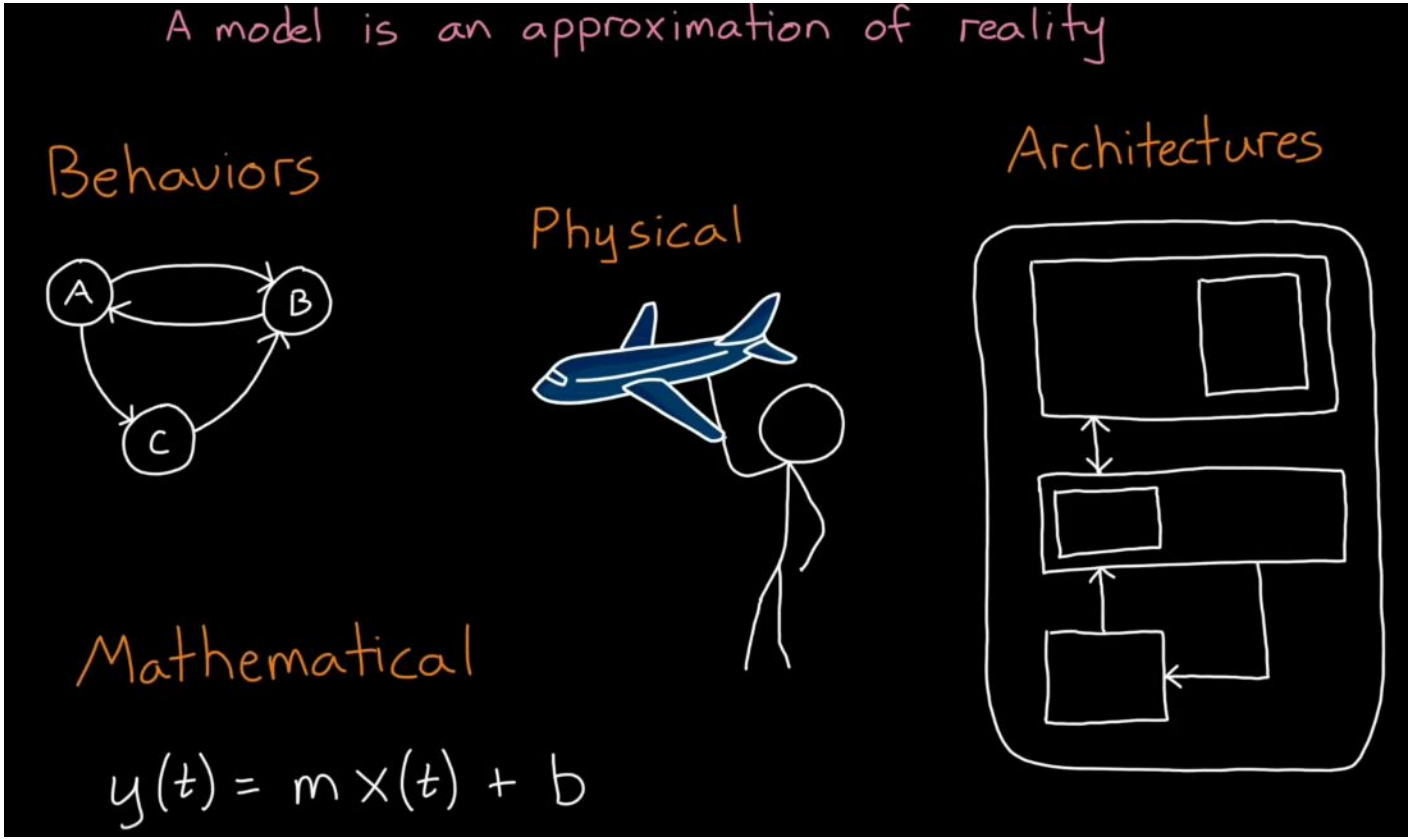
Part I
Modeling Introduction

What is a Model

System Engineering: Model based Approach - https://www.youtube.com/watch?v=bckHxdGjtQc

Model is an abstract representation, description and definition of the structure, processes, information and resources.

Its a physical, mathematical, or otherwise **logical** representation of a system, entity, phenomenon, or process



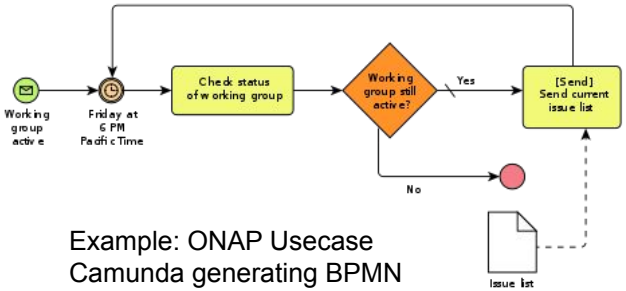
Model of what?

Types of Model

1. Business Process Model
2. Structural Model
3. Behavioral Model
4. Information/Data Model

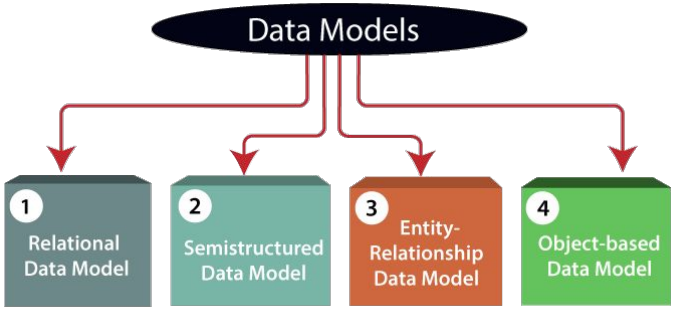
Note: Model could be of many more types, but they are not related to us.

Business Process Model



Example: ONAP Usecase Camunda generating BPMN (ONAP SO)

https://www.javatpoint.com/data-models

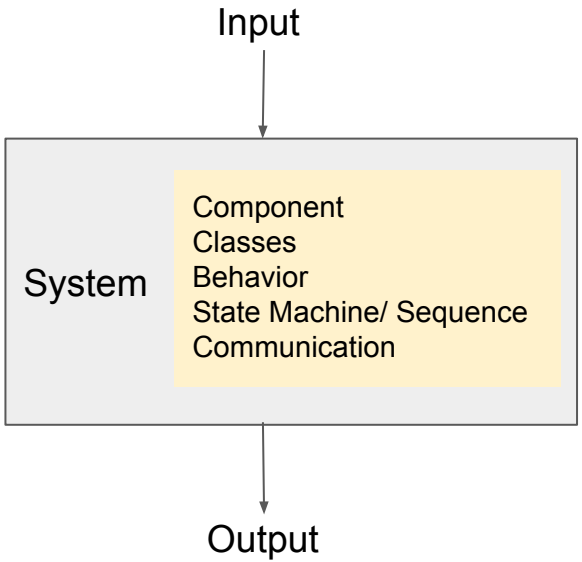


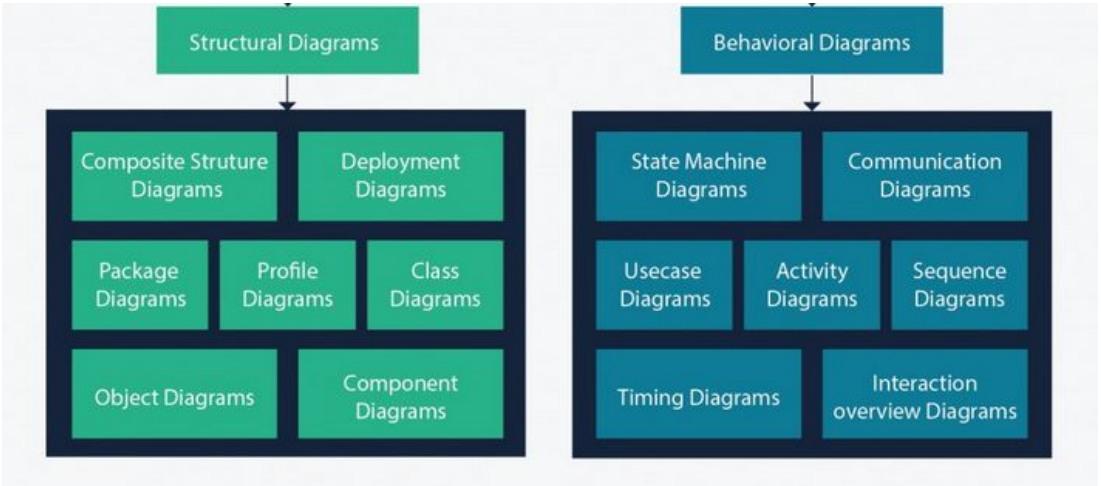
Row Column

Structures with hierarchies

Entities and their relationship

Extension of ER (Inheritance)





blob:https://web.whatsapp.com/b3f71518-14ea-4866-819e-c3cc4d302be9

Models in software life cycle

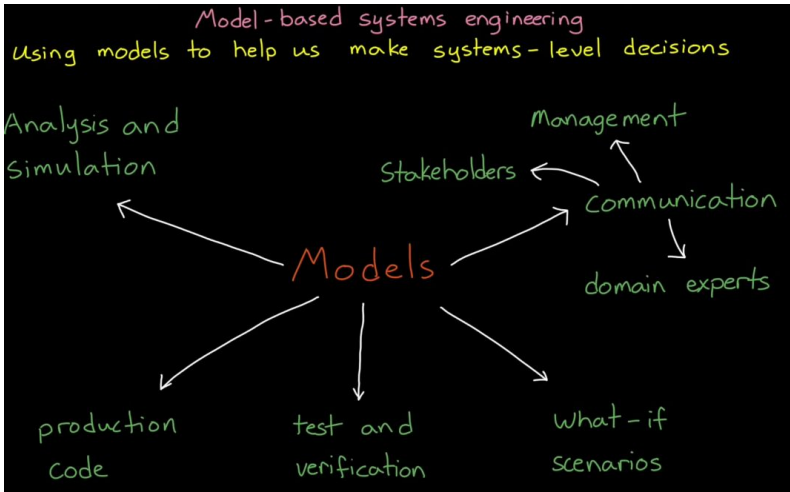
Life-Cycle Activity	Models	Methods & Tools
Software Management	• Life-Cycle Process Model • Work Breakdown Structure • Constructive Cost Model (COCOMO) • Project Plan • Configuration Management (CM) Plan • Risk Management Plan	• Effort, Schedule and Cost Estimation • Risk Analysis • Data Collection • Project Tracking • CM Management • Iterative/Incremental Development • Agile Development
Software Requirements	• Functional Model • User Class Model • Data Flow Diagram • Object Model • Formal Model • User Stories	• Requirements Elicitation • Prototyping • Structural Analysis • Data-Oriented Analysis • Object-Oriented Analysis • Object Modeling Language (OML) • Formal Methods • Requirements Specification • Requirements Inspection
Software Design	• Architectural Model • Structure Diagram • Object Diagram • Class Specification • Data Model	• Structured Design • Object-Oriented Design • OML • Modular Design • Integrated Development Environment (IDE) • Database Management System (DBMS) • Design Review • Refinement
Software Construction	• Detail Design Document • Pseudocode • Flow Chart • Program Code • Unit Test Plan • Integration Test Plan	• Detailed Design • Functional Programming • Object-Oriented Programming • IDE • DBMS • Black Box/White Box Testing • Basic Path Testing • Unit Testing • Code Review • Proof of Correctness • Software Reuse • Integration • Integration Testing
Software Testing	• System Test Plan • Reliability Model • Software Maintenance Process	• Usability Testing • System Testing • Acceptance Testing • Regression Testing • Reliability Testing • Non-Functional Software Testing
Software Maintenance	• Software Maintenance Process	• Automated Testing Tools • Maintenance Change • Impact Analysis • Inventory Analysis • Restructuring • Reverse Engineering • Re-engineering

Above image shows some examples of models and tools used in different phases of software life cycles.

Why Model

Models are representations that can aid in defining, analyzing, and communicating a set of **concepts**. System models are specifically developed to support analysis, specification, **design**, **verification**, and **validation** of a system, as well as to communicate certain information.

- **A means for Analysis**
  a. Formal approach considering all possible scenarios ensure faster error detection.
- **A means for communication**
  a. Better coordination among different stakeholders (Developers, testers, architects, manager etc)
  b. Easier requirements gathering.
- **Continuous evaluation**
  a. Avoid breaking existing system
  b. Faster existing system understanding (system visualization via model)
- **Well defined contract with external world with help to Data modeling**
- **A means for Automation**
  a. Auto code generation
  b. Automated verification (design, code)
  c. Automated implementation



What is a Modeling Language

A modeling language is **any artificial language that can be used to express information or data or knowledge or systems in a structure that is defined by a consistent set of rules**. Or in simple terms, modeling language is a language to define model.

A modeling language can be graphical or textual.

- *Graphical* modeling languages use a diagram technique with named symbols that represent concepts and lines that connect the symbols and represent relationships and various other graphical notation to represent constraints.
- *Textual* modeling languages may use standardized keywords accompanied by parameters or natural language terms and phrases to make computer-interpretable expressions.

A modeling language can also to executable and non executable

- **Non Executable** modeling language is useful for representations (pictoral/textual) of system requirements, structures and behaviors, which can be useful for communication, design, and problem solving but cannot be used programmatically.
- **Executable** modeling languages applied **with proper tool support**, however, are expected to automate system verification and validation, simulation and code generation from the same representations.

Domain Specific Modeling Language: Languages which are specific to a Domain.

Modeling Language: BPMN (Business Process Modeling), TOSCA (Service Modeling), YANG (Data Modeling), UML (Graphical Modeling Language), XSD, SMI (for MIB)
Data Representation/Serialization Language: XML, JSON, YAML, ProtoBuf

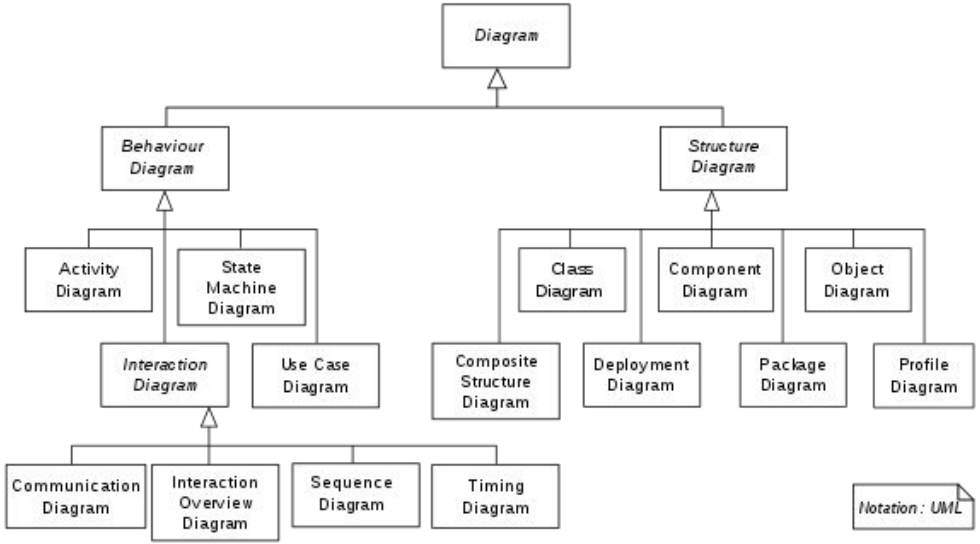
UML - Graphical Modeling Language

https://www.tutorialspoint.com/uml/uml_overview.htm
https://www.youtube.com/watch?v=WnMQ8HlmeXc

The **Unified Modeling Language** (**UML**) is a general-purpose, developmental, pictorial modeling language in the field of software engineering that is intended to provide a **standard** way to **visualize the design of a system.**[1]

UML diagrams represent two different views of a system model:[26]

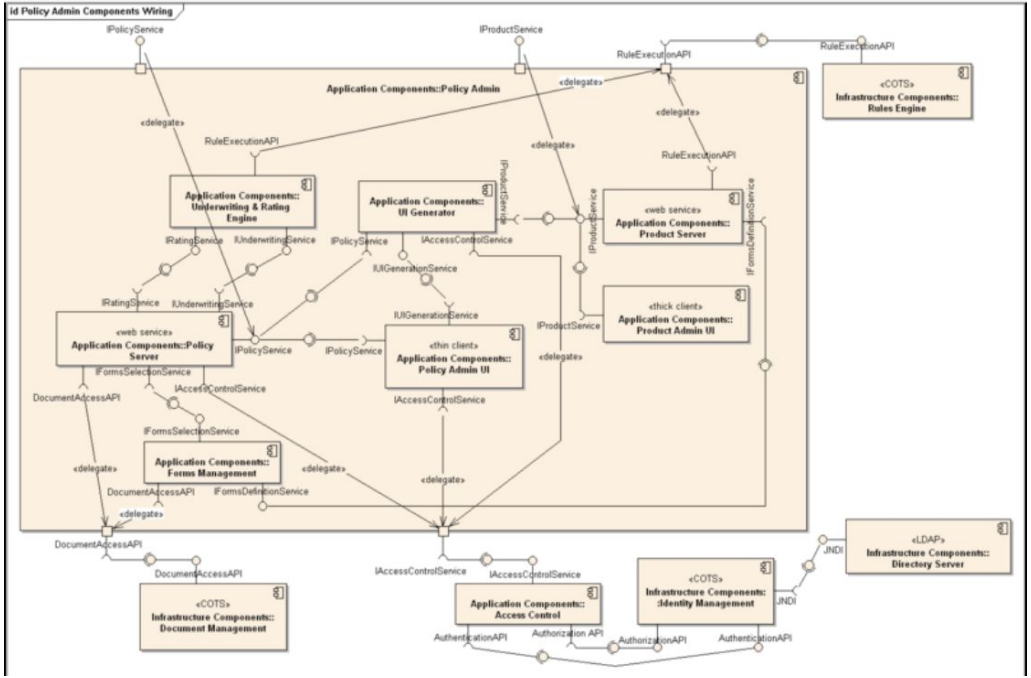
- Static (or ***structural***) view: emphasizes the static structure of the system using objects, attributes, operations and relationships. It includes class diagrams and composite structure diagrams.
- Dynamic (or ***behavioral***) view: emphasizes the dynamic behavior of the system by showing collaborations among objects and changes to the internal states of objects. This view includes sequence diagrams, activity diagrams and state machine diagrams.



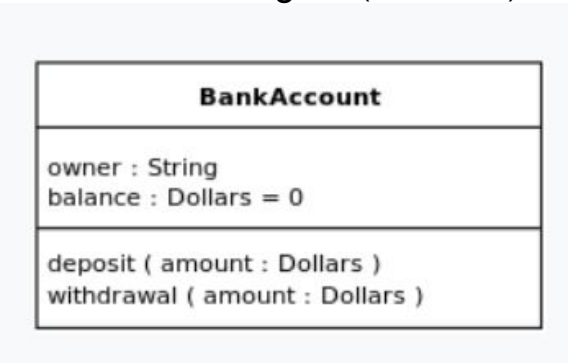
- UML is different from the other common programming languages such as C++, Java, COBOL, etc. UML is a pictorial language used to make software blueprints.
- UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design.
- Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc.

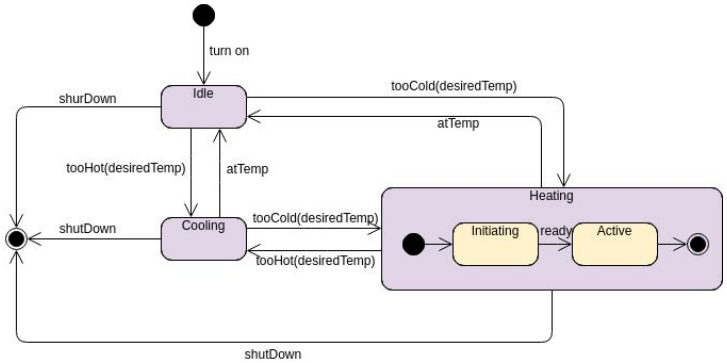
UML examples

Component Diagram (Structural)
Note: Even ONAP Architecture diagrams are UML



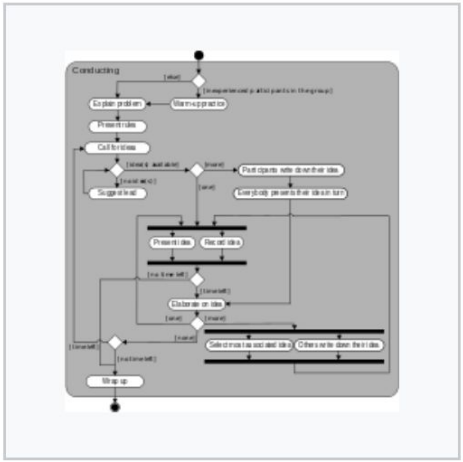
Class Diagram(Structural)



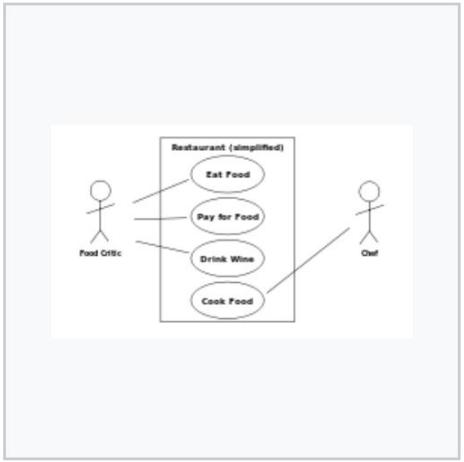


State Machine Diagram (Behavioural)

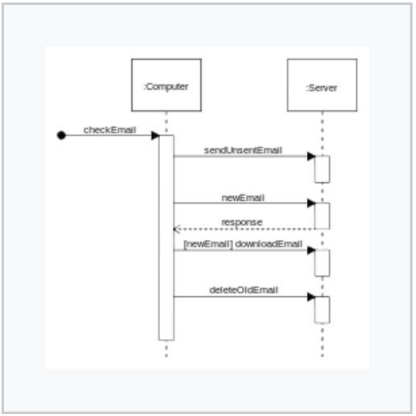
UML examples



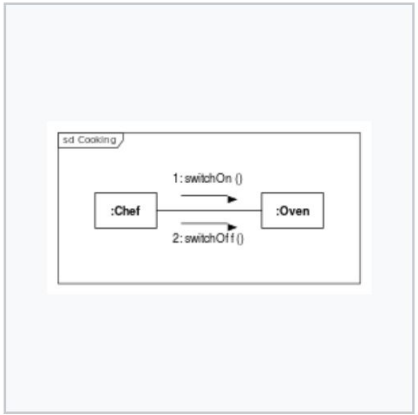
Activity diagram



Use case diagram



Sequence diagram



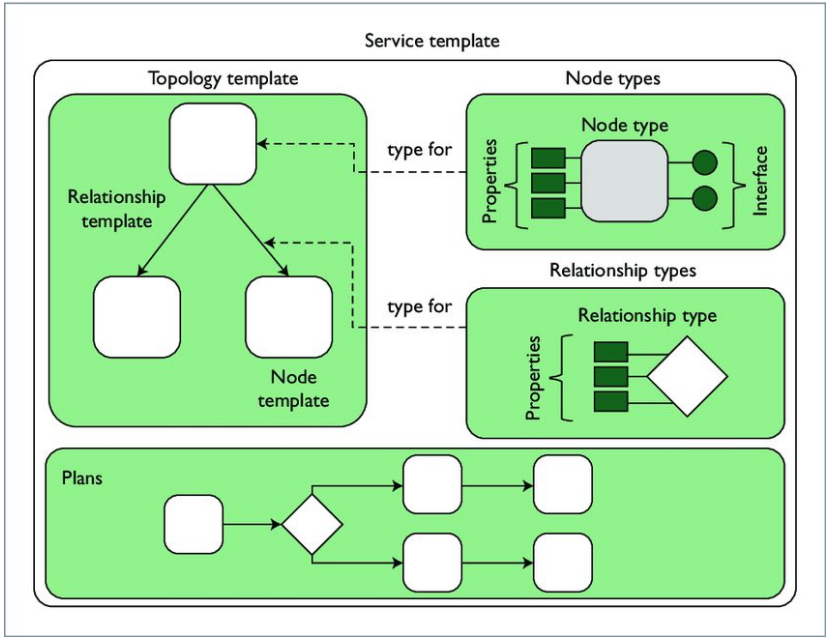
Communication diagram

Tools to try out UML: https://geekflare.com/about-uml-diagram-and-tools/
https://moqups.com/templates/diagrams-flowcharts/uml-diagrams/?gspk=Y2hhbmRhbmt1bWFy&gsxid=jedm2W62eNoV&partnerstack_group=Partner

TOSCA: Textual Modeling Language

Topology and Orchestration Specification for Cloud Applications, is an OASIS standard language to describe a topology of cloud based web services, their components, relationships, and the processes that manage them. The TOSCA standard includes specifications of a file archive format called CSAR.
It can model service/application in a standardized manner, improve automation, enable portability, and overcome interoperability issues.

```
topology_template:
  description: Wordpress deployment template
  inputs:
    os_distribution:
      type: string
      constraints:
        - valid_values: [ debian, ubuntu, knoppix ]
      description: The host Operating System (OS) architecture.
      default: ubuntu
  node_templates:
    wordpress:
      type: tosca.nodes.Wordpress
      requirements:
        - host: apache
        - database:
            node: mysql
            capability: tosca.capabilities.Endpoint.Database
        - php:
            node: php
            capability: tosca.capabilities.Root
    php:
      type: tosca.nodes.PHP
      requirements:
        - host: computeWww
    computeDb:
      type: tosca.nodes.Compute
      capabilities:
    computeWww:
      type: tosca.nodes.Compute
      capabilities:
        os:
          properties:
            type: linux
            architecture: x86_64
            distribution: { get_input: os_distribution }
        host:
          properties:
            num_cpus: 1
            disk_size: 1 GB
            mem_size: 1024 MB
            cpu_frequency: 1 GHz
    apache:
      type: tosca.nodes.Apache
      properties:
      requirements:
        - host: computeWww
    mysql:
      type: tosca.nodes.Mysql
      requirements:
        - host: computeDb
```



Part II
Network Management

Why: Network Management

Networks becomes larger, complex, supporting more applications and more users. Challenge which it brings in is:

- More things can go wrong, disabling the network or a portion of the network or degrading performance to an unacceptable level.

Know when to upgrade

- Is your bandwidth usage too high?
- Where is your traffic going?
- Do you need to get a faster line, or more providers?
- Is the equipment too old?

Keep an audit trace of changes

- Record all changes
- Makes it easier to find cause of problems due to upgrades and configuration changes

Keep a history of your network operations

- Using a ticket system let you keep a history of events.
- Allows you to defend yourself and verify what happened

Accounting

- Track usage of resources
- Bill customers according to usage

Know when you have problems

- Stay ahead of your users! Makes you look good.
- Monitoring software can generate tickets and automatically notify staff of issues.

Trends

- All of this information can be used to view trends across your network.
- This is part of baselining, capacity planning and attack detection.

Network Management Requirements

There are literally dozens of management "items." The following is a brief list of the information that can be reported from most networking devices:

- Track statistics for the network
- Keep track of status of machinery
- Enable/disable ports
- Graphically display information (ports)
- Maintain security (users, login, blocking traffic from unknown device)
- Provide resilience
- Handle messaging/polling
- Implement software upgrades
- Handle traffic/topology control
- Configure IP address
- Set to defaults
- Monitor other devices
- Send messages (to monitor)
- View faults

FCAPS

TABLE 1-1 ITU-T FCAPS Model

Management Functional Areas (MFAs)	Management Function Set Groups
Fault	Alarm Surveillance; Fault Localization & Correlation; Testing; Trouble Administration; Network Recovery
Configuration	Network Planning & Engineering, Installation; Service Planning & Negotiation; Discovery; Provisioning; Status & Control
Accounting	Usage Measurement, Collection, Aggregation, Mediation; Tariffing & Pricing
Performance	Performance Monitoring & Control; Performance Analysis & Trending; Quality Assurance
Security	Access Control & Policy; Customer Profiling; Attack Detection, Prevention, Containment and Recovery; Security Administration

Network Management System

Network management system is a **collection of tools, protocols etc.** for **network monitoring** and **control**

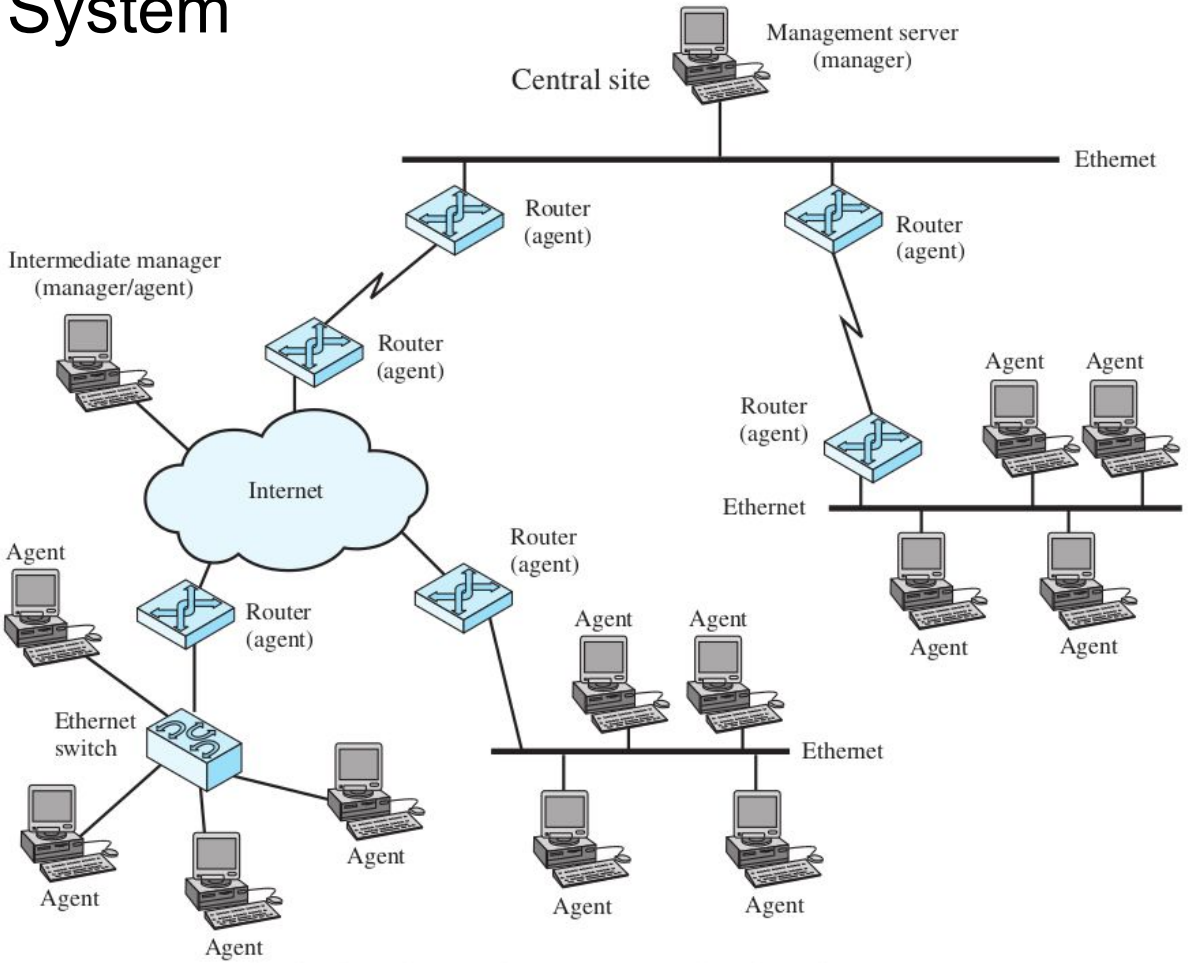


Figure 20.2 Example Distributed Network Management Configuration

Why: Network Management Automation/Programmability

Up to 95 percent of network changes today are done manually.

Manual changes lead to configuration errors and inconsistencies in the network.

Expanding network changes at scale can be problematic.

Network downtime and nonremote troubleshooting time are detrimental.

Personal Experience

At some point in time, more than 15 years ago, I had to support a network management system (NMS) product managing Layer 3 virtual private networks (L3VPNs) from a configuration, monitoring, and accounting point of view. Being a Cisco Certified Internetwork Expert (CCIE), I enjoyed playing with the router's CLI to configure an MPLS core of Provider Edge (PE) and Provider (P) routers, Virtual Routing and Forwarding (VRF) on the PE routers, with different routing options between customer edges (CEs) and PE routers, such as a default gateway, Routing Information Protocol (RIP), Open Shortest Path First (OSPF), or Border Gateway Protocol (BGP). Those configurations, mixed with route targets and route distinguishers, were fun to play with in the lab. I enjoyed demoing this solution to customers in pre-sale projects. Then, as a logical next step, the operators wanted to deploy those services in production networks. I recall one of the operator's constraints: "Now, we want to go from the L3VPN customer order to production in 20 minutes!" That 20-minute goal sounded like a stretch at the time, considering all the tasks involved—from the service order to the service validation. Next to the configuration of new L3VPN services, the product maintained a topology of the networks (which implies a network discovery), a mapping of L3VPN to specific customers, IP address mapping, VRF naming, location, customer flow record monitoring, and so on. The automation at that time used a template-based mechanism composed of multiple variables populated on the fly, and then a Telnet-based set of scripts to push the configuration to the routers and screen-scrape "show" commands. That project not only involved a greenfield environment, where all L3VPN services were new, but also a brownfield environment, where existing L3VPN services had to be discovered. —Benoit Claise

A fundamental shift started in the industry some years ago: A transition from network operators managing the network to operations engineers automating the network

Why: Network Management Automation/Programmability

Following industry **trends** driving Network Automation/Programmability

Reduced Deployment Time Expected/ Faster TTM

Hardware Commoditization and Disaggregation: Expects more automation

DevOps

SDN

NFV

Elastic Cloud: Pay As You Grow

Data Model–Driven Management

Intent-Based Networking

CLI Scripting

CLI scripting is the **primary approach** to making **automated configuration** changes to the network, at least prior to NETCONF. CLI Scripting is **NOT** suitable **for monitoring**.

Unfortunately, CLI scripting has various limitations and the most important one is the **lack of transaction management**. Configuring a device maybe a complex task, involving multiple actions i.e. configuration changes, that need to take place. Usually these multiple actions can not be done partially, as this would leave the device in an undefined state. In case one action fails, we need to rollback, i.e. undo all previous actions. This requires extensive programming when transaction management is not supported and this is the case with CLI.

Lack of structured error management is the second big issue of CLI. Usually whenever a new device software is released, new CLI commands are added but unfortunately in some cases existing commands are modified or deleted. This means that every other software that was using CLI as an application programming interface (API) will fail, and most probably without gracious handling of the errors.

This **ever changing structure and syntax** of CLI commands is what makes CLI scripts fragile and costly to maintain. These are all side-effects of the basic fact that CLIs are designed to be used by humans and not an API for programmatic access.

The CLI is **not standardized.** While the networking device configuration CLI is similar, it is not consistent from a syntax and semantic point of view across vendors or across a specific vendor's set of operating systems.

There are **dependency issues** when configuring devices via the CLI. In some cases, a CLI command for configuring an interface must be entered before configuring a VLAN. If those steps aren't executed in the proper order, the configuration fails—or worse, it's only partially completed.

Conclusion: CLI is not suited for automation. It is not machine-friendly and lacks a well-defined format. **Need an API based approach, which is standardized, machine friendly and can solve above issue**

What about SNMP?

SNMP works "reasonably well for device monitoring"

RFC 3535: Overview of the 2002 IAB Network Management Workshop – 2003
https://tools.ietf.org/html/rfc3535

- Typical config: SNMPv2 read-only community strings
- Typical usage: interface statistics queries and traps
- Empirical Observation: SNMP is not used for configuration
  - Lack of Writeable MIBs
  - Security Concerns
  - Difficult to Replay/Rollback
  - Special Applications

Reasons for this include the lack of a defined discovery process that makes it hard to find the correct MIB modules, limitations inherent in the use of the UDP protocol, and the lack of useful standard security and commit/transaction mechanisms.

SNMP has always done a good job in terms of monitoring devices. However, it fails at device configuration

Operator Requirements (RFC3535)

1. Ease of use is a key requirement...
2. …clear distinction between configuration data, data that describes operational state and statistics.
3. …be able to fetch separately configuration data, operational state data, and statistics from devices, and to be able to compare these between devices.
4. It is necessary to enable operators to concentrate on the configuration of the network as a whole rather than individual devices.
5. Support for configuration transactions across a number of devices…
6. …it should be possible to generate the operations necessary to get from A to B with minimal state changes and effects on network and systems.
7. …mechanism to dump and restore configurations …pulling and pushing configurations from/to devices…
8. …consistency checks of configurations over time…
9. …common database schema for network configuration….
10. …text processing tools such as diff, and version management tools such as RCS or CVS, can be used to process configurations…
11. …role-based access control model and the principle of least privilege…
12. …consistency checks of access control lists across devices.
13. Devices should be able to hold multiple configurations.
14. …support both data-oriented and task-oriented access control.

The Meaning of Transactions

The four properties that define a transaction: ACID

Atomicity

- Transactions are indivisible, all-or-nothing

Consistency

- Transactions are all-at-once
- There is no internal order inside a transaction, it is a **set** of changes, **not a sequence**
- Implies that { create A, create B } and { create B, create A } are identical
- Implies that a system behaving differently with respect to the sequence is not transactional

Independence

- Parallel transactions do not interfere with each other
- Transactions appear to happen always-in-sequence

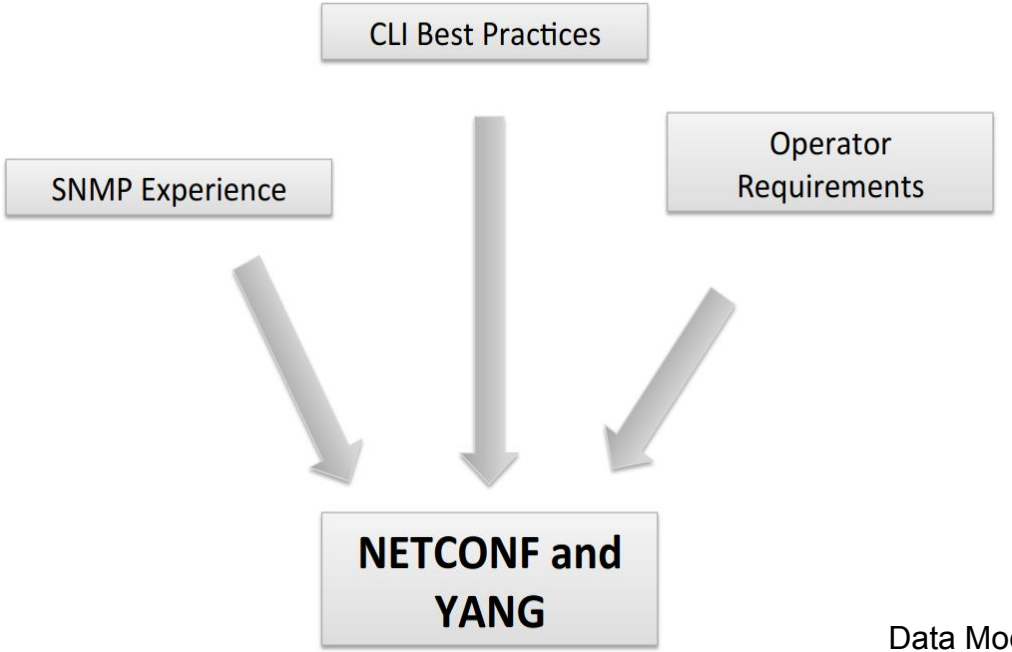
Durability

- Committed data always-sticks, i.e. remains in the system even in the case of a fail-over, power failure, restart, etc

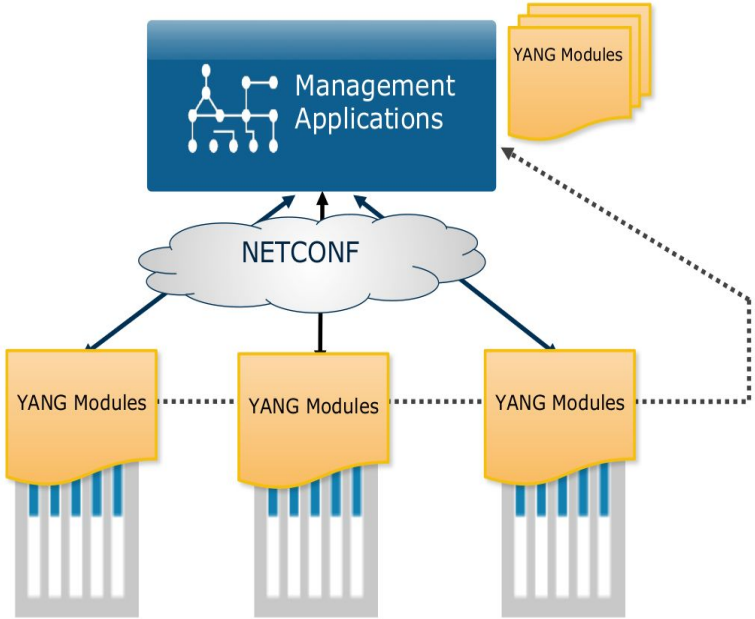
Netconf/YANG - Data Model Driven Management

RFC 3535
Implementation

Best Practices Coming Together



NETCONF and YANG in Context



Data Model - YANG
Network Protocol - Netconf
YANG model-driven management protocols: NETCONF, **RESTCONF, and gNMI**

Key Definitions

Data model-driven management builds on the idea of specifying in models the semantics, the syntax, the structure, and constraints of management objects. From there, scripts use **APIs rendered from those models via tooling**. The advantage is that, as long as the models are updated in a backward-compatible way, the previous set of APIs is still valid.
An important advantage of data model–driven management is the **separation** of the **models** from the **protocols** and **encodings**, which means that it is easier to add protocols and encodings. Data model driven management was initially built on NETCONF and XML, but other protocols/encodings have since seen the light: RESTCONF with JavaScript Object Notation (JSON), gRPC Network Management Interface (gNMI) with protobuf, and so on

What is a Data-Model? What is a Network Management Protocol?

What is a Data Model?

A data model is simply a well understood and agreed upon method to describe " something" . As an example, consider this simple " data model" for a person.

- *Person*
  - Gender – male, female, other
  - Height – Feet/Inches or Meters
  - Weight – Pounds or Kilos
  - Hair Color – Brown, Blond, Black, Red, other
  - Eye Color – Brown, Blue, Green, Hazel, other

- Data-Model
  - A data-model explicitly and precisely determines the structure, syntax and semantics of the data…
  - …that is *externally* visible
  - Consistent and complete
- Protocol
  - Remote primitives to view and manipulate the data
  - Encoding of the data as defined by the data-model

Netconf/YANG & SNMP Stack

	SNMP	NETCONF	SOAP	REST
Standard	IETF	IETF	W3C	-
Resources	OIDs	Paths		URLs
Data models	Defined in MIBs	YANG Core Models		
Data Modeling Language	SMI	YANG	(WSDL, not data)	Undefined, (WSDL), WADL, text…
Management Operations	SNMP	NETCONF	In the XML Schema, not standardized	HTTP operations
Encoding	BER	XML	XML	XML, JSON, …
Transport Stack	UDP	SSH TCP	SSL HTTP TCP	SSL HTTP TCP

"RESTConf"

Netconf/YANG vs SNMP

tail-f

What makes NETCONF/YANG different?

This is where the difference is:
In the supported use cases!

Use Case	SNMP	NETCONF
Get collection of status fields	Yes	Yes. Bulk xfer up to 10x faster. Really.
Set collection of configuration fields	Yes, up to 64kB	Yes
Set configuration fields in transaction	No	Yes
Transactions across multiple network elements	No	Yes
Invoke administrative actions	Well…	Yes
Send event notifications	Yes	Yes, connected
Backup and restore configuration	Usually not	Yes
Secure protocol	v3 is fair	Yes
Test configuration before final commit	No	Yes

Part III
Network Data Modeling using YANG

YANG

YANG is a data modeling language used to model network configuration & state data. Modeling languages such as SMI (SNMP), UML, XML Schema (XSD), and others already existed. However, none of these languages were specifically targeted to the needs of configuration management. They lacked critical capabilities like being easily read and understood by human implementers, and fell short in providing mechanisms to validate models of configuration data for semantics and syntax.

YANG serves as API contract between client-server. Also its an executable modelling language and tools can be build to achieve automation.

YANG – A Data Modeling Language for Networking

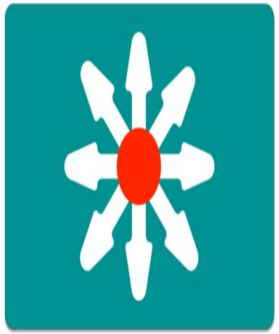
- Human readable, and easy to learn representation
- Hierarchical configuration data models
- Reusable types and groupings (structured types)
- Extensibility through augmentation mechanisms
- Supports definition of operations (RPCs)
- Formal constraints for configuration validation
- Data modularity through modules and sub-modules
- Well defined versioning rules

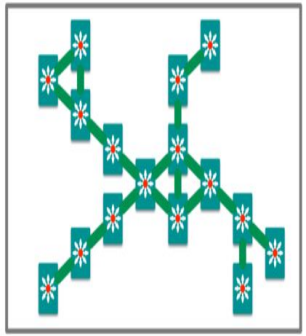
Why you should care:

YANG is a full, formal contract language with rich syntax and semantics to build applications on

```
list interface {
        key "name";
        unique "type location";

        leaf name {
          type string;
          reference
            "RFC 2863: The Interfaces Group MIB - ifName";
        }

        leaf description {
          type string;

...

  container statistics {
          config false;
          leaf discontinuity-time {
            type yang:date-and-time;
          }

          leaf in-octets {
            type yang:counter64;
            reference
              "RFC 2863: The Interfaces Group MIB - ifHCInOctets";
          }
```

What might a YANG Data Model describe?





Device Data Models

- Interface
- VLAN
- Device ACL
- Tunnel
- OSPF
- etc

Service Data Models

- L3 MPLS VPN
- MP-BGP
- VRF
- Network ACL
- System Management
- Network Faults
- etc

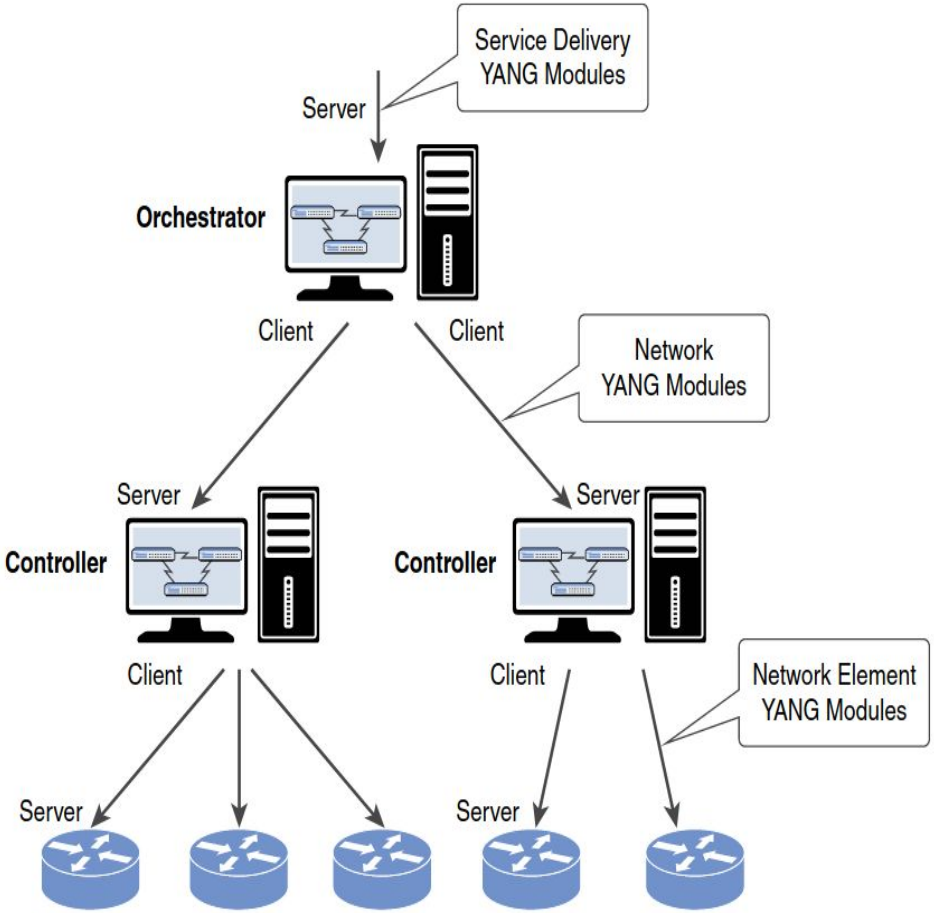
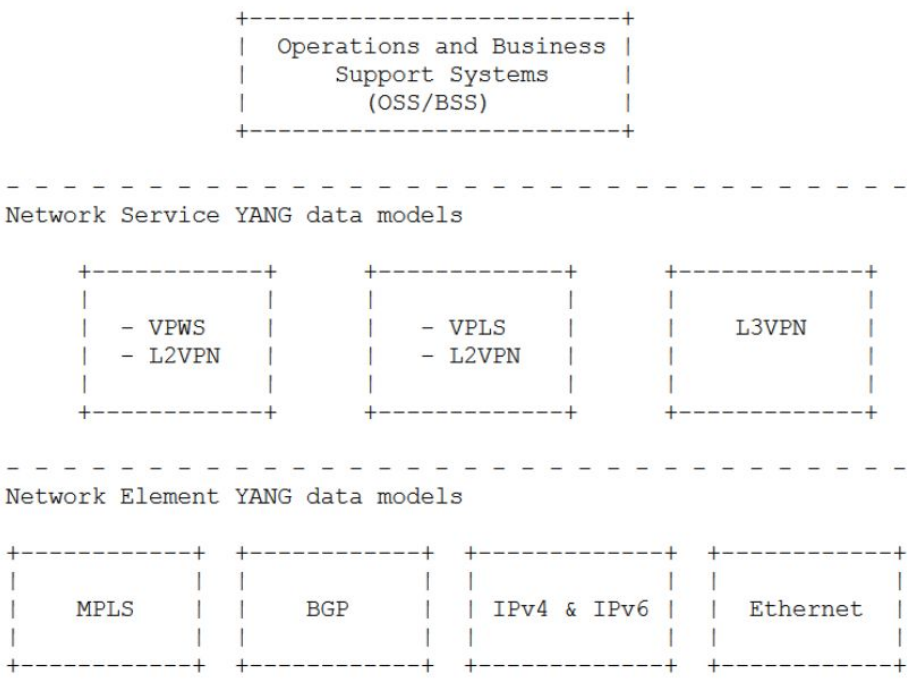


FIGURE 2-3 The Management Architecture

SDOs Alignment and Trajectory



MEF

MEF

OPEN DAYLIGHT

IETF

DMTF

Open ROADM

IETF

broadband forum

CableLabs

OPENCONFIG

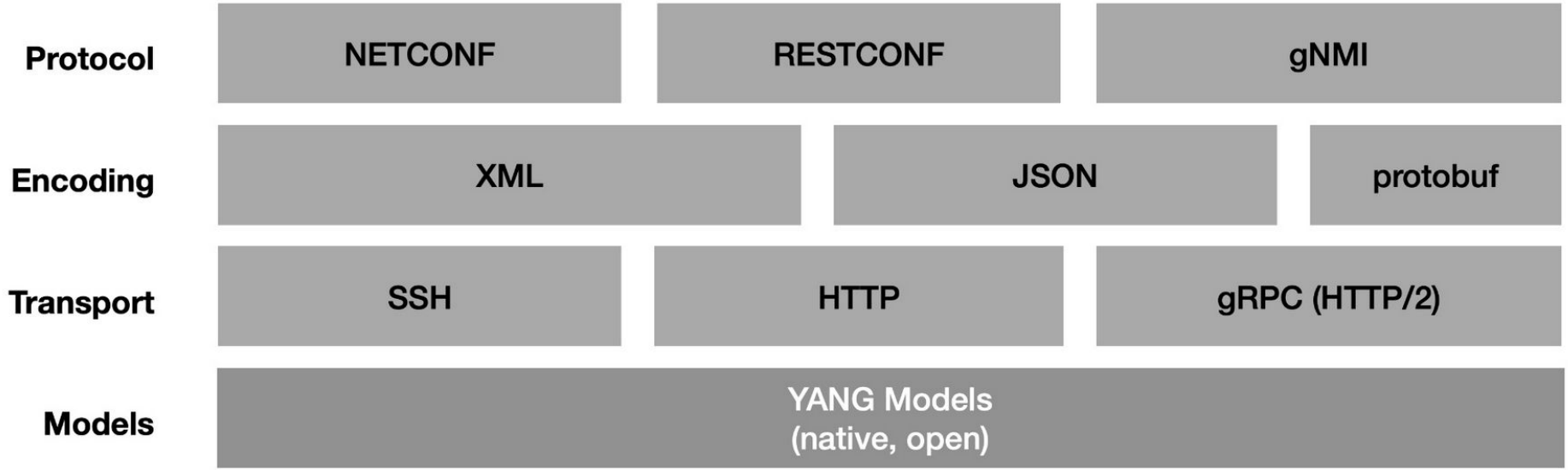
IEEE Advancing Technology for Humanity

ITU

Network Management Protocols using YANG

TABLE 2-1 Quick NETCONF/RESTCONF/gNMI Protocol Comparison

	NETCONF	RESTCONF	gNMI
Message and Payload Encoding	XML.	JSON or XML.	gNMI notifications with JSON (two variants) or protobuf payload.
Operation Semantics	NETCONF specific; network-wide transactions.	RESTCONF specific, based on HTTP verbs. Single-target, single-shot transactions.	gNMI specific. Single-target, single-shot, sequenced transactions.
RPC Mechanism	NETCONF specific; XML based.	REST style.	gRPC.
Transport Stack	SSH/TCP/IP.	HTTP/TLS/TCP/IP.	HTTP2/TLS/TCP/IP.



YANG Language
Hands-On using “pyang”
Toolset around YANG

Part IV
Network Automation using YANG (YANG in Network Controllers)

SDN Journey

SDN started with the concept of the **separation of the control and data plane entities**, **with a focus on OpenFlow** as the open standard to configure the data plane Forwarding Information Base (FIB), or Media Access Control (MAC) table.

Through the years, the notion of SDN has been evolved to other protocols with focus on **programmability/automation**.

PCEP, the Path Computation Element communication Protocol (RFC 5440), is a communication protocol between a Path Computation Client (PCC) and a Path Computation Element (PCE), or between two PCEs, which includes path computation requests and path computation replies as well as notifications of specific states related to the use of a PCE in the context of Multiprotocol Label Switching (MPLS) and Generalized MPLS (GMPLS) traffic engineering

BGP-LS, which stands for BGP Link State Distribution (RFC 7752), is a mechanism by which link-state and traffic engineering information is collected from networks and shared with external components using the BGP routing protocol.

NETCONF/YANG provides a simple, standardized way to enable a programmable interface to any device or service

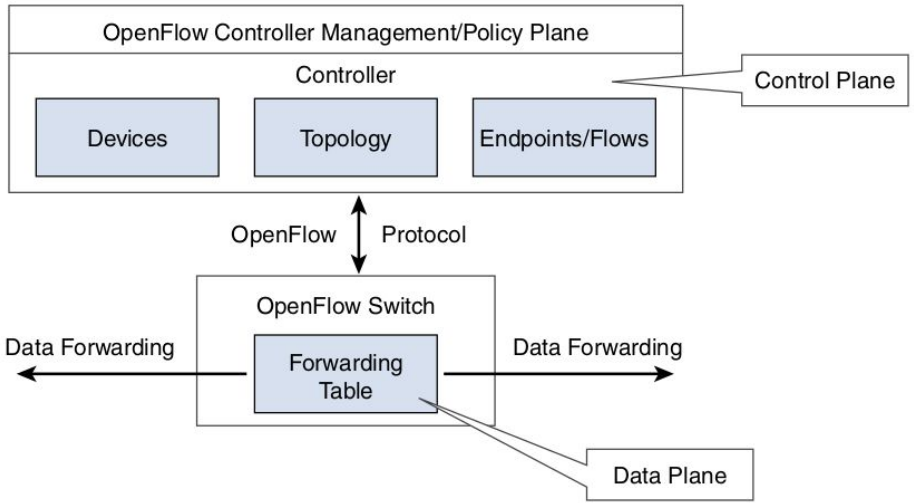


FIGURE 1-2 OpenFlow

YANG and Automation

There are 2 parts in implementing the application:

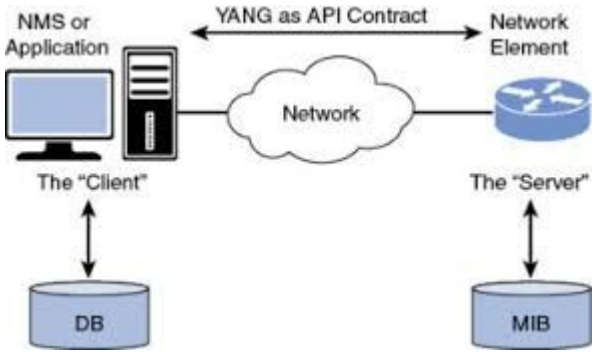
1. Syntax/semantics processing of the request/response being exchanged.
2. Business logic to compute the request.

YANG can automate the first part, thereby application only needs to focus on business logic. It can automate following:

1. Automated API generation & API LCM
2. Automated validation (based on syntax/semantics/constraints) processing of the request/response
3. Automated code generation corresponding to YANG
4. Automated marshalling/unmarshalling (serialization/deserialization) of data.
5. Automatically creating datastore schema (tree based structure).

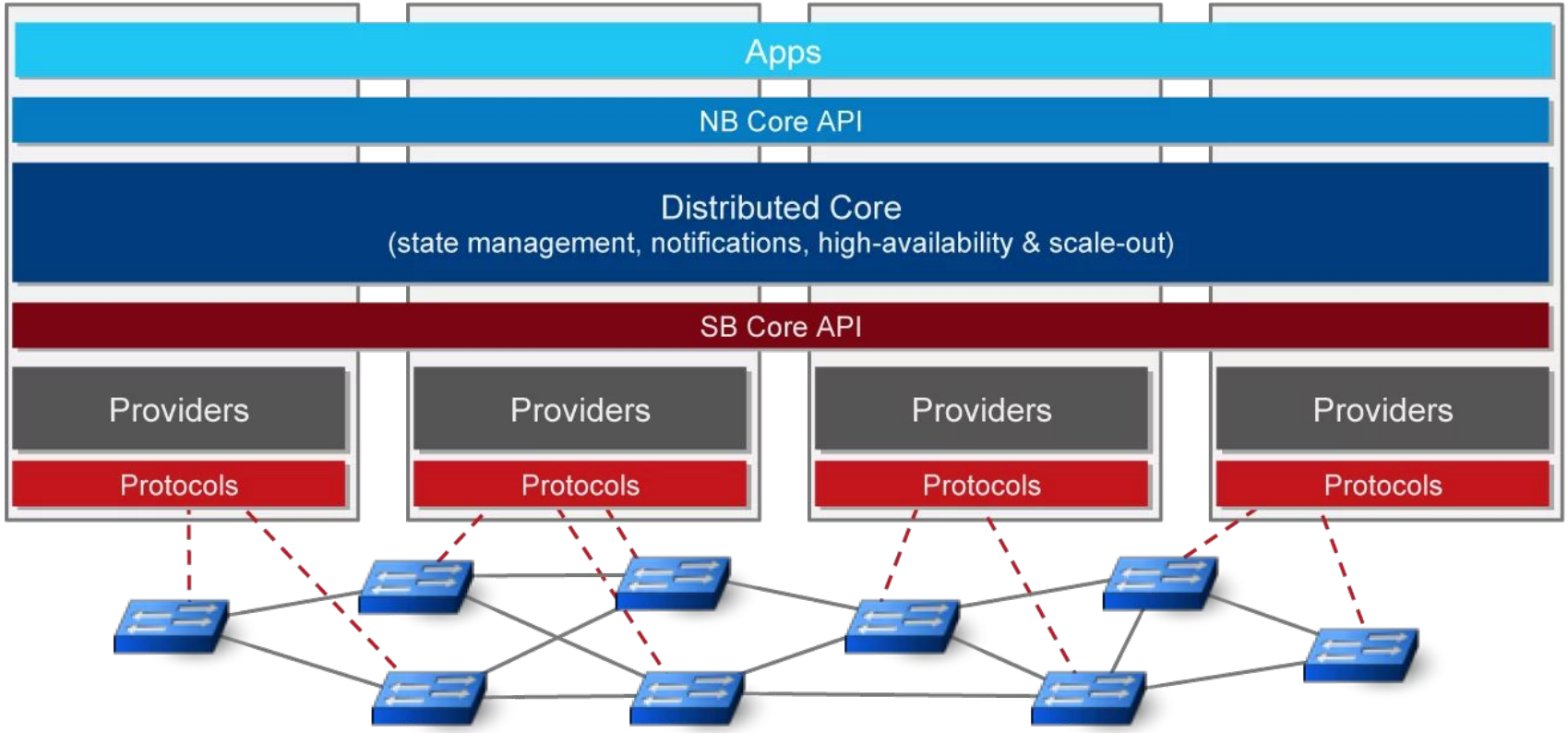
Additionally it be extended to achieve further automation like “model mapping” etc.

Also YANG serves as an API contract between client and server.



ONOS

ONOS Architecture



https://www.youtube.com/watch?v=60wl-05l7kk

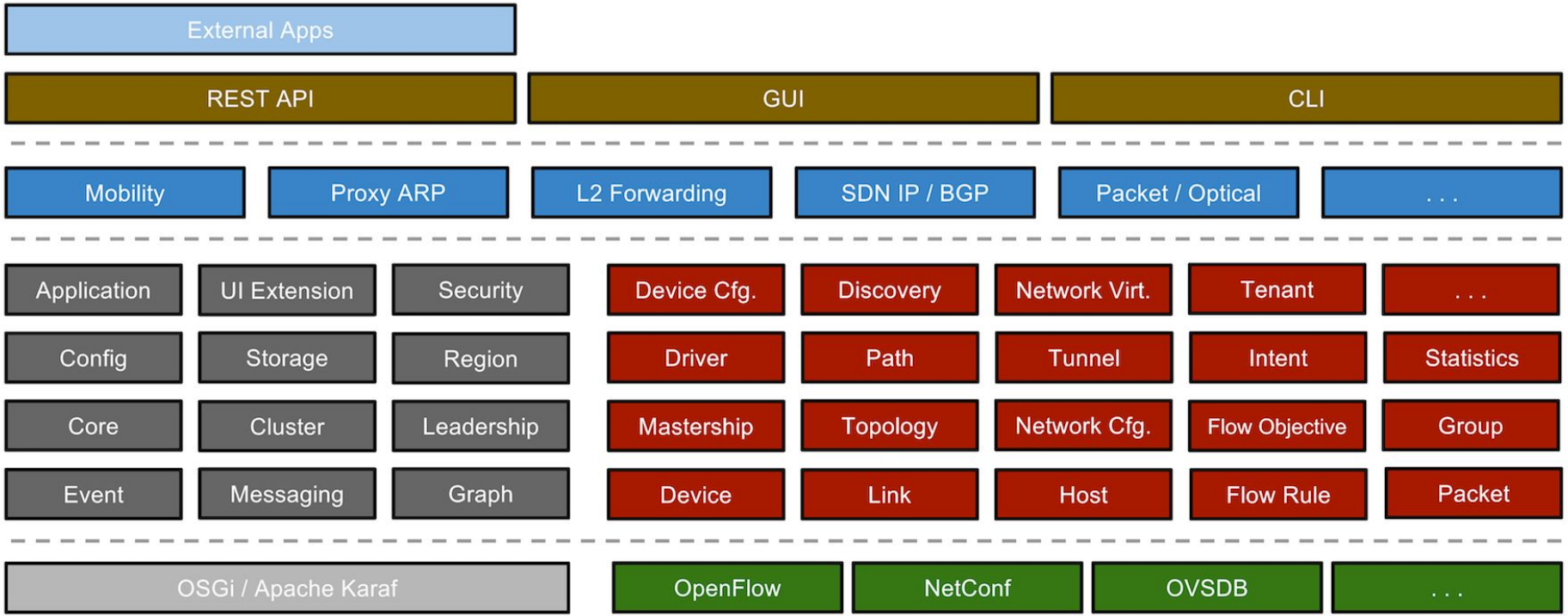
ONOS Architecture

Design Principles

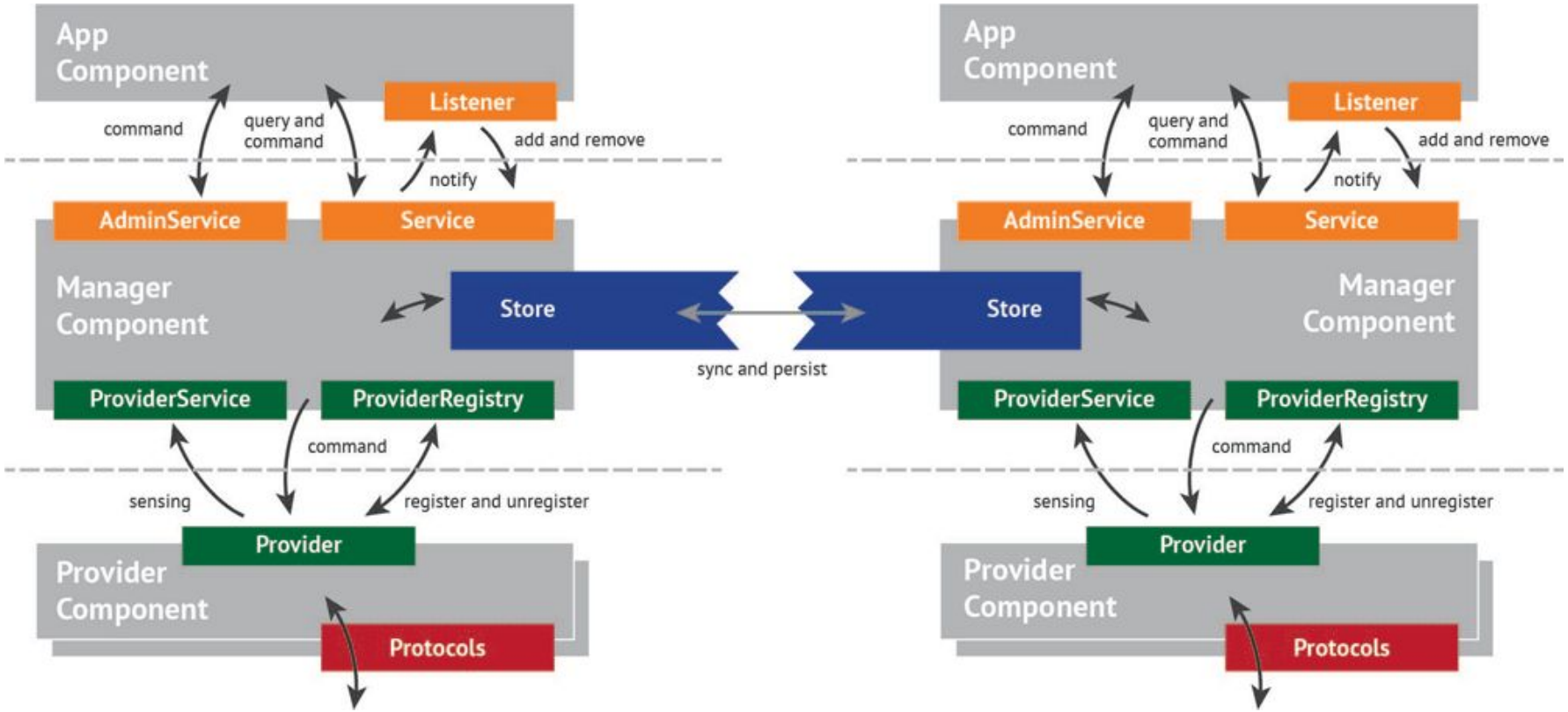
- High-Availability, Scalability and Performance
- Strong Abstractions
- Separation of Concerns and Modularity
- Symmetric Distributed Architecture with logically centralized
- System Tiers (Tiered Architecture)

- **Distributed Core** The pivotal layer is the ONOS core, which while allowing for physical separation of data and control functions, is responsible for presenting a logically centralized view of network state and logically centralized access to network control functions.
- **Southbound API & Providers** The core is separated from the other tiers via two logically distinct interface boundaries. The south-facing interface, is a high-level API through which the core interacts with the network environment. Except, rather than interacting with the environment directly, ONOS core relies on protocol-specific adapters to conduct these interactions using the protocol of their choice, whether it is OpenFlow, Netconf, OVSDB, TL1 or even other available means, such as CLIs. The southbound API then acts as a bridge for core to send edicts to and receive sensory data from the various protocol providers. The protocol-independent nature of this API guarantees that no protocol-specifics leak into the ONOS core and to the applications. Furthermore, its high-level nature avoids merely shrink-wrapping a specific protocol library and at the same time, it allows ample maneuvering room for the protocol providers to translate protocol-specific behaviours into protocol-agnostic ones.
- **Northbound API** On the north-facing side, ONOS core exposes a set of abstractions to applications and additional network services via its northbound API. These abstractions provide a range of access to network information starting from the usual low-level topology abstractions, such as devices, links and hosts, to higher-level abstractions, such as the network topology graph. Similarly, they provide a range of abstractions for affecting the network state via flow objective programming and intent-based programming.

ONOS Core Subsystem



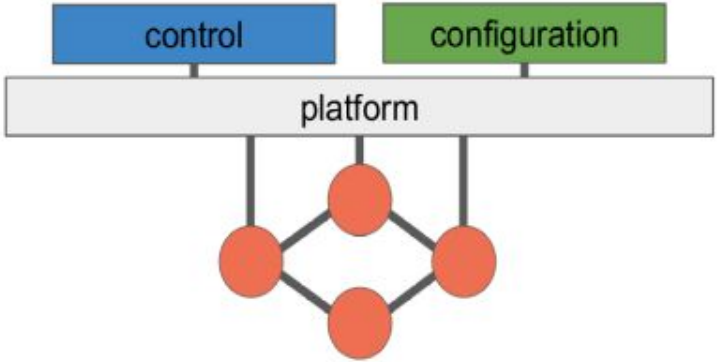
ONOS Core Subsystem Design Pattern



YANG in ONOS: Why?

Control and Configuration

- Operators need a resilient and scalable platform capable of *both* control and configuration



Configuration still critical

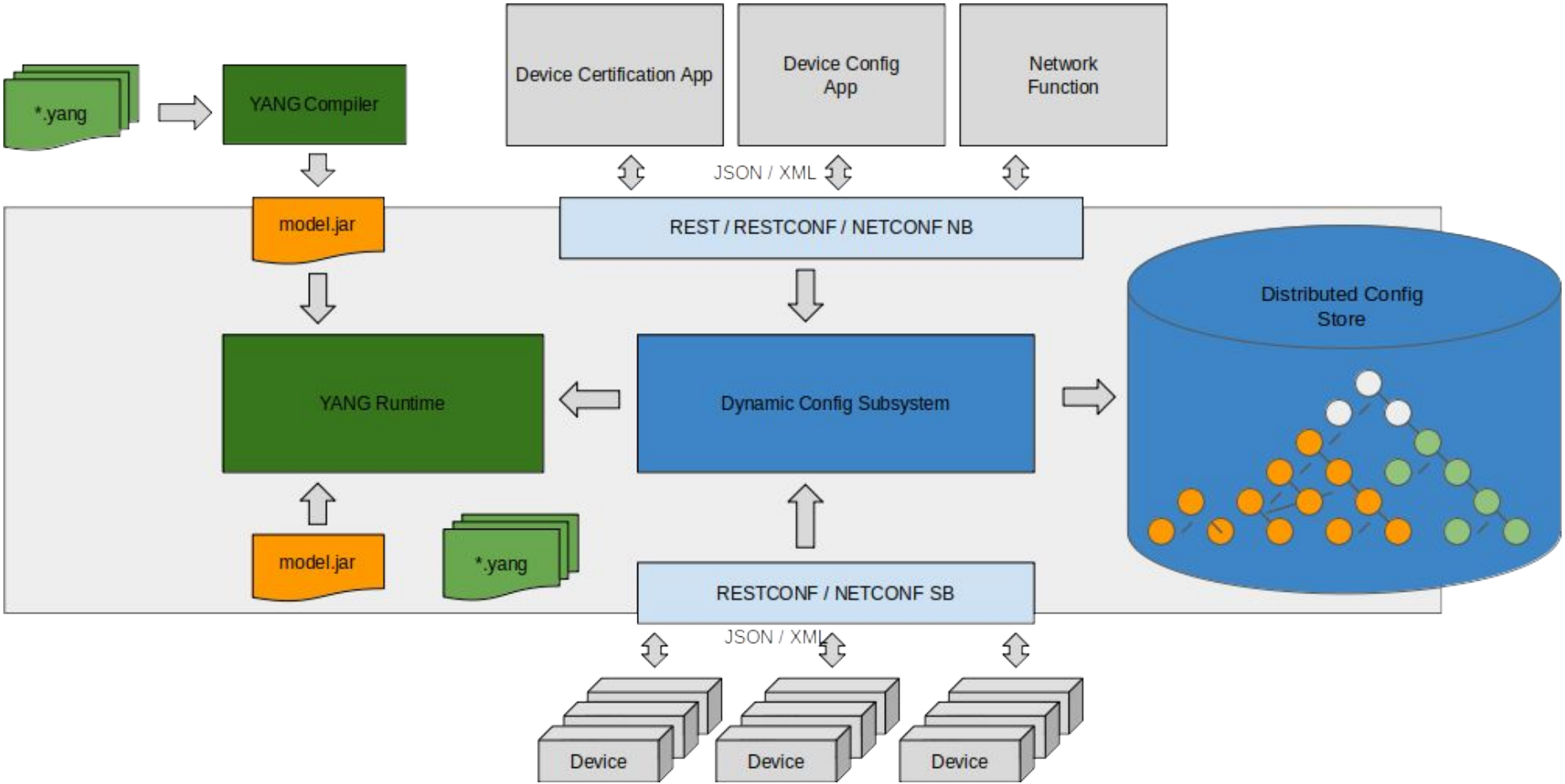
- Dynamic (re)configuration continues to be critical
  networks still need to be managed and configured
  if nothing else, configured to be controlled
- Configuration even more important in brown-fields
  devices may expose only limited control capabilities

Service Configuration

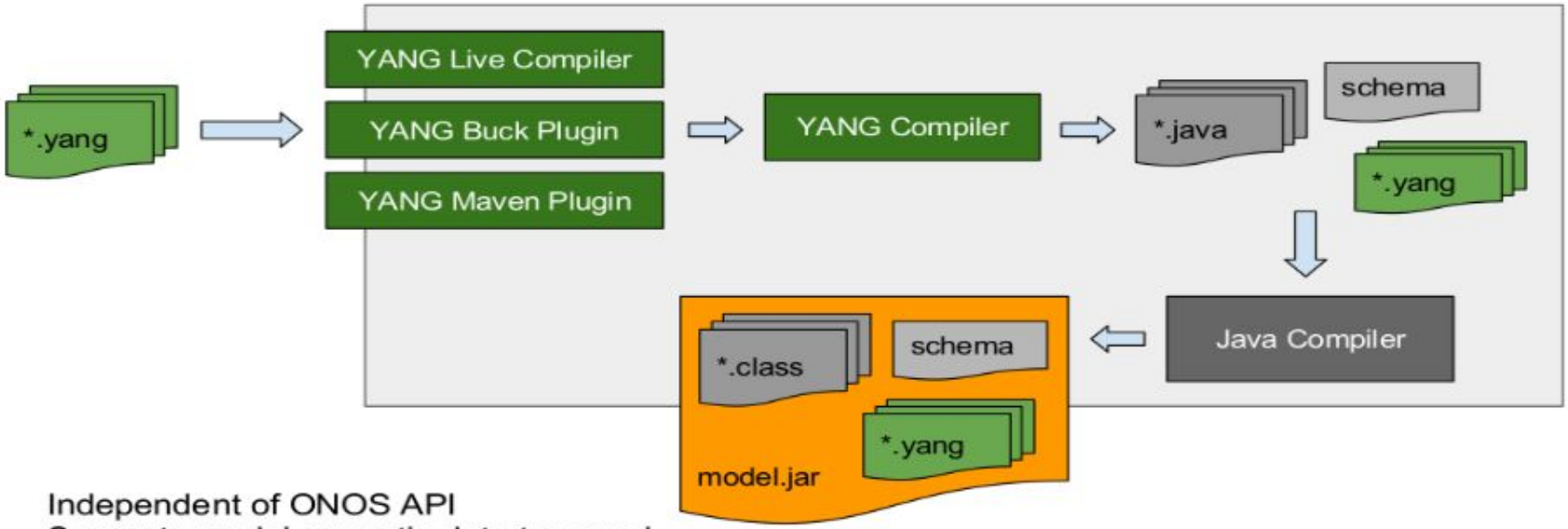
- Operators want to create & sell customized services
  do this with agility and minimal human intervention
  create automated ways to instantiate such network services
- Services comprise both configuration & control

- Goal is to enable a network operator to seamlessly configure and provision a service on the network comprising many devices from many vendors **with minimal or no human intervention**
- Provides network operators with agility to deploy new services with **reduced OPEX**
- Offers vendors opportunity to support many services on their devices.

YANG Subsystem in ONOS: Overall Architecture



YANG Tool Chain

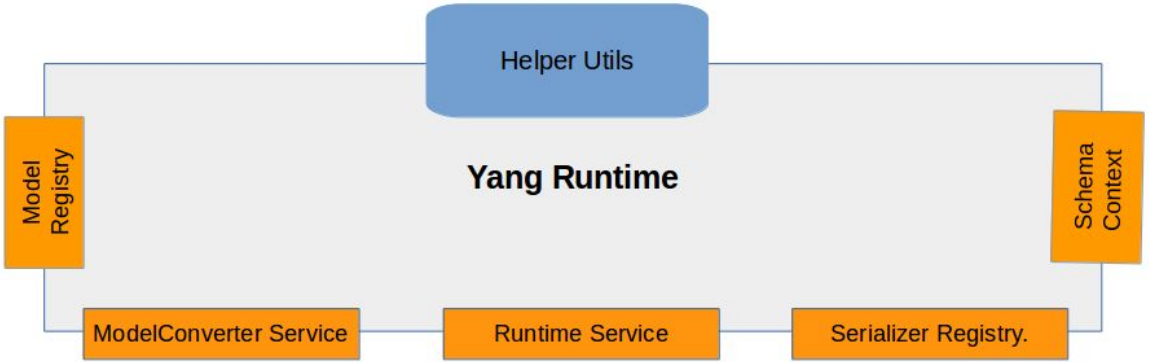


- ✓ Independent of ONOS API
- ✓ Supports model-agnostic data traversal
- ✓ Generates schema for run-time validation and encoding/decoding
- ✓ Generates model-specific rich data types

https://wiki.onosproject.org/display/ONOS/YANG+Compiler

YANG Compiler example

YANG Subsystem in ONOS: YANG Runtime



Extensible to support any new data format.

Independent of NB/SB protocol.

Provides schema context information can be used to build value added applications over the top.

1) **YANG Schema Registry (YSR)** - Maintains the registered schema registry and provides related services.

2) **YANG Model Registry (YMR)** - Maintains the registered models registry.

3) **YANG Object Builder (YOB)** - utilities to obtain YANG modeled objects from DataNode.

4) **YANG Tree builder (YTB)** – utilities to get DataNode from YANG modeled objects.

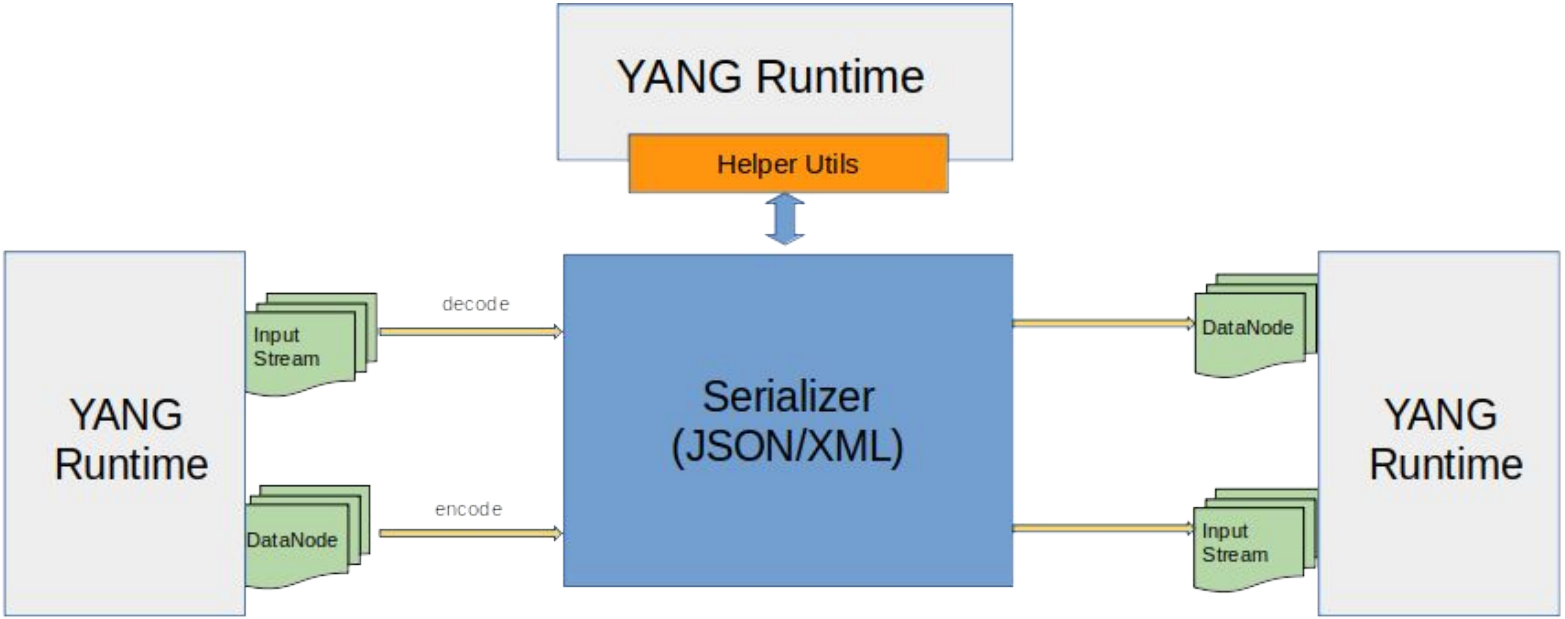
5) **YANG Schema Context Provider (YSCP)** – schema context provider.

6) **Helper Utils** – utilities to support serializers and applications.

7) **YANG Runtime Manager (YRM)** – YANG runtime manager.

https://wiki.onosproject.org/display/ONOS/YANG+Runtime

YANG Subsystem in ONOS: YANG Serializers



YANG Serializers: Facilities to convert Input stream to DataNode and vice versa.

Default serializers available with YANG App itself.

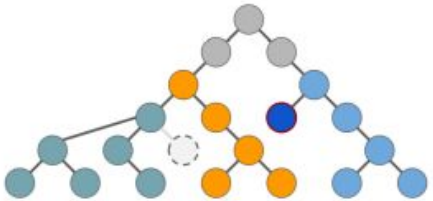
1) JSON Serializer: Convert to/from JSON to DataNode

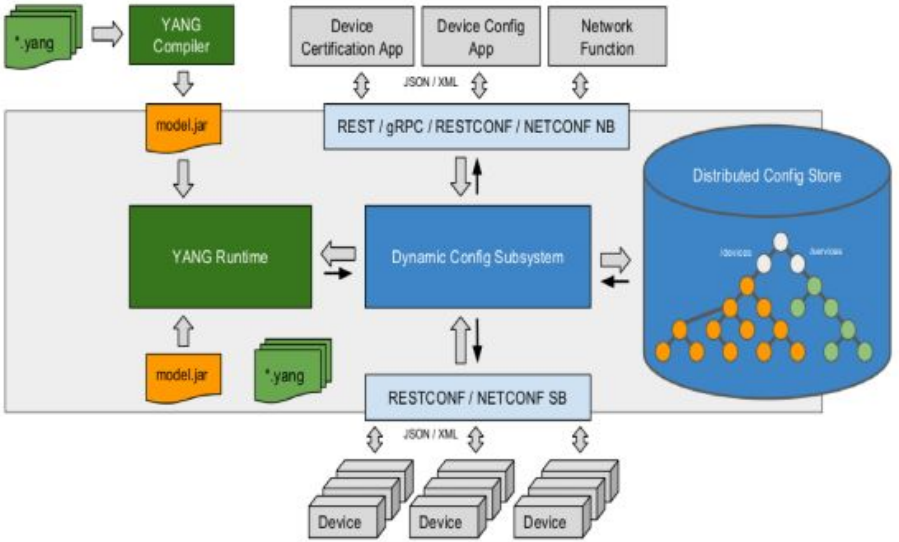
2) XML Serializer: Convert to/from XML to DataNode

YANG Subsystem in ONOS: Dynamic Config App/Store

Information Store as a Tree

- Adjustable to model augmentations & deviations
  - new nodes can be introduced, some can be removed
- Can be logically extended to aggregate information
  - many devices, many services under a unified tree structure



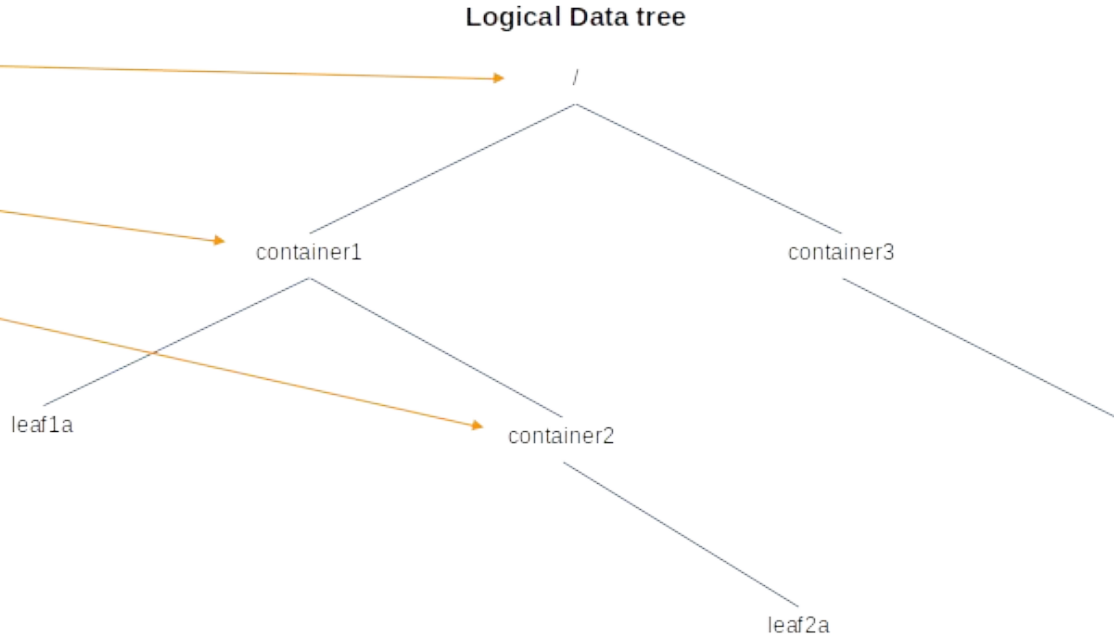


- Stores data in abstract DataNode fomat
- Currently rely on post filtering mechanism for data change notifications.

YANG Subsystem in ONOS: YANG to Data Tree

Sample YANG

```
module sample1 {
    yang-version 1;
    namespace urn:sample1;
    prefix s1;
    revision 2016-11-11;
    container container1 {
        leaf leaf1a{
           type string;
        }
        container container2{
            leaf  leaf2a{
                type string;
            }
        }
    }

    container container3 {
        leaf leaf3a{
           type string;
        }
    }


    rpc set-node-limit {
        input {
            leaf node-limit {
                type int16;
            }
        }
    }

    notification node-limit-reached {
        leaf node-limit {
            type int16;
        }
    }
}
```



YANG Subsystem in ONOS: Model/App type

Base Models which will be used by different YANG components. It includes following:

1) **DataNode:** Provides a basis for holding information in a tree whose structure corresponds to a set of YANG models (base & augmentations).
- Is immutable
- Provides a mechanism for mutation (Builder pattern, etc.)
- Understands the schema that was used to construct it.
- Store will be based on DataNode.

2) **ResourceId:** Resource identifier, path till the resource.
- Is immutable
- Provides a mechanism for mutation (Builder pattern, etc.)
- Used to operate on a given resource.

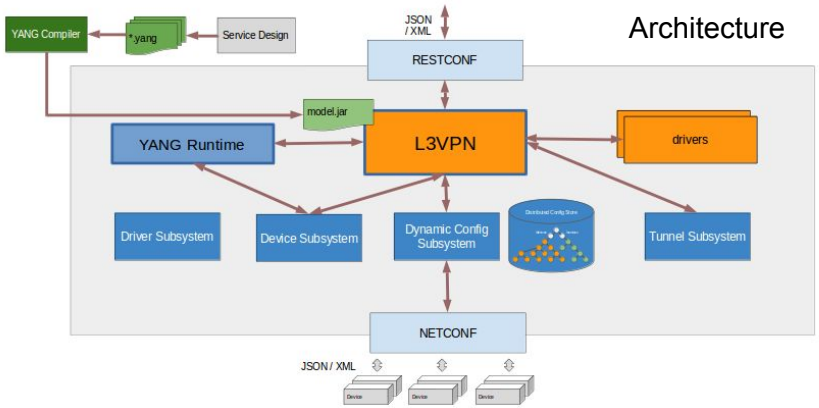
3) **ModelObject:** Provides a common basis for all POJOs which are generated from a YANG model. It is mutable.
- The binding between *ModelObject* POJO and *DataNode* is performed by the YANG Runtime.

Types of Applications:

Schema Aware Applications: Applications who are aware about the YANG data structure and operates on that
Example: L3VPN, PCE, SFC

Schema Un-aware Applications: Applications who are not aware about the YANG data structure and wanted to operate in schema agnostic way.
Example: RESTCONF, NETCONF, Store

L3VPN Example

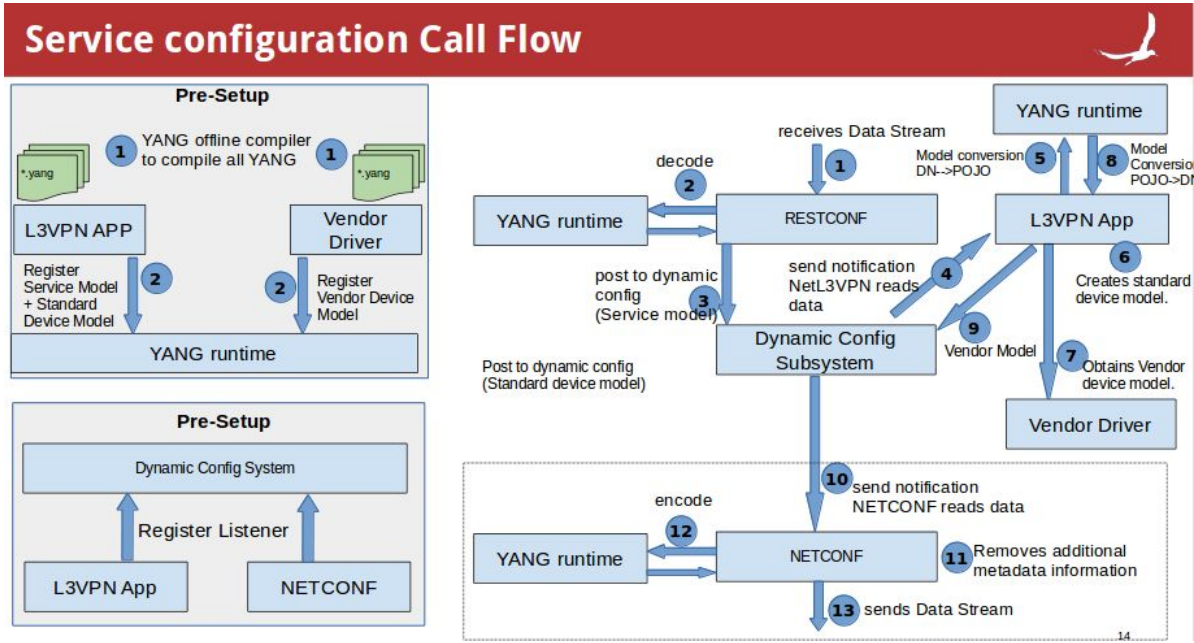


This project in ONOS implements the L3VPN creation requests from App. It provides a Rest Api for north app to use. When getting creation requests , the L3VPN component prepares the configuration informations and downloads the configuration informations into PE devices

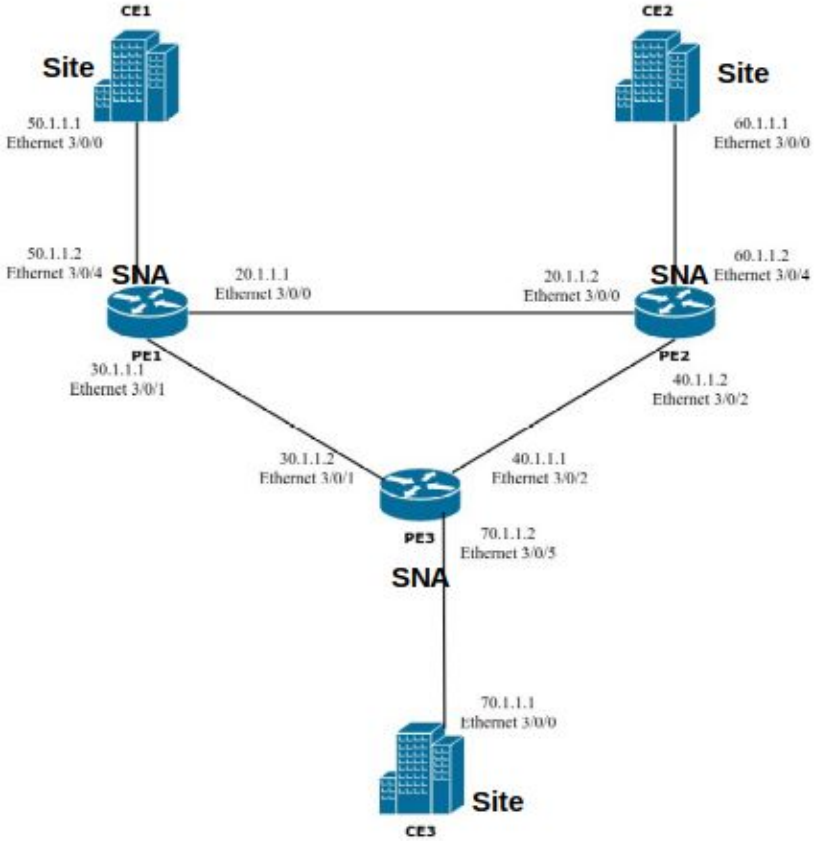
https://wiki.onosproject.org/display/ONOS/L3VPN

L3VPN Service YANG
https://github.com/opennetworkinglab/onos/tree/2.4.0/models/l3vpn/src/main/yang

Device YANG
https://github.com/opennetworkinglab/onos/tree/2.4.0/models/huawei/src/main/yang



L3VPN Example: L3SM YANG




```
module: ietf-l3vpn-svc
  +--rw l3vpn-svc
   +--rw vpn-services
   | +--rw vpn-service* [vpn-id]
   |    +--rw vpn-id                 svc-id
   |    +--rw customer-name?         string
   |    +--rw vpn-service-topology?  identityref
   |    +--rw cloud-accesses {cloud-access}?
```

```
   +--rw sites
     +--rw site* [site-id]
      +--rw site-id                  svc-id
      +--rw requested-site-start?    yang:date-and-time
      +--rw requested-site-stop?     yang:date-and-time
      +--rw locations
      | +--rw location* [location-id]
      |    +--rw location-id    svc-id
      |    +--rw address?       string
      |    +--rw postal-code?   string
      |    +--rw state?         string
      |    +--rw city?          string
```

```
      +--rw site-network-accesses
        +--rw site-network-access* [site-network-access-id]
         +--rw site-network-access-id    svc-id
         +--rw site-network-access-type?  identityref
         +--rw (location-flavor)
         | +--:(location)
         | | +--rw location-reference?     leafref
         | +--:(device)
         |    +--rw device-reference?      leafref
```

L3VPN Example: L3SM YANG

```
| +--rw oam
|   +--rw bfd {bfd}?
|    +--rw enabled?    boolean
|    +--rw (holdtime)?
|      +--:(profile)
|      | +--rw profile-name?  string
|      +--:(fixed)
|       +--rw fixed-value?  uint32
+--rw security
| +--rw authentication
| +--rw encryption {encryption}?
|   +--rw enabled?       boolean
|   +--rw layer          enumeration
|   +--rw encryption-profile
|    +--rw (profile)?
|      +--:(provider-profile)
|      | +--rw profile-name?  string
|      +--:(customer-profile)
|       +--rw algorithm?    string
|       +--rw (key-type)?
|         +--:(psk)
|         |   ...
|         +--:(pki)
+--rw service
| +--rw svc-input-bandwidth?  uint32
| +--rw svc-output-bandwidth?  uint32
| +--rw svc-mtu?         uint16
| +--rw qos {qos}?
| | +--rw qos-classification-policy
| | | +--rw rule* [id]
| | |   +--rw id              uint16
| | |   +--rw (match-type)?
| | |   | +--:(match-flow)
| | |   | | +--rw match-flow
| | |   | |    ...
| | |   | +--:(match-application)
| | |   |   +--rw match-application?  identityref
| | |   +--rw target-class-id?   string
| | +--rw qos-profile
| |   +--rw (qos-profile)?
| |    +--:(standard)
| |    | +--rw profile?  string
| |    +--:(custom)
| |      +--rw classes {qos-custom}?
| |       +--rw class* [class-id]
| |          ...
```

```
  | +--rw carrierscarrier {carrierscarrier}?
  | | +--rw signalling-type?  enumeration
  | +--rw multicast {multicast}?
  |   +--rw multicast-site-type?    enumeration
  |   +--rw multicast-address-family
  |   | +--rw ipv4?  boolean {ipv4}?
  |   | +--rw ipv6?  boolean {ipv6}?
  |   +--rw protocol-type?       enumeration
  +--rw routing-protocols
  | +--rw routing-protocol* [type]
  |   +--rw type   identityref
  |   +--rw ospf {rtg-ospf}?
  |   | +--rw address-family*  address-family
  |   | +--rw area-address?   yang:dotted-quad
  |   | +--rw metric?      uint16
  |   | +--rw sham-links {rtg-ospf-sham-link}?
  |   |   +--rw sham-link* [target-site]
  |   |    +--rw target-site  svc-id
  |   |    +--rw metric?    uint16
  |   +--rw bgp {rtg-bgp}?
  |   | +--rw autonomous-system?  uint32
  |   | +--rw address-family*   address-family
  |   +--rw static
  |   | +--rw cascaded-lan-prefixes
  |   |   +--rw ipv4-lan-prefixes* [lan next-hop] {ipv4}?
  |   |   | +--rw lan     inet:ipv4-prefix
  |   |   | +--rw lan-tag?  string
  |   |   | +--rw next-hop  inet:ipv4-address
  |   |   +--rw ipv6-lan-prefixes* [lan next-hop] {ipv6}?
  |   |    +--rw lan     inet:ipv6-prefix
  |   |    +--rw lan-tag?  string
  |   |    +--rw next-hop  inet:ipv6-address
  |   +--rw rip {rtg-rip}?
  |   | +--rw address-family*  address-family
  |   +--rw vrrp {rtg-vrrp}?
  |    +--rw address-family*  address-family
  +--rw availability
  | +--rw access-priority?  uint32
  +--rw vpn-attachment
    +--rw (attachment-flavor)
     +--:(vpn-policy-id)
     | +--rw vpn-policy-id?  leafref
     +--:(vpn-id)
       +--rw vpn-id?     leafref
       +--rw site-role?    identityref
```

ODL

https://www.opendaylight.org/about/platform-overview
https://docs.opendaylight.org/en/latest/getting-started-guide/concepts_and_tools.html
https://slideplayer.com/slide/5876657/

THANKS

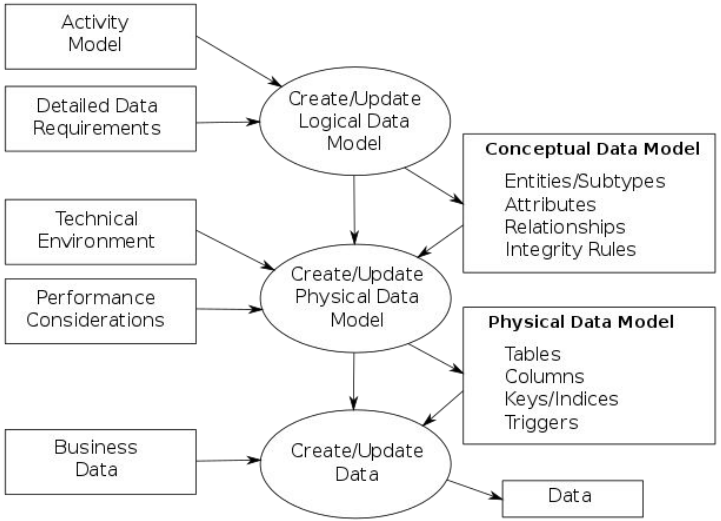
References

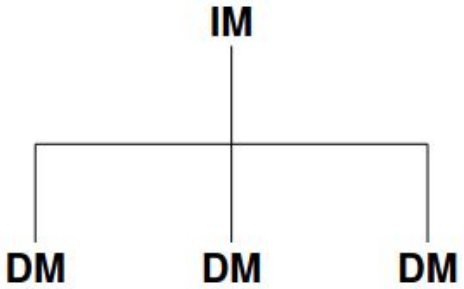
Tail-F Videos

https://www.youtube.com/watch?v=Vr4kB1_6fLQ
https://www.youtube.com/watch?v=m6spTjQyTEo
https://www.youtube.com/watch?v=xoPZO1N-x38
https://www.youtube.com/watch?v=33VBb6N4yOY

Other Video
https://www.youtube.com/watch?v=zy9QA-uU0u4

Information Model vs Data Model





conceptual/abstract model for designers and operators

concrete/detailed model for implementors

The relationship between an IM and DM is shown in the Figure above. Since conceptual models can be implemented in different ways, multiple DMs can be derived from a single IM.

Although IMs and DMs serve different purposes, it is not always possible to precisely define what kind of details should be expressed in an IM and which ones belong in a DM. There is a gray area where IMs and DMs overlap -- just like there are gray areas between the models produced during the analysis, high-level design and low-level design phases in object-oriented software engineering. In some cases, it is very difficult to determine whether an abstraction belongs to an IM or a DM.

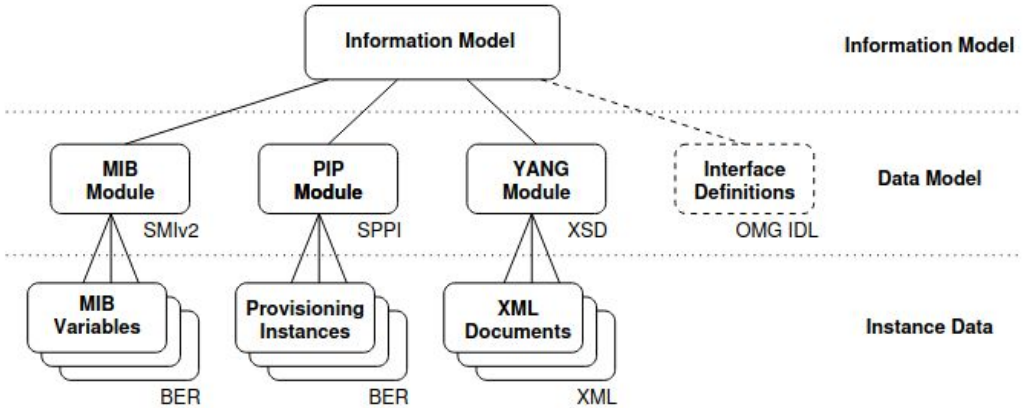
Information Model vs Data Model cont.. (RFC 3444)

Information Models

- Information Models are used to model managed objects at a conceptual level, independent of any specific protocols used to transport the data (protocol agnostic).
- The degree of specificity (or detail) of the abstractions defined in the information model depends on the modeling needs of its designers.
- In order to make the overall design as clear as possible, information models should hide protocol and implementation details.
- Information models focus on relationships between managed objects.
- Information models are often represented in Unified Modeling Language (class) diagrams, but there are also informal information models written in plain English language.

Data Models

- Data Models are defined at a lower level of abstraction and include many details (compared to information models).
- They are intended for implementors and include implementation and protocol-specific constructs.
- Data models are often represented in formal data definition languages that are specific to the management protocol being used.



What is SDN?

Software-Defined Networking (SDN) is a network architecture approach that enables the network to be intelligently and centrally controlled, or ‘programmed,’ using software applications. This helps operators manage the entire network consistently and holistically, regardless of the underlying network technology.

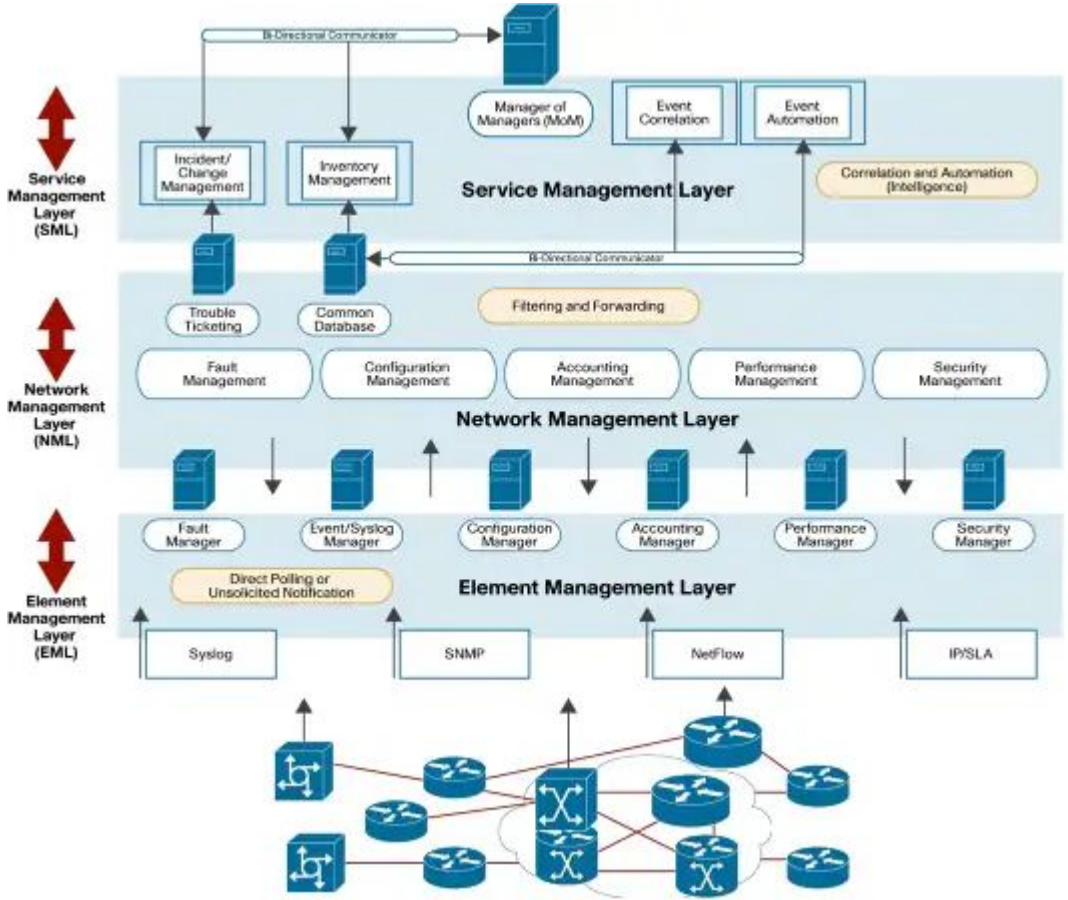
SDN. Is about separating control plane from data plane.

NV. Is about separating hardware from software using a network hypervisor**.**

NFV. Is about separating Network Functions (NAT, DPI, etc.) from hardware**.**

Hierarchical Network Management

https://www.geeksforgeeks.org/difference-between-ems-and-nms/

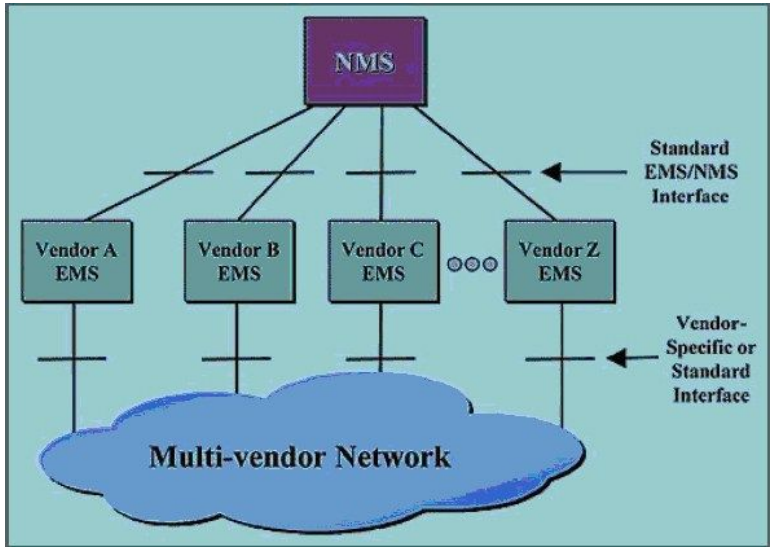


1. Element Management System (EMS) :

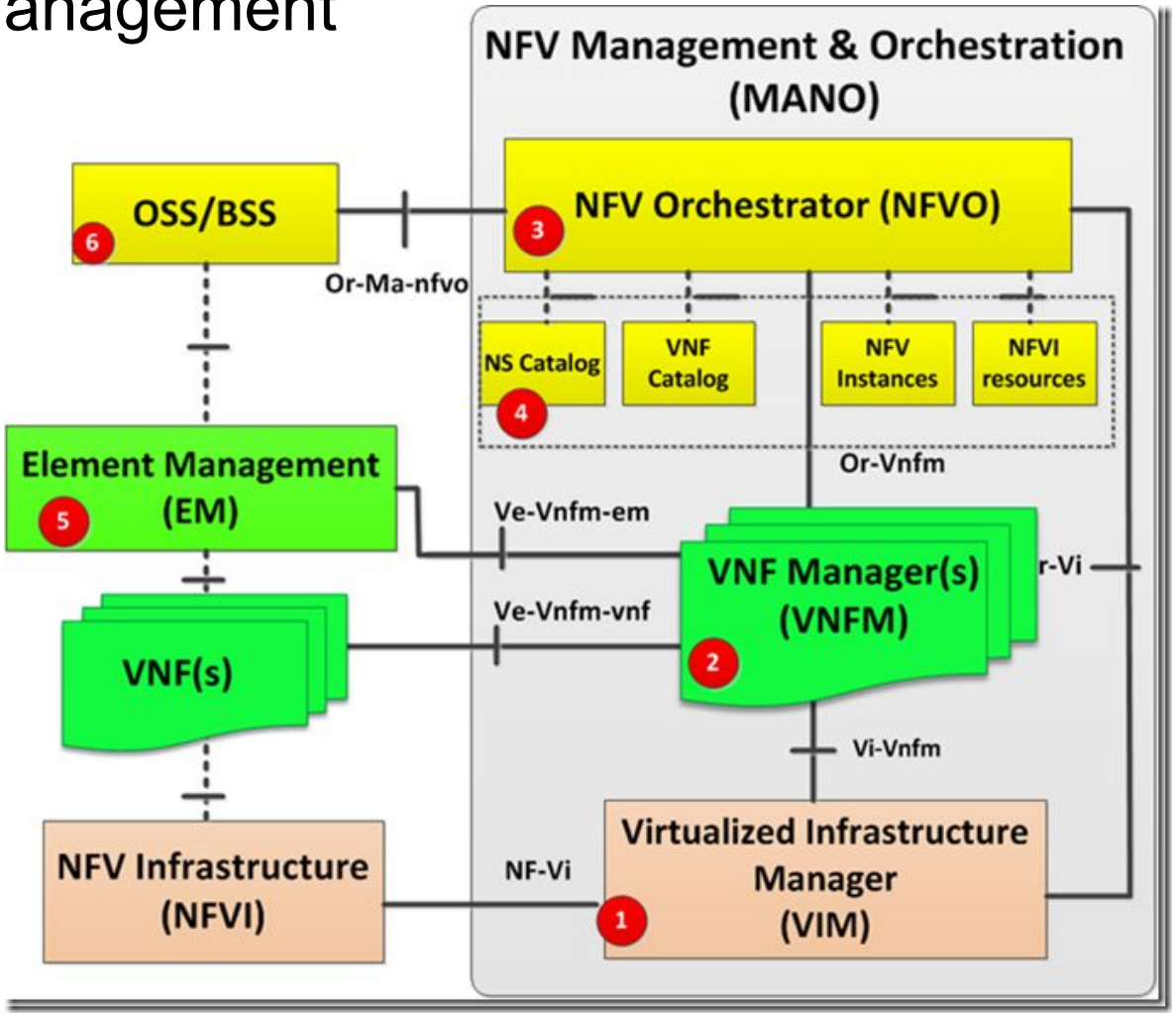
Element Management System (EMS), as name suggests, is a software tool that is generally used to manage elements. Here element represent node i.e. device such as computer, router, etc. EMS usually manages group of similar elements (same category or same vendor)

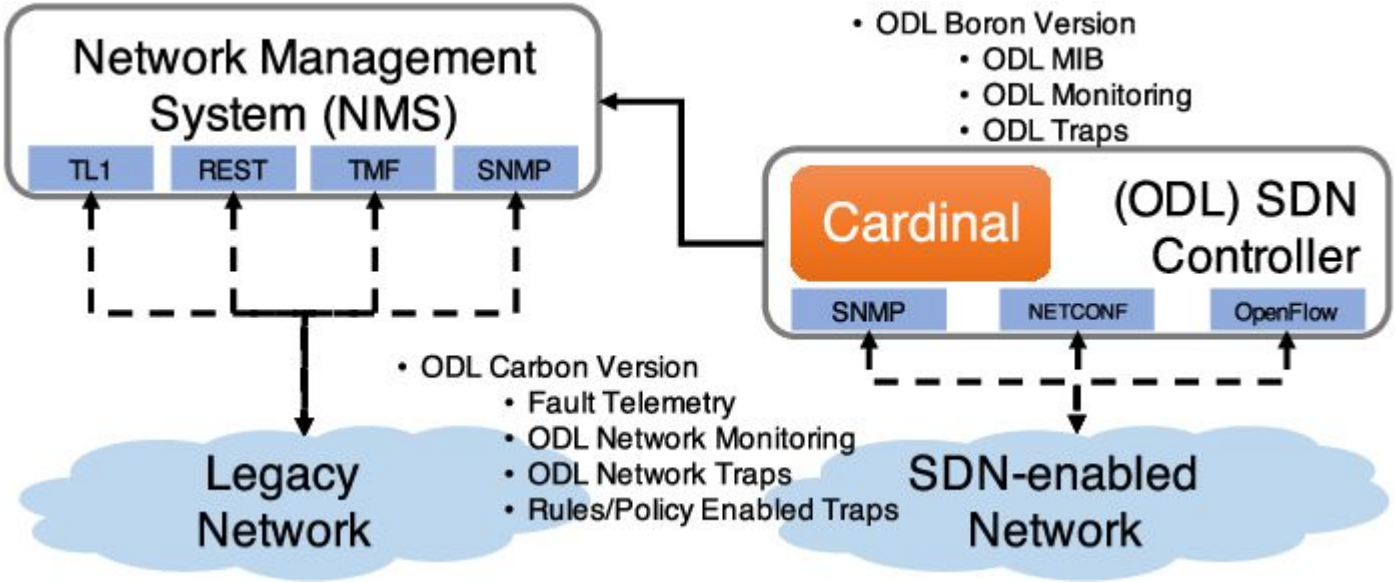
2. Network Management System (NMS) :

Network Management System (NMS), as name suggest, is a software tool that is generally used to manage and control network i.e. interconnected nodes/elements



MANO & Network Management




Network Management System (NMS)
TL1
REST
TMF
SNMP
• ODL Boron Version
• ODL MIB
• ODL Monitoring
• ODL Traps
Cardinal
(ODL) SDN Controller
SNMP
NETCONF
OpenFlow
• ODL Carbon Version
• Fault Telemetry
• ODL Network Monitoring
• ODL Network Traps
• Rules/Policy Enabled Traps
Legacy Network
SDN-enabled Network