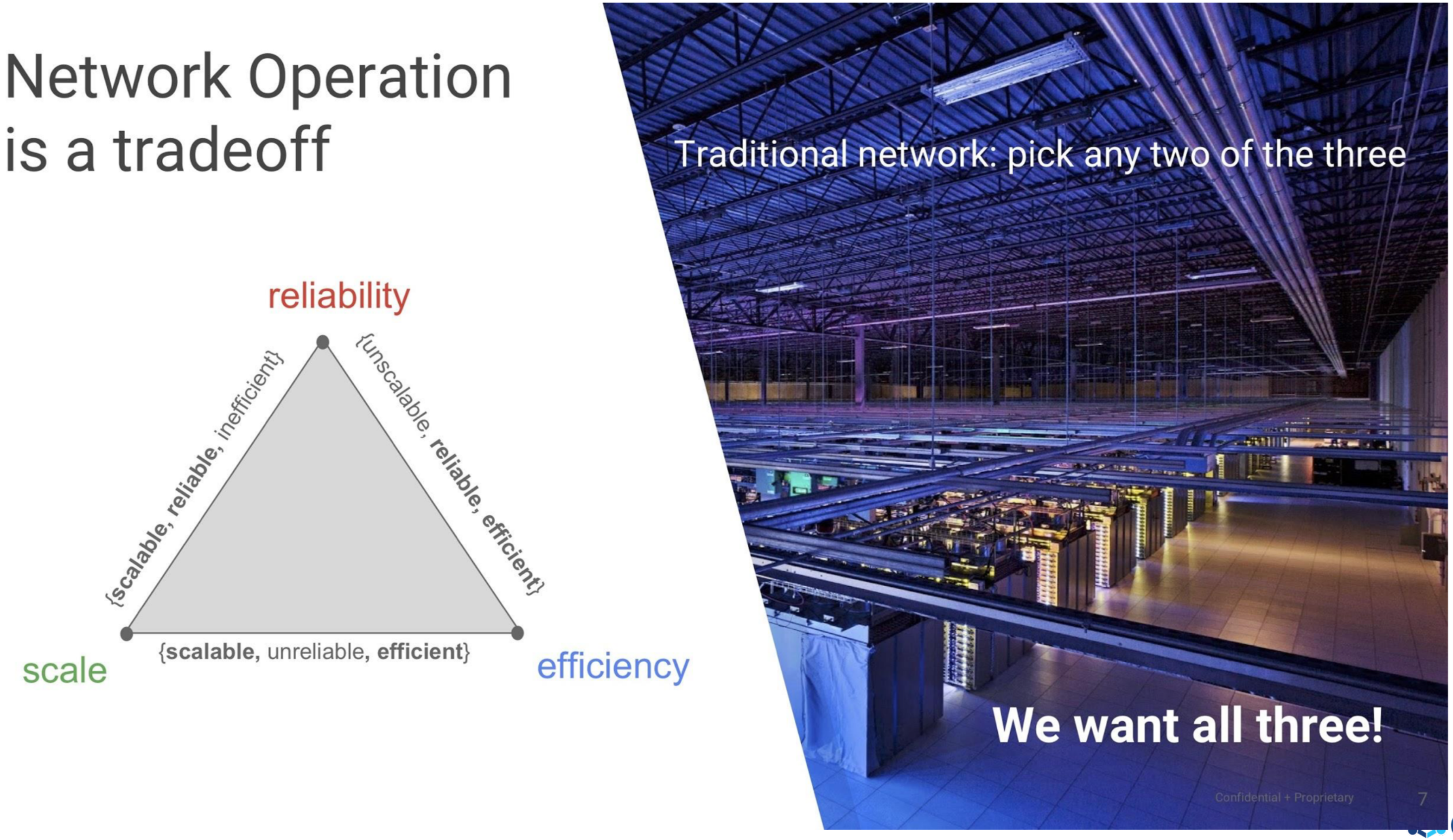
Device Configuration in µONOS

Andrea Campanella, Sean Condon
Open Networking Foundation
<andrea,sean>@opennetworking.org

Overview

- Motivation & background
- System design & goals
- Core components
- Northbound interface
- Southbound interface
- Configuration GUI
- Model plugins
- End to end workflow
- Next steps
- How to contribute

Vision: Zero Touch Networking



Network Operation is a tradeoff

reliability
{scalable, reliable, inefficient}
{unscalable, reliable, efficient}
scale
{scalable, unreliable, efficient}
efficiency
Traditional network: pick any two of the three
We want all three!
Confidential + Proprietary
7

Vision: Zero Touch Networking

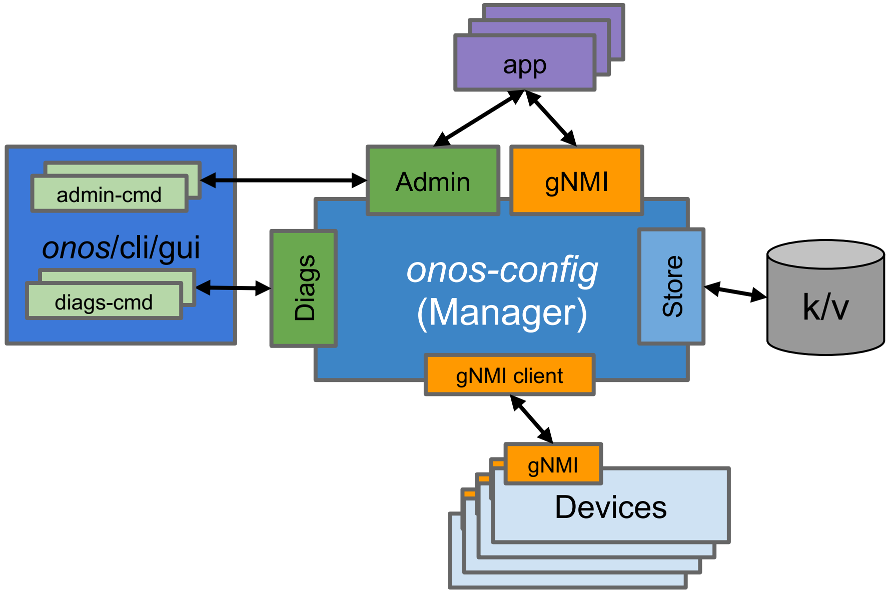
70% of network failures happen during management operations, due to the high level of complexity of such operations across a wide variety of network types, devices, and services

[1] Google lessons
https://ai.google/research/pubs/pub45623

Credit to Google

onos-config Capabilities Overview

- Implements gNMI n/b and s/b APIs
- Multi-device configuration transactions
- Model driven, multi version support
- Rollback to previous points in time
- Device state through subscription
- Flat storage of configuration data in k/v store
- Configuration validation against YANG models
- Cloud Native architecture and deployment as 1 microservice



Core concepts

Network change

- A collection of Configurations

Configuration

- A collection of Changes
- For an online/ offline/ future/ past device

Change

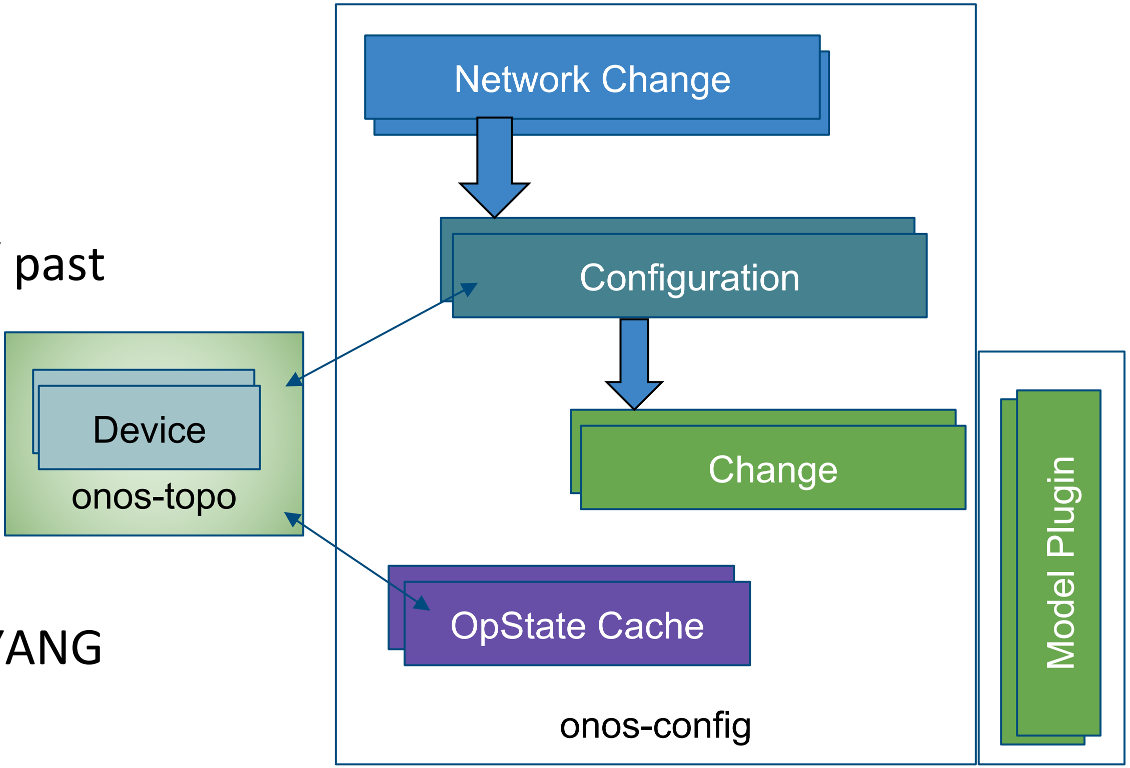
- A collection of keys/values
- Immutable

Model Plugin

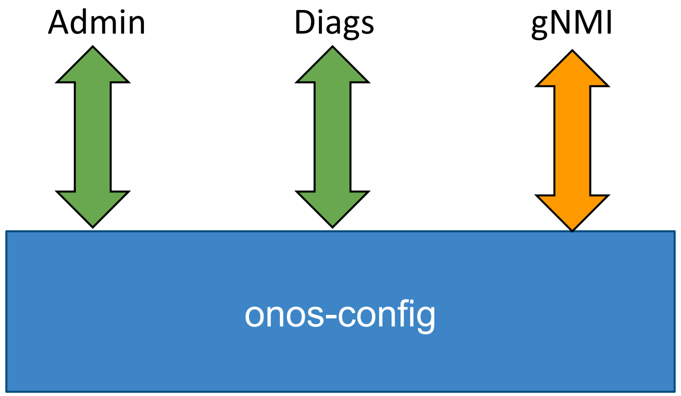
- A shared library of compiled YANG files

Opstate Cache

- Cache of devices Operational & State data



Northbound interface



3 gRPC interfaces

- defined as proto files

Admin

- Manage model plugins
- Manage Network Changes
- Rollback

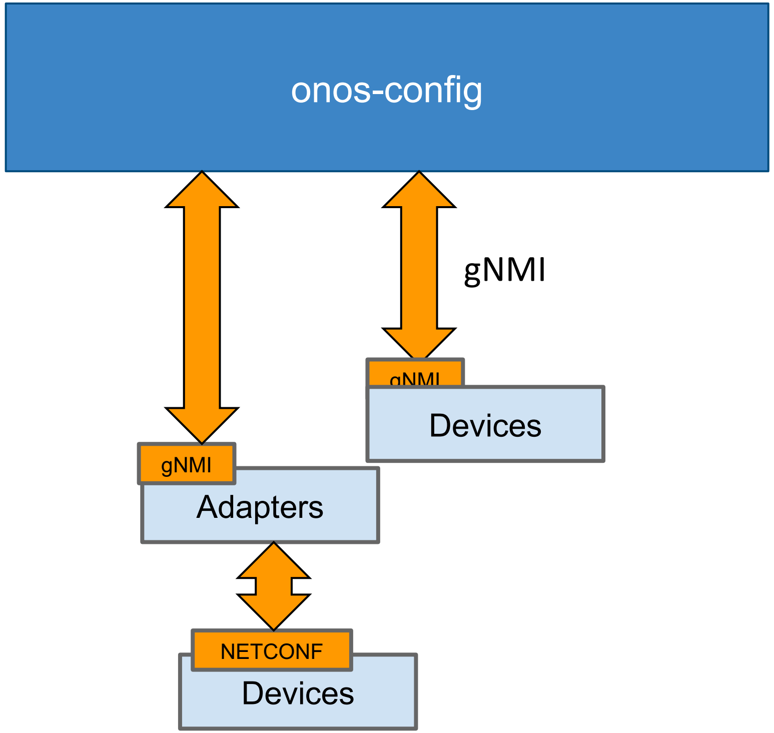
Diags

- Access core objects (Configurations & Changes)

gNMI Server

- General Network Management Interface (from Open Config initiative)

Southbound interface



gNMI Client

- Read capabilities
- Set configuration
- Get state
- Subscribe to state

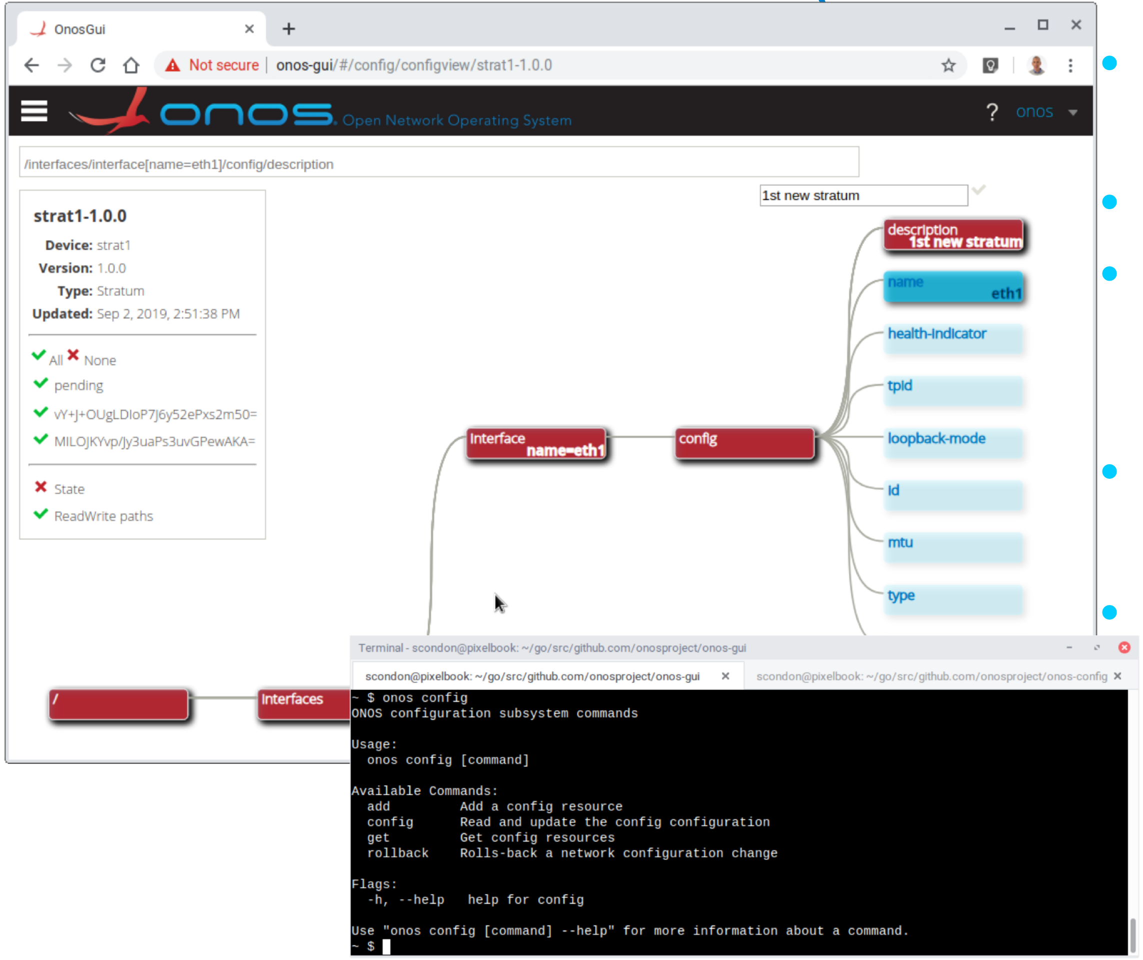
Operational state cache

- Driven by model plugin
- After connection reads all state attributes
- Then subscribe for further changes
- Expose as state to other subsystems

Other protocols (e.g. NETCONF)

- Build external adapter/controller

User Interfaces (Web or command line)



OnosGui
Not secure | onos-gui/#/config/configview/strat1-1.0.0
onos Open Network Operating System
? onos

/interfaces/interface[name=eth1]/config/description

1st new stratum

strat1-1.0.0

Device: strat1
Version: 1.0.0
Type: Stratum
Updated: Sep 2, 2019, 2:51:38 PM

✓ All ✗ None
✓ pending
✓ vY+J+OUgLDIoP7J6y52ePxs2m50=
✓ MILOJKYvp/Jy3uaPs3uvGPewAKA=

✗ State
✓ ReadWrite paths

/
Interfaces
Interface name=eth1
config
description 1st new stratum
name eth1
health-indicator
tpid
loopback-mode
id
mtu
type
Terminal - scondon@pixelbook: ~/go/src/github.com/onosproject/onos-gui
scondon@pixelbook: ~/go/src/github.com/onosproject/onos-gui
scondon@pixelbook: ~/go/src/github.com/onosproject/onos-config

```
~ $ onos config
ONOS configuration subsystem commands

Usage:
  onos config [command]

Available Commands:
  add         Add a config resource
  config      Read and update the config configuration
  get         Get config resources
  rollback    Rolls-back a network configuration change

Flags:
  -h, --help   help for config

Use "onos config [command] --help" for more information about a command.
~ $
```

- Connects to onos-config through gRPC (northbound)
- Driven by model plugins
- Configuration editor
  - Create or rollback network wide changes
- Modular extension of classic ONOS GUI
- Core views
  - Network changes
  - Configurations
  - Models

ModelPlugin

- User creatable plugin for onos-config
- YANG model driven
  - YGOT generated **go** code
- Bundled as shared object library
  - <xyz>.so.1.0.0
- Can be loaded dynamically or at startup (define in helm chart)
- Drives validation of configuration on each change
- Drives GUI options/prompts/metadata

https://github.com/onosproject/onos-config/blob/master/docs/modelplugin.md

End To End Device Workflow 1/2

1. Design YANG model of real device capability
2. Build Model Plugin from template
3. Load the model plugin
4. Create initial configuration for device(s) (optional)

End To End Device Workflow 2/2

1. Connect to a device(s)
   a. linked by device type and version of model
2. Configure parameters on device(s)
   a. View Read Write paths as guide for new attributes
3. Observe and subscribe to state of device through gNMI NBI
4. Add a new version of the model
   a. Migrate configuration and push to device

Next steps

- Fully distributed implementation
  - based on Atomix Go client
- Device specific driver mechanism (plugin driven)
- Platform hardening
  - Security
  - Performance
  - Events handling at scale
- Extend GUI functionality
  - Search bar
  - Time series data plotting
- Documentation, tutorials

How to get involved

- Join *#micro-onos* channel on *onosproject.slack.com*
- Attend ONOS TST meetings
  - bi-weekly Wednesdays at 9:00 PST/PDT
- Fork and send pull-requests to *https://github.com/onosproject* repositories
- Participate in *onos-dev@onosproject.org* mailing list
- Questions?
  - email andrea@opennetworing.org, sean@opennetworking.org

Thank You

Follow Up Links:

µONOS repositories

Atomix repositories

Stratum repositories

gNMI specification