NANOG 75 Hackathon
Team ID(Pod)10T

Our Team

Mercy Onuoha, Ball State University
Jaime Botello, Riot Games
Damien Garros, Roblox
Jonathan Ballinger, Uber
Adam Mills, Roblox
Antoine Delannoy, Cloud Temple
Robert Becker, Ball State University
Anas Tarsha, Nutanix

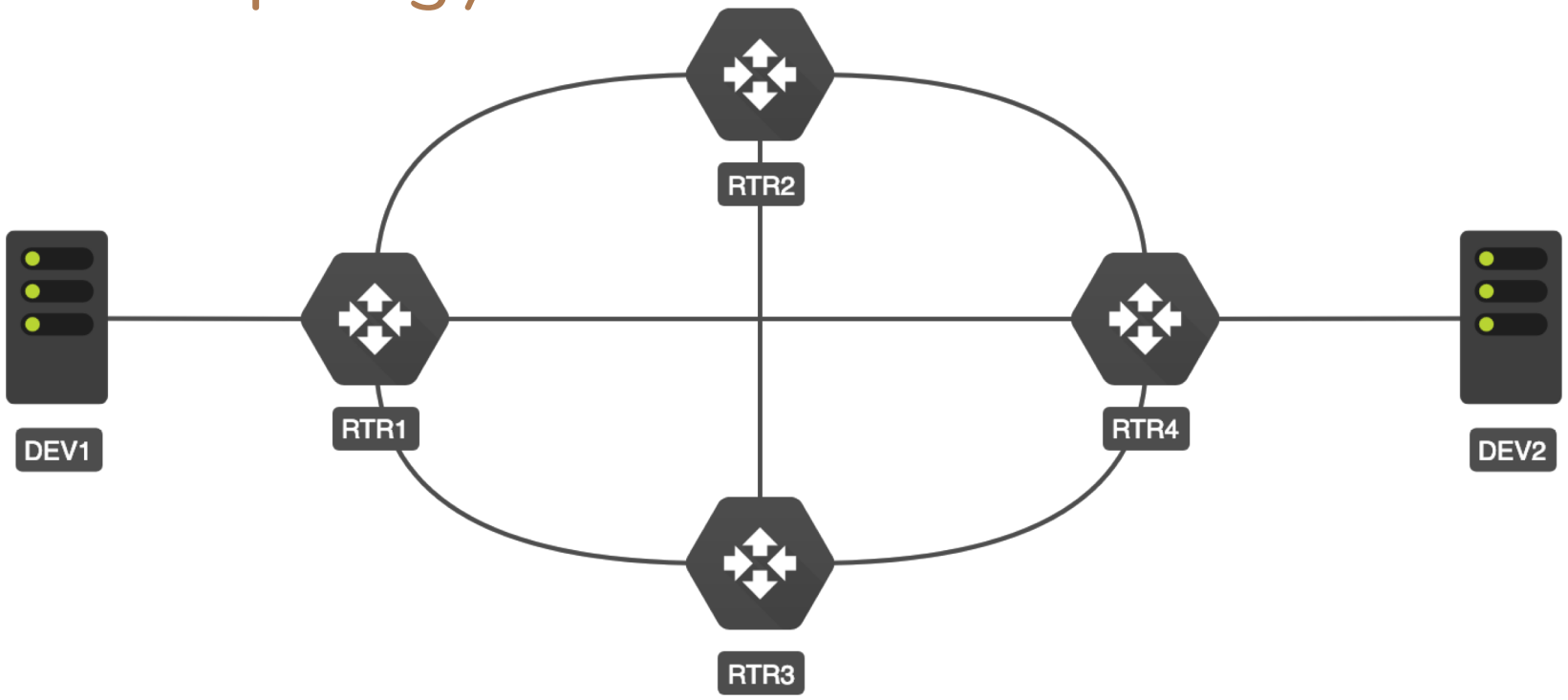
Hack Goals

Day 0 - ZTP configuration of routers via a Python bootstrap script

Day 1 - Generate streaming telemetry from the router using gNMI and send it to a Kafka database

Day 2 - Inject static LSPs into device RIBs via gRIBI, with the optimal paths being determined via the telemetry data from Day 1

Lab Topology



Day 0

```
XML_FILE_LIST =  ['grpc_config.xml',
                  'hostname.xml',
                  'mpls_static.xml',
                  'intafaces_oc.xml',
                  'lldp_oc.xml']
```

- Create snippets for device interfaces and LLDP using YANG models
- **Don't be afraid to use vendor specific models, they are there for a reason**

- Router uses its own serial number to determine which location to pull config snippets from, eliminating the need to have MAC addresses for each device
- Router can be queried by the script to gain other information such as vendor, model, etc

```
ztp_serial = ztp_script.get_serial_number()
if ztp_serial is None:
        ztp_script.syslogger.info("Serial Number not Extracted, aborting...")
        sys.exit(1)
ztp_hostname = SERIAL_NO_MAP[ztp_serial]
for config in ZTP_CONFIG_XML:
    cmd = "wget " + SERVER_XML_URL + ztp_hostname + "/"+ config
    response = ztp_script.run_bash(cmd)
    ztp_script.load_diff(nc_mgr, config)
nc_mgr.close_session()
```

Day 0

```
def load_diff(self, nc_mgr, diff_file):
    try:
        conf_file = open(diff_file)
        x_read = conf_file.read()
        rpc_obj = nc_mgr.edit_config( config=x_read, format='xml')
        nc_mgr.commit()
    except Exception:
        self.logger.exception("Exception in loading config")
```

- Modify provided ZTP bootscript to load XML config snippets and apply them to the device configuration

DON’T FORGET TO COMMIT THE CONFIG!

Day1 | OpenConfig BGP

```
# Configure global bgp parameters
afi_safi = protocol.bgp.global_.afi_safis.AfiSafi()
afi_safi.afi_safi_name = oc_bgp_types.IPV4UNICAST()
afi_safi.config.enabled = True
protocol.bgp.global_.afi_safis.afi_safi.append(afi_safi)

# Configure the peer-group and address family
peer_group = protocol.bgp.peer_groups.PeerGroup()
peer_group.peer_group_name = bgp["peer-group-name"]
peer_group.config.peer_group_name = bgp["peer-group-name"]
peer_group.config.peer_as = int(bgp["peer-as"])
peer_group.transport.config.local_address = bgp["peer-group-local-address"]
afi_safi = peer_group.afi_safis.AfiSafi()
afi_safi.afi_safi_name = oc_bgp_types.IPV4UNICAST()
afi_safi.config.enabled = True
peer_group.afi_safis.afi_safi.append(afi_safi)

# Configure the bgp neighbor with peer-group
neighbor = protocol.bgp.neighbors.Neighbor()
neighbor.neighbor_address = bgp["neighbor"]
neighbor.config.neighbor_address = bgp["neighbor"]
neighbor.config.peer_group = bgp["peer-group-name"]

# Fill out your BGP object properly using the Openconfig Yang Model
# You will need to bring up the IBGP neighbor between rtr1 and rtr4
protocol.bgp.peer_groups.peer_group.append(peer_group)
protocol.bgp.neighbors.neighbor.append(neighbor)
ni.protocols.protocol.append(protocol)
```

- BGP configuration on the edges, RTR1 and RTR4
- Configuration for BGP peers using OpenConfig YANG models and YDK

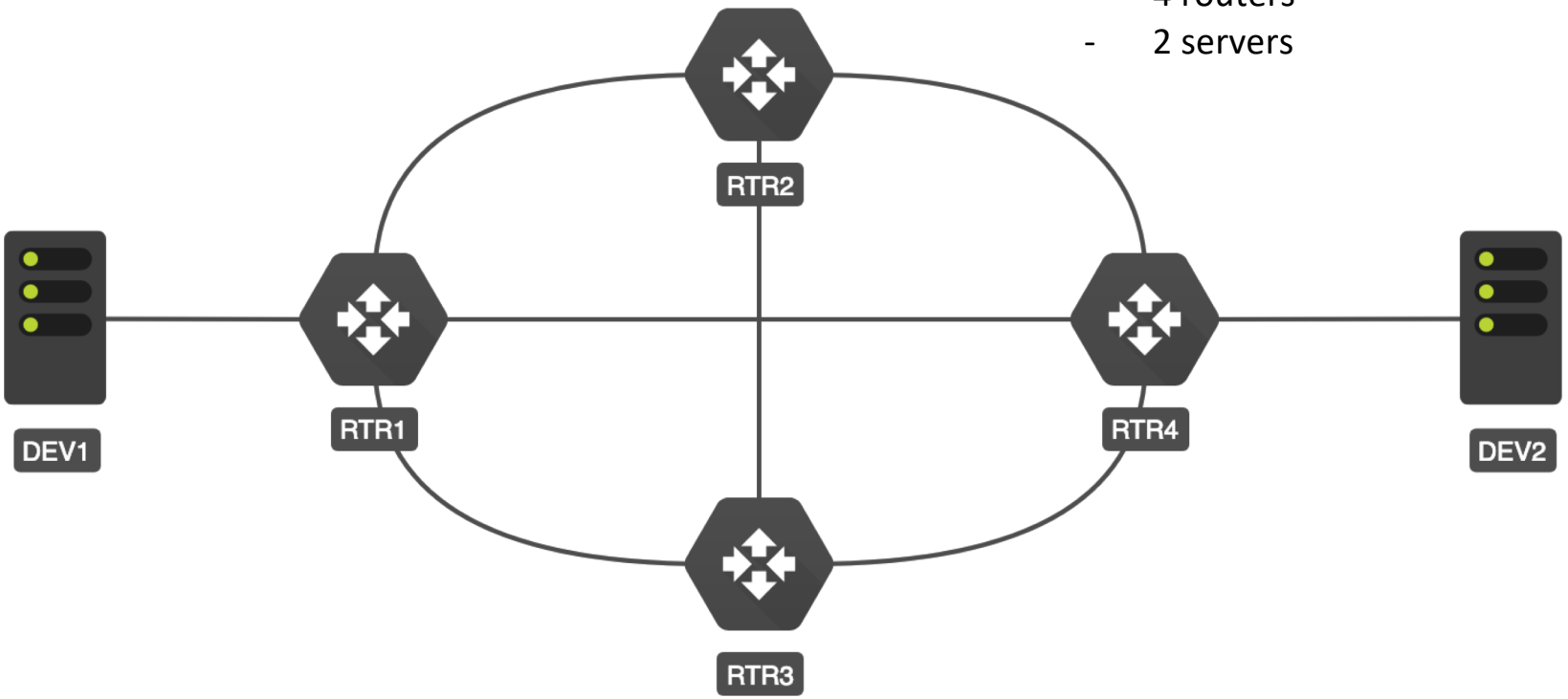
Day1 | OpenR

```
- name: Start OpenR container
  docker_container:
    image: akshshar/openr-xr
    command: /root/run_openr.sh
    restart: yes
    name: openr
    volumes:
      - /var/run/netns:/var/run/netns
      - /misc/app_host:/root
      - "/misc/app_host/hosts_{{inventory_hostname}}:/etc/hosts"
    hostname: "{{ inventory_hostname }}"
    docker_host: "unix:///misc/app_host/docker.sock"
    capabilities: [ SYS_ADMIN, NET_ADMIN]
```

- Open/R running on all routers as IGP
- Open/R docker container pushed to all routers using Ansible native modules

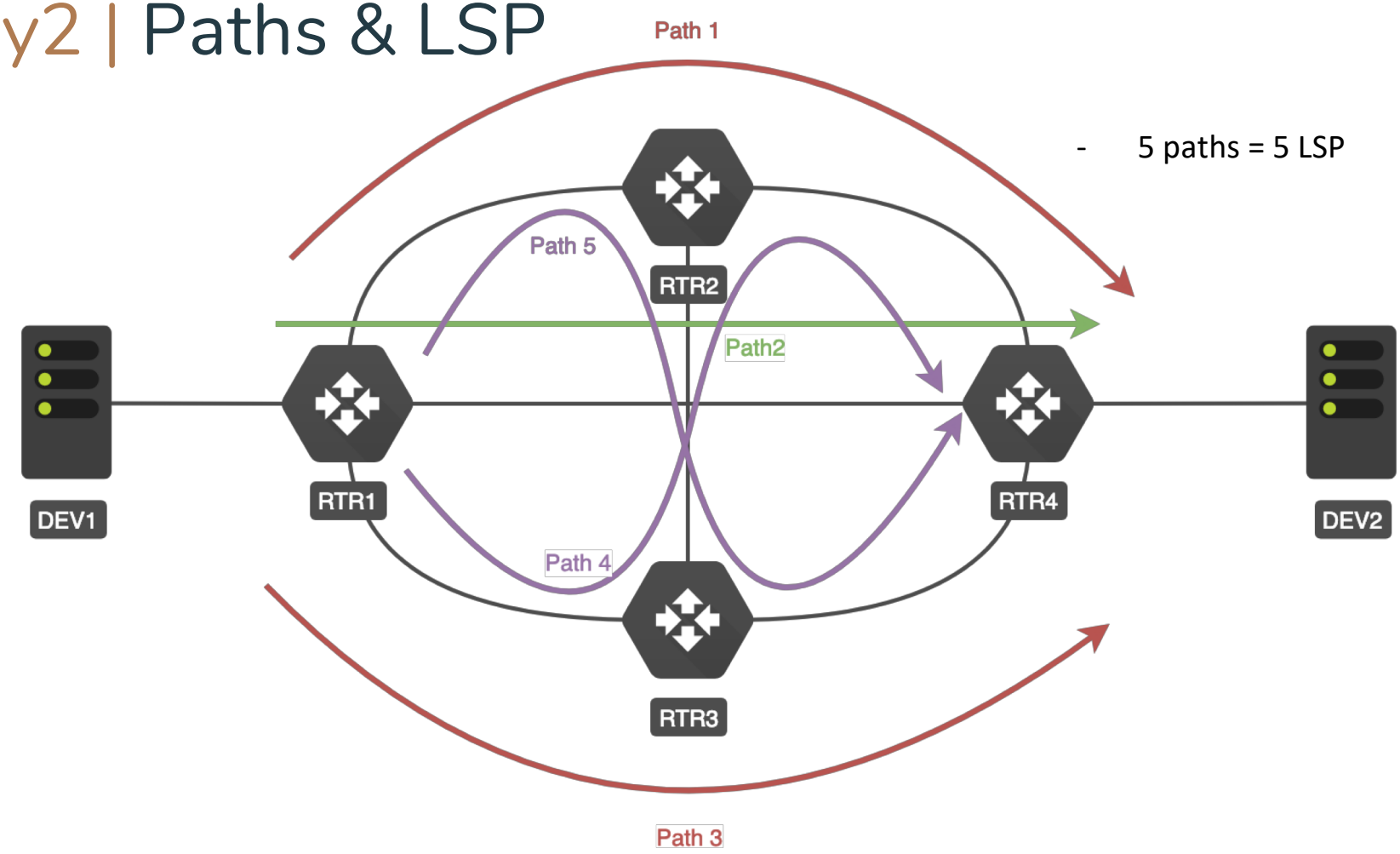
Day2 | Topology

- MPLS Network
- 4 routers
- 2 servers

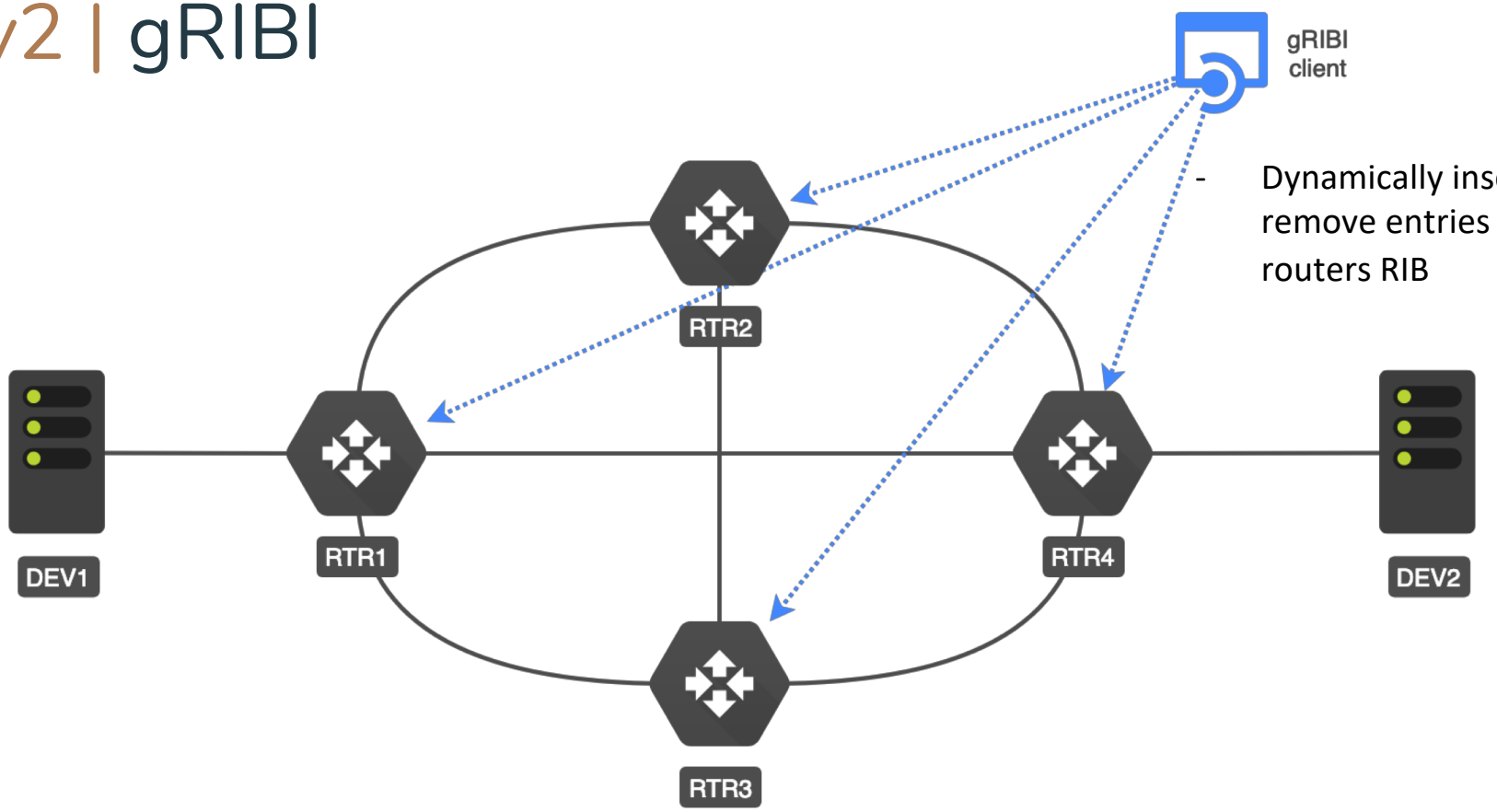


Day2 | Paths & LSP

- 5 paths = 5 LSP

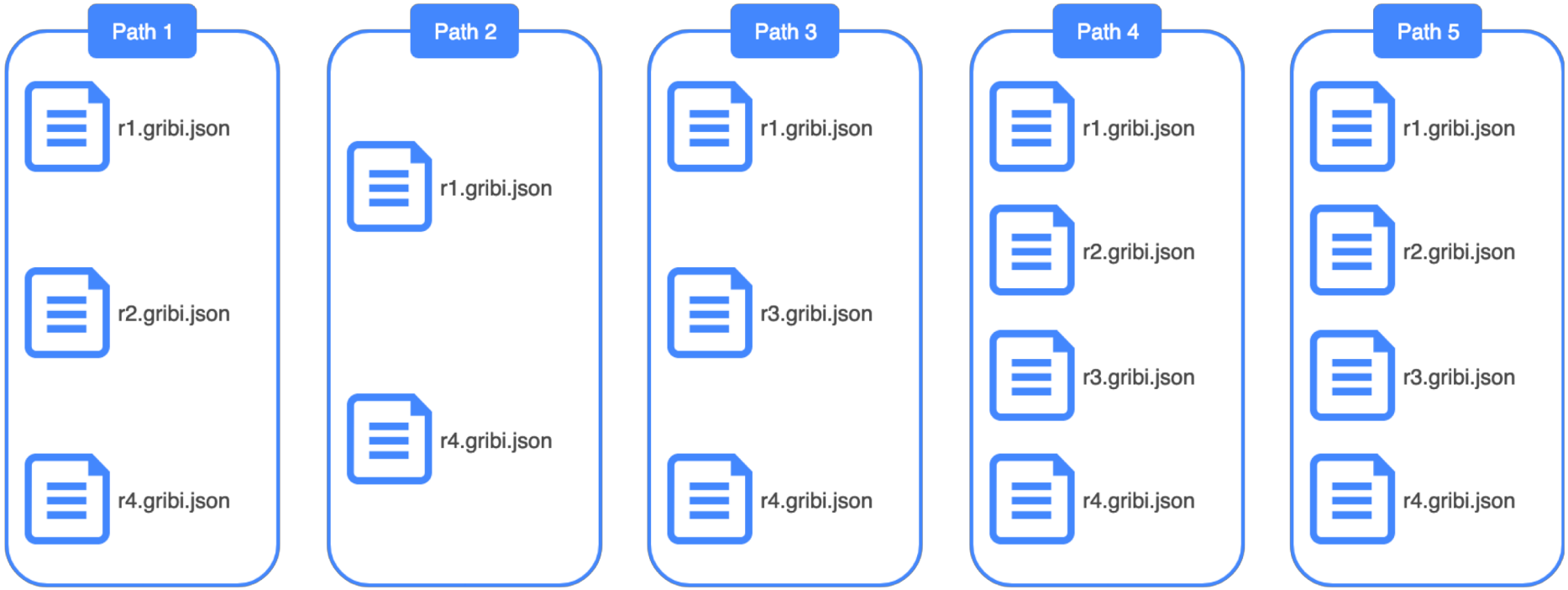


Day2 | gRIBI



- Dynamically insert or remove entries in routers RIB

Day2 | LSP templating



Day2 | LSP templating

- One JSON template per router per LSP
- The template specifies which entry to insert in the RIB :
  - IP / MPLS route
  - Label (static)
  - Operation (PUSH, SWAP, POP)

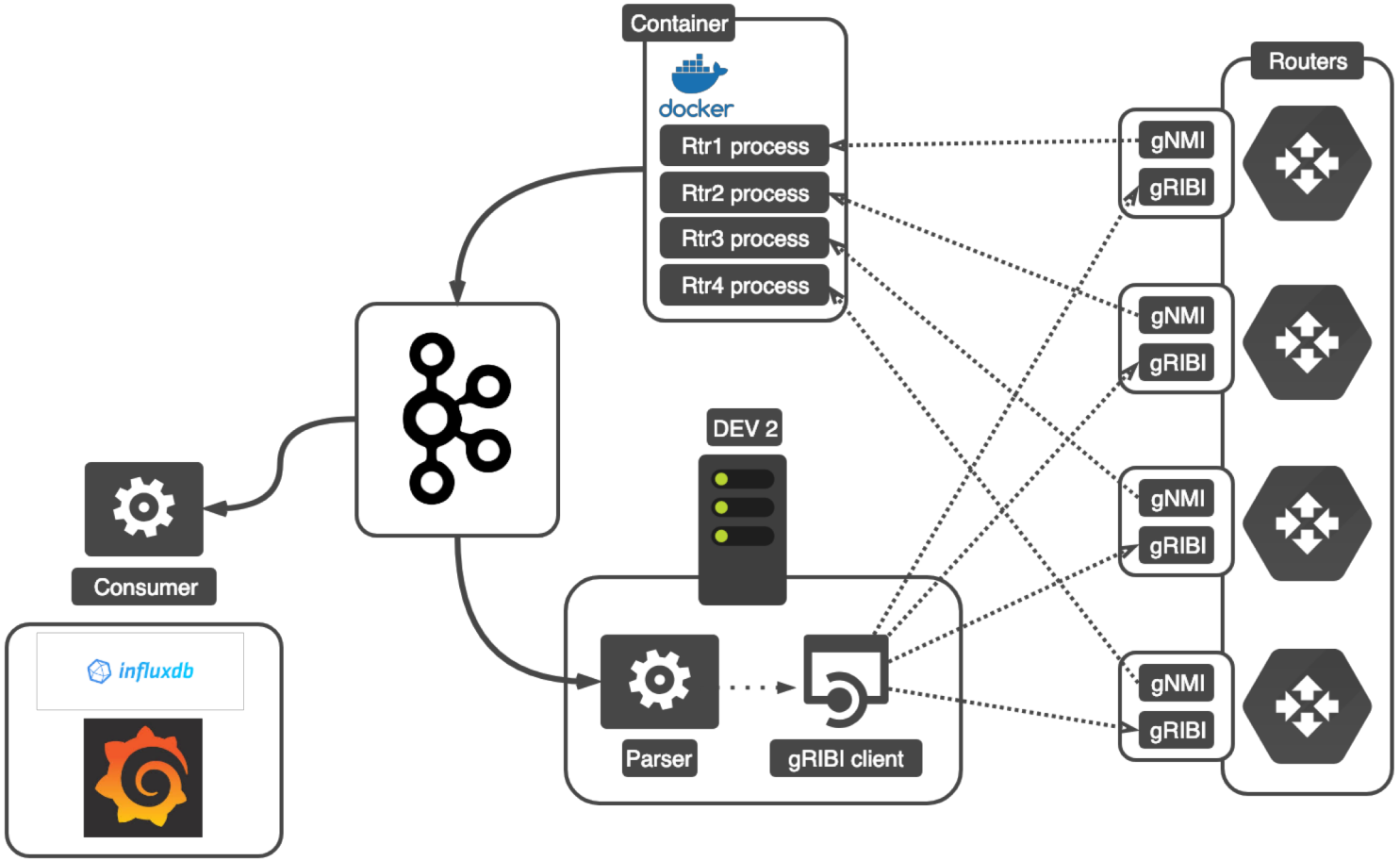
```
    "gribi_nhs" : [
        {
            "id" : 3000,
            "inst_name" : "default",
            "key_index" : 1,
            "entry" : {
                "encap_header_type" : "OPENCONFIGAFTENCAPSULATIONHEADERTYPE_IPV4",
                "int" : "GigabitEthernet0/0/0/1",
                "subint" : 0,
                "ip_address": "10.5.1.10",
                "mac_address" : "",
                "origin_protocol": "OPENCONFIGPOLICYTYPESINSTALLPROTOCOLTYPE_STATIC",
                "pushed_mpls_label_stack" : [
                    33333
                ]
            }
        },
```

```
    "gribi_routes" : [
        {
            "id" : 3200,
            "inst_name" : "default",
            "entry" : {
                "prefix" : "10.1.1.0/24",
                "type" : "v4",
                "encap_header_type" : "OPENCONFIGAFTENCAPSULATIONHEADERTYPE_IPV4",
                "next_hop_group" : 1,
                "octets_forwarded" : 0,
                "packets_forwarded" : 0
            }
        }
    ],
    "gribi_mpls" : [
        {
            "id" : 3300,
            "inst_name" : "default",
            "label" : 33333,
            "entry" : {
                "nh_group" : 2,
                "octets_forwarded" : 1,
                "packets_forwarded" : 1,
                "popped_mpls_label_stack" : [
                    33333
                ]
            }
        }
    ]
```

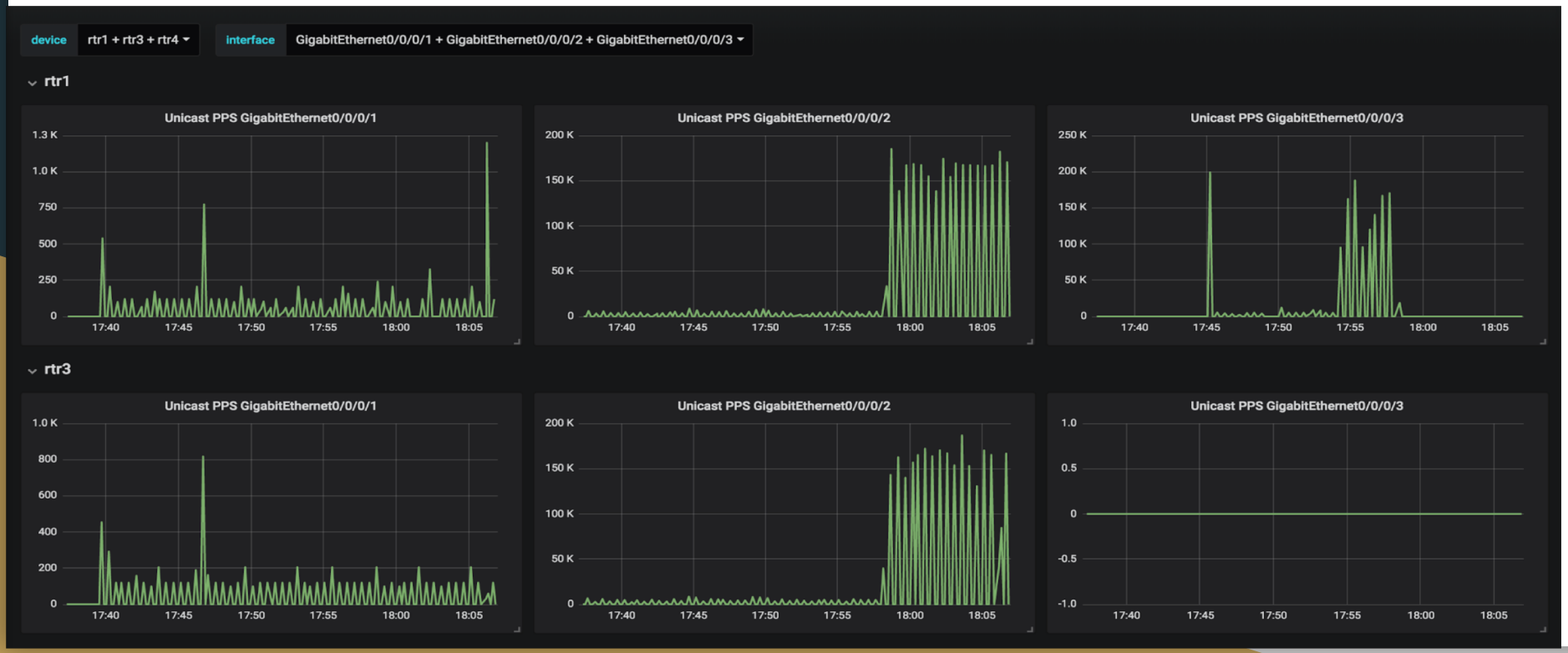
Day2 | Telemetry Parsing

- Parser written in Python, running on Dev2 server
- Consumes telemetry data from Day 1 as JSON messages using Kafka
- Maintains a view of the current network state using interface oper-status change events
- Triggers LSP setup/deletion according to network state
- Path selection determined by a priority map

Day2 | Macro scheme

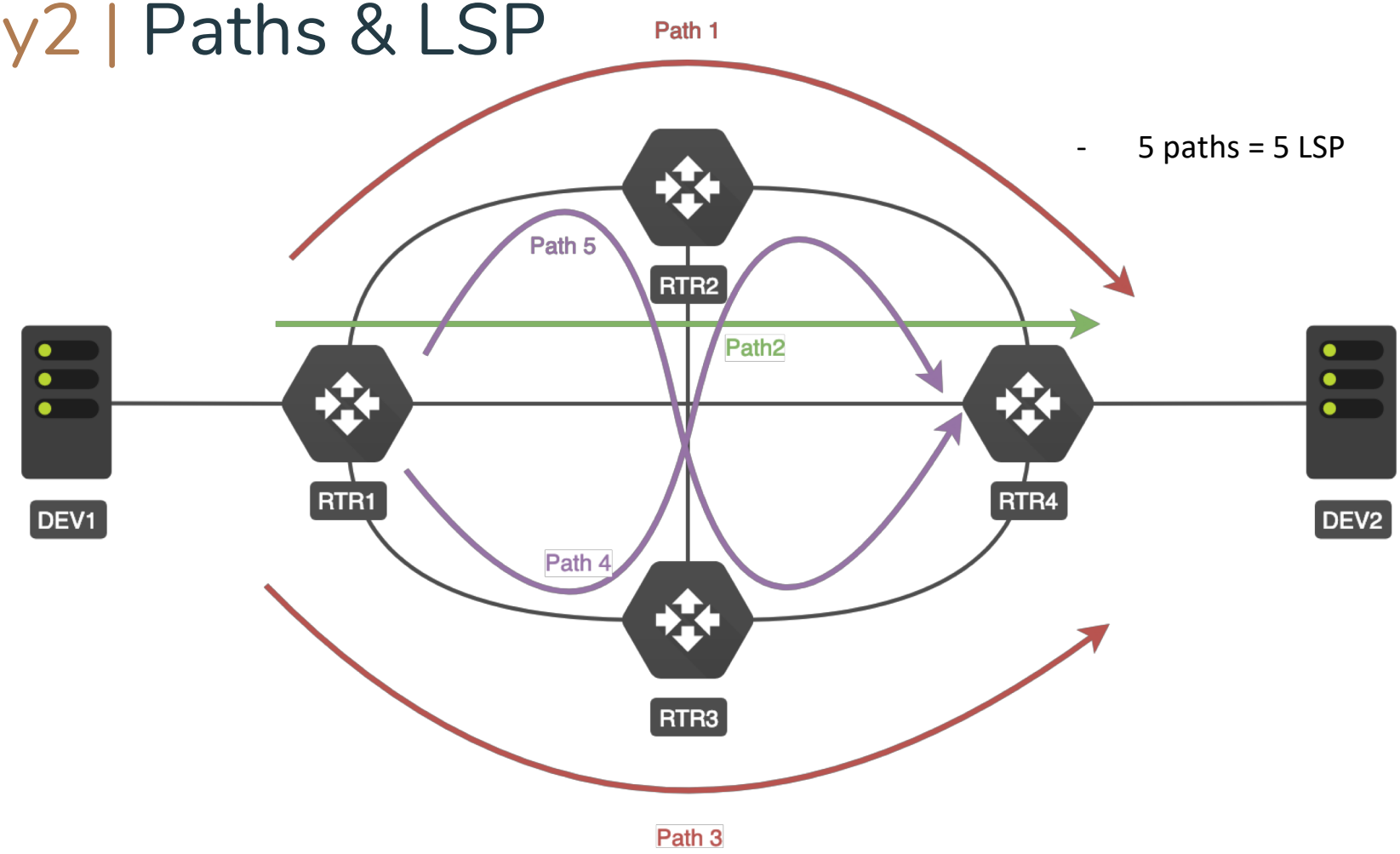


Day1 | Telemetry Collector to TSDB

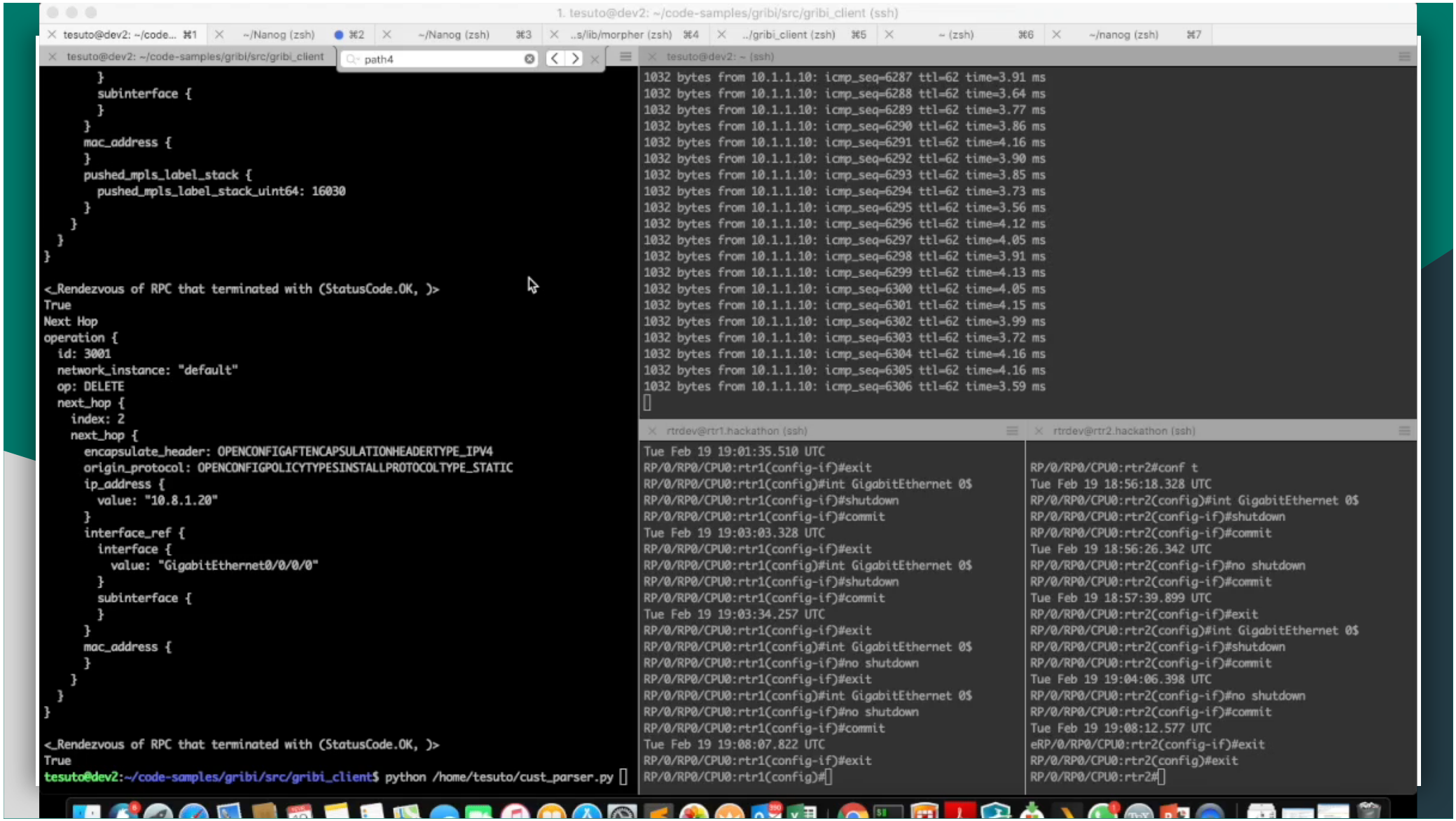


Day2 | Paths & LSP

- 5 paths = 5 LSP



Demo Time!


1. tesuto@dev2: ~/code-samples/gribi/src/gribi_client (ssh)
tesuto@dev2: ~/code... ⌘1
~/Nanog (zsh) ⌘2
~/Nanog (zsh) ⌘3
../lib/morpher (zsh) ⌘4
../gribi_client (zsh) ⌘5
~ (zsh) ⌘6
~/nanog (zsh) ⌘7
tesuto@dev2: ~/code-samples/gribi/src/gribi_client
path4
subinterface {
}
}
mac_address {
}
pushed_mpls_label_stack {
pushed_mpls_label_stack_uint64: 16030
}
}
}
}
<_Rendezvous of RPC that terminated with (StatusCode.OK,)>
True
Next Hop
operation {
id: 3001
network_instance: "default"
op: DELETE
next_hop {
index: 2
next_hop {
encapsulate_header: OPENCONFIGAFTENCAPSULATIONHEADERTYPE_IPV4
origin_protocol: OPENCONFIGPOLICYTYPESINSTALLPROTOCOLTYPE_STATIC
ip_address {
value: "10.8.1.20"
}
interface_ref {
interface {
value: "GigabitEthernet0/0/0/0"
}
subinterface {
}
}
mac_address {
}
}
}
}
<_Rendezvous of RPC that terminated with (StatusCode.OK,)>
True
tesuto@dev2:~/code-samples/gribi/src/gribi_client$ python /home/tesuto/cust_parser.py
tesuto@dev2: ~ (ssh)
1032 bytes from 10.1.1.10: icmp_seq=6287 ttl=62 time=3.91 ms
1032 bytes from 10.1.1.10: icmp_seq=6288 ttl=62 time=3.64 ms
1032 bytes from 10.1.1.10: icmp_seq=6289 ttl=62 time=3.77 ms
1032 bytes from 10.1.1.10: icmp_seq=6290 ttl=62 time=3.86 ms
1032 bytes from 10.1.1.10: icmp_seq=6291 ttl=62 time=4.16 ms
1032 bytes from 10.1.1.10: icmp_seq=6292 ttl=62 time=3.90 ms
1032 bytes from 10.1.1.10: icmp_seq=6293 ttl=62 time=3.85 ms
1032 bytes from 10.1.1.10: icmp_seq=6294 ttl=62 time=3.73 ms
1032 bytes from 10.1.1.10: icmp_seq=6295 ttl=62 time=3.56 ms
1032 bytes from 10.1.1.10: icmp_seq=6296 ttl=62 time=4.12 ms
1032 bytes from 10.1.1.10: icmp_seq=6297 ttl=62 time=4.05 ms
1032 bytes from 10.1.1.10: icmp_seq=6298 ttl=62 time=3.91 ms
1032 bytes from 10.1.1.10: icmp_seq=6299 ttl=62 time=4.13 ms
1032 bytes from 10.1.1.10: icmp_seq=6300 ttl=62 time=4.05 ms
1032 bytes from 10.1.1.10: icmp_seq=6301 ttl=62 time=4.15 ms
1032 bytes from 10.1.1.10: icmp_seq=6302 ttl=62 time=3.99 ms
1032 bytes from 10.1.1.10: icmp_seq=6303 ttl=62 time=3.72 ms
1032 bytes from 10.1.1.10: icmp_seq=6304 ttl=62 time=4.16 ms
1032 bytes from 10.1.1.10: icmp_seq=6305 ttl=62 time=4.16 ms
1032 bytes from 10.1.1.10: icmp_seq=6306 ttl=62 time=3.59 ms
rtrdev@rtr1.hackathon (ssh)
Tue Feb 19 19:01:35.510 UTC
RP/0/RP0/CPU0:rtr1(config-if)#exit
RP/0/RP0/CPU0:rtr1(config)#int GigabitEthernet 0$
RP/0/RP0/CPU0:rtr1(config-if)#shutdown
RP/0/RP0/CPU0:rtr1(config-if)#commit
Tue Feb 19 19:03:03.328 UTC
RP/0/RP0/CPU0:rtr1(config-if)#exit
RP/0/RP0/CPU0:rtr1(config)#int GigabitEthernet 0$
RP/0/RP0/CPU0:rtr1(config-if)#shutdown
RP/0/RP0/CPU0:rtr1(config-if)#commit
Tue Feb 19 19:03:34.257 UTC
RP/0/RP0/CPU0:rtr1(config-if)#exit
RP/0/RP0/CPU0:rtr1(config)#int GigabitEthernet 0$
RP/0/RP0/CPU0:rtr1(config-if)#no shutdown
RP/0/RP0/CPU0:rtr1(config-if)#exit
RP/0/RP0/CPU0:rtr1(config)#int GigabitEthernet 0$
RP/0/RP0/CPU0:rtr1(config-if)#no shutdown
RP/0/RP0/CPU0:rtr1(config-if)#commit
Tue Feb 19 19:08:07.822 UTC
RP/0/RP0/CPU0:rtr1(config-if)#exit
RP/0/RP0/CPU0:rtr1(config)#
rtrdev@rtr2.hackathon (ssh)
RP/0/RP0/CPU0:rtr2#conf t
Tue Feb 19 18:56:18.328 UTC
RP/0/RP0/CPU0:rtr2(config)#int GigabitEthernet 0$
RP/0/RP0/CPU0:rtr2(config-if)#shutdown
RP/0/RP0/CPU0:rtr2(config-if)#commit
Tue Feb 19 18:56:26.342 UTC
RP/0/RP0/CPU0:rtr2(config-if)#no shutdown
RP/0/RP0/CPU0:rtr2(config-if)#commit
Tue Feb 19 18:57:39.899 UTC
RP/0/RP0/CPU0:rtr2(config-if)#exit
RP/0/RP0/CPU0:rtr2(config)#int GigabitEthernet 0$
RP/0/RP0/CPU0:rtr2(config-if)#shutdown
RP/0/RP0/CPU0:rtr2(config-if)#commit
Tue Feb 19 19:04:06.398 UTC
RP/0/RP0/CPU0:rtr2(config-if)#no shutdown
RP/0/RP0/CPU0:rtr2(config-if)#commit
Tue Feb 19 19:08:12.577 UTC
eRP/0/RP0/CPU0:rtr2(config-if)#exit
RP/0/RP0/CPU0:rtr2(config)#exit
RP/0/RP0/CPU0:rtr2#