Containerized testbed deployment of SURFnet8 service layer network

Pim Paardekooper & Iñigo Gonzalez

Supervisors: Migiel de Vos, Marijke Kaat & Peter Boers

Introduction

SURF

- Collaborative organisation for ICT in Dutch education and research.
- SURF network
  - more then 300 nodes
  - Juniper MX routers
- SURFnet7 -> SURFnet8

Virtual Testbed

- Separate from a production environment
- Malleable

SURFnet8 Virtual Testbed

- Makes use of vMX: a virtual router developed by Juniper
- High resource usage
- Scalability bottleneck

Project Purpose

Containerized routing protocol process (cRPD)

Research question:

- How can a containerized testbed using cRPD be scalable in terms of number of router instances, to help SURF engineers test their network setup?

Background

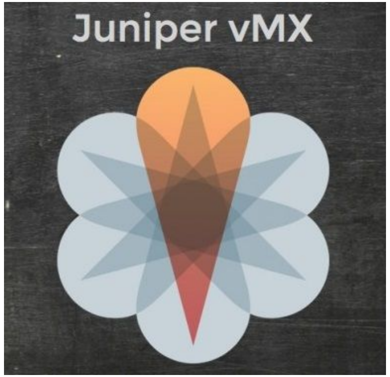
vMX virtualized testbed

Previous research on virtualized testbed using Juniper's vMX

- Virtual router running Junos OS
- Operational consistency of physical MX series routers

Results:

- High resource allocation required (4 cores and 3GB of memory)
- Constrained resource availability
- Scalability bottleneck



Container RPD (cRPD)

- Juniper's routing protocol process decoupled from Junos OS
- It learns route state through various protocols and keeps that state in the RIB
- Does not feature a data plane
- Packet forwarding is handled by the Kernel

Minimum resource requirements:

CPU	1 core
Memory	256 MB

Kubernetes

- Orchestrator for containerized applications
- Open source project
- Automates deployment, scaling and management of containers

- A Pod represents a set of running containers
- By default Pods are interconnect on a flat network setup

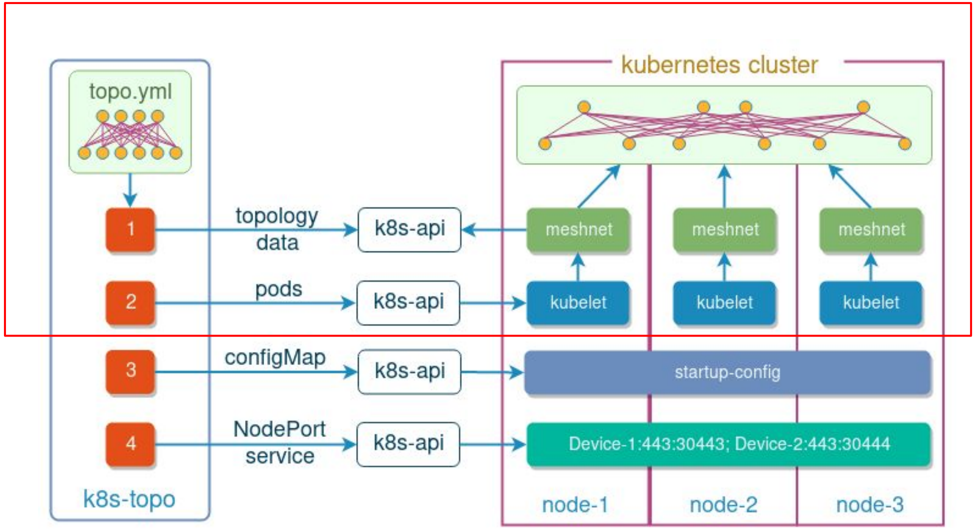
Meshnet CNI

- Allows creating point-to-point links between containers
- Configuration deployed through Topology custom resource
- Links can be created between pod running in different nodes (hosts)

```yaml
apiVersion: networkop.co.uk/v1beta1
kind: Topology
metadata:
  name: r1
spec:
  links:
  - uid: 1
    peer_pod: r2
    local_intf: eth1
    local_ip: 12.12.12.1/24
    peer_intf: eth1
    peer_ip: 12.12.12.2/24
```

K8s-topo

- Simplifies the interaction with Meshnet
- Helps create arbitrary network topologies
- Builds **Topology** and **Pod** manifests from lightweight configuration files



Defining the use case

Interviews

Interviewing SURF engineers to find out:

- Most relevant use cases
- Used protocols
- Required tool integration
- Manageability requirements

Use case: eBGP route convergence time

Path vector routing protocol that allows autonomous systems to exchange routing information

- Data maintained in Routing Information Base (RIB) tables
- RIB maintained through ‘update’ and ‘keepalive’ messages

Route convergence time:

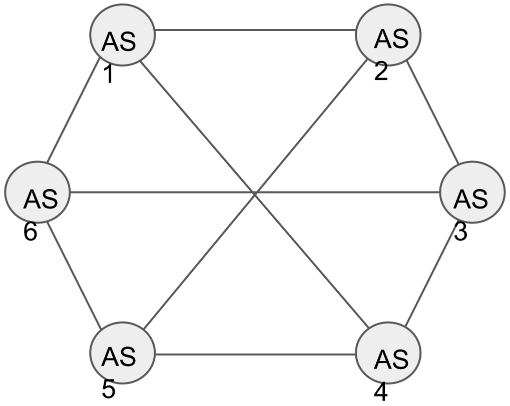
- time elapsed from the moment when a change of a route occurs until all routers accordingly adjust their routing tables

Single protocol, testing scalability and good case to compare against previous studies

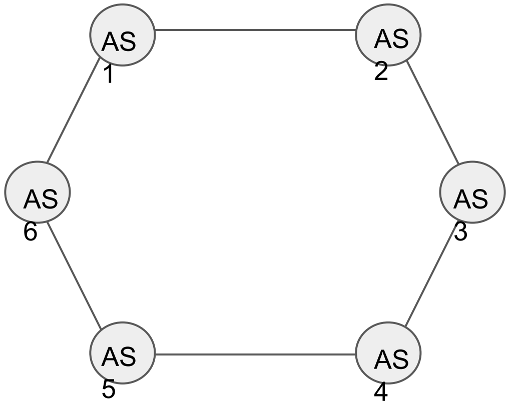
Creating the virtual testbed

Creating the topology

Mesh



Ring



cRPD configuration

- BGP peering
- Load configuration
- License

```
policy-options {
    policy-statement send-direct {
        term 1 {
            from protocol direct;
        }
    }
}
routing-options {
    autonomous-system {{ item.node_number }};
}
protocols {

    bgp {
        traceoptions {
            file bgp-traces-crpd-{{ item.node_number }} size 4294967295;
            flag update receive;
        }
        group external-peers {
            type external;
            export send-direct;
            neighbor {{ item.peer1_ip4 }} {
                peer-as {{ item.peer1_number }};
            }
            neighbor {{ item.peer2_ip4 }} {
                peer-as {{ item.peer2_number }};
            }
        }
    }
}
```

Building the routing table

- ExaBGP
  - “The BGP swiss army knife”
  - Setup BGP peering
  - announce routes
- Prefix generator
- Docker image deployed with k8s-topo

```
FROM python:3.7

EXPOSE 179

RUN pip install exabgp
RUN apt-get update && apt-get -y install vim

CMD mkfifo //run/exabgp.{in,out}
CMD chmod 666 //run/exabgp.{in,out}
CMD /bin/bash

ENTRYPOINT /bin/bash
```

Experiment setup

Experiment setup

- Ring topology
- Number of nodes: small 6, medium 30, big 100
- Number of routes: 0, 10, 100, 1000, 10000
- 5 iterations
- Azure cloud service
  - not Azure k8s service
  - VMs with k8s connected with a weave CNI
  - Meshnet, kubectl -f apply meshnet.yml
  - Docker images: cRPD and ExaBGP
  - Experiment
  - Enough VMs for 100 CPU cores

Measuring BGP route convergence

- Inject one route with ExaBGP
- Measure looking at update messages from logs

```
policy-options {
    policy-statement send-direct {
        term 1 {
            from protocol direct;
        }
    }
}
routing-options {
    autonomous-system {{ item.node_number }};
}
protocols {

    bgp {
        traceoptions {
            file bgp-traces-crpd-{{ item.node_number }} size 4294967295;
            flag update receive;
        }
        group external-peers {
            type external;
            export send-direct;
            neighbor {{ item.peer1_ip4 }} {
                peer-as {{ item.peer1_number }};
            }
            neighbor {{ item.peer2_ip4 }} {
                peer-as {{ item.peer2_number }};
            }
        }
    }
}
```

Measuring building the topology

- Wait till pods are in ready state
- Wait till pods are configured
- Wait till routing table is filled
    - show route summary

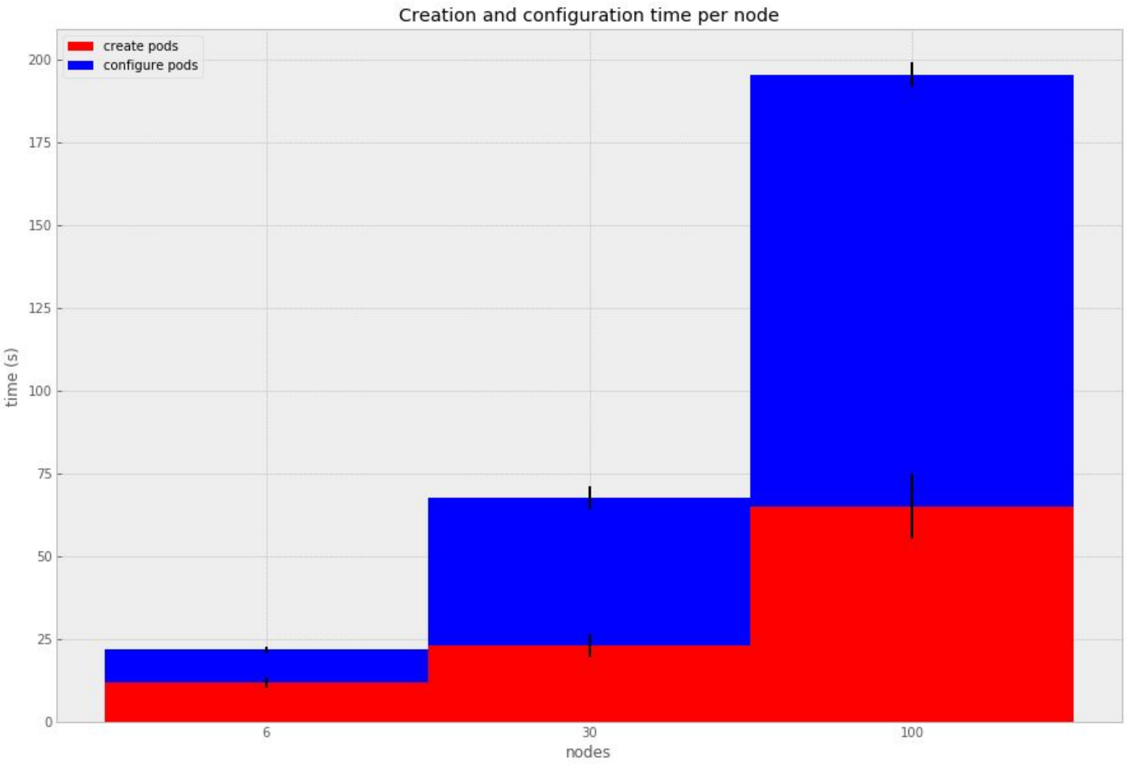
Results

Creation of the full mesh topology

- Long startup time: more than 10 minutes for 10 nodes
- Not even feasible to test 30 nodes
- Start up time increases exponentially due to the amount of links

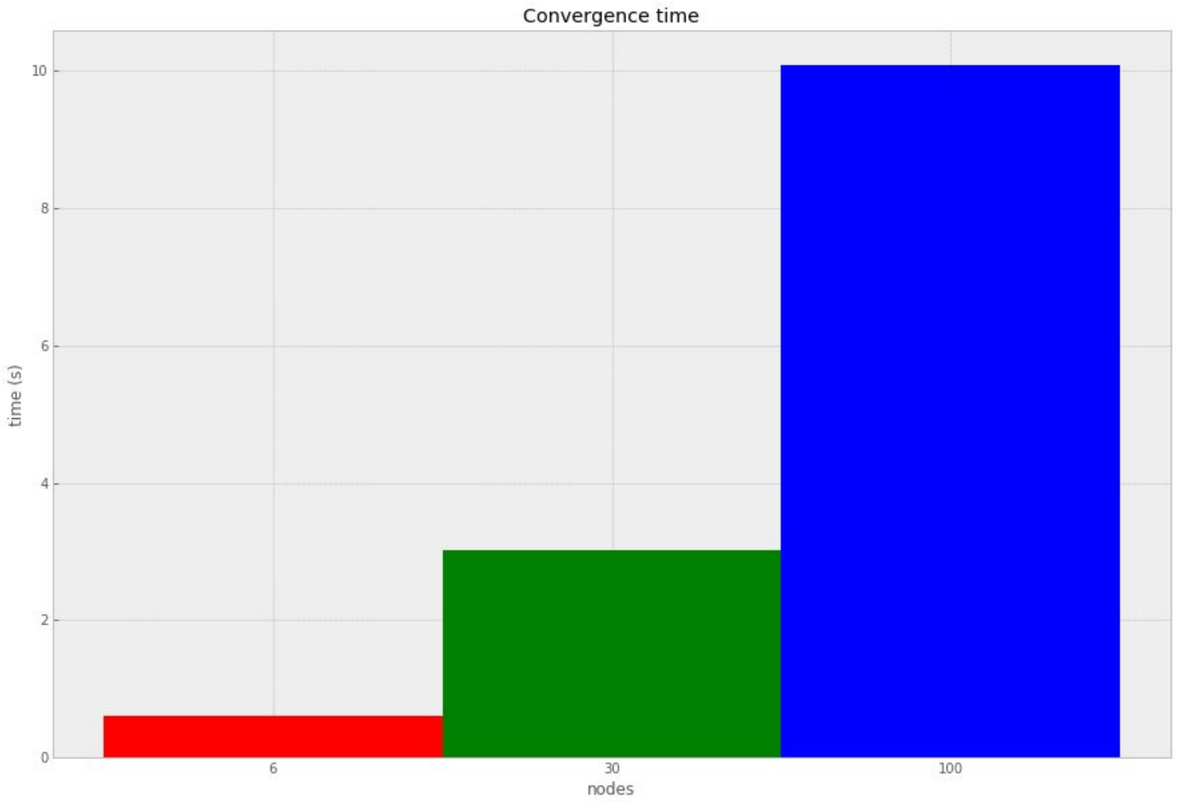
- Slow response from Meshnet with high amount of links needed to be created

Creation of the ring topology (average time)



- Startup time increases linearly
- Configuration is loaded in a sequential manner

Route convergence average time (ring topology)



- Increases linearly with the amount of nodes
- Update messages follow one path
- Consistent time results

Conclusion and Future work

Conclusions

How can a containerized testbed using cRPD be scalable in terms of number of router instances, to help SURF engineers test their network setup?

- Testbed can scale with amount of nodes but not amount of routes
- cRPD responded as expected to BGP route convergence time
- Good startup time which can be optimized further

- Does not scale with amount of links between routers
- Meshnet is a scalability bottleneck

Future Work

- Test a different network plugin instead of Meshnet
- Test Meshnet using a cluster architecture with many small nodes (resource-wise) instead of few big ones
- Test startup time with more efficient configuration loading method
- Test startup time for a configuration with more protocols