Why NETCONF?

An Automator's Perspective

Viktor Leijon

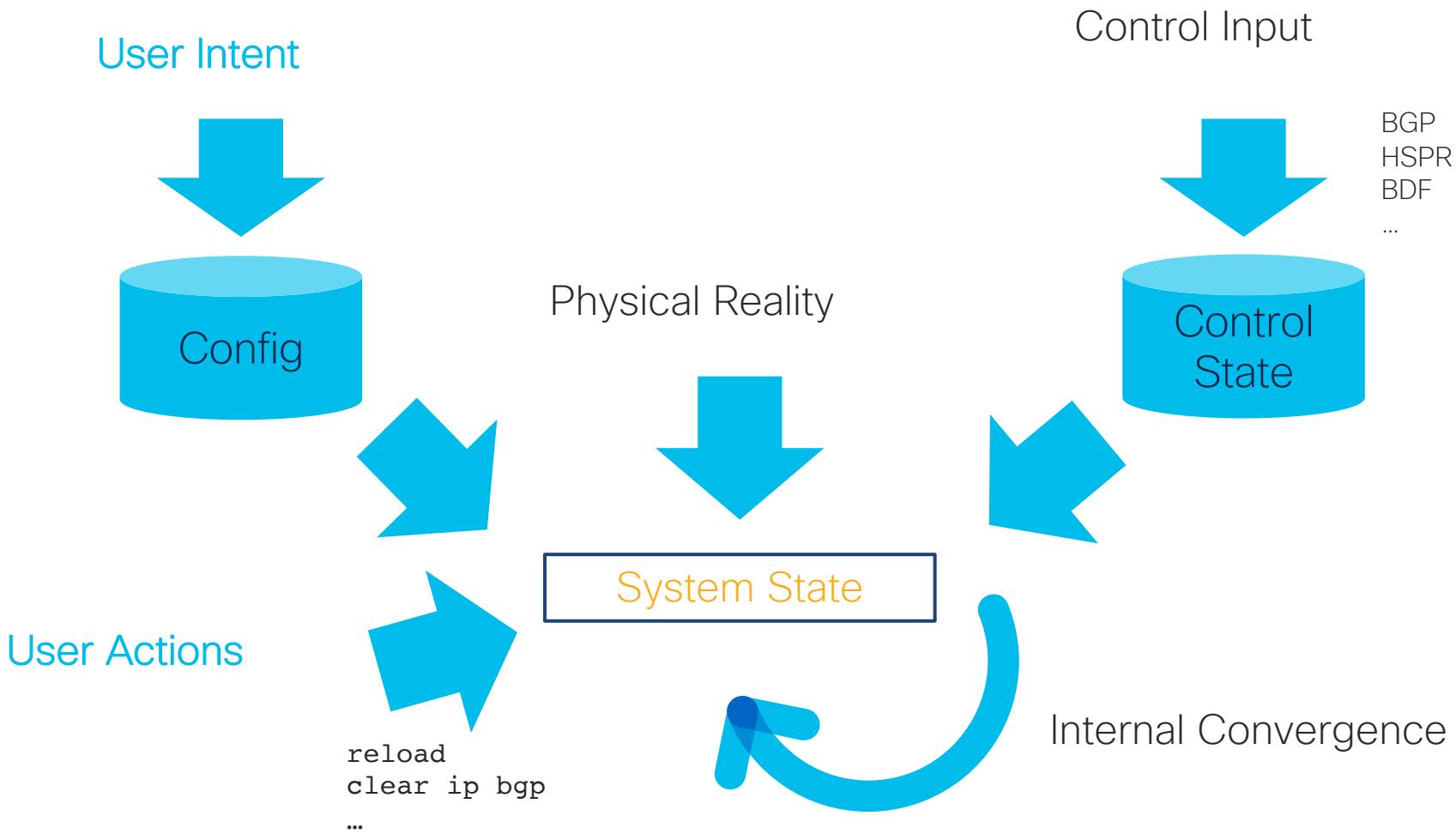
23rd of October, 2019

Agenda

- Intent Based Networking
- The NETCONF/YANG Worldview
- Network Orchestration
- Multi-Domain Orchestration
- Summary

Intent Based Networking

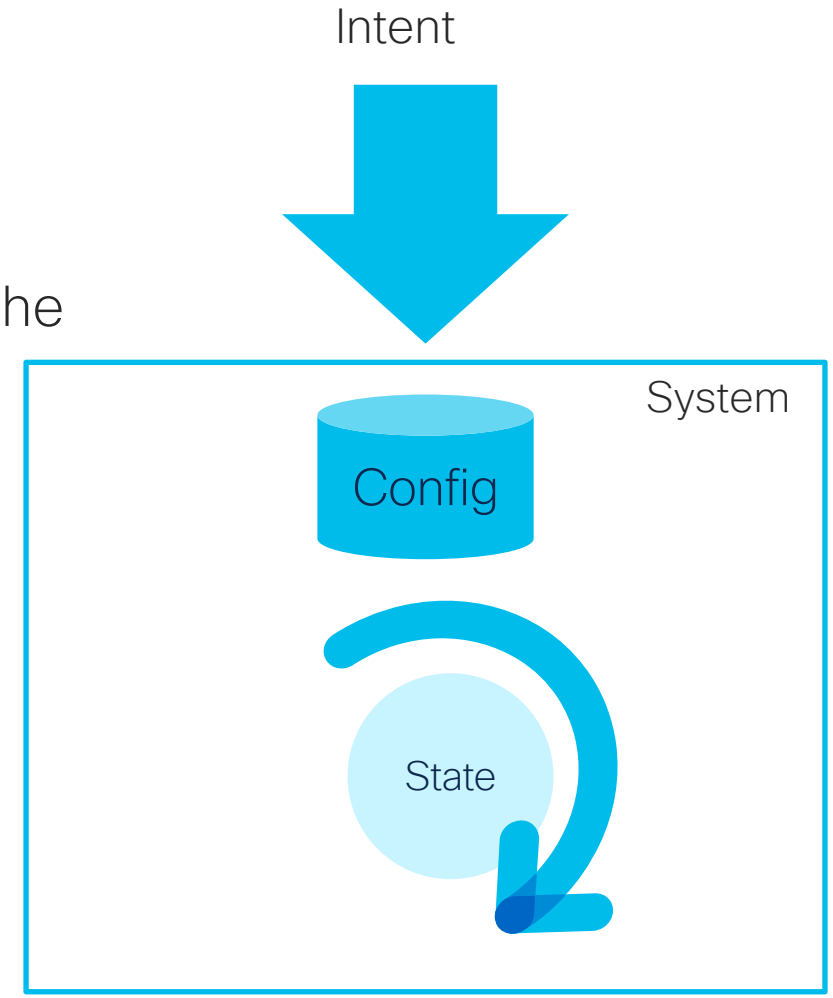
Causes of State



Intent based interface principles

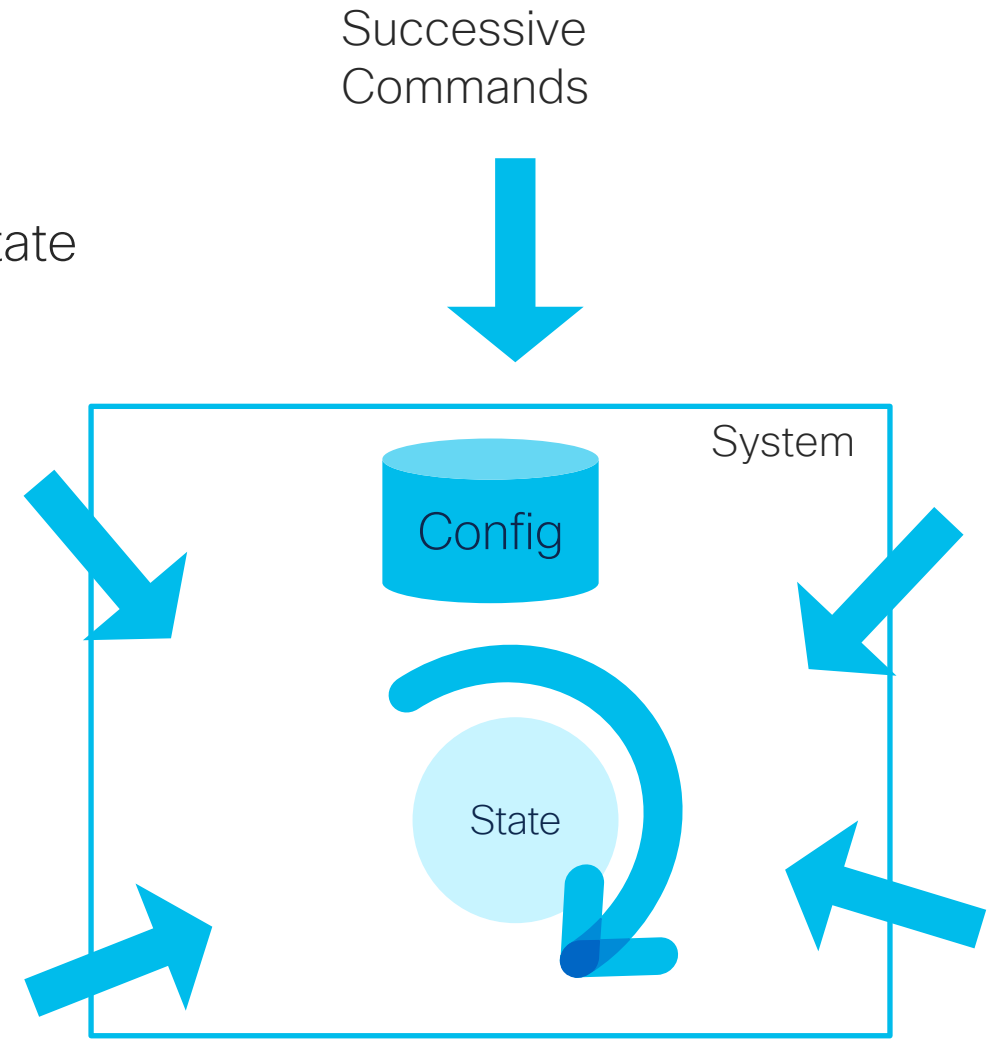
1. Writing your intent is enough
2. The system strives to execute on the intent
3. Intents are idempotent – multiple requests with the same intent has no additional effect
4. You can always write intent regardless of current state
5. The system never modifies received intent

Intent is configuration done right!



Command driven interfaces

1. Explicit commands to move between state
2. The correct command depends on the current state
3. The state is exposed to the user
4. Requires timed sequences of commands to achieve complex effects
5. Automation is by workflows/runbooks



Intent-based or sequences of actions?

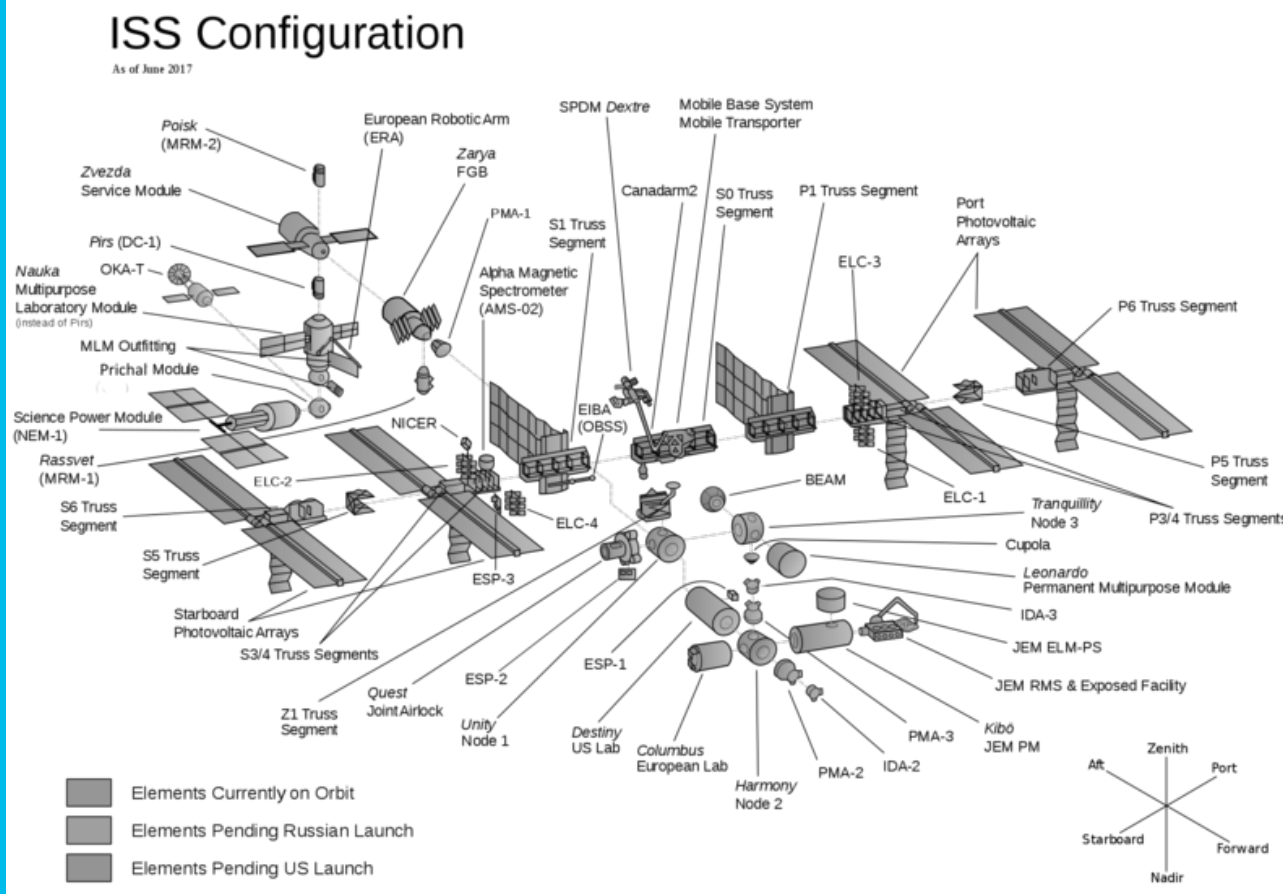
Intent

- Ideal as an interface to a closed-loop system
- Transfer of responsibility to the system
- Allows modularity

Actions

- Good for things that are one-off
- Enforces sequentiality
  - Approval
  - Workforce management

The NETCONF/YANG Worldview



Standardizing Network Management

- 30 year legacy of new protocols
  E.g. CMIP, SNMP, Corba, SOAP, CIM, REST, …
- IETF Realized SNMP wasn't used for configuration
  SNMP → RFC 3535 → NETCONF/YANG
- NETCONF/YANG are nice modular, scalable standards
  Downside: You could implement single RPC "execute-cli-command" and claim NC/YANG compliance
- We may have to teach operators how to ask for NETCONF/YANG

What they are saying about NETCONF/YANG

"Over time I assume that we will vote (fo
NETCONF and YANG) with our checkbook." *Level 3: Jack Waters, CT*

"We are converging on a modeling language, we are converging on configurations in a common way. Getting away from CLI." *AT&T: Margaret Chiosi, Distinguished Network Architect*

"NETCONF/YANG is actually the way forward for us." *DT: Axel Clauberg, VP Network Architecture*

"Propriety interfaces are not what the industry wants. We are now following standardized protocols. In particular in our embedded OSs, where we are converting them to use NETCONF/YANG interface." *Cisco: Dave Ward, CTO*

"We have actually supported NETCONF for all of our routing devices for quite some time." *Juniper: Geoff Mattson, VP Product Development*

"We need to ensure we launch NETCONF/YANG" *Ericsson: Gabriela Styf Sjöman, VP Cloud computing and NMS*

Adapter Tax; The #1 Killer of Cool Telco Business Ideas

NETCONF

- NETCONF/YANG compliant devices
- YANG downloaded from device
- No code required

IOS-XR: 0 loc/leaf
(0 loc / 200,000 leafs)

Junos: 0 loc/leaf
(0 loc / 200,000 leafs)

Huawei 9k: 0 loc/leaf
(0 loc / 50,000 leafs)

CLI

- Hand written YANG with annotations
- Code to log in, recognize prompts, error messages, parse state data

IOS-XR: 0.10 loc/leaf
(6 kloc / 60,000 leafs)

IOS: 0.15 loc/leaf
(12 kloc / 80,000 leafs)

F5BIP: 2.5 loc/leaf
(10 kloc / 4,000 leafs)

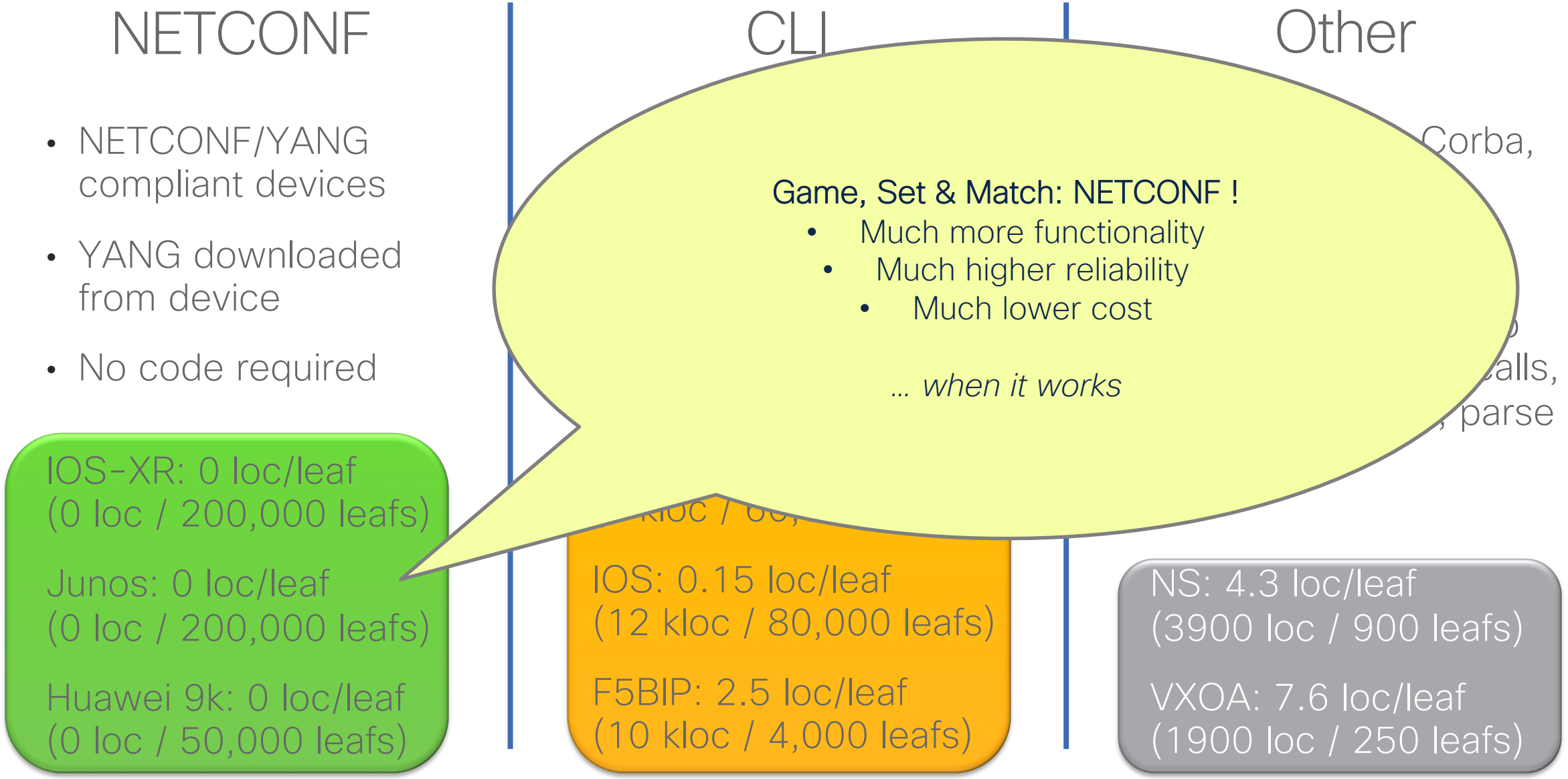
Other

- REST, SOAP, Corba, TL1 …
- Hand written YANG
- Code to log in, map requests to RPC calls, sequence calls, parse responses

NS: 4.3 loc/leaf
(3900 loc / 900 leafs)

VXOA: 7.6 loc/leaf
(1900 loc / 250 leafs)

Adapter Tax; The #1 Killer of Cool Telco Business Ideas



What does "...when it works" mean? How do you know if it "works"?

"...when it works" means

- RFCs
- Service Automation Criteria

How to know?

- Test your use cases in the NSO Interop Lab

- Add ConfD, enable the NETCONF interface and be done?
- Effort, Quality, Coverage varies greatly between products

- Non-conformance often local to one YANG module or subsystem
- Once a use case works, it typically works reliably

Service Automation Criteria

Mandatory NETCONF Basics

- Hello
- Locking
- Sub-tree filters

YANG Model Correctness

- YANG model syntax
- YANG model constraints

YANG Model Compliance

- Types, ranges & constraints

Transactionality

- All-or-nothing semantics
- All-at-once semantics
- Config validity depends on config alone
- No Autoconfig

Network Orchestration Network Programmability

What is Network Orchestration?

- For our purposes:
  Network Orchestration is the process of automatically provisioning the correct configuration state in the network
- By extension it is about
  - Writing the configuration to the devices
  - Creating overlay networks
  - Starting virtual network functions
  - Coordinating resource usage
  - Ensuring compliance to defined standards
- Orchestration can provide different degrees of abstraction

Network Programmability

- Opening up APIs towards the network
- Turning Network Configuration into Software Development
- Opening up to use of Software Principles
- A large cultural transformation
- Networking at the speed of software

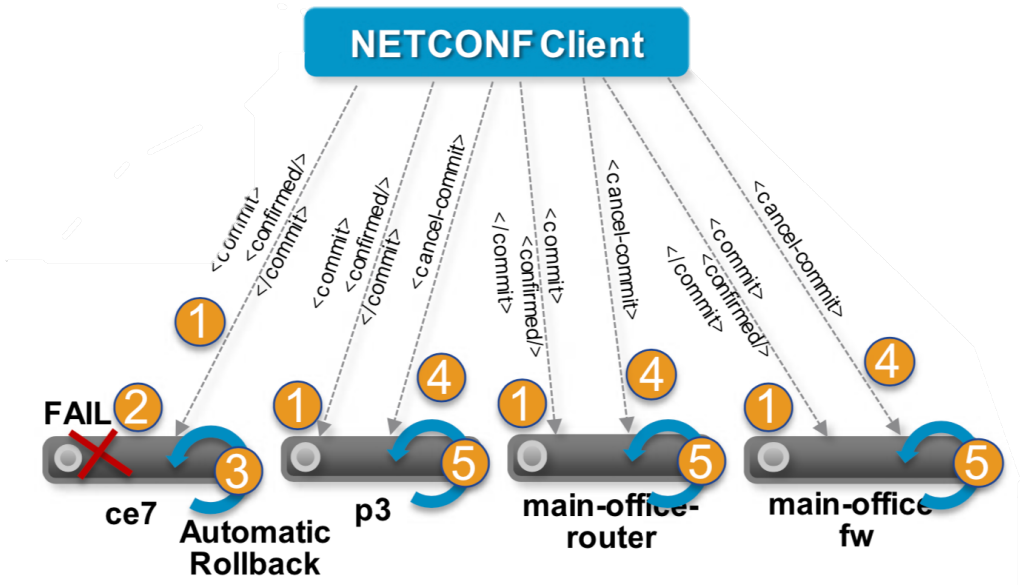
Using Network Programmability

- Lean heavily on software concepts:
  - Modularity
  - Re-use
  - Well-defined abstractions
- Strict separation of concerns
  - Enabling independent work
  - Intent and model-based orchestration
- Layered Service Orchestration
  - Horizontal and vertical scale-out
  - Independent development of features

Model-centric development
API-centric development

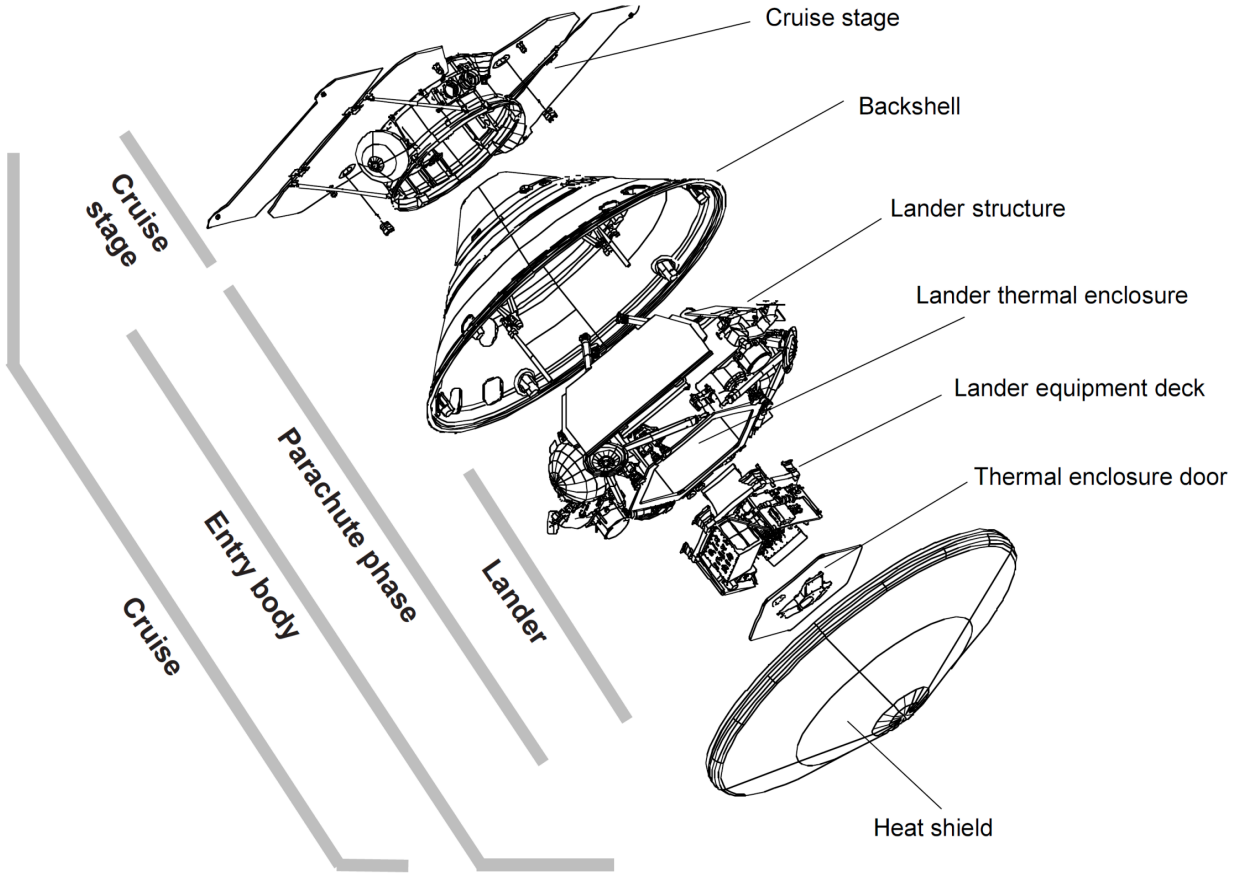
- A service model describes all the things you can do with the service
- The model is the center of gravity in the development process
- The stakeholders share ownership of the model
- Models allow for
  - A common abstraction of intent
  - A standard way of exchanging data (w/protocol)
  - Automated rendering of APIs and automatic data validation
  - Abstract reasoning and discussion of problems and services
- Models reduce the integration complexity

Many Things can Go Wrong
..When Making Changes to Distributed Systems



- The simplest way to handle such faults is to just let the entire service fail and show the user an error message.
- By using transactions, the application can:
  - pretend that there are no crashes (atomicity)
  - that no one else is concurrently accessing the database (isolation)
  - and that storage devices are perfectly reliable (durability).

Multi-Domain Orchestration Cross-Domain Orchestration



Lander flight system

Why cross-domain orchestration?

- Orchestration today is segmented into domains, driven by domain needs
- The future needs cross-domain orchestration for many reasons
  - An API to the network
  - Increasing need to smash the silos!
- Integration costs between domains are a big concern
- Overcoming the zoo of orchestrators
  - A term from Axel Clauberg, Vice President at DT
  - At a light-reading event 68% of the audience foresaw “a zoo of orchestrators that require significant manual effort to work in tandem and/or custom integration”

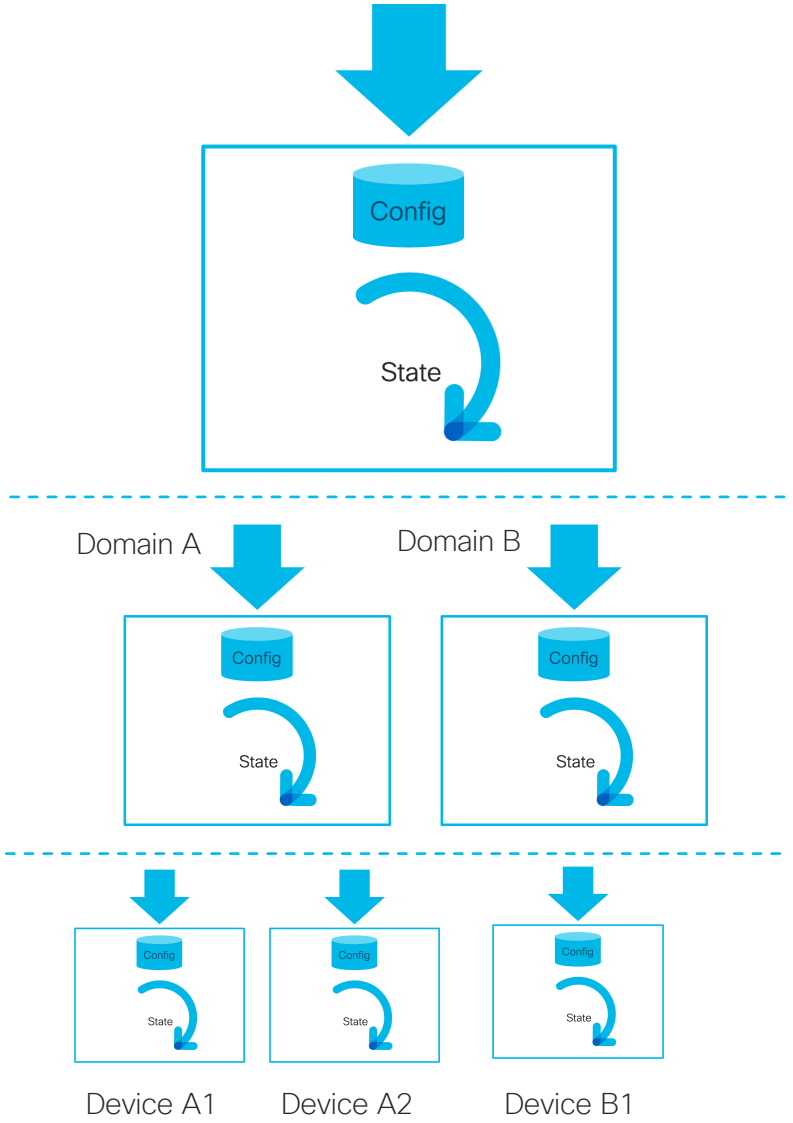
Cross-Domain challenges

- Rigid and inflexible architectures
- High rate of change in the environment
- Expensive integration projects
- Adapter tax for each integration point
- Domain specific context hinders cross-domain usage

- Many different administrative and technical domains
- Different basic paradigms in different domains
- Difficult for staff to maintain end-to-end understanding of systems
- Requires a solid foundation in the domains
- The problems tend to multiply!

Extending Programmability

- Our toolbox should be as generic as possible
  - Same methods everywhere
  - Similar technology on all levels
- Intent based interfaces allow us to hand off responsibility
- Encapsulation is powerful
- Good abstraction helps bring intent to the next level!



Cross-Domain Programmability

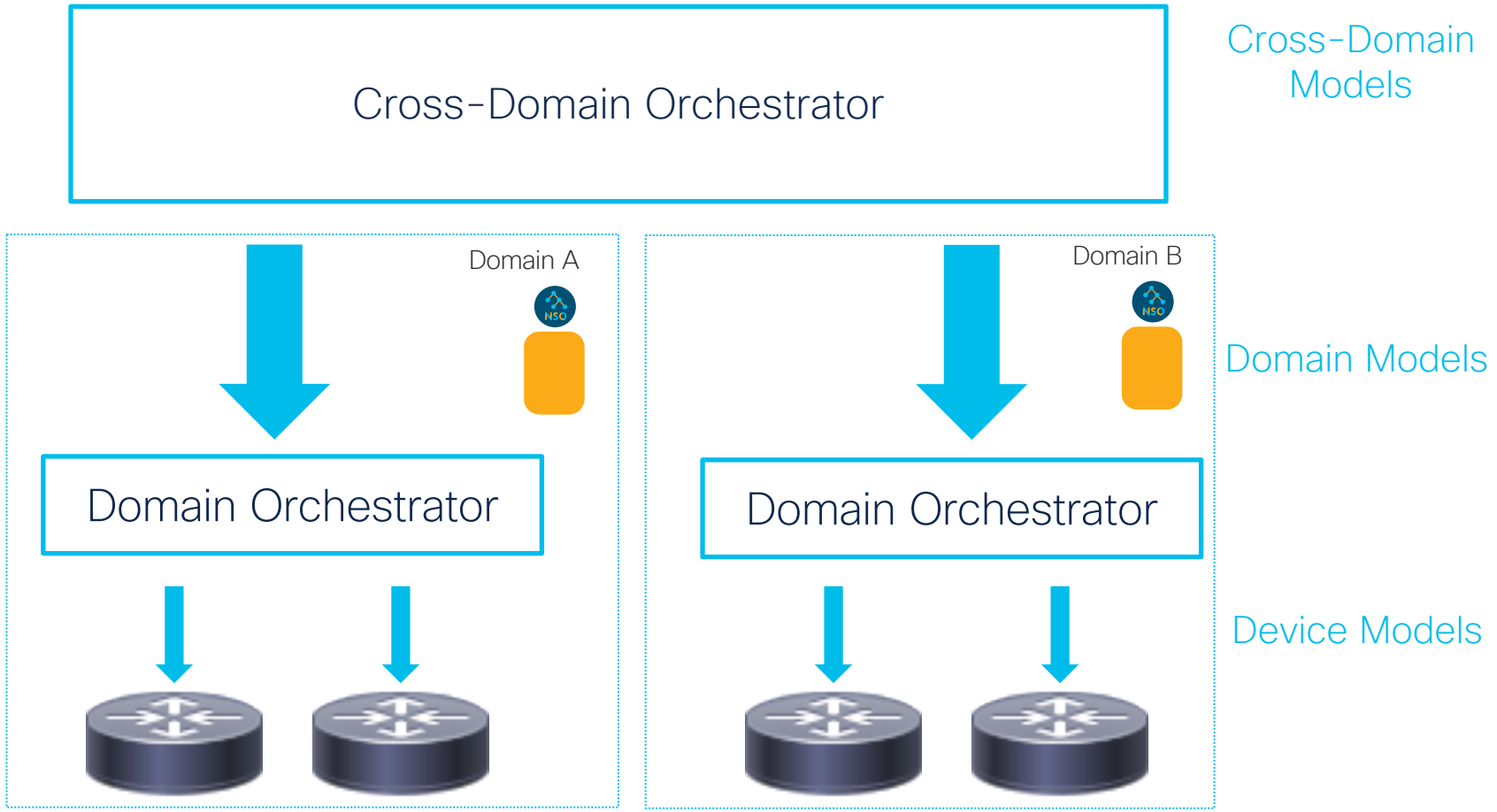
CDO provides API:s to the network

CDO leverages domain API:s

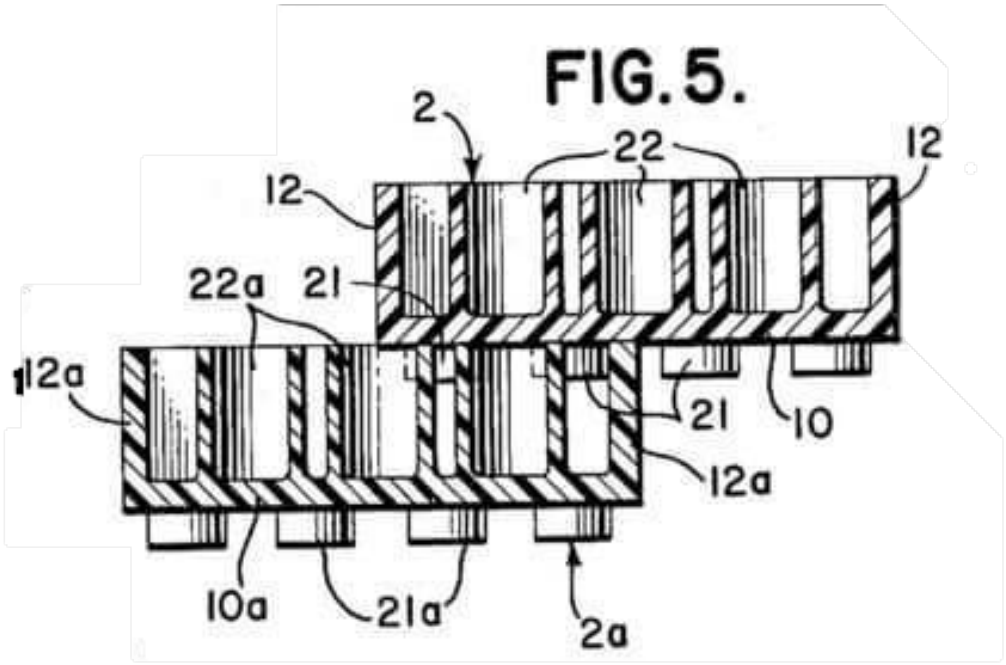
Orchestrators provide domain API:s

Orchestrators leverage device API:s

Programmable devices provide device API:s



Summary



What do we want?

- Fundamentally we want to control the state of the network
- We want to strive for a modular architecture.
- The solution must scale well with increasing complexity.
- Standardization reduces cost
- We want an API to the network

Lessons learned

Principles matter

Early design choices are the foundation to success

History repeats itself

Workflows are still there, and some people have forgotten what they are good at!

Conscious implementation

Implementations must be compatible with the basic design philosophy

It works!

NETCONF great simplifies network automation and is constantly improved


CISCO