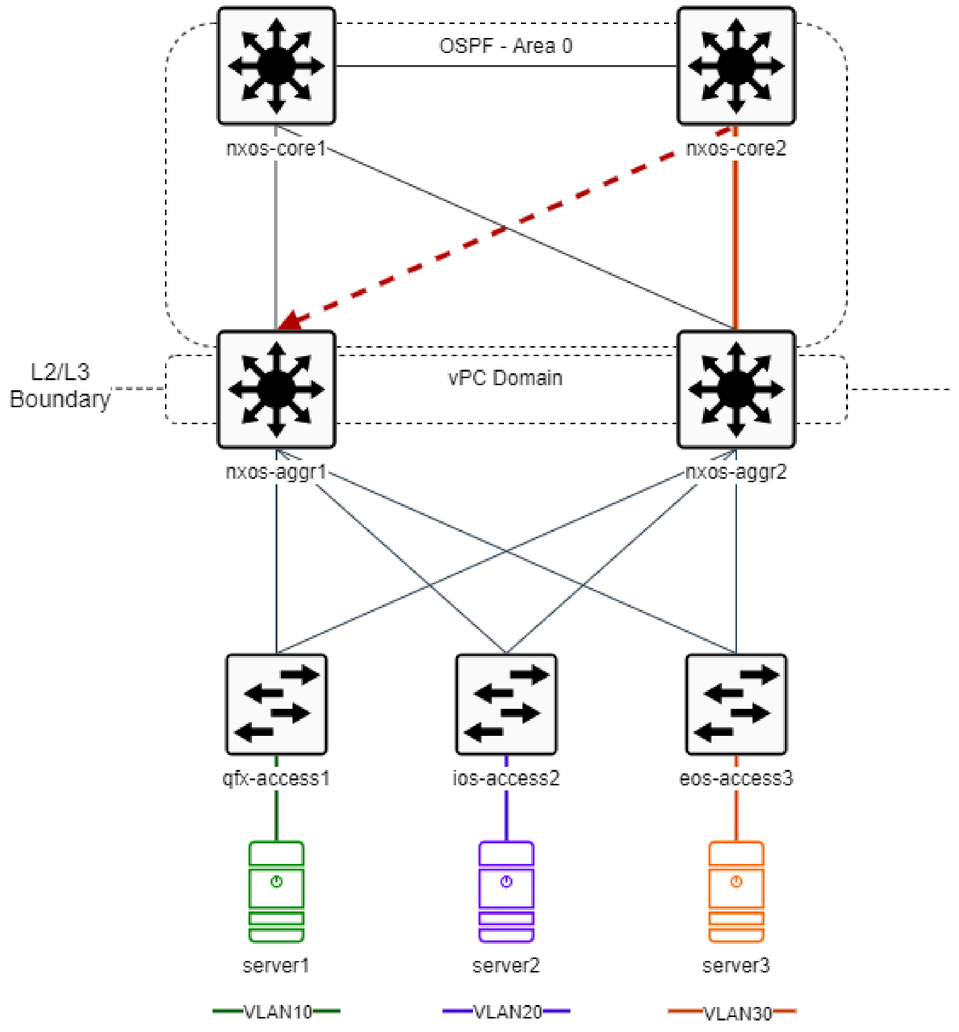
Name	Description	Variables
pathConstraints	Constraint the path a flow can take (start/end/transit locations).	startLocation, endLocation, transitLocations, forbiddenLocations
headers	Packet header constraints.	srcIps, dstIps, srcPorts, dstPorts, applications, ipProtocols, icmpCodes, dscps, packetLengths, tcpFlags
actions	Only return flows for which the disposition is from this set.	Success, Failure, Accepted, No_Route, Null_routed, Loop …. many more
maxTraces	Limit the number of traces returned.	int
invertSearch	Search for packet headers outside the specified headerspace, rather than inside the space.	True/False
ignoreFilters	Do not apply filters/ACLs during analysis.	True/False

Code Walk Through



Demo

Topology Failure



What Else?

- Join the Batfish community
  - NTC Slack - https://networktocode.slack.com #batfish
  - Batfish Slack - https://batfish-org.slack.com

- Batfish Enterprise
  - https://www.networktocode.com/
  - https://www.intentionet.com/

- Demo Code - https://github.com/networktocode/interop2020-batfish

>>> network .toCode()

Thank You…