SCTE CABLE-TEC EXPO 22
SEPTEMBER 19-22 • PHILADELPHIA, PA

2022 Fall Technical Forum
SCTE • CABLELABS • NCTA

Commercial Network Services Lifecycle Management

A Technical Paper prepared for SCTE by

Pattabi Ayyasami
Sr Principal Engineer
Comcast
1050 Enterprise Way #100, Sunnyvale, CA
+1 (408) 212-4383
pattabi_ayyasami@comcast.com

Tirumalesh Ramaiah Reddy
Director, Software Development & Engineering
Comcast
1800 Arch St, Philadelphia, PA
+1 (215) 286-5073
tirumalesh_ramaiahreddy@cable.comcast.com

Table of Contents

List of Figures

List of Tables

1. Introduction

As Software Defined Networking (SDN) and Network Functions Virtualization (NFV) have become modern approaches to quickly deliver and offer network services such as SD-WAN, Security and other value-added services using service chains. Declarative & model driven orchestration technologies like TOSCA has become the fabric to develop Network Automation platform to manage lifecycle of these network service applications and build an ecosystem for multiple vendors to participate & integrate with their VNF solutions.

Comcast architecture leverages several information models and orchestration tools including TOSCA and YANG to enable complete life cycle management of SDN and NFV functionalities and to configure physical and virtual devices that comprise an end-to-end service.

The paper describes how TOSCA is being used to describe orchestration between resources across complex end to end service(s) such as SD-WAN and Cloud Security. Demonstrate the capabilities of TOSCA like substitution/service decomposition for vendor abstraction to realize some of the VNF's, interface and life cycle operations used for the implementation aspects, requirement, capability types, node referencing features, define and visualize end-to-end service as a topology graph of inter-connected nodes and to navigate across connected services.

YANG is a data modeling language used to describe the device configuration. Device/network configuration properties are modeled as YANG. The syntax and semantic constraints and capabilities offered by YANG are made use of during service validation and activation.

2. Overview

At a high level, for Comcast use cases, Service life cycle management comprise of one or more of the following.

1) Staging/provisioning the base configuration for the network devices
2) Amend changes to the configuration.
3) View service configurations
4) Updates service configuration.
5) Add/Remove features to the services
6) Upgrade/Downgrade of services
7) Monitoring the health of services and corrective actions in case of service degrade and/or disruption.

This paper will focus on the model-based service management platform. The platform provides APIs for some of the service life cycle management functionalities. The platform is a middleware layer that interacts with OSS/BSS systems northbound and Network Layer, VNF Managers, Vendor specific EMS/NMS systems southbound.

The paper highlights how TOSCA model and Service Orchestrator enables to support service life cycle management functionalities for different domains such as SD-WAN, Security, Wireless etc.

3. Service Model (TOSCA)

OASIS TOSCA is an orchestration language for automating Service Lifecycle Management. Service designers use TOSCA to create Service Templates that contain, model-based descriptions of services, platforms, infrastructure, and data components, along with their relationships, requirements, capabilities,

and configurations. The TOSCA model optionally supports service-specific orchestration and lifecycle management directives and operational policies that operate on these models. TOSCA model assumes a runtime environment (an "orchestrator") in which service models are processed with the goal of orchestrating the service and managing service lifecycles.

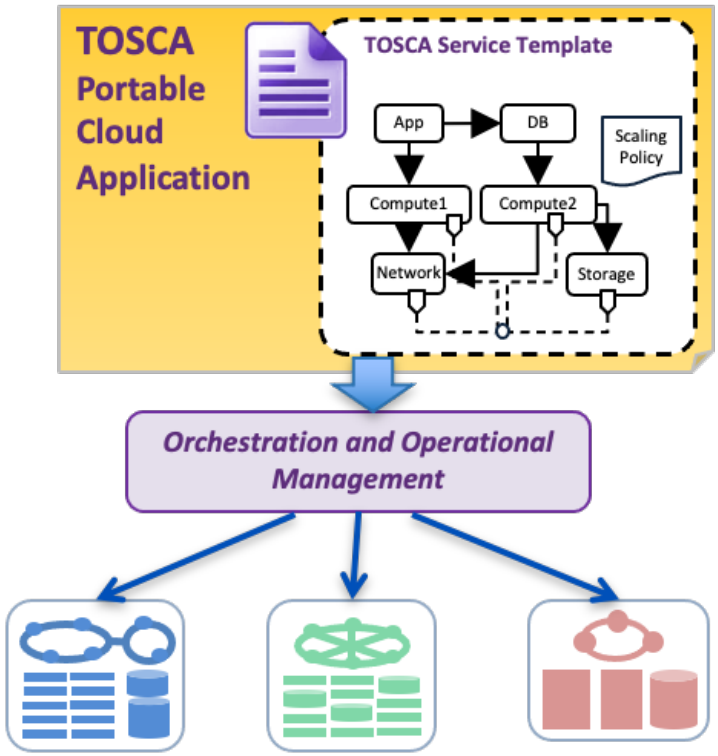

Figure 1 - What is TOSCA

3.1. Why TOSCA

TOSCA provides several features for the automation of service life cycle management via declarative directives. The following tables provide a high-level comparison of declarative versus imperative architecture.

Table 1 – Declarative versus Imperative Architecture

Declarative Architecture	Imperative Architecture
Arbitrary number of levels of recursion	Fixed number of layers
Federation built-in: North-south and east-west interfaces are the same	Federation must be added-on: North-south and east-west interfaces are different
Flexible resource layer: no architectural distinction between resources and services; resources are	Inflexible resources layer: distinction between resource layer and services layer is baked into the architecture

Declarative Architecture	Imperative Architecture
accessed as services, and services can be exposed as resources	
Identical DSL at each level in the recursion	Layer-specific APIs
Identical orchestration functionality at each level in the recursion	Layer-specific orchestration functionality
Interface implementations based on Domain-Specific Language (DSL)	Interface implementations based on APIs
Interface paradigm based on negotiation (request/response) and delegation	Interface paradigm based on management (higher layers control lower layers)
Organizing construct: recursive decomposition	Organizing construct: static layering

3.2. TOSCA Features

The following are some of the high-level features offered by TOSCA. These features are used as part of the service life cycle management.

- TOSCA Specification and TOSCA profiles. Profiles come with pre-defined type definitions.
- Ability to define type definitions (Node Types, Datatypes, Requirements, Interfaces, Capabilities, Artifacts etc.)
- Standard Interface & Lifecycle Operations - create / configure / start / stop / delete
- Custom Interfaces and Operations (Modify, Upgrade)
- Declarative workflows based on node relationships and capabilities
- Service Topology Template (Service inputs and outputs, Service components/nodes and their relationships)
- Substitution Mapping / Service Decomposition
- Node reference via node filter
- Package TOSCA models as a CSAR

3.3. Use of TOSCA and Yang

In certain use cases, there is a need to use multiple modeling languages for example, YANG. In such scenarios, the properties needed for the orchestration functionalities are modeled as TOSCA type definitions. The core feature configurations are modeled in YANG. The device configuration YANG model is parsed and represented as JSON schema and packaged as part of the CSAR. This allows the service orchestrator and implementation to validate input JSON data for the feature configurations against the schema at run time.

3.4. TOSCA Meta Model Phases

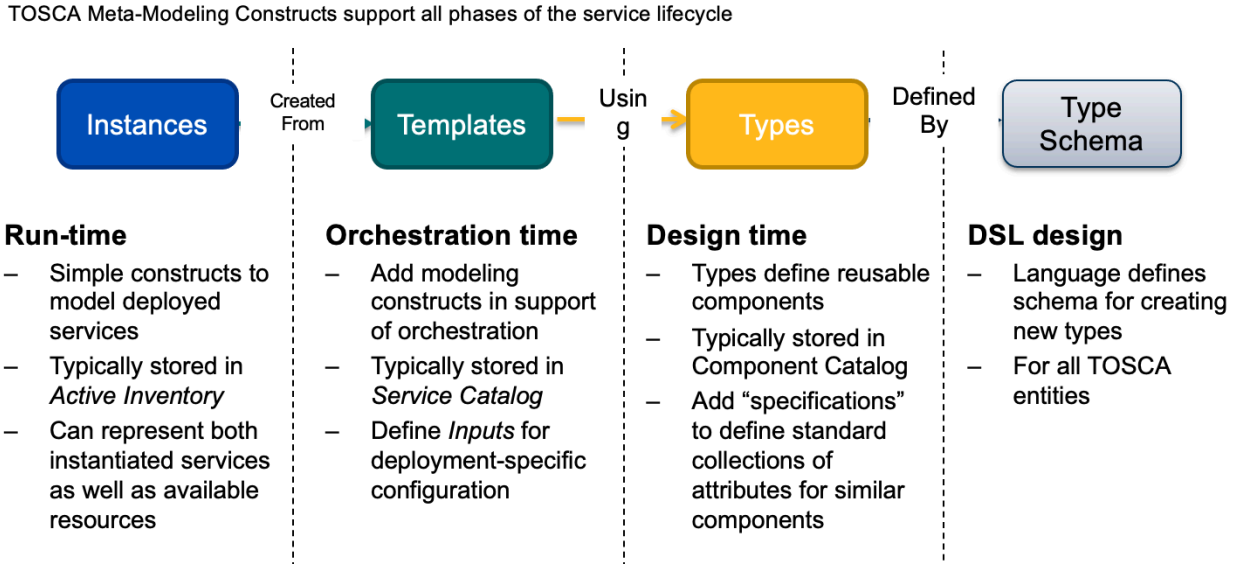

Figure 2 - TOSCA Meta Model Phases

3.5. TOSCA Object Model

Following is the TOSCA Object Model UML

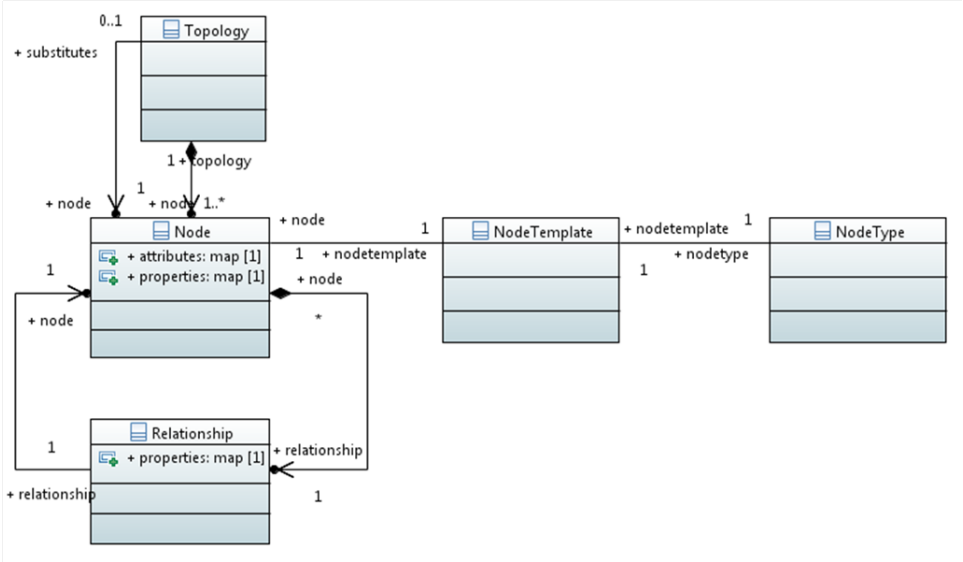

Figure 3 - TOSCA Object Model

4. Service Orchestration

4.1. Challenges with Service Life Cycle Management

The following lists some of the high-level challenges with regards to managing services.

- Service Model – How much to model/when not to model, Vendor specific versus Vendor agnostic, mix/match of model technologies (for example, TOSCA, YANG, SNMP, CLI etc.)
- Handing out of automation changes (Manage out of band changes and synchronization)
- Source of Truth – Network versus Automation System. This is critical in scenarios where out of band changes happen and the automation system does not have the capability to sync with the network.
- Rollbacks in case of failures – Automated versus Manual
- Managing service dependencies – Impact of a service change to another dependent service (Ripple effect)
- Service Upgrades – Model upgrade resulting in service instance(s) upgrades, Defect fixes, new feature rollouts etc.

4.2. Platform Goals

Software platform to unify and automate service lifecycle management functionalities via model driven (TOSCA for orchestration and YANG for configuration) architecture. The platform.

The platform offers capabilities to onboard models, provision/deploy/update services for different domains such as SD-WAN, Security, WIFI etc.

- A software platform to unify and automate complete service lifecycle
- Leverages Model driven Service Automation Language: TOSCA & YANG for configuration.
- Onboarding, Provisioning, Deployment, Monitoring
- Orchestrate multiple services (e.g., SD-WAN, Security, VNFs, other)
- Orchestrate hybrid environments – physical, multiple clouds and on premise
- Modeling and Service design tools
- Dynamic traffic steering and service chaining
- Active Inventory and Service Topology
- Southbound vendor specific implementations
- Enable us to define and implement business logic and functional logic independently

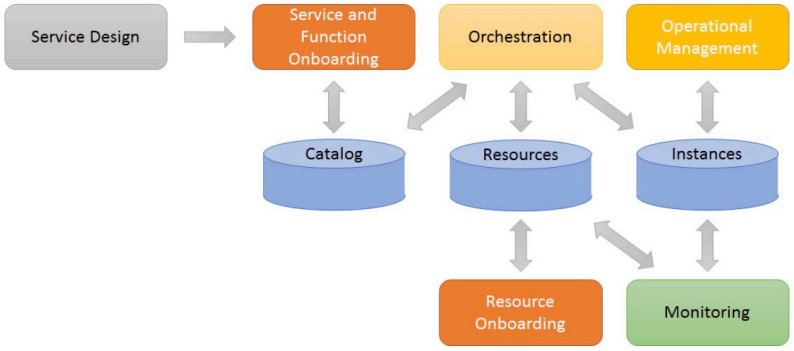

Figure 4 – Functional Architecture of Service Orchestrator

4.3. Service and Function Model Definition (Design)

Network Services and Network Functions are designed by creating TOSCA topology templates from which services can be instantiated and managed.

Service design involves the following steps:

- Defining set of type definitions (Data types, Node types, Relationship types and Capability types etc.). Reuse and/or enhance existing type definitions and add new type definitions.
- Define the topology template that represents the service as a topology of interconnected service components, relationships between the service components
- Define inputs that need to be provided at deployment time
- Define outputs that are expected to be returned upon successful service operation
- Provide implementation details (Plans/Workflows/Scripts references) for the life cycle operations (create, update, delete, upgrade, etc.) of the service components.
- Package as a cloud service archive (CSAR).

4.4. Service and Function Onboarding

The service packaged as a CSAR should be onboarded to the Service Orchestrator catalog repository. Service orchestrator validates (syntax and semantics) the CSAR and store in the catalog repository. Multiple versions of the CSAR can be onboarded.

4.5. Orchestration (Runtime)

At runtime, service(s) can be instantiated using the onboarded service CSAR. The service inputs as defined in the CSAR should be provided as part of service instantiation.

Service orchestrator performs validation of the inputs, invoke the implementations, and create service instances in the instance repository. Internally, TOSCA service orchestrator does the required substitutions, resolves references (matching requirements and capabilities) to the service components and resources. The service instance is a graph of service components nodes connected via relationships.

5. UseCases

5.1. SD-WAN

SD-WAN is an overlay service configured over the underlay infrastructure. Comcast offers physical/virtual CPEs in which SD-WAN is configured. CPEs are configured via VNF Manager. VNF Manager offers REST APIs. Configurations are staged at the VNF Manager and later committed to the CPEs (2 step process). SD-WAN logically modeled as 4 services. Services are linked. Information shared across services to enable configuration. For example, WAN Interface information, CPE details exposed by uCPE Service to VPN Site service. VPN information exposed by VPN Service to VPN Site Service.

Vendor VNF Manager modeled as a node type and node instance in the service (Not as a separate service/resource). Service Referencing (Orchestrator implementation extension, like the node referencing in TOSCA) feature used to link services and share information across services.

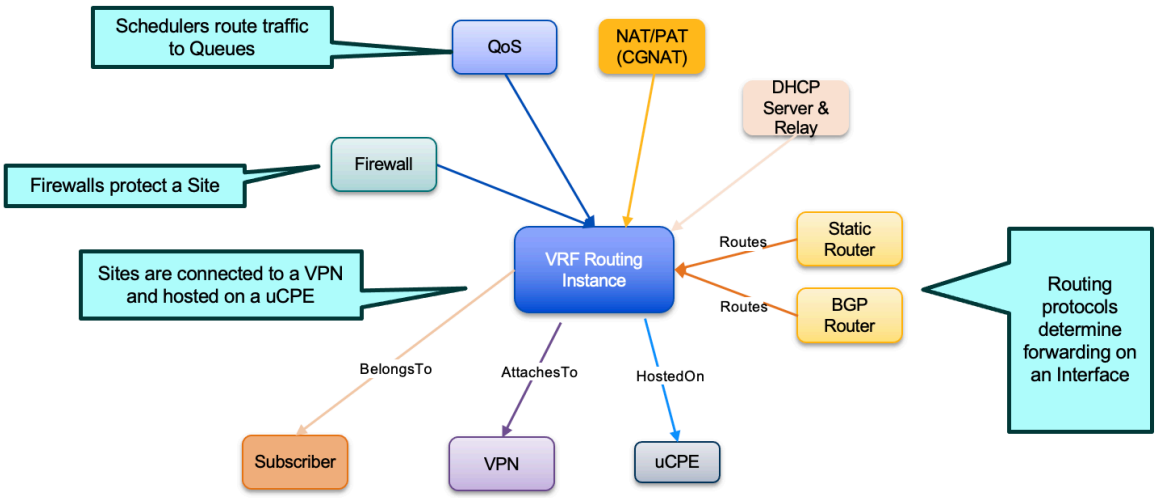

Figure 5 – SDWAN VPN Site Sevice (Sample)

The following are some high-level configurations represented as JSON type.

- Interface (LAN & WAN Interfaces) Configuration
- Routing Configuration (BGP, Static Routes etc.)
- DHCP Server & relay
- Class of Service
- CGNAT
- Firewall
- Traffic Steering Rules/Policies
- Custom Services & URL Filter Definitions

SD-WAN VPN Site Service Topology Template (Snippet only)

Following topology template snippet for a service component/node in a topology template. The sample highlights the node requirements and the life cycle operations of the service component (create, delete, upgrade) and the implementation references.

```
firewall:
      type: comcast.nodes.sdwan.fw.v1.FirewallConfiguration

      properties:
        vnfManager: { get_input: vnfManager }
        customerId: { get_input: customerId }
        firewallConfiguration: { get_input: firewallConfiguration }}
        zones: { get_property: [ vrfRoutingInstance, zones ] }
        interfaces: { get_attribute: [ SELF, ucpeService, interfaces ] }
         ## Properties omitted for clarity


      requirements:
        - protectable:
            node: vrfRoutingInstance
        - device:
            node_filter:
               properties:
                 deviceId: { get_input: deviceId }

      interfaces:
        Standard:
          inputs:
             ## Omitted for clarity
          create:
            implementation:
              primary: Artifacts/Deployment/WORKFLOW/CreateFirewall
          delete:
            implementation:
              primary: Artifacts/Deployment/WORKFLOW/DeleteFirewall

        Upgrade:
          create:
            implementation:
              primary: Artifacts/Deployment/WORKFLOW/UpgradeFirewall
            inputs:
              ## Omitted for clarity
```

5.2. Firewall

Vendor agnostic models are defined as abstract models. The goal of abstraction is to avoid making technology/product decisions at design time. The abstract models will be decomposed (model decomposition feature of TOSCA/Service orchestrator) at runtime to vendor specific implementations.

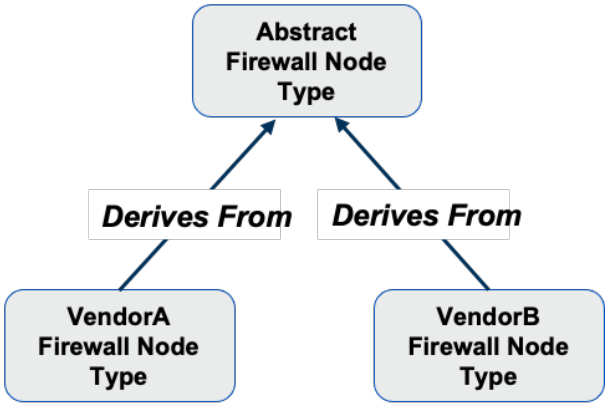

Figure 6 - Firewall Vendor Abstraction

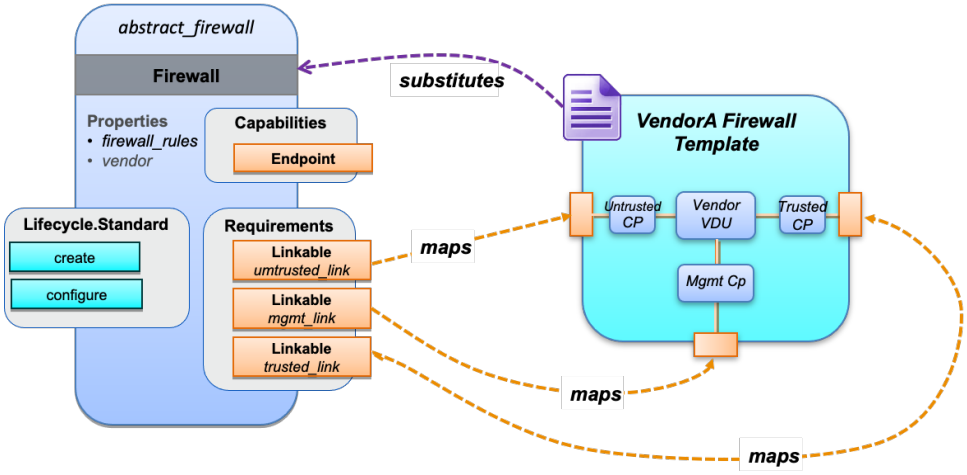

Figure 7 - Firewall Vendor Substitution

Substitution and Service decomposition - One Node instance substituted with a single node / set of nodes defined in the substituting topology template. In the above example, Firewall node substituted with Vendor A Firewall Template containing 4 nodes.

6. Conclusion

This paper highlighted how TOSCA as a modeling language and the TOSCA features combined with a service orchestrator simplifies the deployment of services for different domains.

The type definitions in TOSCA model simplifies the application to perform the validations. The service orchestrator combines all the node dependencies and generates the declarative workflow for the service minimizing the need to deal with dependencies in the application.

The service orchestrator enables to provide implementations for the supported life cycle interfaces and operations and is abstracted from the domain TOSCA models. In addition, it offers the flexibility to provide multiple implementations for a given model (Plug & Play).

Abbreviations

AP	Access Point
BGP	Border Gateway Protocol
CGNAT	Carrier-grade Network address translation
CPE	Customer Premises Equipment
CLI	Command Line Interface
CSAR	Cloud Service Archive
DHCP	Dynamic Host Configuration Protocol
JSON	JavaScript Object Notation
LAN	Local Area Network
REST	REpresentational State Transfer
SCTE	Society of Cable Telecommunications Engineers
SD-WAN	Software-Defined Wide-Area Networking
SDN	Software-Defined Networking
SNMP	Simple Network Management Protocol
TOSCA	Topology and Orchestration Specification for Cloud Applications
uCPE	Universal Customer Premises Equipment
VNF	Virtual Network Function
VPN	Virtual Private Network
WAN	Wide Area Network
YANG	Yet Another Next Generation

Bibliography & References

TOSCA Modeling Overview - OASIS, https://www.oasis-open.org/committees/download.php/62645/TOSCA%20Overview%202018%2002%2026.pdf.

