

Programmability Concepts with NETCONF/YANG

Craig Hill

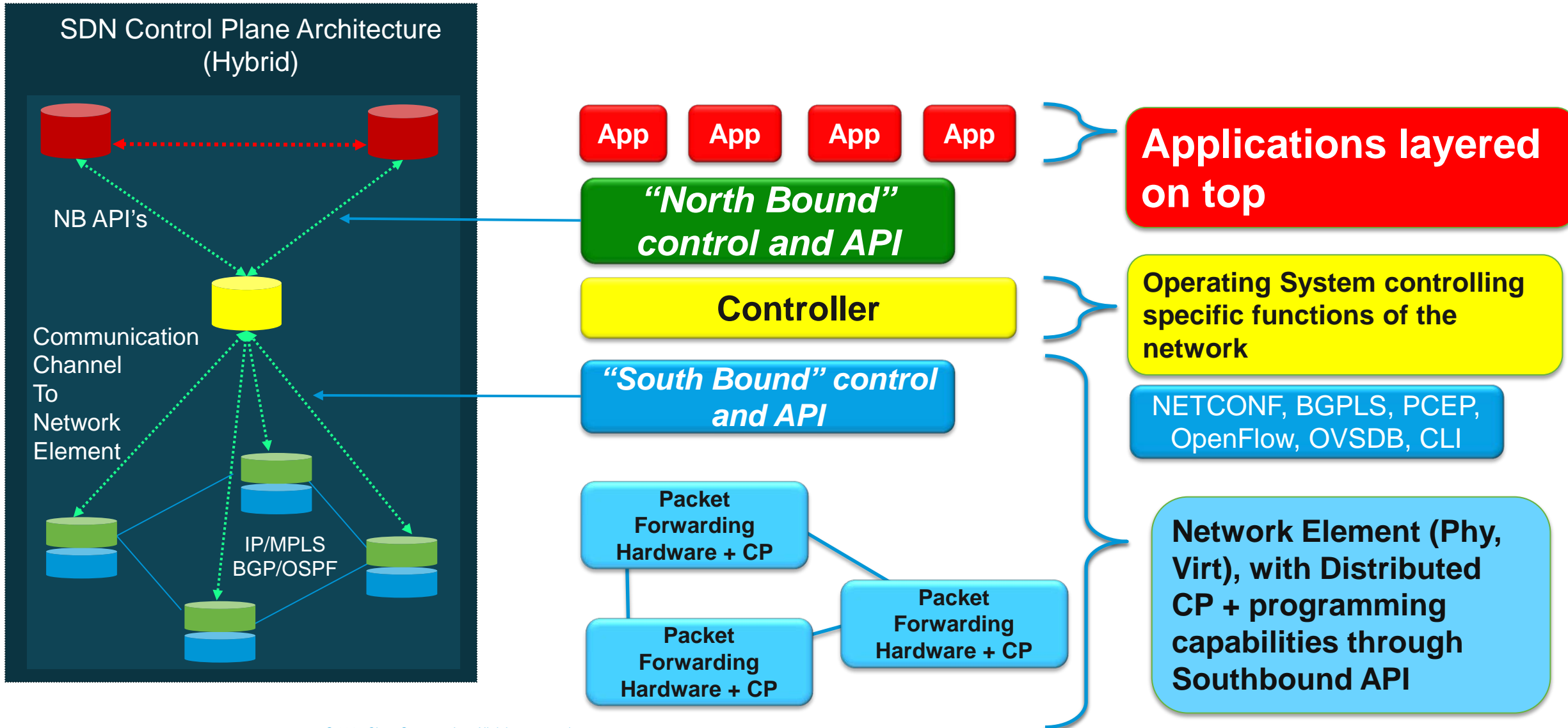
Distinguished SE, U.S. Federal

CCIE #1628

Vers 1.1

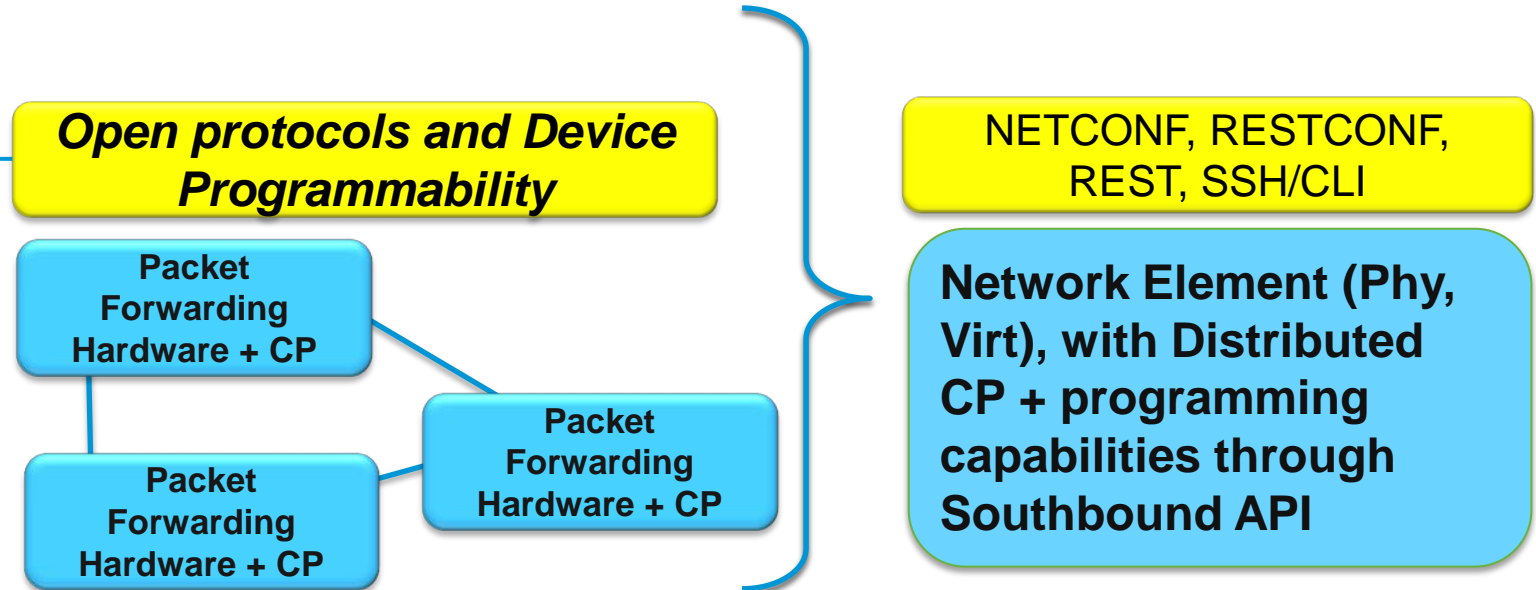
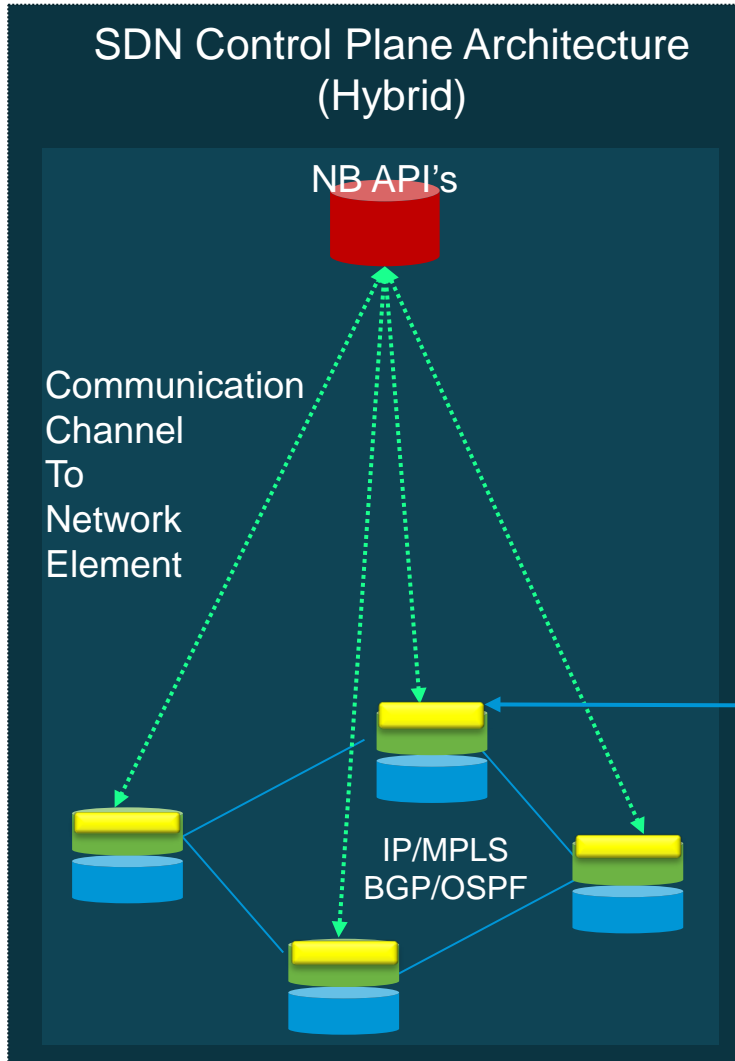
Hybrid SDN Model

Distributed and Centralized (via controller) Control Plane + Standards Data Plane



Hybrid SDN Model

Open Protocols and Direct Device Programmability

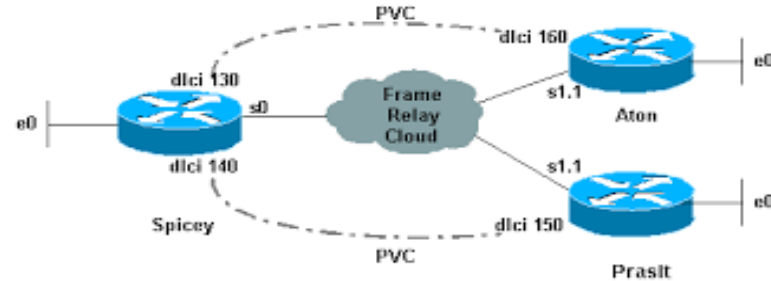


Why Programmability?



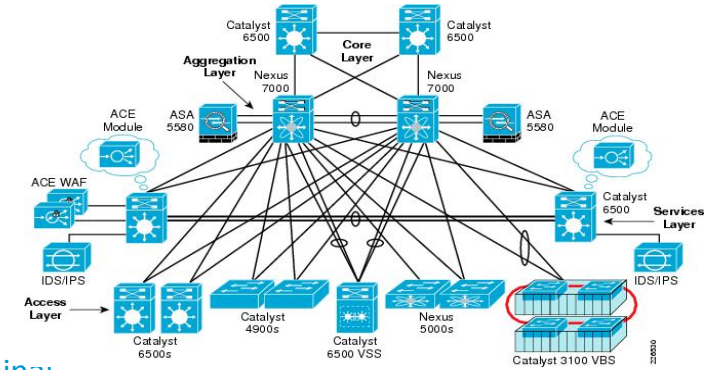
Evolution of Network Configuration

1990s



```
CAT6K>enable
CAT6K# config terminal
CAT6K(config)# interface fastethernet 1/1
CAT6K(config-if)# ip address 1.1.1.1 255.255.255.0
CAT6K(config-if)# no shutdown
CAT6K(config-if)# exit
CAT6K(config)# router eigrp
CAT6K(config-router)# network 1.1.1.0
CAT6K(config-router)# exit
CAT6K(config)# exit
CAT6K# copy run start
```

Today



```
NEXUS>enable
NEXUS# config terminal
NEXUS(config)# interface ethernet 1/1
NEXUS(config-if)# no switchport
NEXUS(config-if)# ip address 1.1.1.1 255.255.255.0
NEXUS(config-if)# no shutdown
NEXUS(config-if)# exit
NEXUS(config)# feature eigrp
NEXUS(config)# router eigrp Test1
NEXUS(config)# interface ethernet 1/1
NEXUS(config-if)# ip router eigrp Test1
NEXUS(config-if)# no shutdown
NEXUS(config-if)# end
NEXUS# copy run start
```

We need to better manage network devices programmatically

Why Programmability is important?



Save Time



Human Error

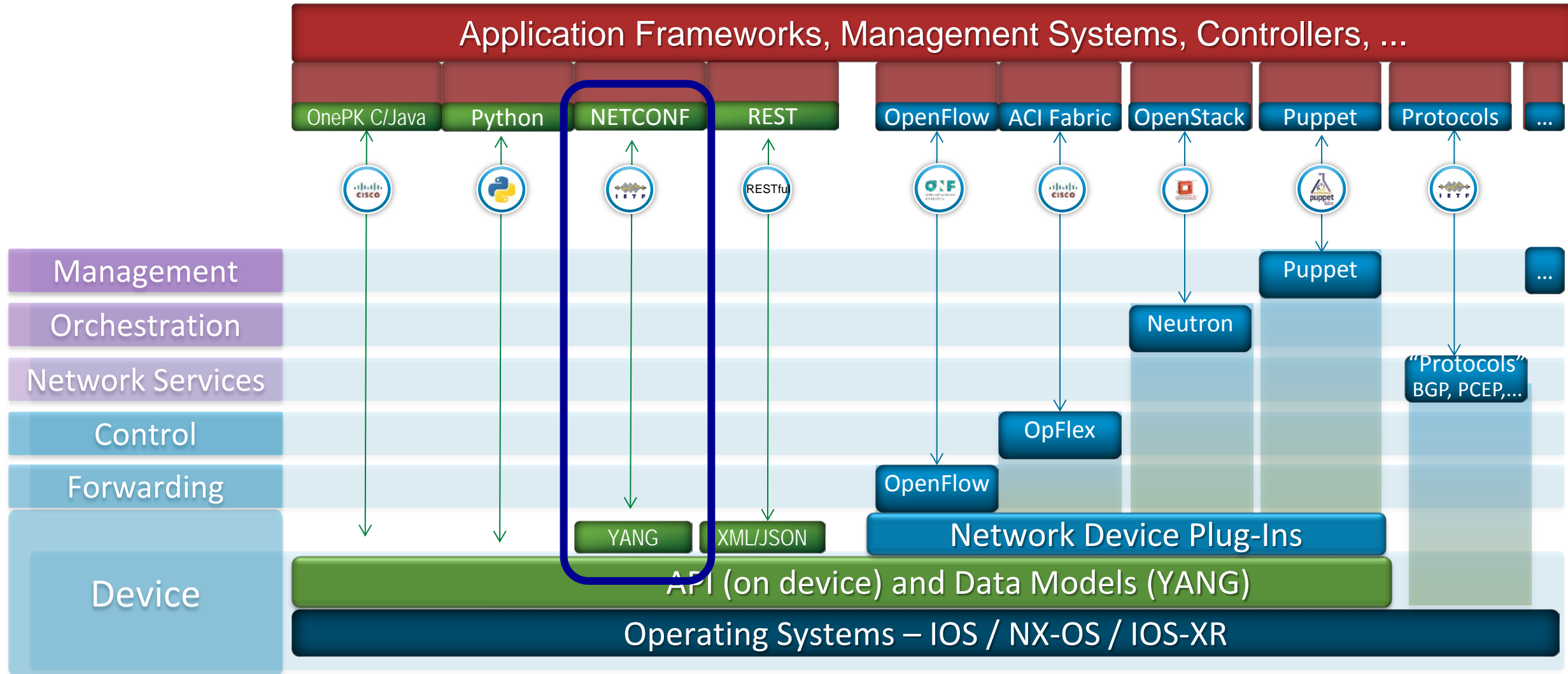


Customize



Innovate

Programmability: Network Automation Interfaces



Why NETCONF/YANG? - Informational RFC 3535

Abstract

This document provides an overview of a workshop held by the Internet Architecture Board (IAB) on Network Management. The workshop was hosted by CNRI in Reston, VA, USA from June 4 thru June 6, 2002. The goal of the workshop was to continue the important **dialog** started between **network operators** and protocol developers, and to guide the IETFs focus on future work regarding network management.

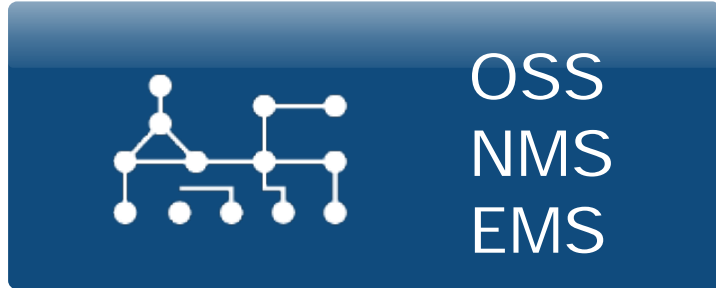
- SNMP had failed
(**For configuration**, that is, **NOT mgmt**)
Extensive use in fault handling and monitoring
- CLI scripting
"Market share" 70%+





configuration



Implications of RFC3535 – Legacy Situation



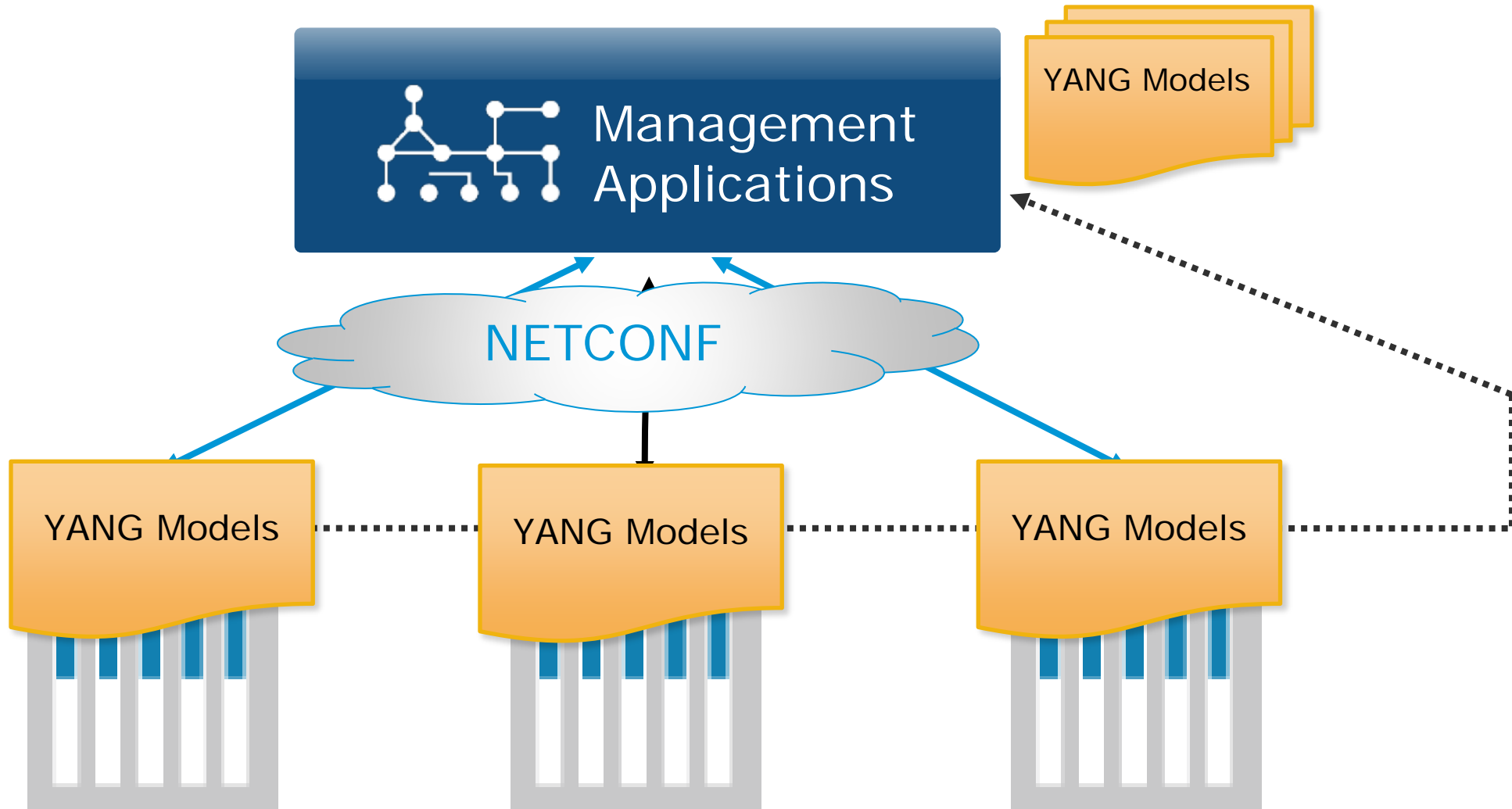
Cost and
complexity

- 
- 
- No well-defined protocols and data-models
 - Lack of atomicity
 - Ordering problem

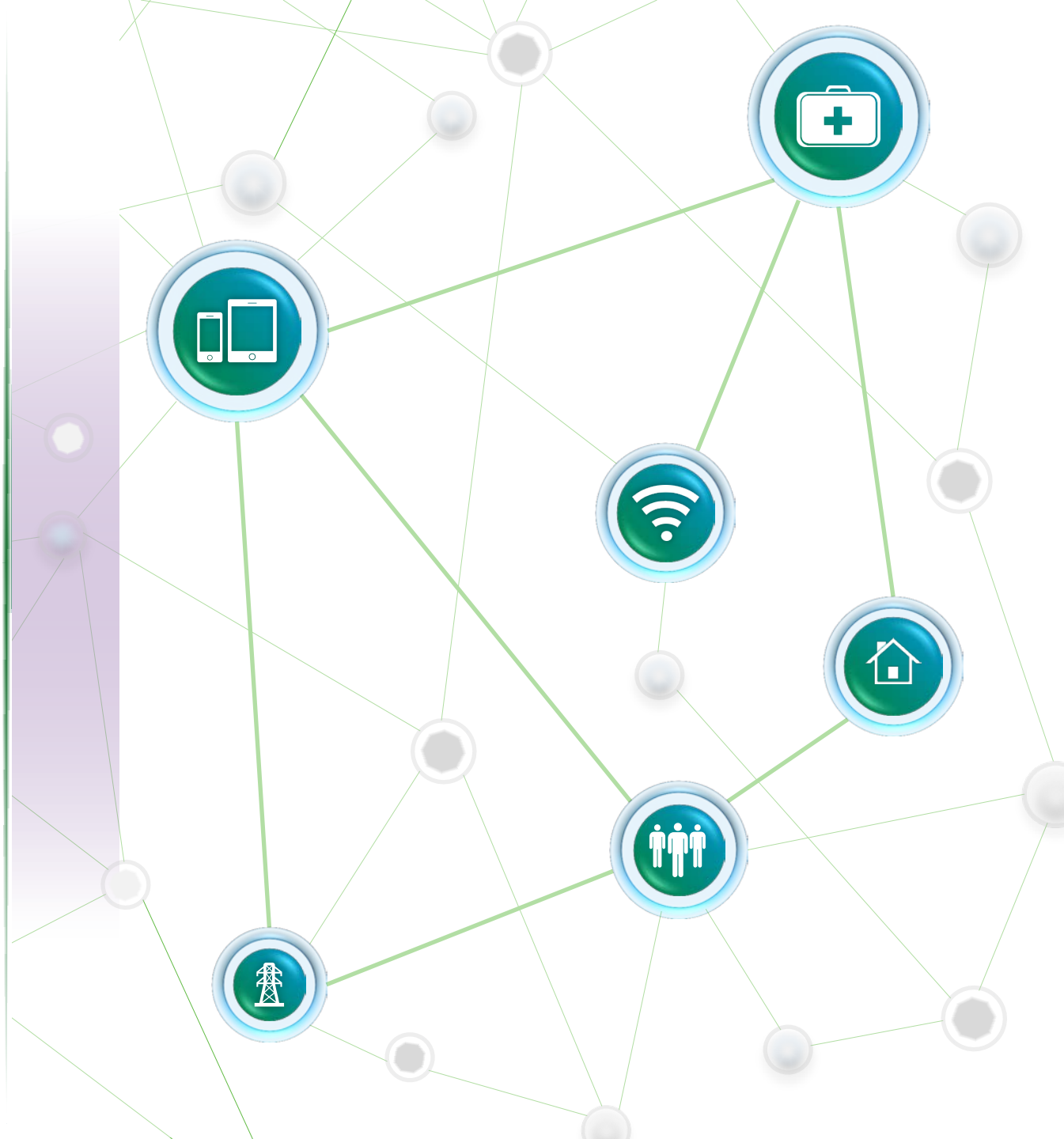
Implications of RFC3535, with transactions



NETCONF and YANG in Context



NETCONF Protocol Introduction



What is NETCONF?

NETCONF (Network Configuration Protocol) is an IETF configuration management protocol

- Not only configuration access, but operational state data
- NETCONF uses XML (as you are going to see)

Why NETCONF?

Easy to Use

Separates Config and Operational Data

Lots of Tooling

Accessible Format

Error Checking

Backup/Restore Capability

Human and Machine Friendly

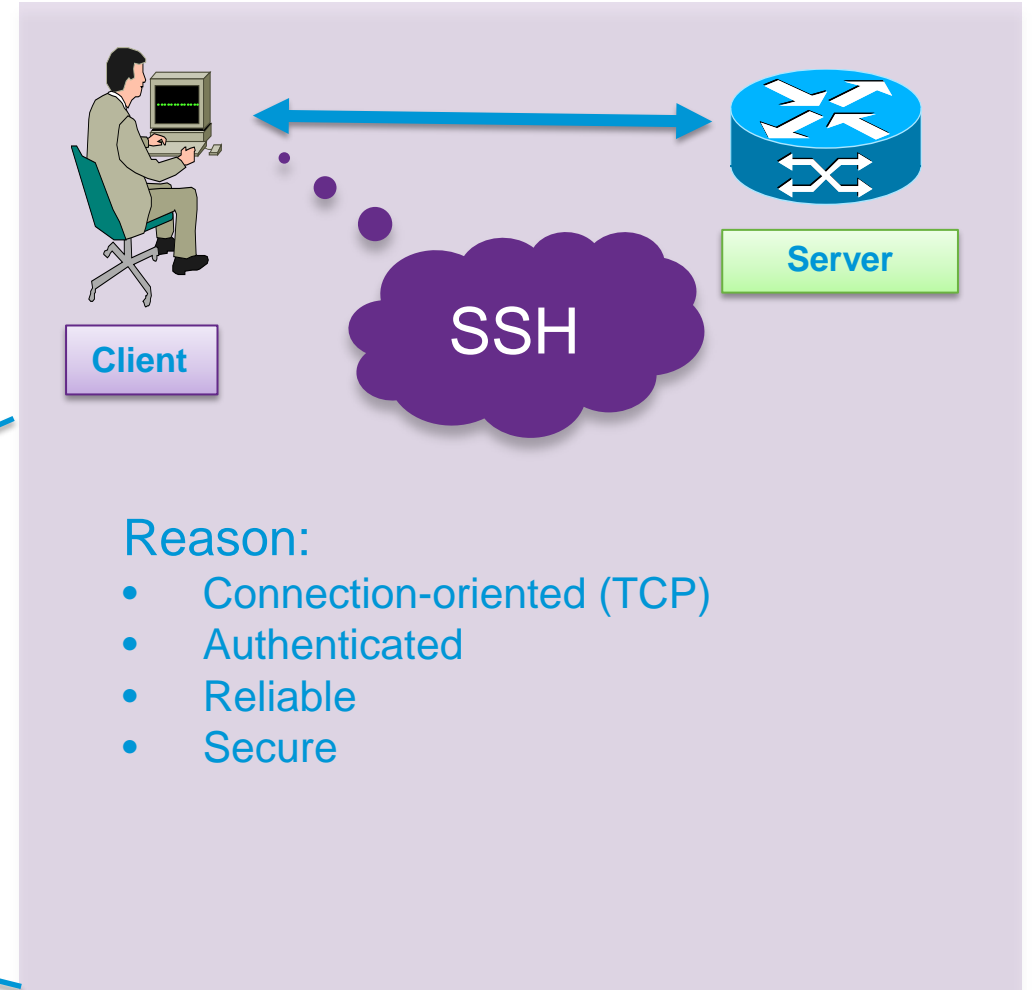
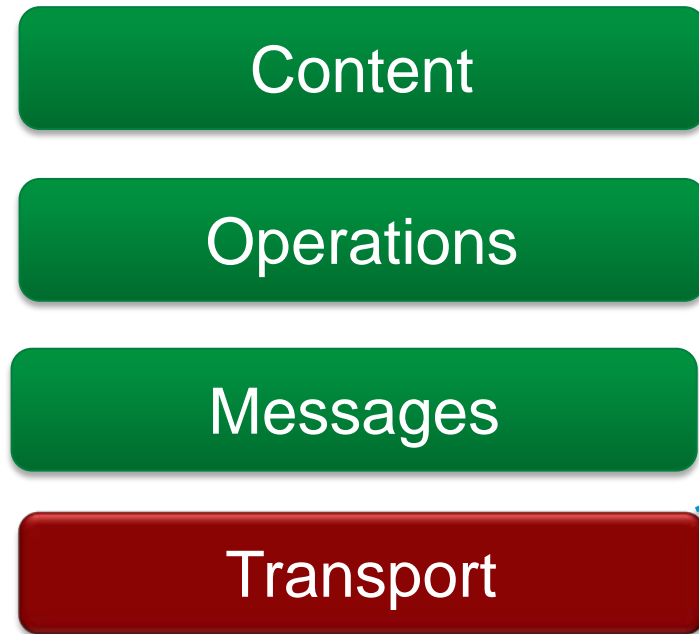
Next-Gen Configuration
Management
Requirements

RFC3535

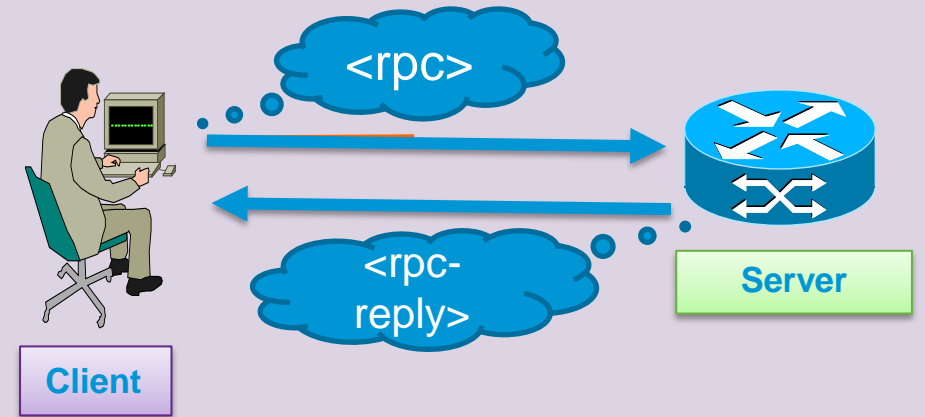
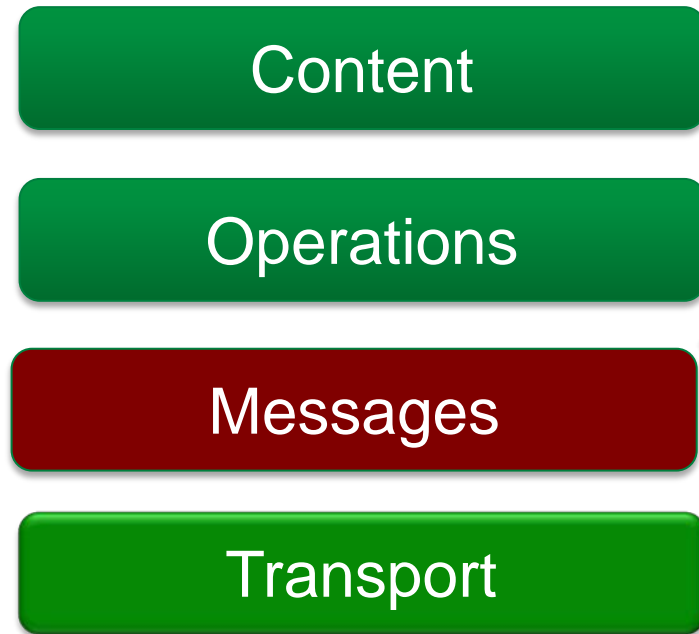
NETCONF IETF Standard Information

V 1.0	V 1.1	Extension
RFC 6535 Background and Requirements	RFC 6241 1.1 Base NETCONF Protocol	RFC 5277 Notifications
RFC 4741 1.0 Base NETCONF Protocol	RFC 6242 NETCONF over SSH	RFC 5717 Partial Locking
RFC 4742 NETCONF over SSH		RFC 6243 With defaults
		RFC 6244 NETCONF + YANG Architectural Overview

NETCONF Protocol Stack



NETCONF Protocol Stack



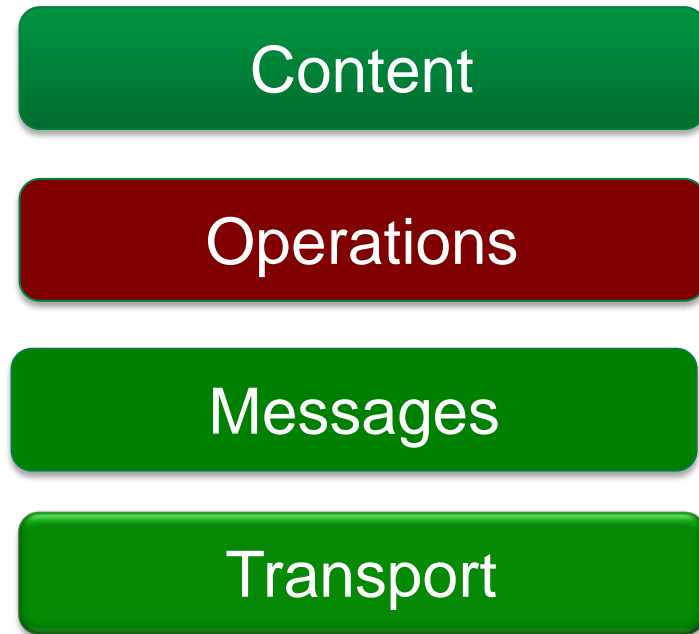
NETCONF Messages:

- Remote Procedure Call (RPC)

RPC Client's Request Methods:

- Controller
- NMS
- Script
- Manual

NETCONF Protocol Stack



OPERATION	DESCRIPTION
<get-config>	Retrieve data from the running configuration
<get>	Retrieve running configuration or device statistic
<edit-config>	Modify a configuration database <ul style="list-style-type: none">- operation = merge (default), delete, create, replace, remove- test-option (:validate), error-option
<copy-config>	Copy a configuration database
<delete-config>	Delete a configuration database
<commit>	Commit candidate configuration to the running db
<lock>/<unlock>	Lock or unlock a configuration datastore system
<close-session>	Terminate this session
<kill-session>	Force Termination of session

NETCONF Protocol Stack

Content

Operations

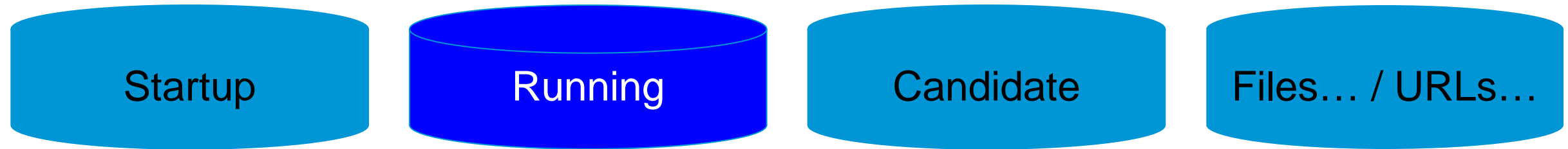
Messages

Transport

```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>
      urn:ietf:params:netconf:base:1.0
    </capability>
    <capability>
      urn:ietf:params:netconf:capability:candidate:1.0
    </capability>
    <capability>
      urn:ietf:params:netconf:capability:notification:1.0
    </capability>
  </capabilities>
  <session-id>285212672</session-id>
</hello>
]]>]]>
```

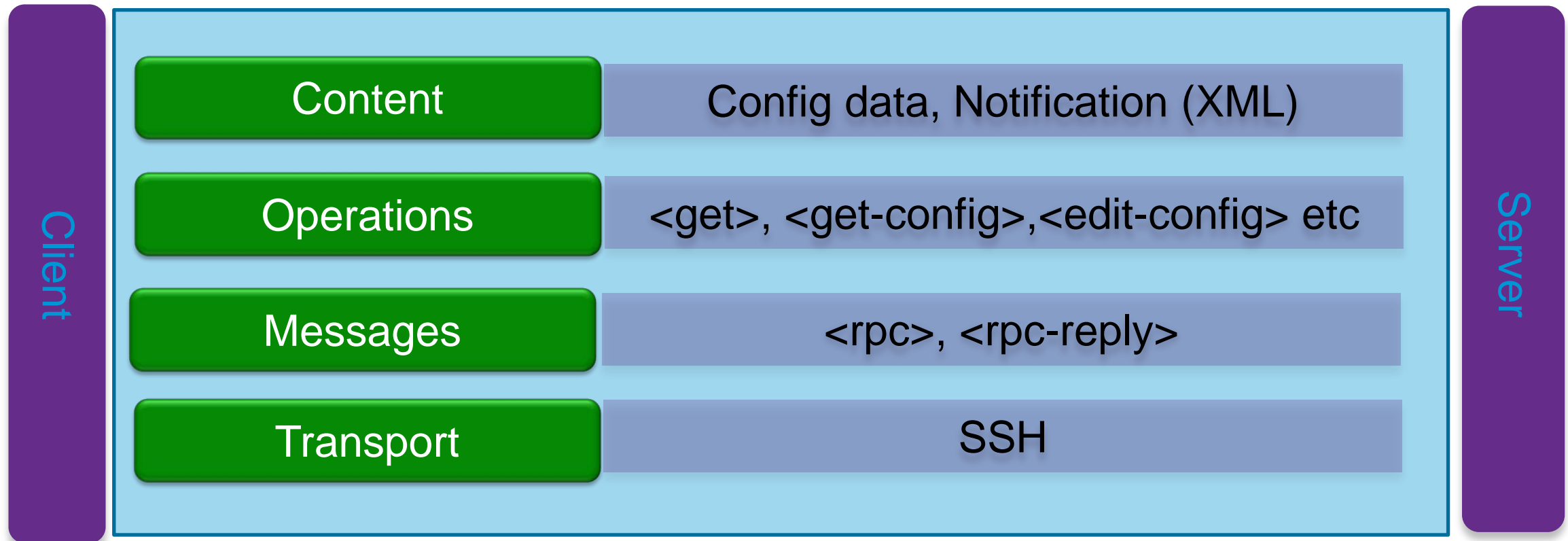


NETCONF Configuration Data Stores

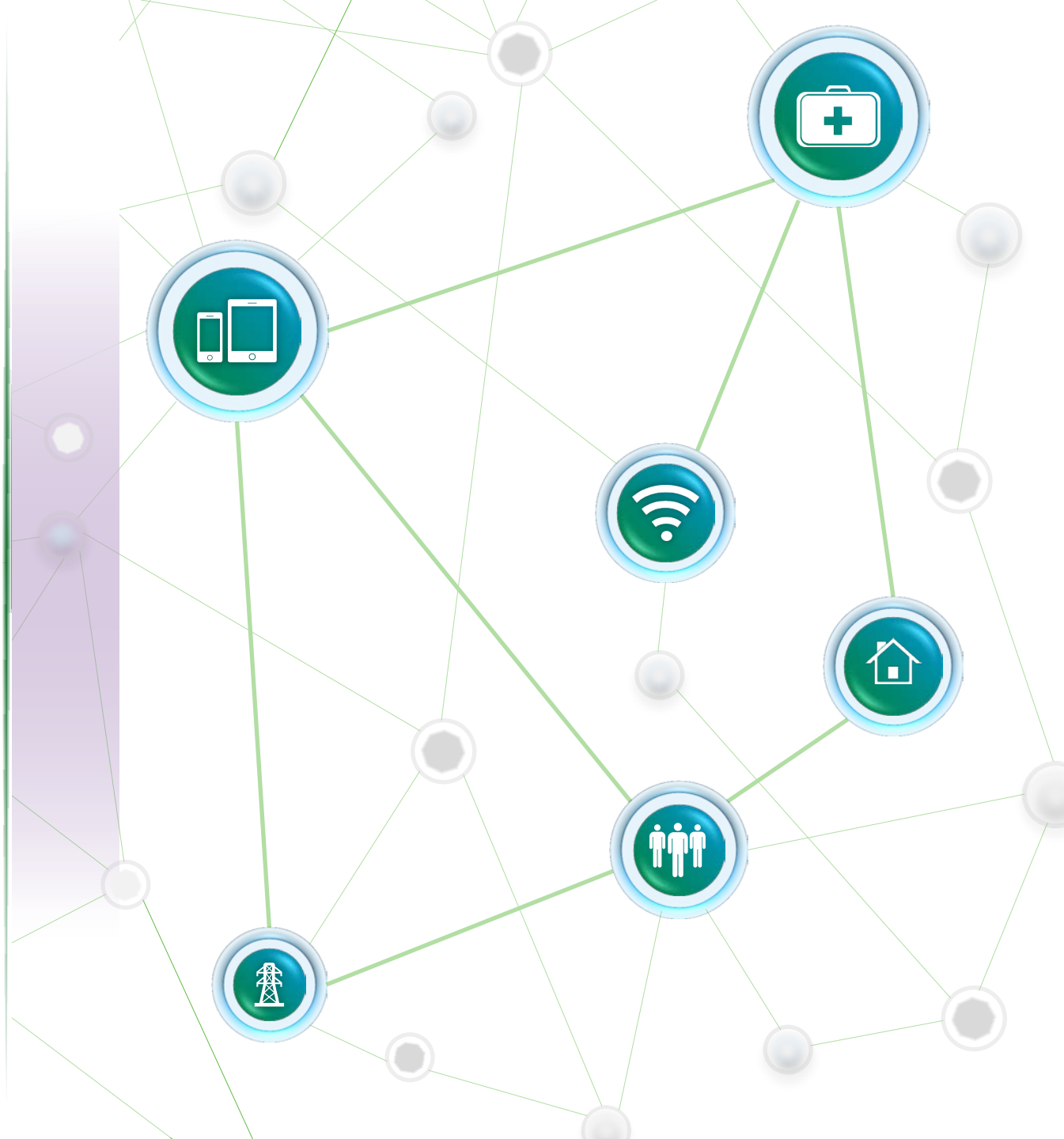


- Named configuration stores
 - Each data store may hold a full copy of the configuration
- Running is mandatory, Startup and Candidate optional (*capabilities :startup, :candidate*)
- Running may or may not be directly writable (*:writable-running*)

NETCONF Protocol Stack Summary



YANG Language Introduction



Overview

- YANG

Data definition language for management data

... Original focus on configuration information, but not any more restricted to it

... Original context: NETCONF

... Can be separated from NETCONF (not an original goal but important!)

Consistent data enabled through models defined in a common language

Model-driven Interfaces

- NETCONF

Network Configuration Protocol

Fill the void between SNMP (monitoring) and CLI (geared towards humans)

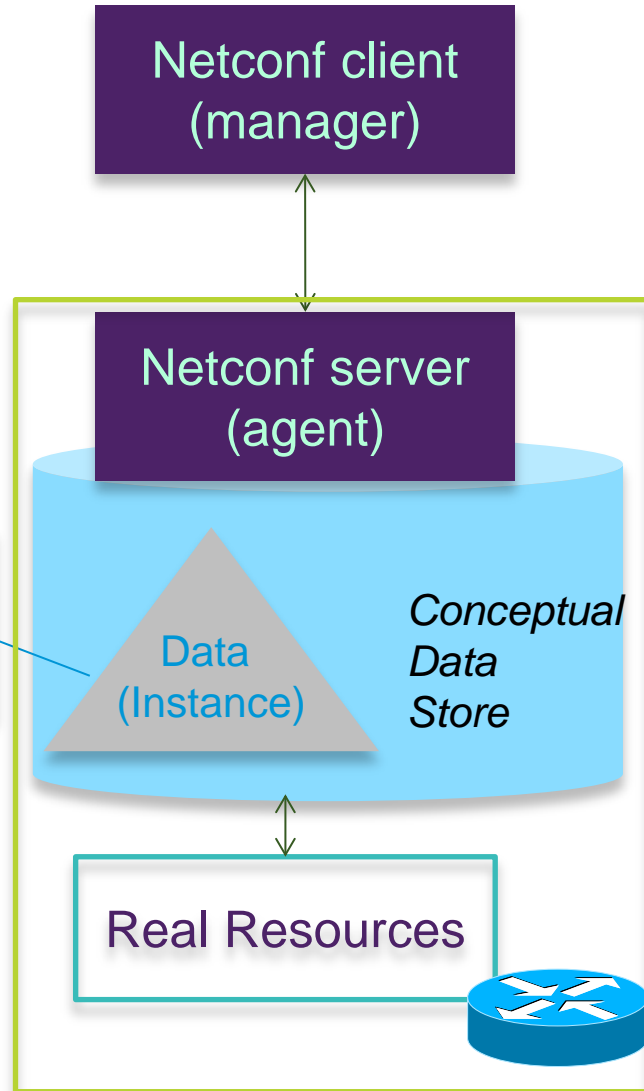
- Standardization through IETF (mainly netconf and netmod WGs)



What can you do with YANG

- Define data models for something that you need to manage. Example:
 - Interfaces and their configuration
 - Access Control Lists (rules that govern policies how to handle packets)
 - Topology
- Specify semantics that allow to validate and maintain consistency
 - Operational or configuration – “read-only” or “read-write”
 - Conditions and constraints – value dependencies, referential integrity, ...
 - Units, defaults, ranges, alternatives, ...
- Extend data models previously defined
 - Simply “insert” new data where applicable (“augmentation”)
 - Add a new sub-tree into the existing model
- Instantiate the data as “conceptual datastore” and manage your device by accessing and manipulating contents of that datastore
 - Datastore: an abstraction of a “real resource” – what you are managing

YANG-based Management Architecture



An NMS, an SDN Controller

*message exchanges include
data encoded per transfer
syntax (e.g. XML)*

Operations retrieve + modify
datastore contents
Data defined per YANG
(the “schema”)

Internal API calls

Interfaces, ACLs, BGP, ...

Note: a device can have several
datastores

“Running”, “Startup”, “Candidate”
Only “running” contains operational data
Operations can target specific datastores
- as a whole (e.g. copy) or data items in it

YANG structure

An index, so a list

```

+--rw if:interfaces
  +--rw if:interface [name]
    +--rw name string
    + ...
    +--rw ipv4?
      +--rw enabled? boolean
      +--rw ip-forwarding? boolean
      +--rw address [ip]
        +--rw ip inet:ipv4-address
        +--rw (subnet)?
          +--:(prefix-length)
            +--rw ip:prefix-length? uint8
          +--:(netmask)
            +--rw ip:netmask? inet:ipv4-address
      +--rw neighbor [ip]
        +--rw ip inet:ipv4-address
        +--rw phys-address? yang:phys-address
    +--rw ipv6?
      +--rw enabled? boolean
      +--rw ip-forwarding? boolean
      +--rw address [ip]
        +--rw ip inet:ipv6-address
        +--rw prefix-length? uint8
      +--rw neighbor [ip]
        +--rw ip inet:ipv6-address
        +--rw phys-address? yang:phys-address
      +--rw dup-addr-detect-transmits? uint32
      +--rw autoconf
        +--rw create-global-addresses? boolean
        +--rw create-temporary-addresses? boolean
        +--rw temporary-valid-lifetime? uint32
        +--rw temporary-preferred-lifetime? uint32

```

Data type, indicative of a leaf item

"()" indicates a choice of alternatives

"?" indicates it may or may not be included

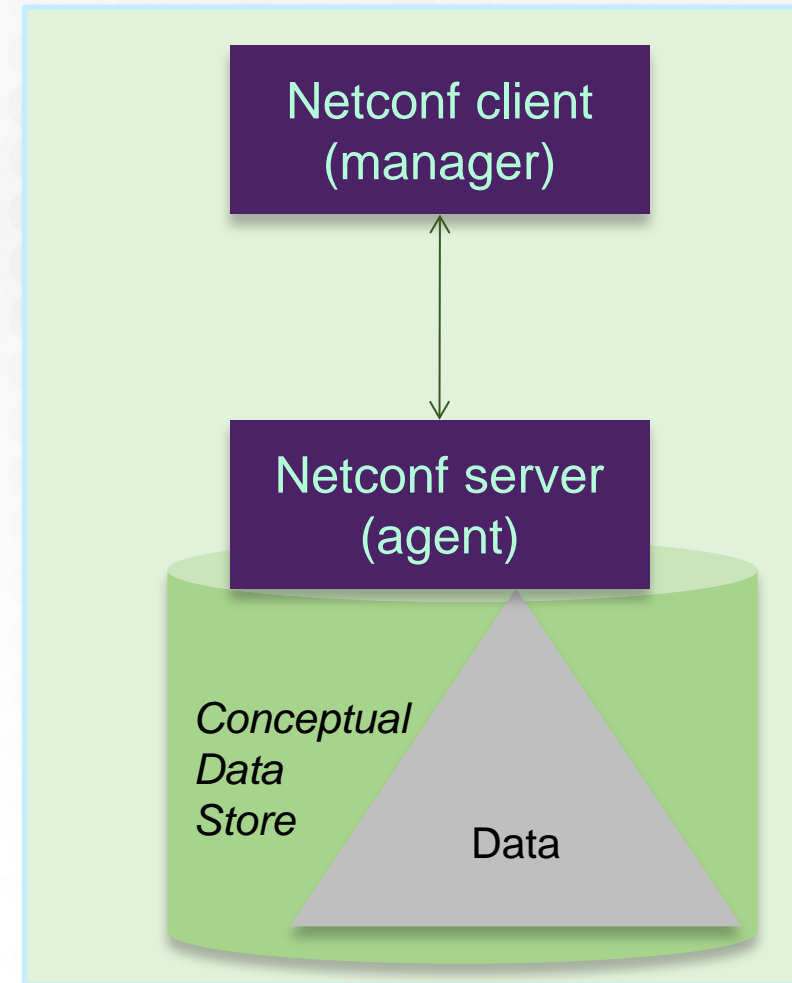
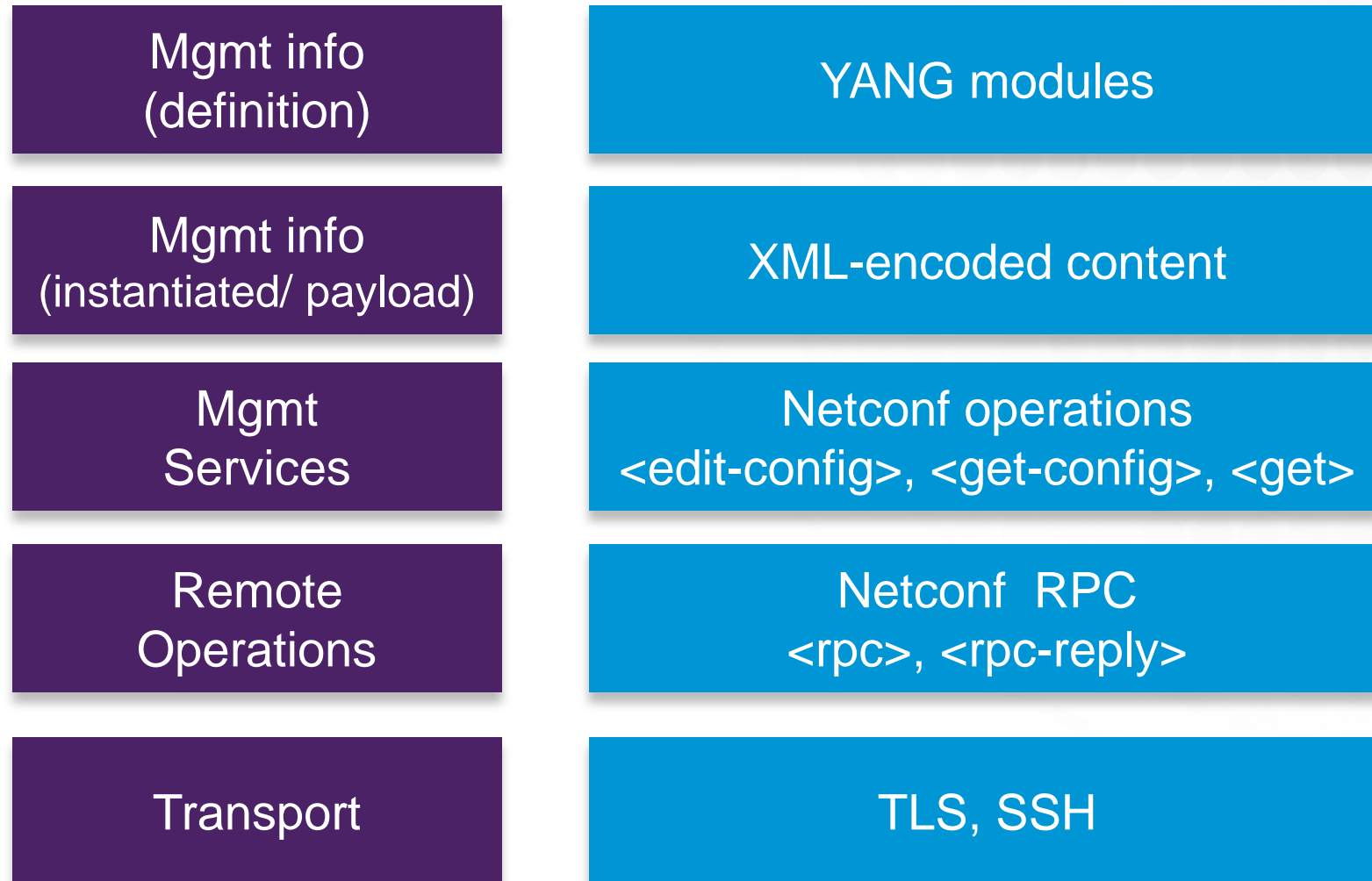
"rw" means it's writable

prefix indicates definition source

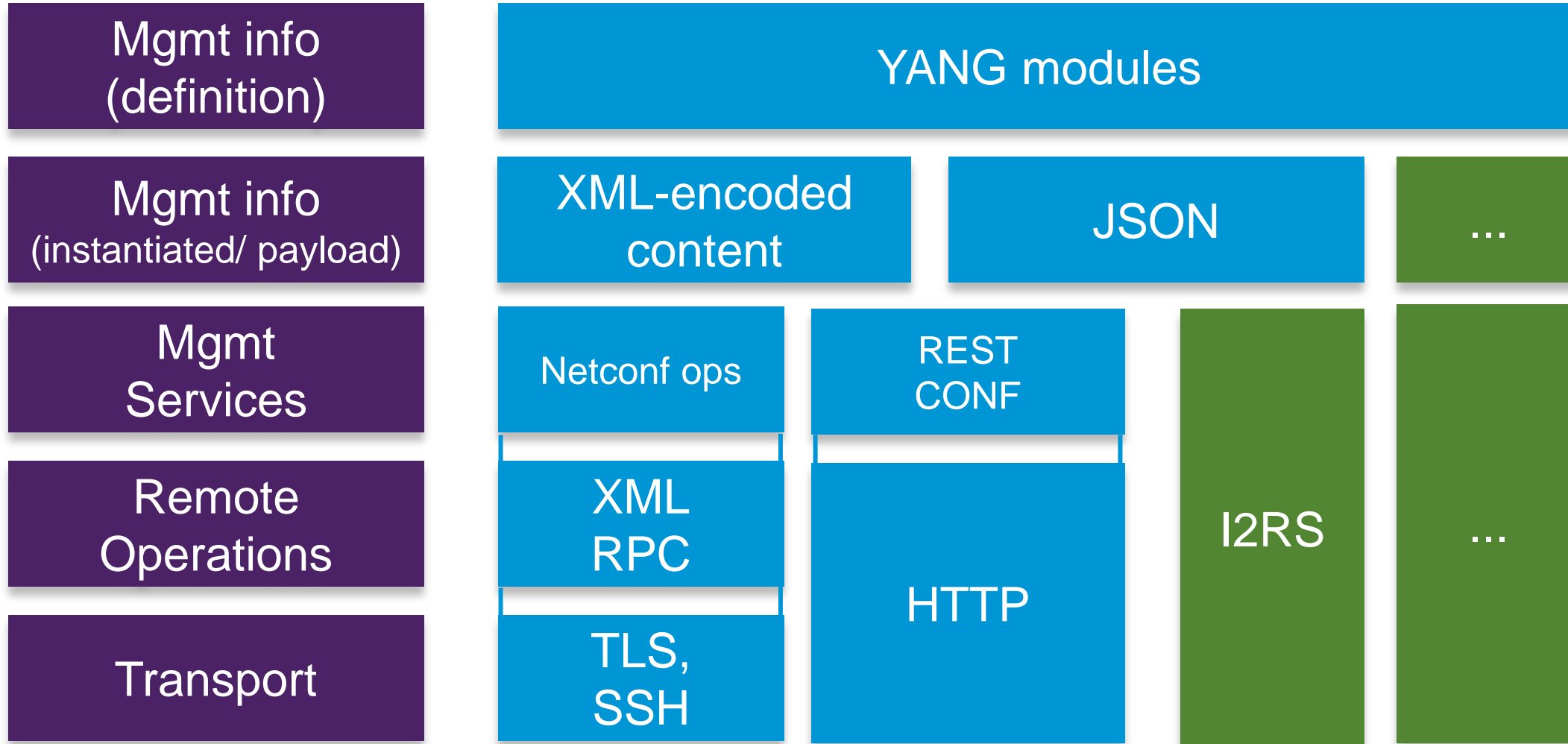
Instance information

```
<interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
  <interface>
    <name>eth0</name>
    <type>ethernetCsmacd</type>
    <location>0</location>
    <if-index>2</if-index>
    <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
      <address>
        <ip>192.0.2.1</ip>
        <prefix-length>24</prefix-length>
      </address>
    </ipv4>
    <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
      <address>
        <ip>2001:DB8::1</ip>
        <prefix-length>32</prefix-length>
      </address>
      <dup-addr-detect-transmits>0</dup-addr-detect-transmits>
    </ipv6>
  </interface>
</interfaces>
```

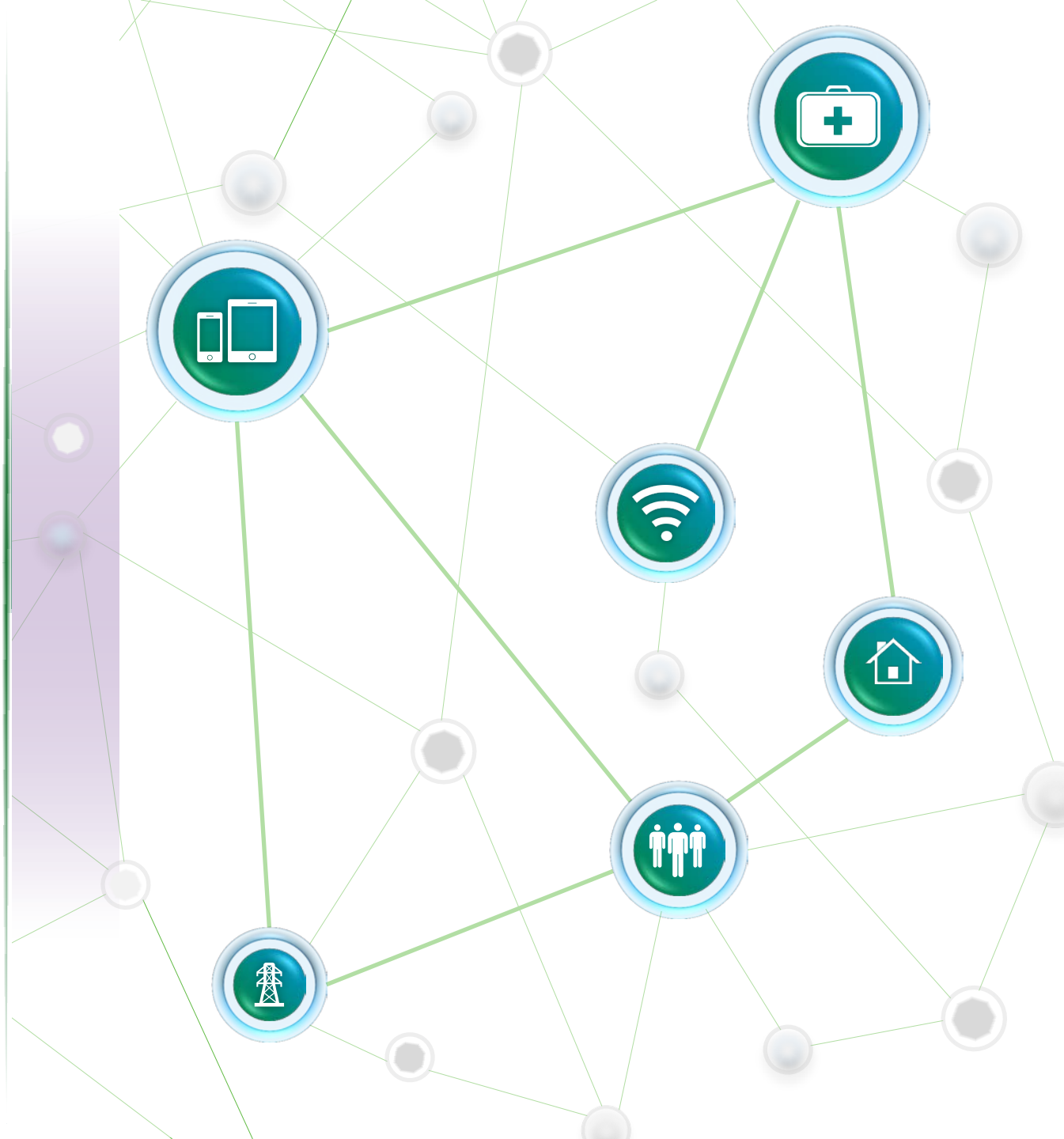
YANG in the context of NETCONF



YANG beyond Netconf



NETCONF/YANG Roadmap



IOS XR YANG Model Support Overview

Release	Components		Platform Support
XR-5.3.0 infra + 20 components	cdp crypto-sam ha-eem ifmgr infra-infra ip-domain lpv4-io ipv4-ma ipv6-ma lib-keychain	manageability-netconf max-xml-ttyagent parser qos-ma rgmgr shellutil subscriber-infra-tmplmgr tty-management tty-server tty-vty	ASR9k

ASR 9000 IOS-XR NETCONF Configuration Link:

http://www.cisco.com/c/en/us/td/docs/routers/asr9000/software/asr9k_r5-3/security/configuration/guide/b-syssec-cg53x-asr9k/Implementing_the_Network_Configuration_Protocol.html

IOS XR NETCONF/YANG Model Upcoming Support

- IOS XR YANG Support

Release	Components		Platform Support
XR-5.3.1 14 additional components	crypto/ssh drivers/lib/media/ether infra/alarm/logger infra/mibs/cedundancymib infra/rsi infra/syslog ip/static	lib/mpp platforms/chassis-control/invmgr snmp/agent snmp/mibs/entitymib snmp/mibs/entstatemib snmp/mibs/frucontrolmib snmp/mibs/itmib	ASR9k CRS

- Platform Support

NCS 4K	RLS 6.0
NCS 6K	RLS 6.0
XRv	RLS 5.3.1

- Tool Support

NCCClient	RLS 5.3.1
-----------	-----------

IOS XE NETCONF/YANG Model Upcoming Support

- IOS XE NETCONF/YANG Support

IOS-XE 3.17 CSR 1000v, ASR 1000, ISR 4400, ASR 9000, Cat 4K, 3K, 2K

- IOS XE REST Support

Thank you.

