



Service Automation Criteria

Getting the value out of NETCONF/YANG

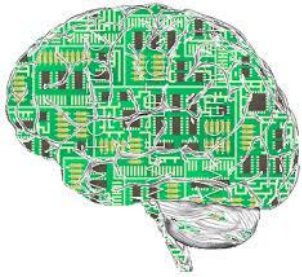
Feb 17, 2016

Objective

- Why do operators want NETCONF/YANG, and what do they mean by that?
- The value of Transactions & SDN Architecture

Management – a communications challenge

Manager



Workers



Orders vs.
Self-organization



*“Any problem, big or small, within a family,
always seems to start with bad communication.
Someone isn’t listening.”*

– Emma Thompson

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
- Lack of interoperability → “Service Automation Criteria”
We may have to teach operators to ask for NETCONF/YANG/SAC

What they are saying about NETCONF/YANG

“Over time I assume that we will vote (for NETCONF and YANG) with our checkbook.” **Level 3: Jack Waters, CTO**

“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**

Operator's Wet Dream

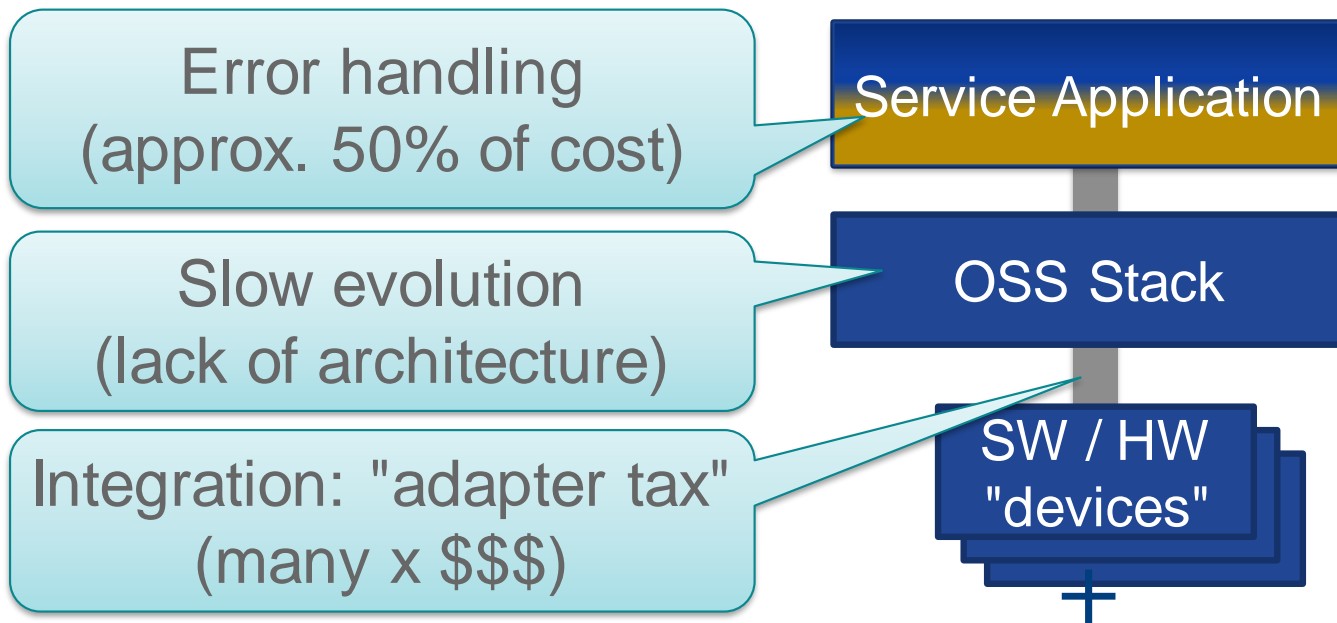
- Manage services (not networks, not devices)
- Service activation (deactivation, modification) code
 - Small, high abstraction
 - Multi-domain, multi-vendor
 - Agnostic of device protocol, location, role, layer and type

→ Transactions

→ SDN Architecture

We taught them the magic word is
NETCONF/YANG
... *the above is what they think it means*

Classical Orchestration Problem Areas



= Way too much
~25% of TCO

That's a lot of money.
More than all the capex!

Transactions

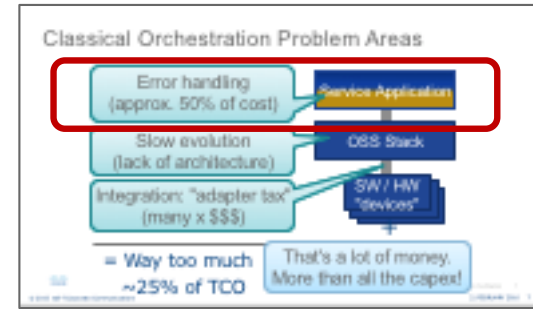
Transaction Definition: the “ACID test”

- Atomicity: all-or-nothing, great for error handling
- Consistency: all-at-once, great for simplicity
- Independence: no-crosstalk, great for many concurrent clients
- Durability: done-is-done, great for reliability

Introduction of transactions + SQL caused a DB industry boom in the 80's. Applications got reliable. Could run against many different DBMS'

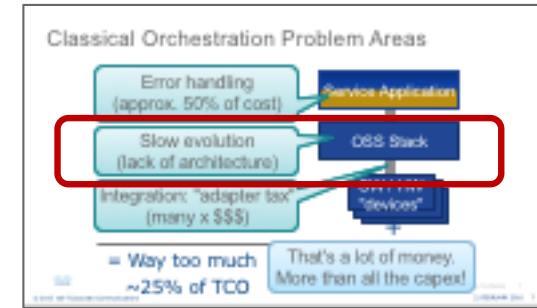
NETCONF makes the network a distributed database

- Network-wide transactions



The SDN Architecture

Three layers of management abstraction



SDN Inventors and Gurus

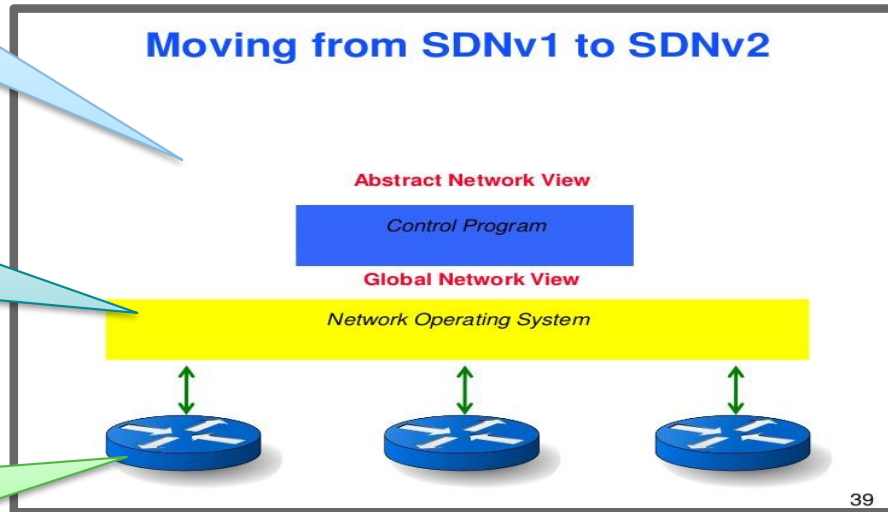
Casado, Shenker, McKeown, Koponen, et al.

describe the SDN architecture like this (2011):

Specification Layer
Logically centralized, transactional, global specification model

Network Operating System Layer
Logically centralized, transactional, global device model

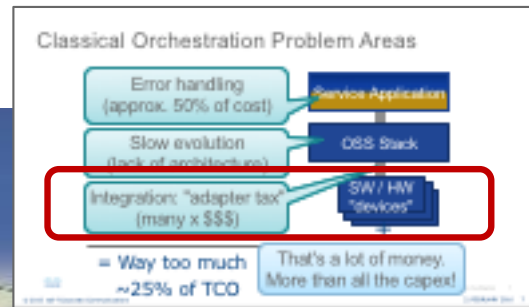
Forwarding Layer
Distributed multi-protocol flow forwarding



www.slideshare.net/martin_casado/sdn-abstractions

No Adaptor Taxation! Demand Standardization!

- I Sign the Declaration of Independence!
- II Download RFP questions for NETCONF and YANG support
- III Encourage participation in Interoperability testing



Sign the Petition Today!

**Get your Copy of the RFP Questions
for NETCONF/YANG!**

<http://info.tail-f.com/noadapertax>

Service Automation Criteria a.k.a. “RFP Questions”

1. Transactional NETCONF
2. Consistent edit-config
3. Validation without activation
4. Confirmed-commit
5. NETCONF over SSH
6. Defaults Handling
7. Standard Models
8. Model Discovery & Events
9. YANG
10. Backwards Compatibility

SAC#1 Transactional NETCONF

Devices **MUST** implement NETCONF 1.1 [RFC6241] or 1.0 [RFC4741]. It is not acceptable with a partial implementation of the mandatory parts of NETCONF. `:rollback-on-error` **MUST** be implemented. At least one of the capabilities `:candidate`, `:writable-running`, or `:startup` **MUST** be implemented. The NETCONF lock operation **MUST** be implemented to specification.

NETCONF isn't completely a smorgasbord. There are mandatory parts that have to be implemented to specification.

Rollback-on-error

“:rollback-on-error MUST be implemented.”

:rollback-on-error is the Atomicity requirement from ACID Transactions.

- The change must be implemented completely or not at all a.k.a. “all-or-nothing”.
- A transaction failure must not be observable from the outside

Many legacy devices implement a try-and-rollback strategy, which technically isn't correct. A failed transaction might in this case upset the network (e.g. lose a BGP peering relation). NSO doesn't know or care, however.

Datastores

“At least one of the capabilities `:candidate`, `:writable-running`, or `:startup` MUST be implemented.”

`:candidate`, `:writable-running` and `:startup` capabilities are all about which NETCONF data stores are supported by the device

- `:candidate` is required for advanced NETCONF use cases, e.g. Network-wide transactions.
- `:writable-running` allows a manager to use a simpler mechanism than configuration through the `:candidate`

A device may very well support both of the above.

Locking

“The NETCONF lock operation MUST be implemented to specification.”

Data stores may or may not be locked by a manager before a configuration change takes place.

Many devices have failed to implement locking correctly according to the NETCONF specification.

SAC#2 Consistent Edit-config

All configuration data MUST be editable through a NETCONF <edit-config> operation. Proprietary NETCONF RPCs that make configuration changes are not sufficient. An <edit-config> operation MUST change the configuration in accordance with the payload, or fail with an error message. In case of failure, there MUST NOT be any change to the configuration on the device. In case of success, all of the payload changes MUST be implemented, and there MUST NOT be any other changes of the configuration than the ones prescribed in the edit-config payload. It MUST be possible to go from any valid configuration to any other valid configuration with a single edit-config operation containing only the minimal delta between the two. A minimal delta MUST only refer to any particular leaf once, i.e. cannot *first* set a leaf to one value, *then* set it to another value.

Edit-config for all data

“All configuration data MUST be editable through a NETCONF <edit-config> operation. Proprietary NETCONF RPCs that make configuration changes are not sufficient.”

Proprietary rpc methods are allowed as a complement to edit-config, and are allowed to have side effects.

Someone might be tempted to only implement a proprietary rpc “execute-cli-command” and no other YANG objects. That’s not good enough.

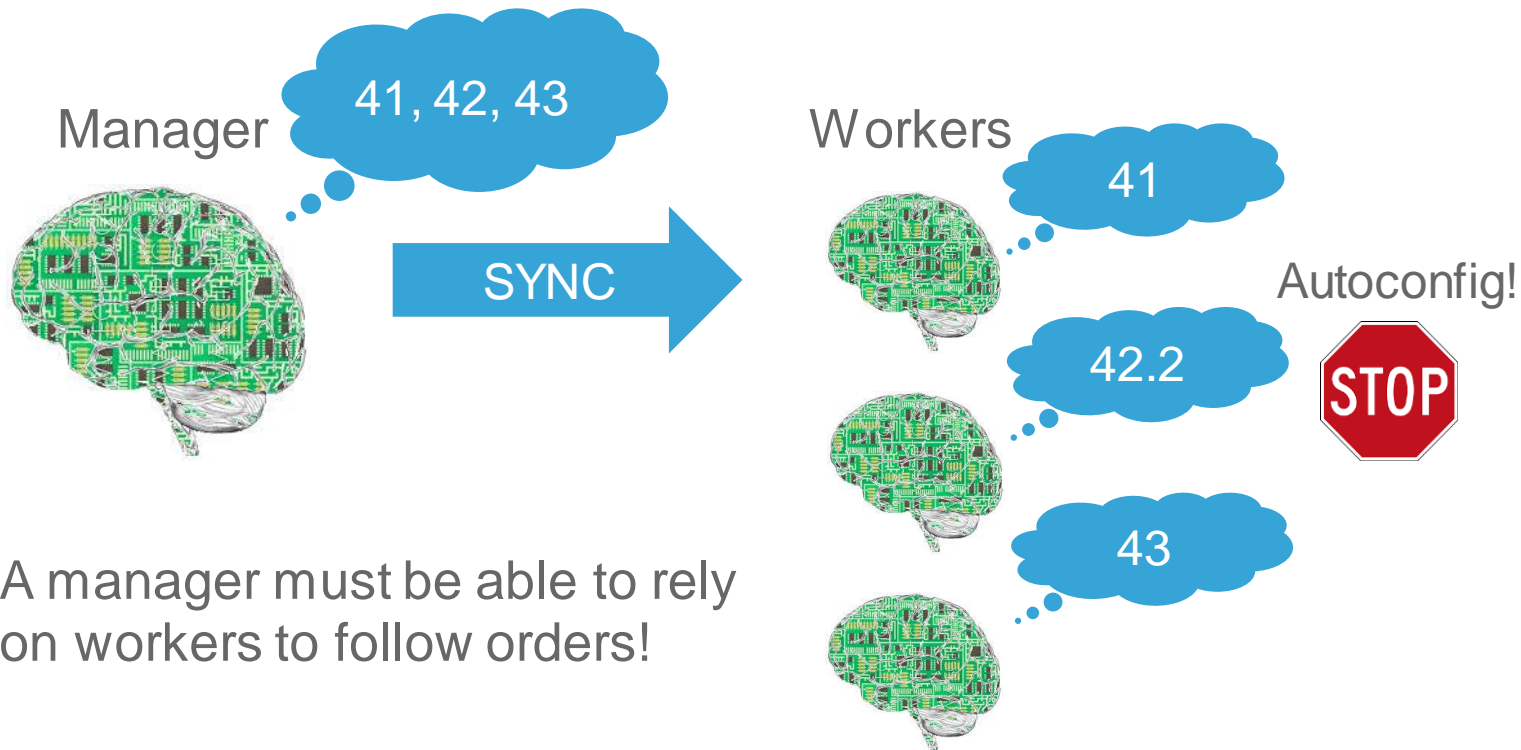
No Autoconfig in Edit-config

“In case of success, all of the payload changes MUST be implemented, and there MUST NOT be any other changes of the configuration than the ones prescribed in the edit-config payload.”

If the device makes other changes, the manager will not know about them, and will get out of sync. The manager will not know that it's out of sync. This will lead to errors down the line.

When side effects are desired, complement with custom rpc operations which modifies the configuration in special ways. The manager will get out of sync, but at least knows it has executed something that might have side effects.

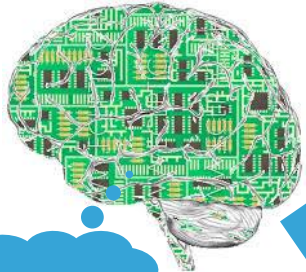
No Autoconfig in Edit-config



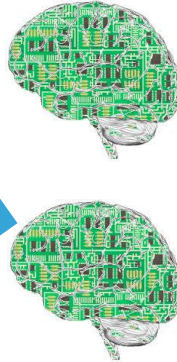
A manager must be able to rely on workers to follow orders!

Autoconfig using RPCs

Manager



Workers



Discovered!

47.5

47.5

rpc discover
SYNC

A manager may use proprietary rpc functions and sync-back.

This is not **edit-config**, however

Validation Dependencies

“It MUST be possible to go from any valid configuration to any other valid configuration with a single edit-config operation containing only the minimal delta between the two.”

The validity of an upcoming configuration must depend on that configuration alone. Specifically the validity must not depend on the currently running configuration, the presence of hardware, the phase of the moon or any other condition that is not part of the configuration.

Operators want to be able to (quickly!) load backups or other saved or computed configurations. This must “work” even if a line card has gone bad.

Validation Dependencies



Since a configuration has been valid in the past, it must be valid now.



Managers rely on being able to go directly to any valid configuration



Sequencing and Admin status

“A minimal delta MUST only refer to any particular leaf once, i.e. cannot *first* set a leaf to one value, *then* set it to another value.”

A transaction is a set of changes, not a sequence. Concepts such as first, then, last, before, and after are meaningless inside a transaction. A transaction can only give each leaf a single value.

Devices will need to properly sequence the work to reach the desired configuration, and cannot rely on operators to specify this order.

Modes which traditionally have prohibited changes (e.g. admin status) must not block transactional configuration changes.



See examples on <https://cisco.jiveon.com/message/285316>

*“The single biggest problem in communication
is the illusion that it has taken place.”*
– George Bernard Shaw

SAC#3 Validation without activation

The :validate capability SHOULD be implemented.

:validate allows a manager to validate the configuration in the candidate data store without activating it. This is essential for network-wide transactions.

SAC#4 Confirmed-commit

If :candidate is supported, :confirmed-commit SHOULD be implemented.

:confirmed-commit is the most reliable form of NETCONF transactions.

If connection is lost during the transaction execution, that is the “command” to roll back to the previous configuration.

Many dreaded forms of network outages would be recovered by implementing this feature.

SAC#5 NETCONF over SSH

NETCONF over SSH [RFC6241] or [RFC4741] MUST be implemented.

NETCONF over TLS may be fine, but NETCONF over SSH is mandatory according to the NETCONF RFCs.

RESTCONF may be a convenient complement to NETCONF. It implements a subset of the NETCONF functionality and aims at being easier to use. It is no replacement for NETCONF.

SAC#6 Defaults handling

The `:with-defaults` capability [RFC6243] MUST be implemented.

Declaring how the device treats default values is essential for mutual understanding of what is being said between a manager and device.

Several default handling modes are possible. The important part is to be clear and consistent about what we mean in situations when things are not spelled out.

*“The most important thing in communication
is hearing what isn’t said.”*
– Peter Drucker

SAC#7 Standard models

Applicable IETF standard YANG models SHOULD be implemented.

<https://datatracker.ietf.org/wg/netmod/documents/>

Using standardized models, possibly extending them with more functionality and control, is key to increased interoperability and lower operator opex cost.

Implementing the models without the other Service Automation Criteria (SAC) is quite pointless, however.

SAC#8 Model Discovery and Events

Data model discovery and download as defined in [RFC6022] SHOULD be implemented. If implemented, all required YANG models should be downloadable. All downloaded YANG models must be free from YANG syntax errors. NETCONF Event Notifications [RFC5277] MAY be implemented.

Being able to download YANG modules directly from the device makes it easier to get the right version of the models, and may enable completely automatic device discovery. If there are syntax errors or some modules missing, this will not work.

NETCONF Notifications are reliable and very useful.

SAC#9 YANG

All data models **MUST** be defined in YANG [RFC6020], and the mapping to NETCONF **MUST** follow the rules defined in this RFC.

Use YANG.

SAC#10 YANG Backwards Compatibility

The data model upgrade rules defined in [RFC6020] section 10 SHOULD be followed. All deviations from section 10 rules MUST be handled by a built-in automatic upgrade mechanism.

Cautions changes to published YANGs should keep us safe. When once in a while something backwards incompatible must be introduced, the device must be able to take care of it.

SAC#11 Operational Data

We should probably add a rule along the lines of

- All operational data **MUST** be retrievable through NETCONF <get> (not only custom rpcs).

We need to define “all operational data”, though.



CISCO

TOMORROW starts here.