White Paper

Service Orchestration Requirements for a New Era of Networking

Prepared by

Caroline Chappell
Principal Analyst, Cloud & NFV, Heavy Reading
www.heavyreading.com

on behalf of

www.cisco.com

May 2015

Introduction

Although software-defined networking (SDN) and network functions virtualization (NFV) are associated with driving a new era of networking, they are in fact responses to customer demands for faster delivery of network services. Every business on the planet is being affected by the need to compete faster and exploit new market opportunities ahead of their rivals. Enterprises are turning to the cloud for the agile infrastructure that boosts operational speed. However, they also need connectivity and associated network-based services to be delivered in similar time scales.

A number of operators are already responding to their customers' hyperfast business requirements by rolling out "programmable network" services, also known as user-defined networks, or network as a service (NaaS). Such services are underpinned by an ability to configure (activate), dynamically and on demand, physical and virtual network elements – in other words, physical and virtual network functions, or PNFs and VNFs.

The key technology that supports on-demand activation of network elements is known as service orchestration. Service orchestration is a fashionable term that is broadly interpreted in ways that are causing considerable confusion. A particular area of misunderstanding is the relationship between a service orchestration system and the NFV Management and Orchestration (MANO) stack. Service orchestration is complementary to SDN and NFV, and is a critical capability for the new era of virtualized networking.

Because service orchestration is interpreted so widely, it is important that operators understand the basic principles that an effective and efficient service orchestration system should support. This paper examines the drivers for service orchestration and the three "extreme" business requirements for flexibility, automation and reliability emerging in a new era of hyperfast business that a service orchestration system must fulfill. It explores the capabilities that a service orchestration system should have to meet these requirements.

Section II looks at the impact of a cloud-driven pace of business on the network and customer requirements for the rapid delivery of network services.

Section III discusses the three "extreme" requirements for programming the network successfully.

Section IV describes the principles that operators need to look for in a future-proof service orchestration system that can coexist with a changing OSS and NFV management landscape.

Business in the Fast Lane

Cloud Enables Hyperfast Business Change

Every business is under pressure today to compete faster. Companies are beginning to need an almost protean ability to reconfigure aspects of themselves in real time to deliver what's been asked of them. Regardless of industry sector, organizations are challenged with the need to rethink the way they operate in an increasingly software-defined world.

To say that the cloud is responsible for hyperfast business change is overly simplistic. However, "the cloud" is shorthand for a set of capabilities that are enabling companies to do business faster: self-service access to business enablers such as cloud-hosted applications and processes, high levels of automation, programmable, scalable infrastructure and ubiquitous network access.

At Mobile World Congress 2015, Equinix pointed out that an average enterprise now needs to connect to 40 to 80 clouds to access the breadth of applications on which it runs its business. The rapid pace of business change means that companies can no longer afford the time it takes to onboard new applications on their private IT infrastructure. Instead, it's quicker to sign up to a software-as-a-service implementation. Which means, Equinix claims, that the average enterprise adds a new cloud every week.

Figure 1: Enterprises Are Moving to the Cloud

Source: Equinix, GE, Accenture Studies

The cloud makes it possible for companies to set up a presence almost instantly in a new market, without having to source an office location or acquire IT systems and staff. Today's speed of business dictates that enterprises should be able to add new virtual offices, partners and customers fast and flexibly, inserting themselves into value chains as soon as they see an opportunity. Opportunities are increasingly transitory and easy to miss: The race goes to those that can respond swiftly, rapidly fail if the market response is disappointing and scale if it proves fruitful.

Business Impact on the Network

In a cloud-based world, where companies can spin up hosted compute and storage resources for minutes or hours at a time to carry out specific processes and/or target new market opportunities, they want the same ability to create and tear

down connectivity. Businesses are beginning to question the need to sustain always-on, fixed network access to resources. They would like to create new connections on demand in near-real time and use them only as long as the process is running in the cloud.

A number of high-profile operators, including AT&T and Pacnet (now Telstra), are responding to these demands, implementing "software-defined" networks that allow business customers and/or their applications to order, add, change and flex connectivity services on demand.

Such networks are variously called "programmable networks," "user-defined networks" or "network as a service" (NaaS), but the principle is the same. They allow customers, through a portal, to logically define where they want connectivity services, their type, bandwidth and duration. Then the system automatically provisions the services, taking minutes to deploy what would once have taken a team of network engineers perhaps weeks to roll out.

The momentum behind the programmable approach to network service delivery is growing and unstoppable. Soon, this will become the "new normal" for connectivity services, and any operator unable to spin up, flex or add services in the same few-minute time scales as their competitors will risk losing customers.

The cloud-savvy next generation of customers will naturally shop around for the supplier that can give them service fastest – an activity that's easier to do in a self-service environment. Not just because such clients are impatient, but because they are themselves under pressure to deliver to their end customers in this new era of hyperfast business change.

The Programmable Network & NFV

NFV makes a key contribution to the hyperfast delivery of network services. NFV enables the rapid deployment of virtual network elements (VNFs) to meet new service requirements, for example to support a new customer-facing service type or to expand an existing service's reach and/or capacity.

New VNFs can be spun up in near-real time because they use the virtualization, orchestration and automation capabilities of the cloud instead of manual installation procedures. Customer-facing services delivered over VNFs, rather than over physical network elements, can therefore be launched much more quickly, and new service instances can be supported on demand.

However, the deployment of a new VNF also requires setting up connectivity – between the VNF's constituent virtual machines (VMs), between the VNF and virtual load balancers if there are multiple instances of the VNF, and between the VNF and other kinds of VNF participating in a customer-facing service.

Configuring network elements to support NFV connectivity must take place in the same time frame as the deployment of the VNF, or it acts as a brake on service delivery. Manually provisioning connections between VMs is not an option in the highly dynamic, high-scale cloud environment. Nor is the manual configuration of VNFs themselves. Once again, the network needs to be programmable to support the speed requirements of NFV. And once again, operators are clear that the end goal of NFV is to support their customers' hyperfast business requirements.

Extreme Network Programming Requirements

Service Complexity Is Growing

Service types are proliferating in response to market demands for different flavors of connectivity at different levels of the network across access networks and the core. This means even more complexity in the network, as new devices are added to support these new services. At the same time, service instances are growing increasingly complicated and varied, as users want to control, for example, their security and performance characteristics. In order to fulfill such services, operators need to link ("chain") multiple vendors' network elements across network layers, activating them at the same time.

The Role of Service Orchestration

In order to deploy, change and flex network services on demand, operators need to activate them in a timely manner across heterogeneous network environments built from layers of devices sourced from many different vendors. Traditional activation involves configuring network elements one by one, mostly by manual manipulation of the command line interfaces (CLIs) of individual network elements. Intermittent automation may be achieved with the help of OSS adapters, but these are slow and expensive to develop.

Service activation can take weeks to accomplish – longer if a specific adapter needs to be written – and each activation process is a "one-off" activity, particular to a network topology, service and customer. Because activation takes place at a physical level, without any kind of abstraction, there is little opportunity for process reuse across services, topologies and customers. Traditional activation is not only inefficient and costly; it is also unsuited to the emerging hyperfast pace of business.

As we pointed out in an earlier white paper, Driving Network Agility Through a Service Orchestration Approach, the term "**service orchestration**" is replacing the concept of network activation. Service orchestration is about **programming** the network: configuring all the elements participating in a service automatically, as a single, end-to-end transaction. The programmability and automation associated with service orchestration enable such transactions to be carried out at significantly lower cost in near-real time. Since the new speed of business requires frequent changes to services, service orchestration should also manage ongoing, in-life configuration changes to services, so that they are responsive to the dynamic nature of customer demand.

Service Orchestration & NFV

Very shortly, at least some of the network elements participating in customer-facing services will be virtualized. VNFs will also need to be activated as part of service fulfillment. This adds a new dimension of configuration complexity and confusion around the roles of service orchestration and the new NFV MANO system.

As we pointed out in a recent white paper, Deploying Virtual Network Functions: The Complementary Roles of TOSCA and NETCONF/YANG, the NFV MANO is responsible for the deployment of a VNF on top of cloud infrastructure. The NFV MANO uses a template approach to configure VNFs initially, leaving them in a "day zero" state of operational readiness. At runtime, when the VNF is needed to support the delivery of a service instance to a specific customer, it is service orchestration that activates (configures) the VNF for this purpose.

The NFV MANO does not have the same idea of network services that a service orchestration system does. As the white paper points out, the NFV MANO uses the term "network service" in an idiosyncratic way to describe a complex VNF consisting of two or more VMs that need to be deployed in a particular relationship ("forward graph") with one another, potentially across multiple locations. A better term for an NFV "network service" would have been "NFV application," to reflect the fact that in the cloud world, IT applications routinely consist of different components that all need to be deployed together.

The NFV MANO is also responsible for subsequent VNF lifecycle operations that are carried out manually in the physical network, such as upgrade, scale, heal and delete. These operations are beyond the scope of service orchestration. However, NFV MANO and service orchestration need a close working relationship. When the NFV MANO scales up a VNF, for example, runtime service configurations need to be applied to the new VNF instance, so service orchestration comes into play.

The Three Extremes

To cope with increasing service variety and complexity, leading operators want to define services in a way that separates them from details of their implementation across vendor-specific network elements. In the future, they envisage customers downloading such service definitions from widely available catalogs on the Internet. Service definitions will likely be written in a standard service modeling language that can be programmatically manipulated, such as YANG. Customers will then look for a network service provider that can instantiate, e.g., a standard L3VPN service model, fastest, for the best price, with a specific QoS and/or according to other criteria.

Operators will need to be able to ingest any service model and programmatically configure it as a service instance on their networks. To do this, a service orchestration system cannot afford simply to target a single service type or configure a vendor-specific set of network devices. Instead, it must meet three "extreme" requirements:

- **Extreme flexibility:** The ability to configure any network element engaged in delivering any service, regardless of element supplier and service type. As the network is becoming virtualized, a service orchestration system must be able to configure both physical and virtual instances of network elements.
- **Extreme automation:** The ability to carry out "hands-free" network element activation through programmatic interfaces and without any interruption to services at runtime. Extreme automation supports the dynamic modification of services and the prevention of manual fallout through the consistent, repeatable execution of automated processes.
- **Extreme reliability:** The ability to apply configurations to the live network in a completely safe and trustworthy manner, using ACID transactionality to protect network consistency. Extreme reliability implies that changes to the network should be made based on a stateful (real-time) understanding of its current configuration.

Service orchestration is a fashionable term and a growing number of systems claim to carry out programmable, model-driven configuration of the network. Operators that want to address the hyperfast business requirements of their customers by investing in a service orchestration capability need to understand the technical principles behind each system's approach and to evaluate how well they meet the "three extreme requirements." It is unlikely that existing OSSs that are merely rebadged as "service orchestration systems" will be able to cope.

Key Service Orchestration Principles

The key principles behind an effective service orchestration system that can add and modify network services in real time, map to the three requirements of flexibility, automation and reliability. **Figure 2** summarizes these principles and their mapping.

Figure 2: Mapping Service Orchestration System Principles to Extreme Requirements

REQUIREMENT	PRINCIPLES
Extreme Flexibility	1. Standards-based configuration with broad industry support from network element vendors and operators/industry bodies 2. Standards-based definition of services 3. **Any** modification of **any** service instance on the fly: e.g., change bandwidth, quality of service (QoS), add/remove links 4. Service independence from the orchestration engine
Extreme Automation	1. Service **and** network element definitions captured in models, enabling programmatic "model to model" manipulation 2. Programmatic means of writing configurations to network elements 3. Programmatic interfaces to other operational systems, including the NFV MANO stack, enabling flow-through process automation, closed feedback loop automation
Extreme Reliability	1. Stateful understanding of network configuration on a per device/service instance basis 2. Support for ACID transactionality 3. Support for high-scale operations

Source: Heavy Reading

Support for Standards

At Mobile World Congress 2015, four leading network operators – AT&T, Deutsche Telekom, Equinix and Level 3 – talked of their requirement for a standards-based way of programming network devices from any vendor, replacing the manual configuration of tens or hundreds of devices that service providers must undertake to deliver a service to an individual customer. They and other operators worldwide are in the process of adopting Network Configuration Protocol (NETCONF), the Internet Engineering Task Force (IETF) standard protocol for reading and writing network configurations.

NETCONF's associated data modeling language, YANG, has, as we pointed out in a previous white paper, The Business Case for NETCONF/YANG in Network Devices, been designed to support the configuration of network elements. YANG's structure and syntax reflects the way that configuration data is represented on network devices. Device data models created in YANG are able to be programmed through NETCONF quickly and easily. A growing number of network element vendors, including Alcatel-Lucent, Brocade, Cisco and Juniper are producing YANG data models for both their physical devices and those devices' virtual counterparts: VNFs.

In a standards-based service orchestration environment, all the participating network elements must first be modeled in YANG, if not by their vendor, then by the

operator or a systems integrator. Device models take a matter of weeks to create, but unlike network adapters, they are non-proprietary, since they are written in a standard language. They can easily be replaced with vendor versions when these become available, without affecting services or orchestrations.

Given the industry momentum behind the NETCONF/YANG standard, a service orchestration system engineered natively to support them provides operators with a future-proof, standardized way of programming their networks, replacing the multiple proprietary interfaces to the network that have grown up over past decades.

However, NETCONF/YANG adoption is a work in progress: The vast majority of network elements today still need to be programmed through proprietary CLIs, or occasionally through proprietary REST APIs. A service orchestration system will still need to support specific, non-standard interfaces southbound in order to programmatically configure devices that have a YANG model associated with them, but which don't yet natively support NETCONF.

As a semantically rich modeling language, YANG can also be used to define services. Modelling services formally in YANG eliminates problems with the current method of service definition. This typically involves a product manager describing a service in natural language using a high-level, paper-based information model. Programmers then need to interpret the model, creating workflows to instantiate it for specific customers across network resources (elements).

This method leads to proliferation of workflows, each tightly coupled to a specific service instance, a specific operation and set of resources. Such workflows, or templates using CLI snippets are time-consuming to create and unwieldy to manage. Further service changes require new templates to be written, so workflow creation becomes an unending and expensive activity.

A YANG service model shares with a natural language-based service model the fact that it does not contain details of the network resources used to fulfill it, so it can be reused to meet different customer requirements in different network circumstances. However, the similarity ends there. Unlike a natural language description, a YANG service model is formally described in a human and computer-readable language, so it can be programmatically manipulated ("parsed") and instantiated automatically by a service orchestration system. A service orchestration system that can ingest any YANG service model and understand how to orchestrate it across network resources, without first defining vendor-specific CLI snippets (workflows), will provide operators with extreme flexibility. The ability to automate this process delivers extreme speed.

In the future, the use of a standards-based, computer-readable service modeling language will also support the emerging NaaS trend, which will see business customers bringing their own service definitions to operator portals for orchestration.

Separating Service Models & Orchestration

It is key that service models are decoupled from the service orchestration engine. To aid flexibility and automation, a service orchestration system needs to remain "stupid" with regard to domain-specific service knowledge. All service knowledge should be captured by service domain experts in formally described service models that are external to the orchestrator, not hard-coded into it. Customer-specific requirements are fulfilled by creating instances of the service model with customer-specific parameters, rather than by writing special, hard-coded orchestration cases (CLI snippets).

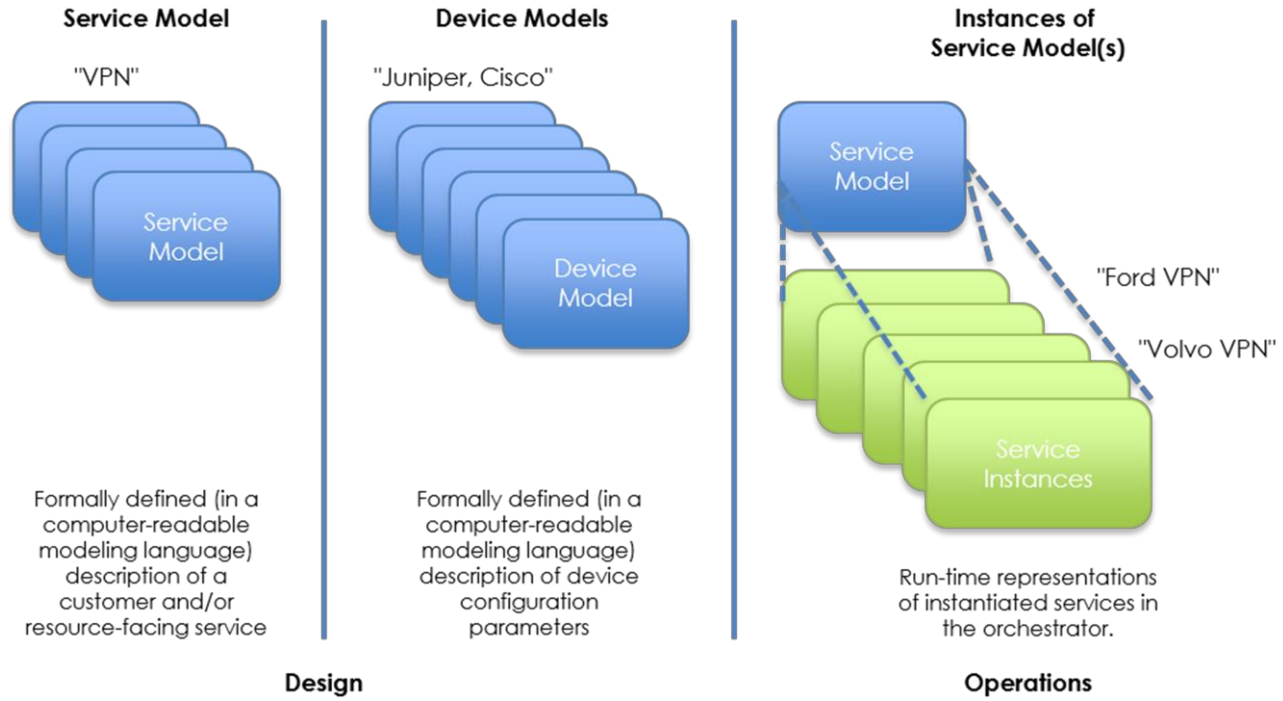
When service models are owned by service domain experts, not the operations team responsible for activating services, service design time can be reduced from months and weeks to days and hours. Creating a new service type is a matter of writing a new YANG model. Since the orchestrations needed to activate instances of the model are rendered automatically at runtime by the service orchestration system and don't need to be created in advance by a small army of programmers, the fulfillment of customer-facing services is much faster.

For complete flexibility, product/service managers should be able to add new service models to the service orchestration system at any time with no impact on the non-stop operation of the system. The service orchestration system nevertheless needs to understand the network context in which it is orchestrating services. It should take an engineering view of the network, with a deep understanding of how to configure multiple types of network device in order to create an appropriate orchestration for each service instance.

Model-Model Driven Automation

A service orchestration system that can work with both YANG service models and YANG network element models is in a powerful position natively to translate one into the other without an intervening, proprietary data transformation step. This supports rapid, reliable and standards-based automation of the configuration process. It also provides operators with the flexible ability to reuse the same service models against devices from different vendors, for example, in different network locations, and/or interchangeably use physical or virtual network elements.

Figure 3: The Role of Models



Source: Heavy Reading/Cisco

Programmatic translation between formally defined service and device models within a service orchestration system supports "no hands" automation of network configuration, which eliminates unnecessary manual fallout.

For full NaaS automation, however, the service orchestration engine should allow individual customers to modify service instances on the fly, for example, changing bandwidth or QoS parameters, or adding or removing links. The service orchestration system should be able to implement the new configurations in the network seamlessly from a customer experience point of view, without any service downtime. This should remain the case even if the service instance is re-instantiated on a different set of network resources and/or in a new location to fulfill the new customer requirements.

Service orchestration (activation) is critical to the service fulfillment process, but only part of the end-to-end process that converts customer service orders to cash. In order to support the process automation needed for a NaaS use case, for example, the service orchestration system needs open APIs northbound through which it can communicate with other B/OSSs and be triggered by them.

At the same time, a service orchestration system also needs to be able to drive the NFV MANO stack, for example, programmatically instructing the NFV MANO to spin up VNFs and their associated resources when these are needed to deploy a service. The service orchestration system is then responsible for applying service instance-specific configurations to the VNF(s).

Stateful Understanding of the Network

Operators need to be able to trust automation, particularly when it is applied to such a critical and sensitive process as activation. In order to make reliable decisions about what to configure in the network, a service orchestration system will need a real-time understanding of the network's current configured state, for all services and all devices.

Such a stateful view makes it possible to minimize actions on the network – a desirable goal, since the application of large configuration changes can have unpredictable effects on network performance, and can even cause network outages.

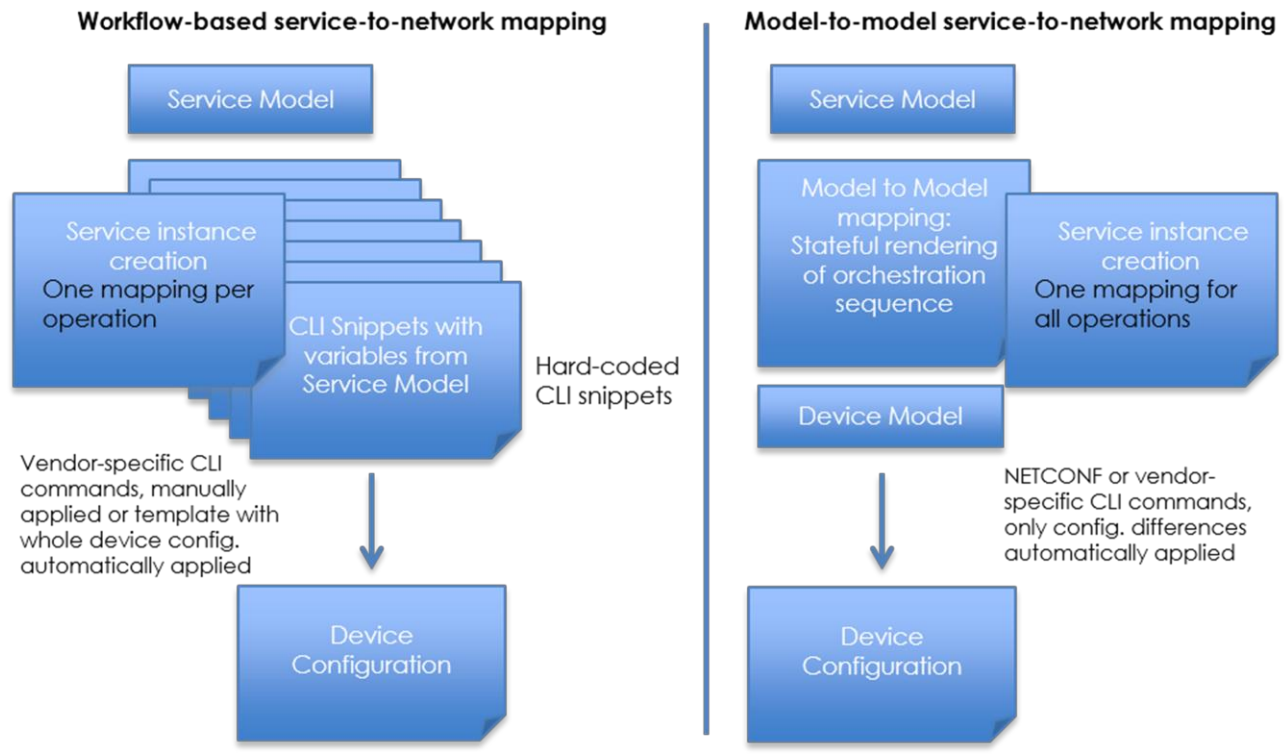
The service orchestration system will therefore need to maintain a real-time repository of all live service instances with their mappings to network elements, physical and/or virtual, and those elements' live configurations. Such a repository represents a "single view of the truth" for the network, a valuable source of stateful data for the service orchestration system, and potentially for other OSSs as well.

When a YANG-based service orchestration system needs to activate a configuration change, it must first check what the desired service change means for its associated devices. The service orchestration system should consult its repository of current service instances and device configurations to understand which device(s) need reconfiguration, and compare the latter's configured state with the desired state.

To make the configuration change, the service orchestration system can take advantage of the declarative mapping possible between YANG service and device models. Since both models are written in the same language, declarative mapping between fields that have been changed in the service model and related fields that now need to be modified in network element model(s) involves a transformation that is easily automated.

In contrast, workflow-based transformations consist of a series of CLI commands that need to be carried out manually and each command sequence ("snippet") has to be defined for each possible change and error case. In a YANG model-model mapping approach, only one declarative mapping is defined and the service orchestration system renders the sequence of operations needed to effect a particular change at run-time.

Figure 4: Comparison of Model-Model & Workflow-Based Service-to-Network Mapping



Source: Heavy Reading/Cisco

An efficient and highly reliable service orchestration system will take a surgical approach to changing fields in network element model(s). It will have an intelligent algorithm that can calculate, from the service instance's changed parameters, exactly which fields in a device model need to be reconfigured. It then sends only those minimal changes to the network. The simplicity and automation made possible through such an algorithm enables large numbers of service attributes to be modified dynamically in a programmable network without affecting network performance.

The surgical application of configuration changes is very different from, for example, operators' use of programmers working from stateless (paper-based) service model descriptions to add configurations manually to network devices.

It also contrasts with other automated approaches, for example, CLI templates that have predefined command sequences ("snippets") hard-coded within them. The traditional CLI template approach may support dynamic changes to service parameters within the template and its reapplication to network devices, but it has a number of shortcomings:

- Traditional CLI templates are often created on a per-activity basis, so a service instance may have multiple different templates associated with it, depending on the "create, delete, modify" or other actions that may be applied. Template proliferation is inelegant, and new template creation is an ongoing and expensive activity.
- Traditional CLI templates typically lack knowledge of the current configured state of the network. They don't work out the "minimal differences" between desired and current states, and therefore rewrite a device's entire configuration. This may involve applying hundreds of lines of CLI code, whereas a state-aware, model-driven service orchestration system might render the exact change in two lines. Since large configuration changes to individual devices may have a destabilizing effect on the network, a template-driven approach is inherently less reliable.
- Traditional CLI templates can struggle with the rollback of non-symmetric configuration changes if an orchestration fails for any reason. In this case, a service orchestration system should be able to apply its intelligent algorithm in reverse, identifying the minimal differences between the changed and previous states of the network and rolling back only those changes, in the correct order. This is difficult unless the service orchestration system understands the semantics of its models, including specific dependencies between configuration items, and has a stateful view of network configuration.

Transactionality & Scale

NETCONF is a transactional protocol by design, but the service orchestration system needs to take advantage of this, supporting the two-phase commit of all configuration changes to the network. That is, either all changes are committed or, in the event of failure, none of them are, and the transaction is completely rolled back. Transactionality protects network consistency and underpins the reliability of service orchestrations. It is important that the service orchestration system can reliably carry out network configuration on a large scale, given current growth trends for network services and elements and the impact of SDN and NFV.

Conclusion

The ability to reprogram the network at speed to meet the hyperfast and increasingly complex service needs of customers is set to become the top competitive differentiator between operators over the next few years.

NFV is part of this speed equation, as network infrastructure is virtualized to enable the agile deployment of network functions on which customer-facing service delivery depends. Programmable connectivity between VNFs, which requires the rapid and accurate configuration of both VNFs and underlying network elements, is a key ingredient of NFV-based service delivery.

Traditional service activation methods that rely on the manual configuration of network elements are no longer fast enough in physical network scenarios, let alone in the cloud environment that underpins NFV. Instead, service orchestration, which supports the programmable configuration of all the network elements/VNFs participating in a service, as a single end-to-end transaction, is in vogue.

However, the growing number of systems that claim to carry out programmable configuration of the network have subtle differences that affect how flexibly and reliably they can activate services, and how effectively they can automate the configuration process at scale. Operators need to delve under the hood to understand how far a particular service orchestration system will free them from network element and OSS vendor dependencies, the cost and time involved in creating orchestration sequences for different service instances, and the risks involved in making network configuration changes on the fly.

Operators will need to assess how far a service orchestration system:

- Uses a standards-based approach to service modeling that natively enables service models to be programmatically manipulated.
- Supports a standard protocol for reading and writing network configurations alongside pragmatic support for vendor-specific CLIs.
- Separates service models from service orchestrations, so that no hard-coded orchestrations are associated with service models and specific service instances are maintained in the service orchestration system. The system should be able to render orchestrations dynamically at service runtime.
- Supports network device abstraction in the form of standards-based network device models so that translation between service and network device/VNF models can be automated rapidly and reliably.
- Has a stateful understanding of the configured network, which it uses, together with model-to-model translation capabilities, to support a surgical approach to making network configuration changes. Rewriting the configuration of an entire device just to change a couple of parameters can have an undesirable impact on network performance.
- Supports transactionality at scale, so that automated programming of the network can be trusted.

Service orchestration is a key capability that every operator will need in the future. Getting it right will help operators ensure their success in a hyperfast world.

About Cisco

Cisco (Nasdaq: CSCO) is the worldwide leader in IT that helps companies seize the opportunities of tomorrow by proving that amazing things can happen when you connect the previously unconnected. For ongoing news, please go to newsroom.cisco.com. Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. A listing of Cisco's trademarks can be found at www.cisco.com/go/trademarks. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company.