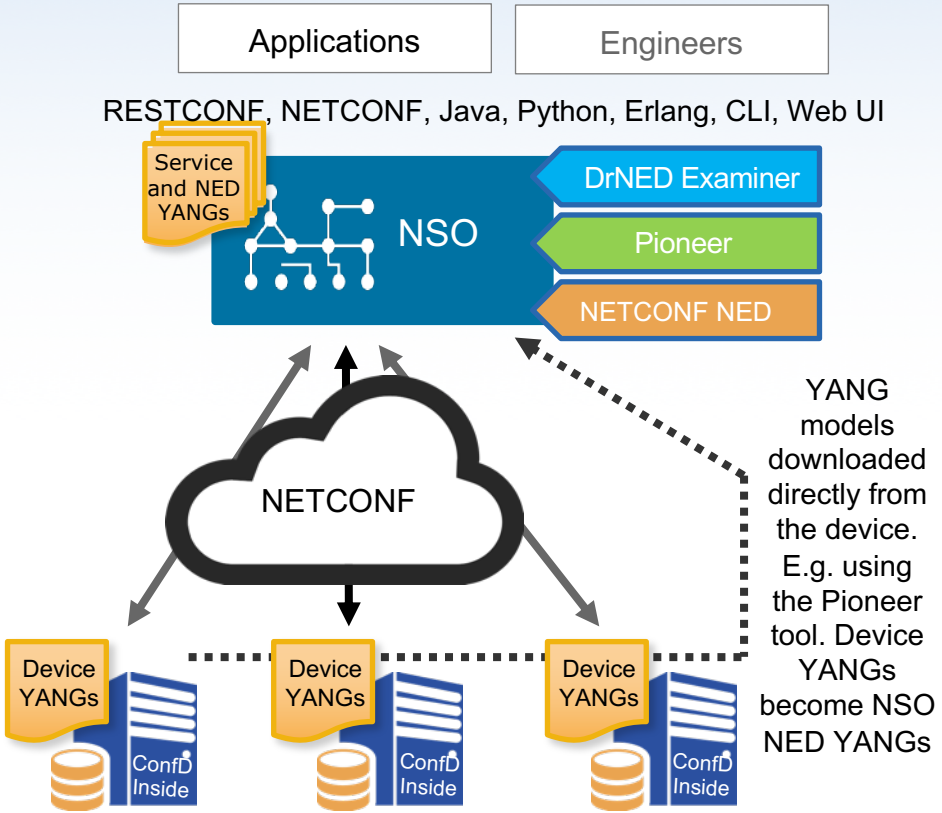
Distributed Network Wide Transactions

Integrate & Test a ConfD Network Element with NSO

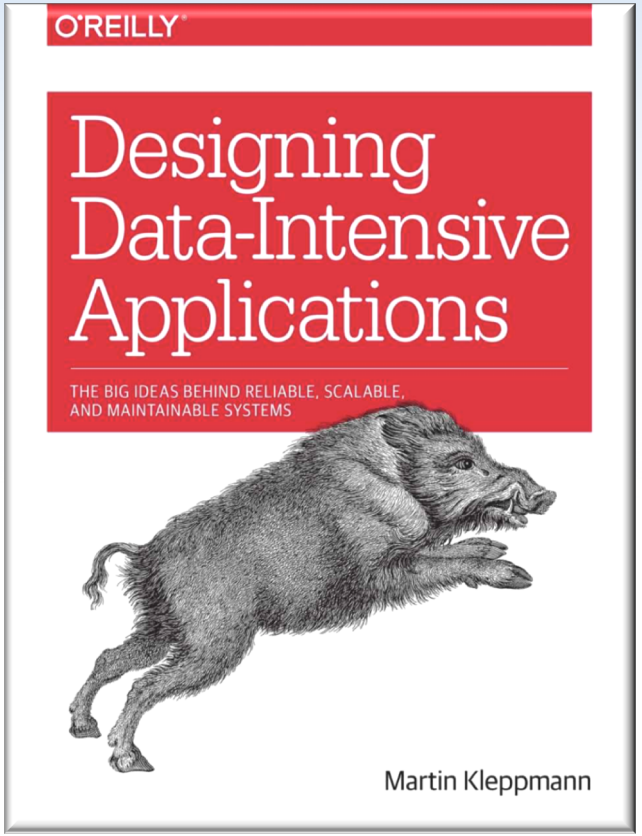
1. Setup ConfD
2. Update the NSO configuration to plug in the new NETCONF device
3. Build a NETCONF NED for the device using
   a. NSO >= 5.2 - NETCONF NED Builder
      NSO < 5.2 - Pioneer NSO package
4. Verify that our expectations are met using the DrNED Examiner NSO package



Inspired by:

- Great book to help navigate the diverse and fast-changing landscape of technologies for processing and storing data.
- Examples of technologies that form the foundation of many popular applications and that have to meet …
  - Scalability
  - Performance
  - Reliability
- ... requirements in production every day.

- Kleppmann, Martin. Designing Data-Intensive Applications O'Reilly Media.



Network Operators and Equipment Vendors Need Machine-to-Machine Communication Principles

- A few years ago, timelines to activate or change a network service could span weeks or even months.
- To meet escalating customer requirements, operators need much faster, more flexible service delivery.
- Drive out slow, error-prone manual processes from service provisioning to reduce operating expenses (OpEx) and increase service agility.
- Replace human-to-machine interfaces and CLI scripting

Automation

Challenges

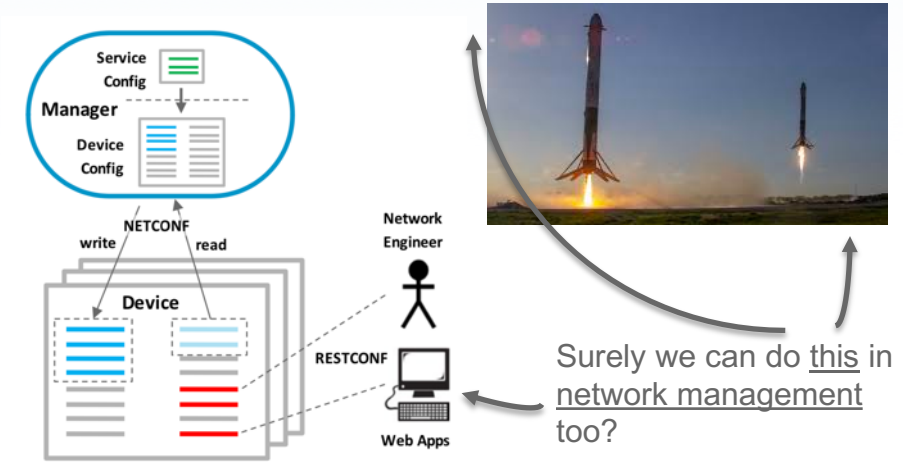
- Wide variety of legacy equipment
- Human-to-machine tinkering
  - CLI scripting
  - Error-prone
  - Non-standard
- Device centric

Opportunities

- Programmability
- Device abstraction through a common network API
- Standards based
- Machine-to-machine
- Scale

To Achieve Full Automation, Operators Need a Network Composed of Programmable Physical and Virtual Devices

- Manageable via precise machine-to-machine transactions.
- Provision services by pushing out new device configurations through networkwide transactions.
- The industry already has a model to enable programmable networks and services:
- ✓ The NETCONF protocol and YANG data models.



Standards for Automation

NETCONF & RESTCONF APIs

- The protocol allows the device to expose a full, formal application programming interface (API).
- Applications can use this straightforward API to send and receive full and partial configuration data sets.
- Make it easier to manage multivendor networks
- Transactional
- Provision services across the network in a single transaction (NETCONF)

YANG data modeling language

- The model defines a contract between the NETCONF client and server
- Allows both parties to have faith the other knows the syntax and semantics behind the modeled data.
- Define devices and services in a consistent, parsable manner.
- Easily parsed by a computer

Managing Distributed Systems with Transactions

A

Atomicity is the ability to abort a transaction on error and have all writes from that transaction discarded.

C

Consistency means that, to a system using transactions, all actions within a transaction are instantaneous.

I

Isolation, means that concurrently executing transactions must be isolated from each other, so they cannot step on each other's toes.

D

Durability implies that once a transaction has committed successfully, any data it has written will not be forgotten, even if there is a hardware fault or the database crashes.

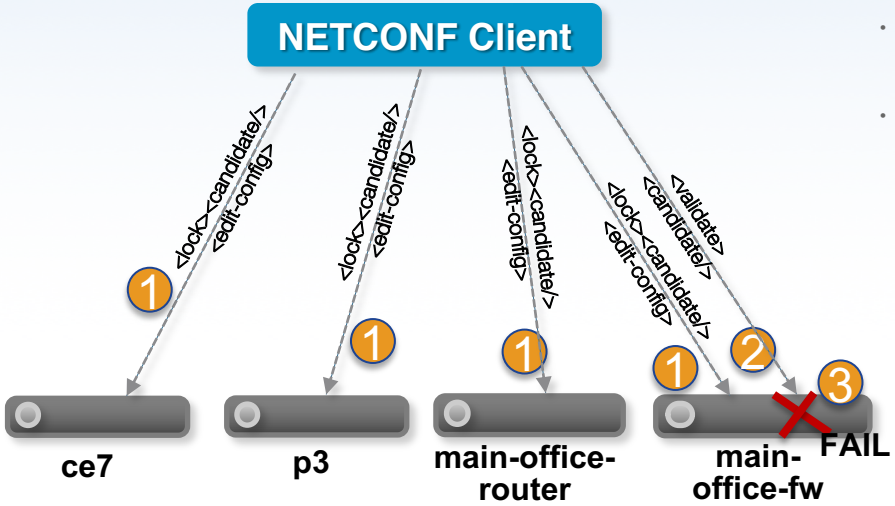
- Systems that do not meet the ACID criteria are sometimes called BASE
- Stands for Basically Available, Soft state, and Eventual consistency.
- It's important to note that not all applications are susceptible to the kinds of problems that ACID is meant to solve, and therefore may not require transactional safety guarantees.
- For programmable network devices that will implement services via automated configuration changes, transactions are a critical requirement.
- Without them, a wide range of error scenarios (processes crashing, network interruptions, power outages, disk full, unexpected concurrency, and others) can cause data to become inconsistent in various ways.

Transactions and Tradeoffs

Some authors have claimed that general two-phase commit is too expensive to support, because of the performance or availability problems that it brings. We believe it is better to have application programmers deal with performance problems due to overuse of transactions as bottlenecks arise, rather than always coding around the lack of transactions.

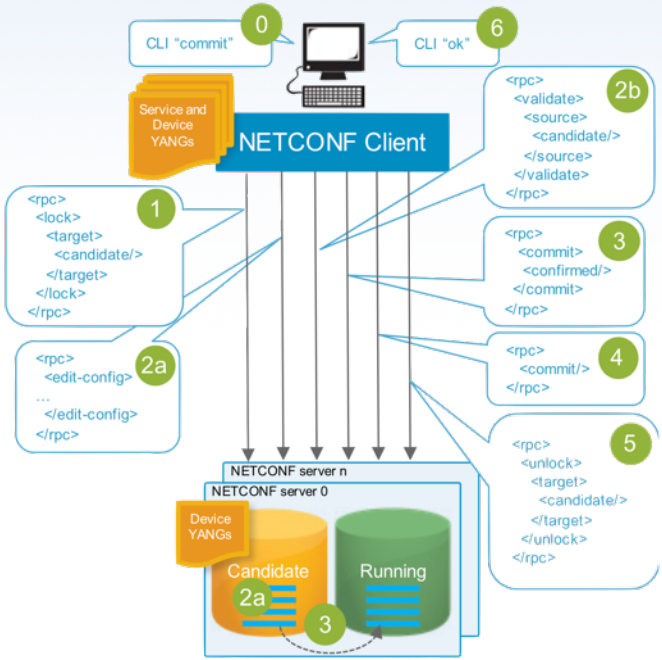
James Corbett et al., *Spanner: Google's Globally-Distributed Database* (2012)

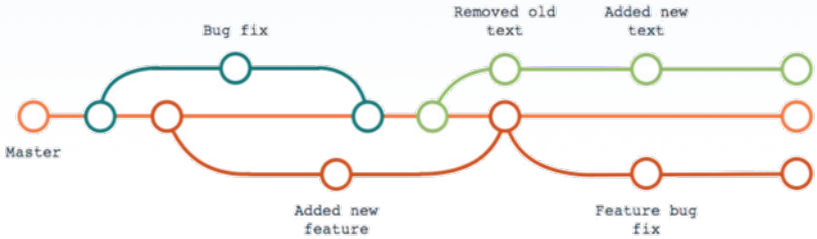
Many Things can Go Wrong
..When Making Changes to Distributed Systems



- 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 NETCONF client can:
  - Pretend that there are no crashes (atomicity) – NETCONF validate candidate and confirmed-commit.
  - Assume that the device will sort out the order of things (consistency) – ConfD CDB subscriber priorities.
  - That no one else is concurrently accessing the database (isolation) – NETCONF lock and ConfD's CDB transaction lock.
  - That storage devices are perfectly reliable (durability) – ConfD's transaction manager writing changes to disk.
- Even though crashes, race conditions, and disk failures do occur, transactions hide those problems, so the NETCONF client application doesn't have to worry about them.
- ConfD hide those problems with other devices for the application that integrated with ConfD with full support for the NETCONF candidate datastore, confirmed-commit:1.1, validate:1.1, rollback-on-error, and writable-running false capabilities.

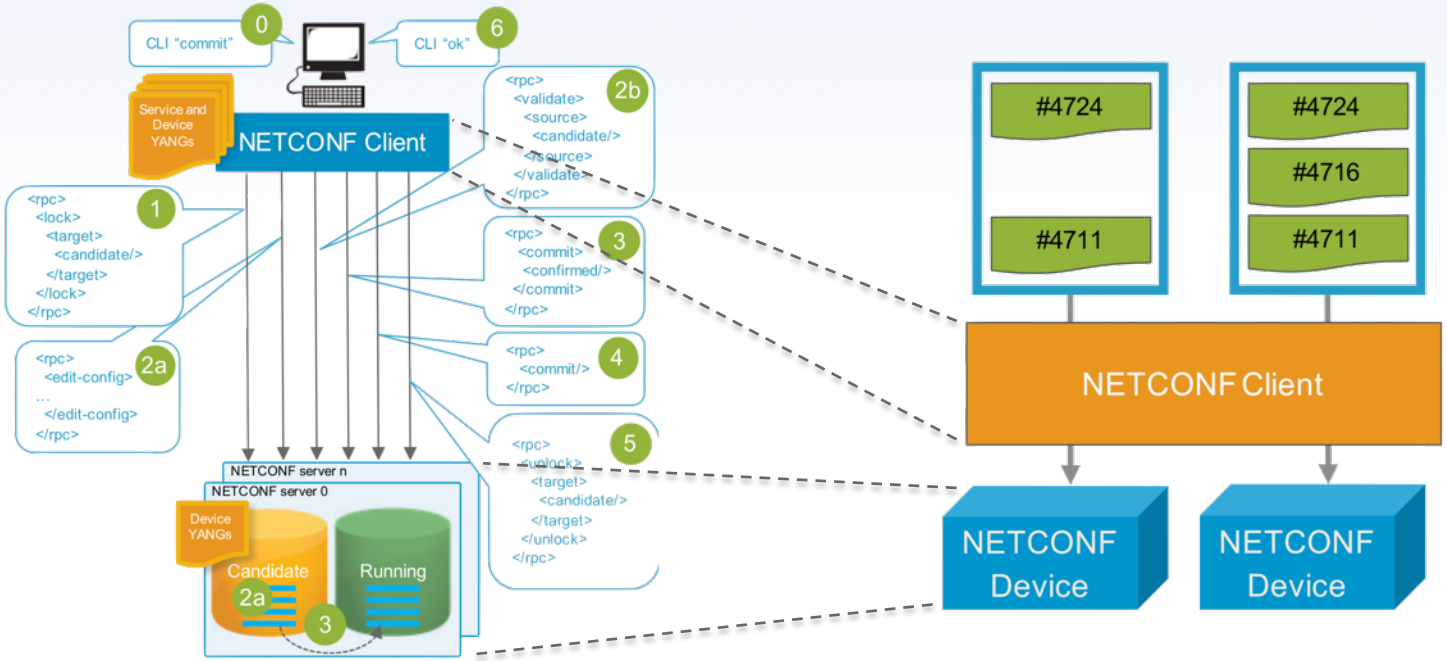
Linearizability or Eventual Causal Consistency?





Linearizability or Eventual Causal Consistency?

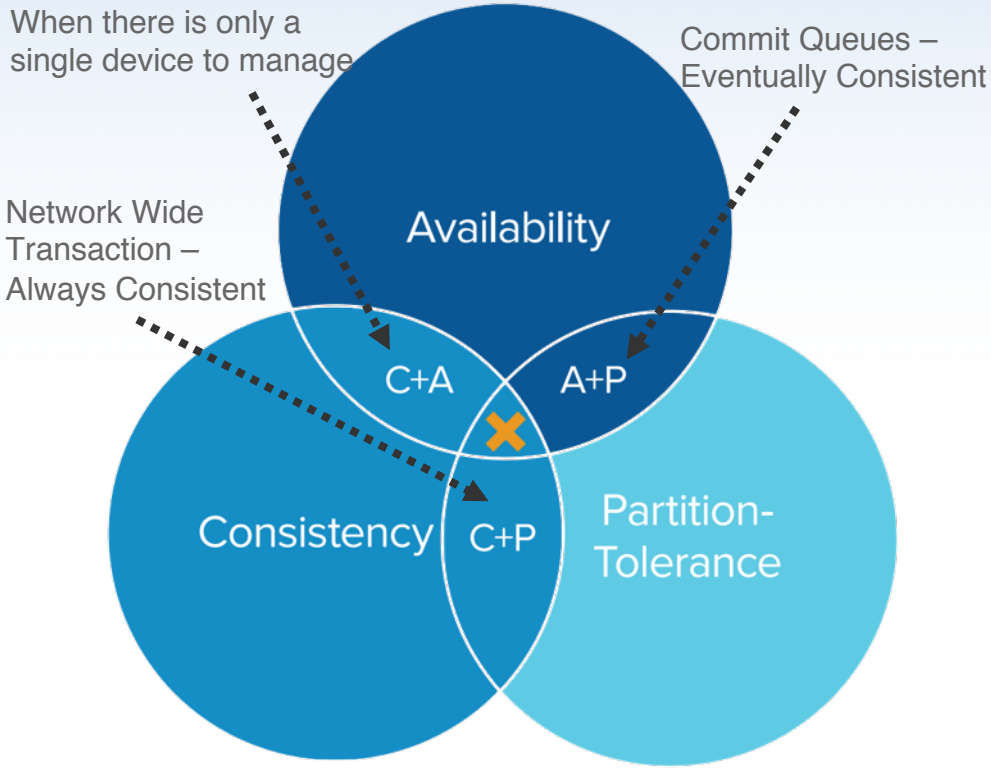
Sacrifice Some Consistency for Faster Service Deployment (Availability) Using Atomic Commit Queues



Keep all devices but the slow less important in network wide transaction

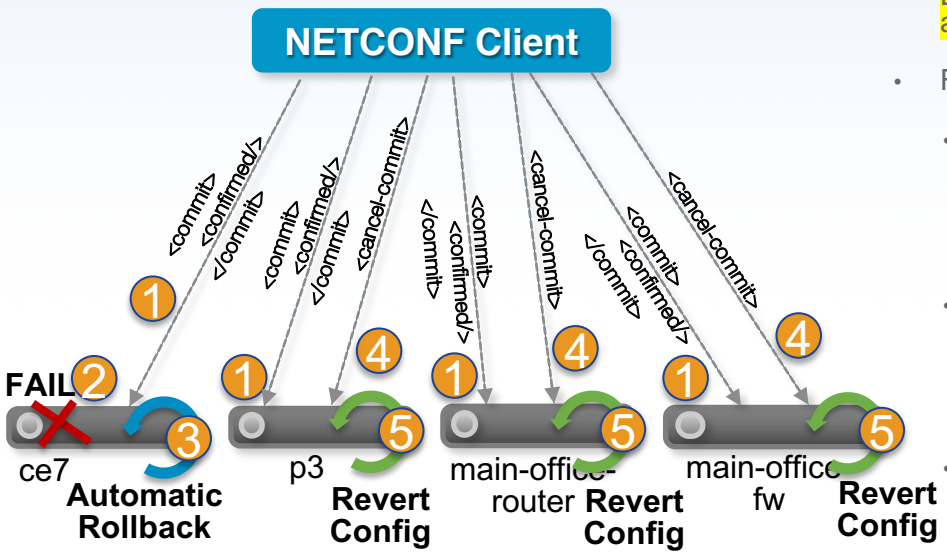
CAP Theorem

Why Combining a Networkwide Transaction with Commit Queues Makes Sense



- Formulated in 1998 by Eric Brewer, with the mathematical proof published in 2002.
- Consistency (C) equivalent to having a single up-to-date copy of the data – not to be confused with Consistency in ACID.
- Availability (A) of that data – for updates
- Tolerance to network partitions (P) – here “devices”
- If there is more than one device to manage, we automatically have P in a distributed system if, so we can then at best choose between of A+P or C+P
- Commit Queues – The manager, e.g. NSO, sacrifice some consistency for availability. Less locking.
- Network Wide Transactions – Sacrifice some availability for consistency.

Consensus: Do we All Agree?



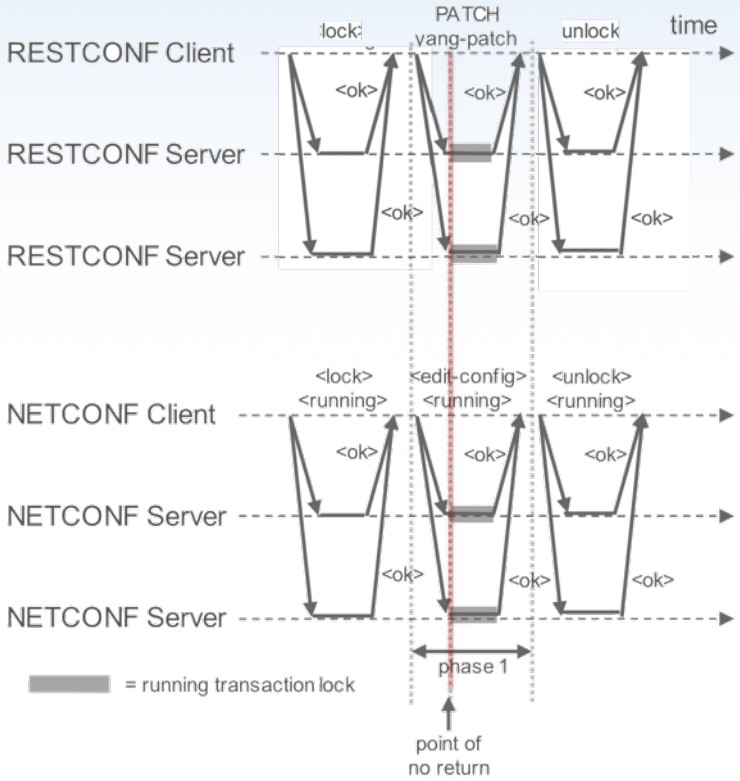
- Distributed Atomic Commit aka Network Wide Transactions – will avoid violations to the atomicity guarantee.
- For example:
  - Some nodes may detect a constraint violation or conflict, making an abort necessary, while other nodes are successfully able to commit.

    :candidate + :validate1.1
  - Some of the commit requests might get lost in the network, eventually aborting due to a timeout, while other commit requests get through.

    :candidate + :confirmed-commit1.1
  - Some nodes may crash before the commit record is fully written and roll back on recovery, while others successfully commit.

    :rollback-on-error + :candidate + :confirmed.commit1.1

Transactions: Single Phase & The Point of No Return

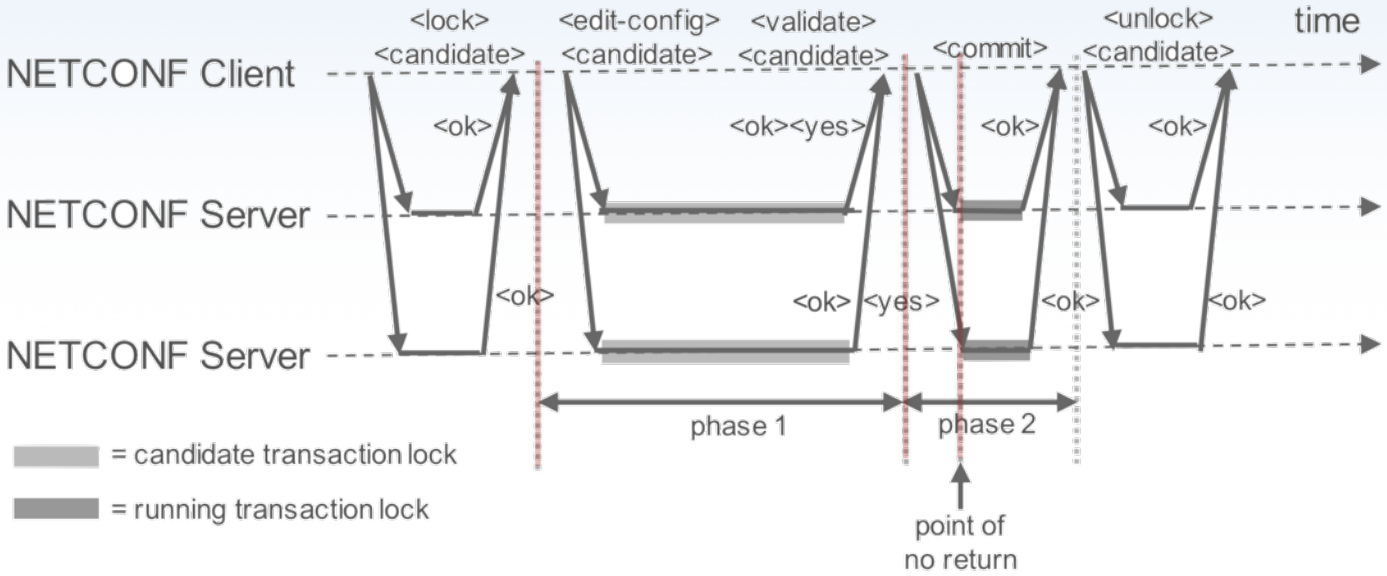


RESTCONF server
:yang-patch capability
enabled

NETCONF server
:rollback-on-error,
and :writable-running
capabilities enabled

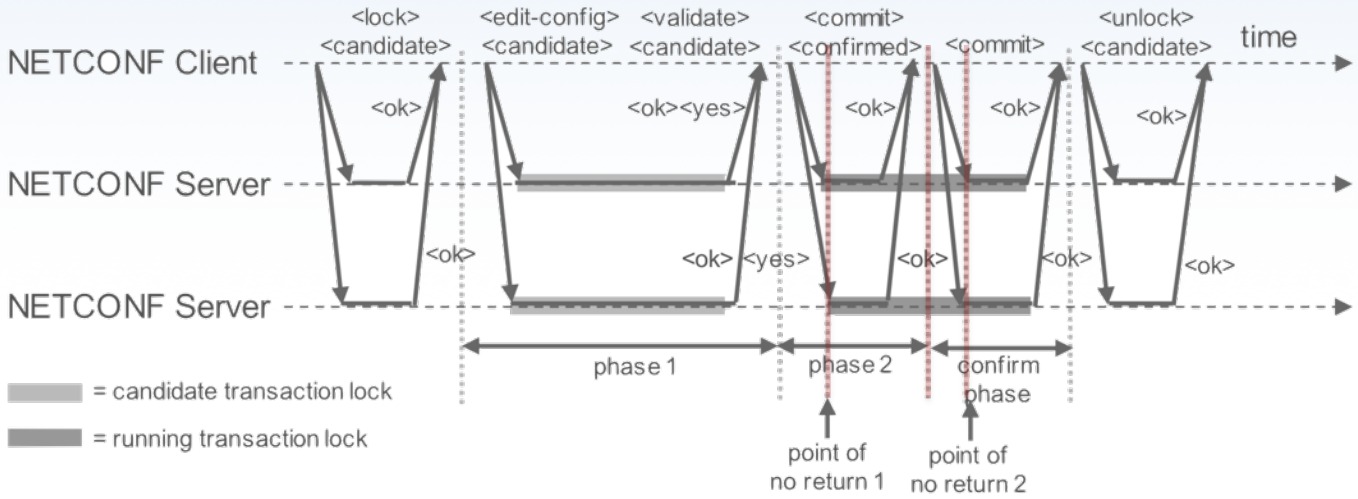
Two-Phase Commit (2PC) - A Classic Algorithm in Distributed DBs

An atomic transaction commit across multiple nodes



NETCONF server :rollback-on-error, :candidate, and :validate:1.1 capabilities enabled :writable-running capability disabled

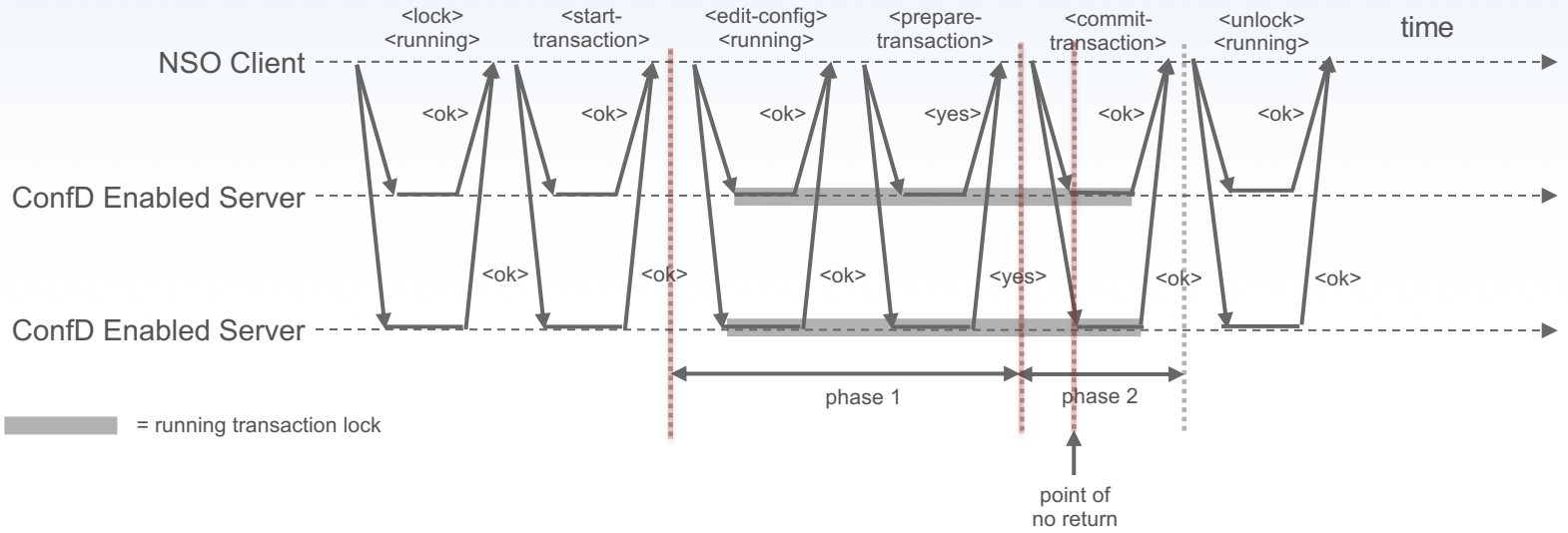
Confirmed Commit – Automatic Rollback



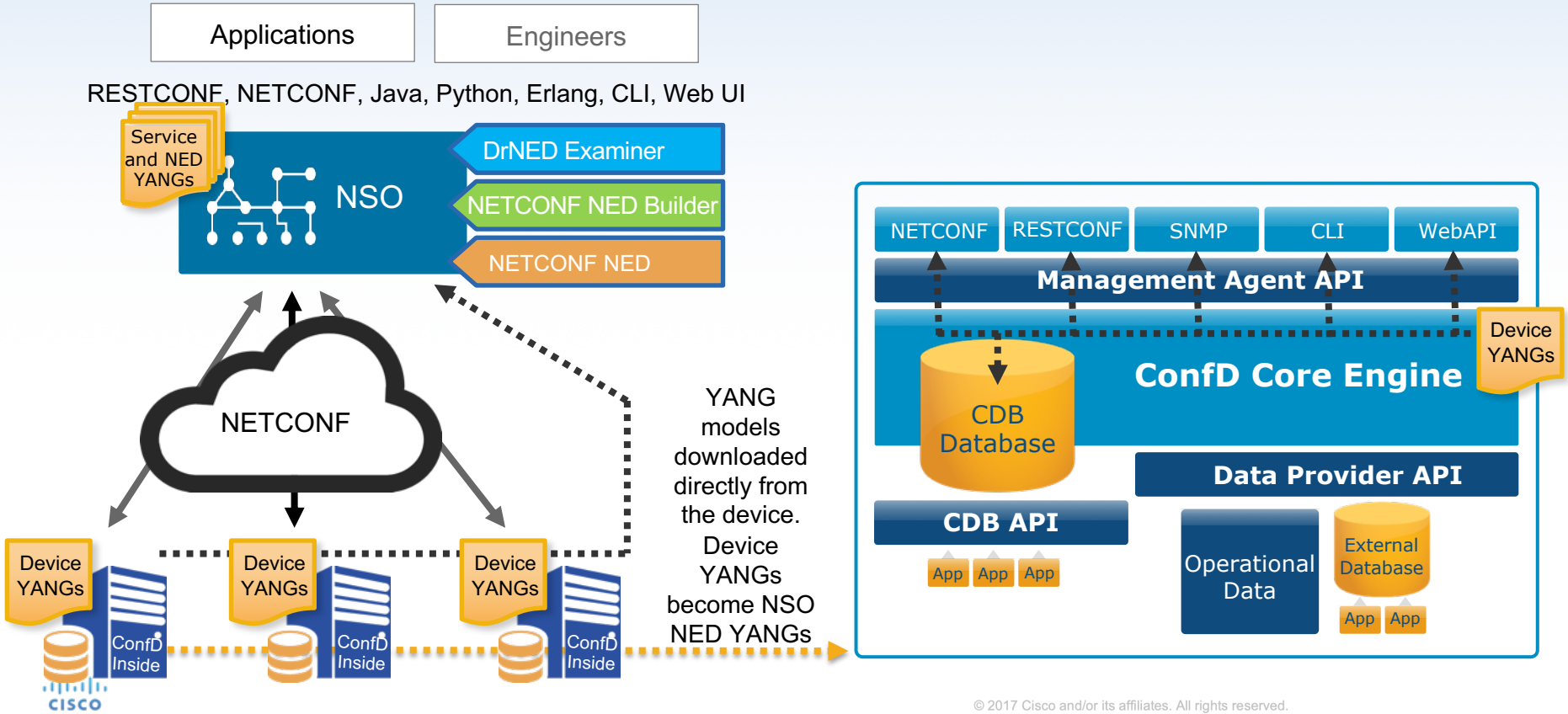
NETCONF server :rollback-on-error, :candidate, :confirmed-commit:1.1, and :validate:1.1 capabilities enabled :writable-running capability disabled

2PC using Tail-f

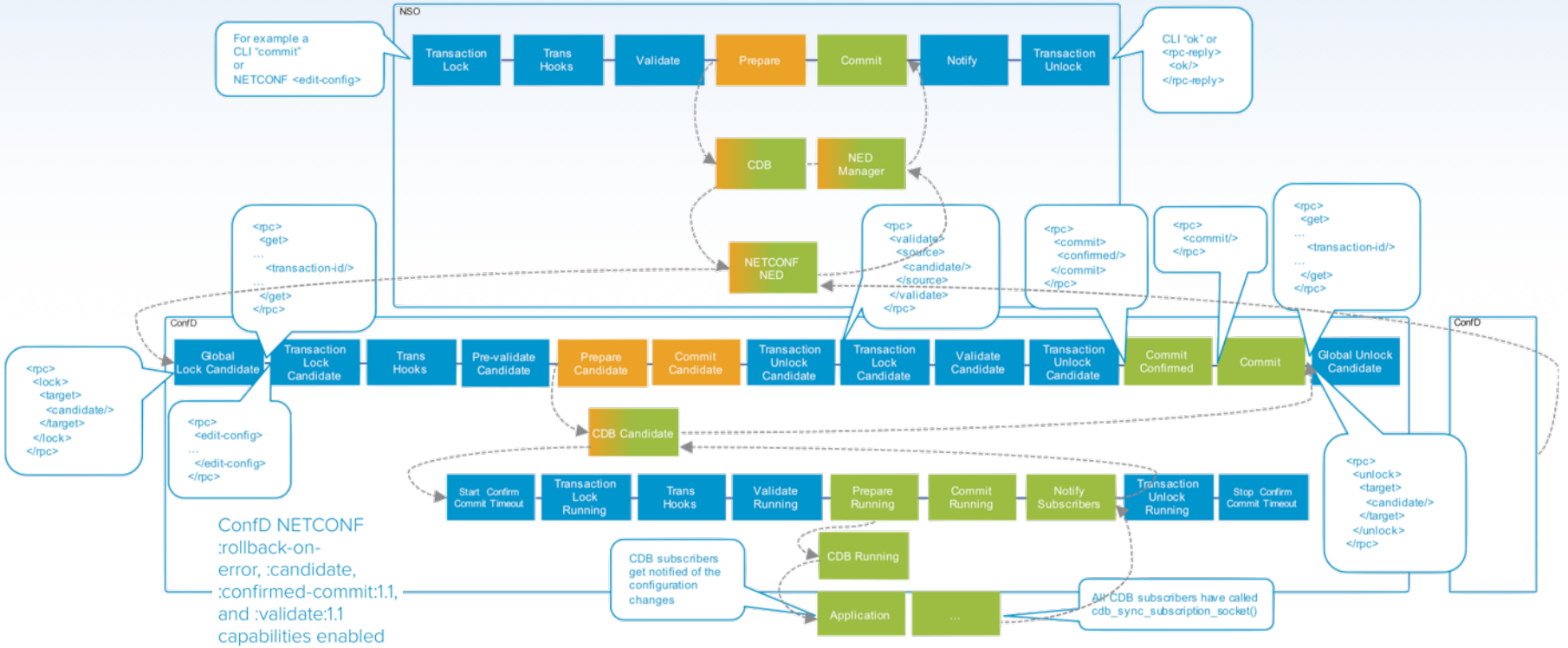
NETCONF server
:rollback-on-error, Tail-f
:transactions, and
:writable-running
capabilities enabled



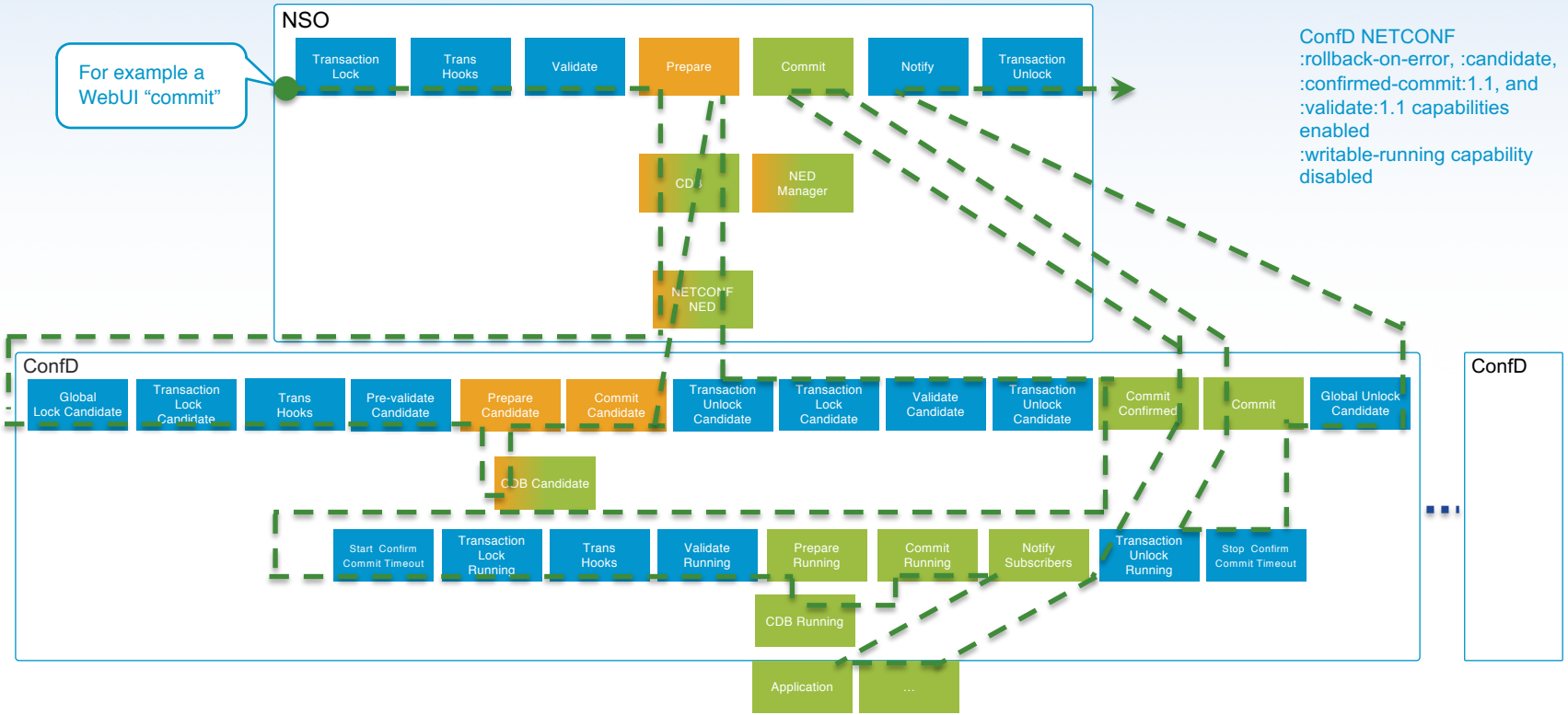
NSO - ConfD Demo Overview



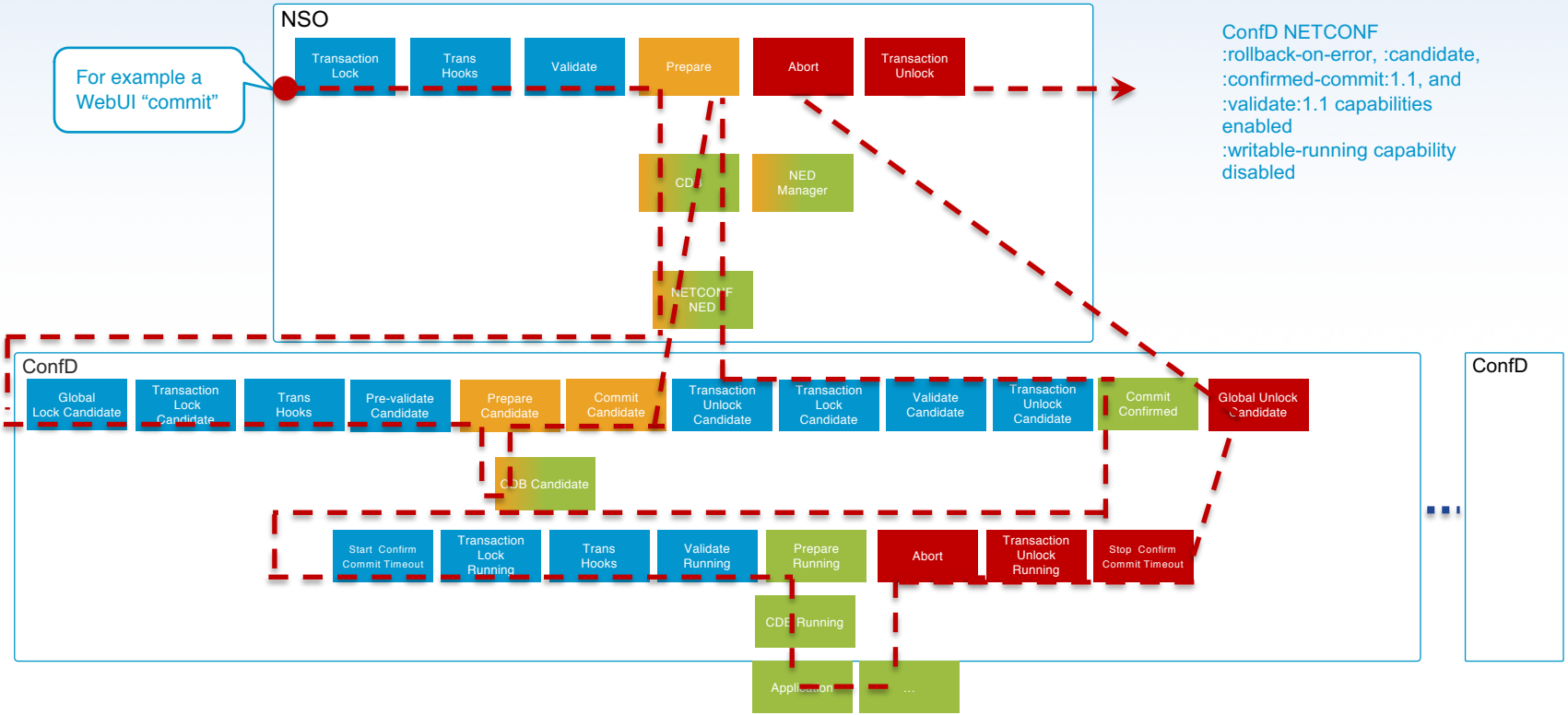
Inside Networkwide NETCONF Transactions



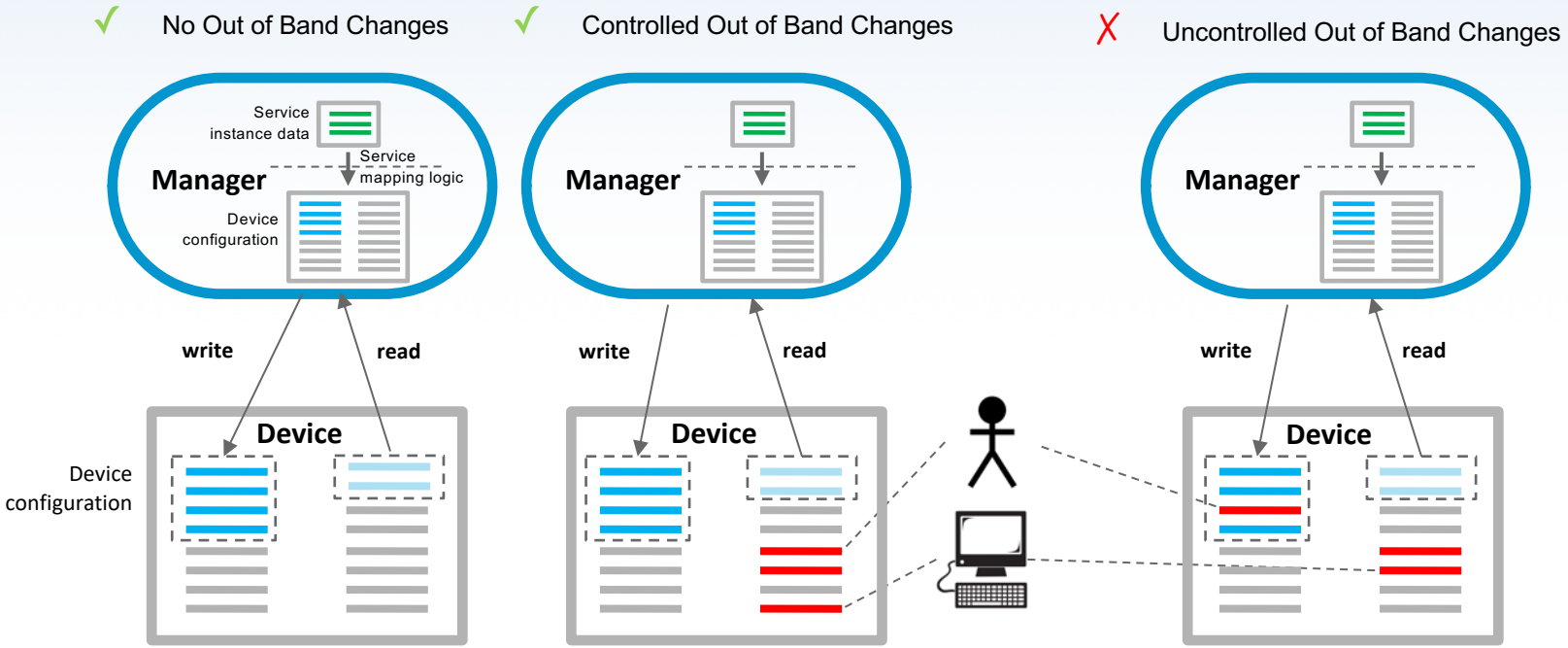
Inside Networkwide NETCONF Transactions



Inside Networkwide NETCONF Transactions



M2M Kryptonite – Uncontrolled Out of Band Changes



Summary

- There can be great benefit for service providers to automate service provisioning in precise machine-to-machine transactions.
- Equipment and service providers have a shared responsibility to make the switch from human-to-machine management.
- Goal is to enable automated operations to reduce OPEX and increase service agility to reduce time to complete the deployment of a service.
- A ConfD based network element integrated with NSO following the Service Automation Criteria is an automation enabler.

For More Information

- ConfD - https://www.tail-f.com
  - The ConfD Developer space on GitHub: https://github.com/ConfD-Developer
- NSO - https://www.cisco.com/c/en/us/solutions/service-provider/solutions-cloud-providers/network-services-orchestrator-solutions.html
  - NSO Developer Space on GitHub (e.g. DrNED Examiner): https://github.com/NSO-developer

ConfD - NSO Interoperability Testing program https://info.tail-f.com/nso_interop_lab

TOMORROW starts here.