Full Automation of CLOS Networks

Zoe Blevins, Yihua He

Agenda

- Motivation
- Components Needed
  - Network Model
  - Live State
  - Device Agents
  - Config Generation System
  - Config Delivery System
- TOR Turnup Use Case
- Results
- Future Application

Motivation

- Scale of CLOS Networks is not manageable without automation
- Problem: Networks are not static
    - The desired state of the network is mutable
    - The desired configuration is a function of desired state and current state
- Solution:
    - A state machine can be used to track the progression from current state to desired state

Network Model

- Dictates the desired state of the network
- Deterministically defines:
  - IP Addressing Scheme
  - Topology
  - Prefix Lists
  - Etc
- Inputs
  - Aggregate Host Subnets
  - TOR Host Subnets
  - Cluster Level Settings
    - Auth hosts, Log servers, etc

Live State Telemetry

- Live metrics and metadata are sent into our collection system at least every 30s by switch agents
- Metrics are used to:
  - Calculate port state to advance incremental config changes
  - Monitor system health
- Metadata is used for tracking the advancement of the config delivery system
- Allows for the system to react to the live state

Device Agent

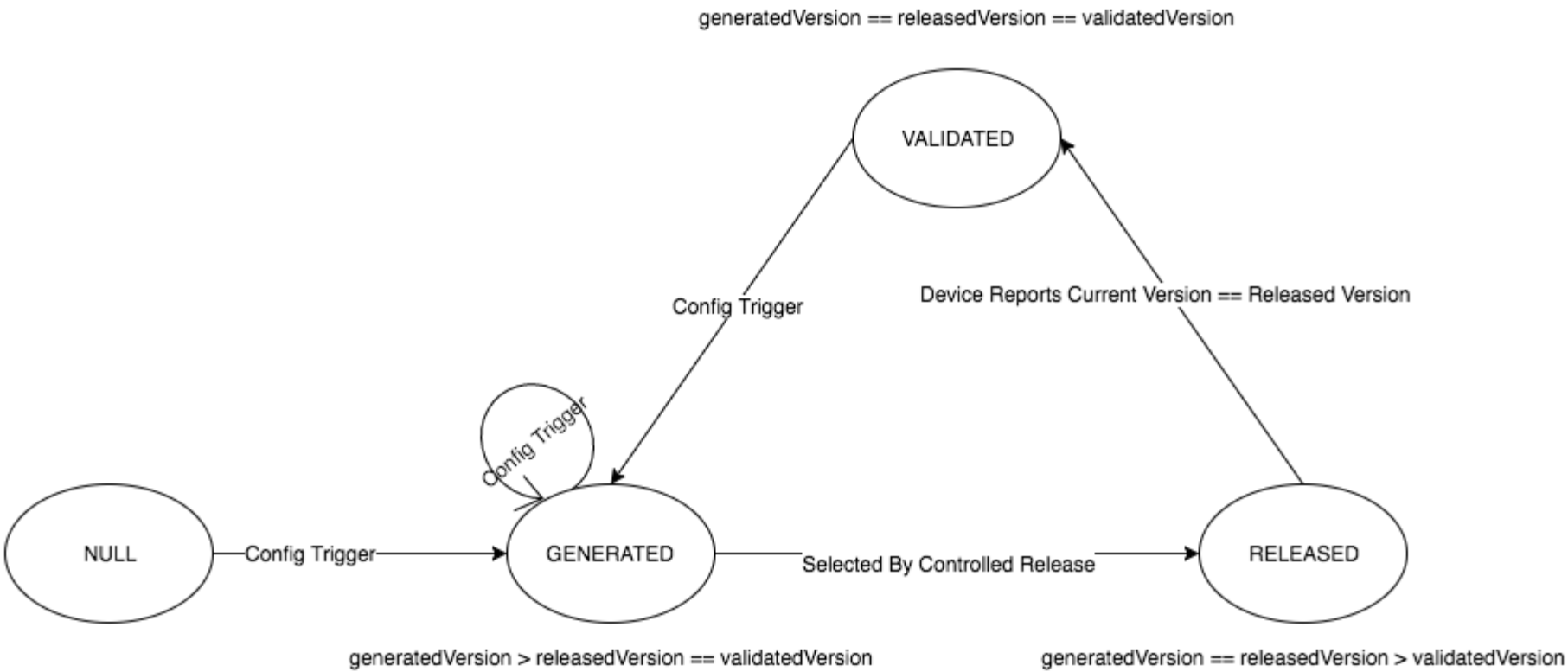
- Metrics
  - Every 10-30s metrics are POSTed to our live state collection system
  - Metrics include: Link/Admin State, Input/Output/PLoss Rate, LLDP Data etc.
- Config Application
  - Loads configuration file over HTTP
  - Performs Config Replace
  - Runs Healthchecks to validate config:
    - Config Replace Succeeded
    - Default Route Present
    - % change in Prefixes received/advertised
    - Ping Success to select targets
  - Performs rollback and reports failure to system

Config Generation System

- Goal: Reconcile desired state with live state to produce a full config for each device in the cluster
- Inputs
  - Network Model
  - Live State
  - External Data Sources (Asset Management, IPAM, etc)
  - Device Templates (Apache Velocity)
- Triggers
  - Cluster Creation
  - New TOR Allocation
  - Port State Changes
  - Anchored Subnet Changes
  - Template Changes

Config Delivery System

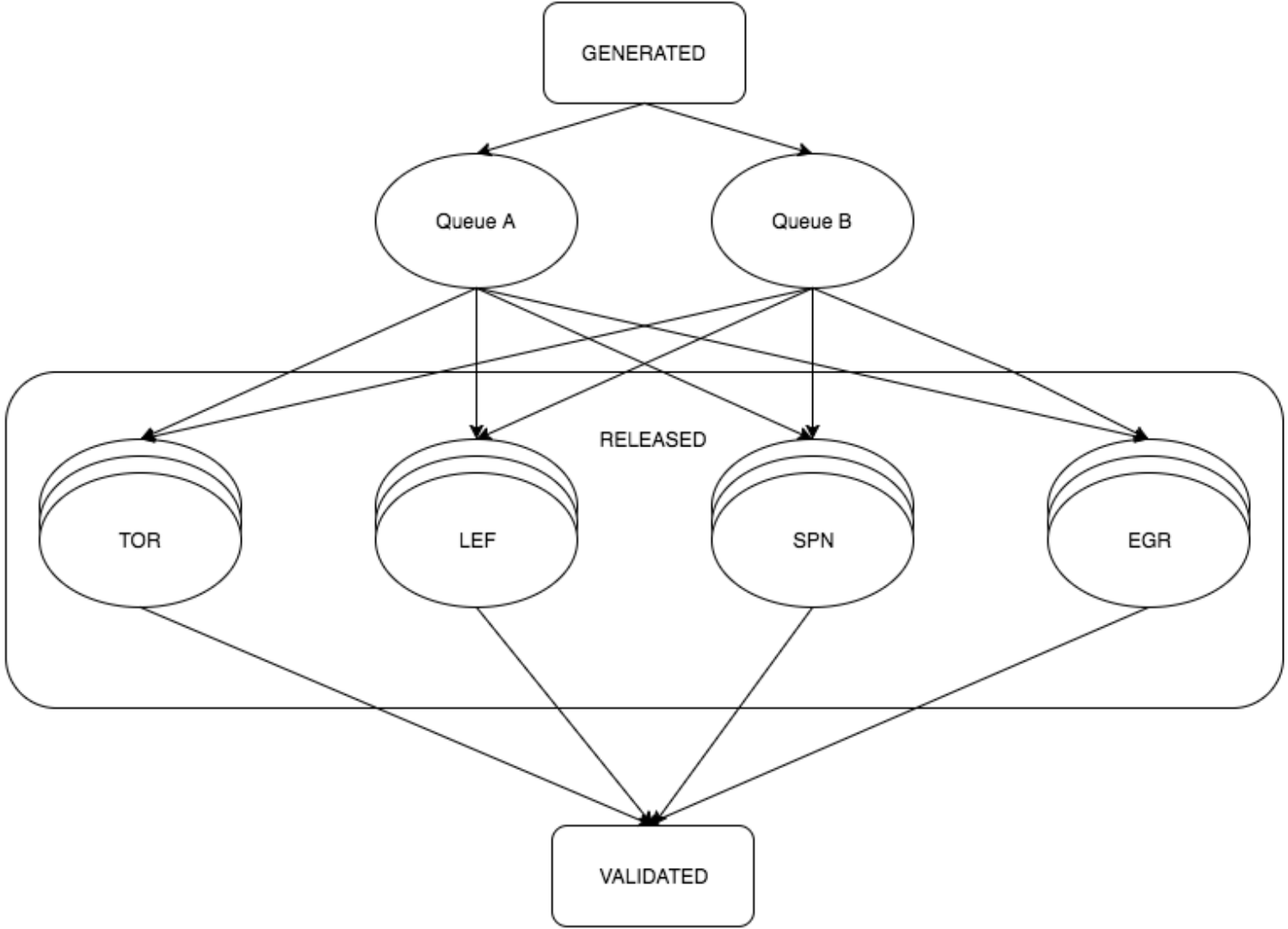
- Goal: Release configuration changes in a rapid and controlled manner
- Solution: State Machine
  - 3 distinct states determined by config versions
    - GENERATED
      - generatedVersion > releasedVersion == validatedVersion
    - RELEASED
      - generatedVersion == releasedVersion > validatedVersion
    - VALIDATED
      - generatedVersion == releasedVersion == validatedVersion
  - Fully Asynchronous and Parallelized
  - State is persisted in DB


generatedVersion == releasedVersion == validatedVersion
VALIDATED
Config Trigger
Device Reports Current Version == Released Version
Config Trigger
NULL
Config Trigger
GENERATED
Selected By Controlled Release
RELEASED
generatedVersion > releasedVersion == validatedVersion
generatedVersion == releasedVersion > validatedVersion

Controlled Release

- Devices are sorted into two queues
  - Planes are separated based on physical redundancy
  - QueueA contains all switches in plane A
  - QueueB contains all switches in plane B
  - Queues do not advance unless healthchecks pass

- Existing configurations in the RELEASED state are sorted into sub-queues by device type
  - A limited number of each device type may be in the RELEASED state at any given time
  - When a slot in the sub-queue opens, a switch in the GENERATED state may then enter the queue


GENERATED
Queue A
Queue B
RELEASED
TOR
LEF
SPN
EGR
VALIDATED

Safeguards

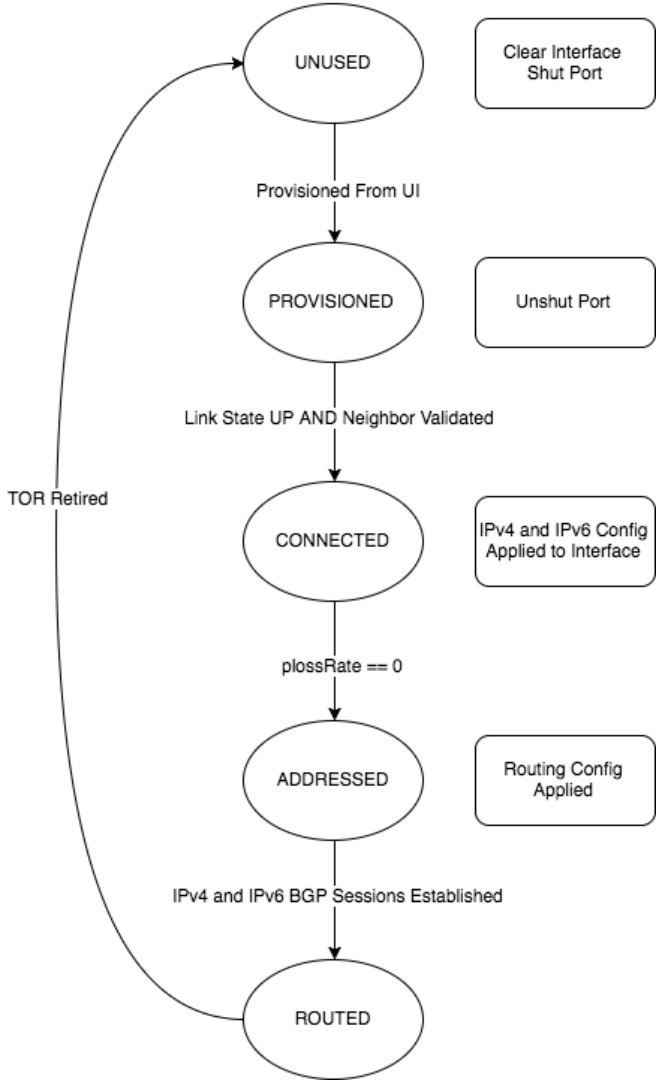
- Input Validation for Model Changes
- Peer-Review Mechanism for Cluster Setting Changes
- Granular Enable/Disable Switch
  - “Big Red Button”
  - System can be enabled/disabled at both the switch level and cluster level
  - Allows for:
    - Preventing further propagation of errant changes
    - Canary Testing large scale changes
    - Removing faulty gear from blocking queues
    - Postponing changes until a maintenance window
- Manual Rollback CLI on Switches
- Monitoring UIs

Automated Tasks

- Cluster Creation
- Template Changes
- Aggregate Subnet Changes
- New Device Allocation
- Host Subnet Changes
- TOR Retire
- Cluster Setting Changes

TOR Turnup Use Case

1. SysEng requests X new TORs with /Y subnets
2. New TOR switches are allocated in the model
    i. This includes external DB population
3. TOR Switch Configs Get Generated and Released
4. LEF Switch Configs Get Generated and Released one plane at a time
5. LEF loads new configuration
6. TOR is plugged in by SiteOps
7. TOR loads new configuration
8. LEF Reports updated interface state data
9. New LEF Config Generated and Released
10. LEF loads new configuration
11. Steps 8-10 are repeated until BGP Sessions are Established




```
#foreach ($interface in $switch.links)##
        interface $interface.name
        #if ($interface.stateIndex == 0)##
                shutdown
         #else##
                 [...]
                no shutdown
                #if ($interface.stateIndex > 1) ##
                                [...]
                        ip address ${interface.sourceIPv4}/31
                        ipv6 address ${interface.sourceIPv6}/127
                        [...]
                #end## ENDIF interface state
        #end## ENDIF interface state
#end## END Foreach
[...]
#foreach($interface in $switch.links)##
        #if ($interface.stateIndex > 2)##
                                 [...]
                                neighbor $interface.destIPv4 description
$interface.destSwitch
                                no neighbor $interface.destIPv4 shutdown
                                neighbor $interface.destIPv6 description $interface.destSwitch
                                no neighbor $interface.destIPv6 shutdown
                                [...]
                 #end## ENDIF interface state
#end## END foreach
```

Results

- No Manual Configuration or Config Application
- No NetOps intervention needed for common tasks
- Switch Config Changes converge in ~30s - 2m
- Cluster wide changes propagate in ~10 - 15m
- **Capable of provisioning a 20,000 node cluster in under 10 minutes with *no* errors**

Future Applications

- Expand model beyond CLOS Network
- Self-Healing based on Live State Changes
- Dynamic Provisioning of Host IPs
- Dynamic Allocation for Increasing Capacity

Q&A

Further Questions/Concerns?

- zblevins@yahoo-inc.com
- hyihua@yahoo-inc.com