Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: September 6, 2018

J. Arkko
Ericsson
J. Tantsura
Nuagenetworks
J. Halpern
B. Varga
Ericsson

March 5, 2018

Considerations on Network Virtualization and Slicing
draft-arkko-arch-virtualization-01

Abstract

This document makes some observations on the effects of virtualization on Internet architecture, as well as provides some guidelines for further work at the IETF relating to virtualization.

This document also provides a summary of IETF technologies that relate to network virtualization. An understanding of what current technologies there exist and what they can or cannot do is the first step in developing plans for possible extensions.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 6, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

Network virtualization is network management pertaining to treating different traffic categories in separate virtual networks, with independent lifecycle management and resource, technology, and topology choices.

This document makes some observations on the effects of virtualization on Internet architecture, as well as provides some guidelines for further work at the IETF relating to virtualization.

This document also provides a summary of IETF technologies that relate to network virtualization. An understanding of what current technologies there exist and what they can or cannot do is the first step in developing plans for possible extensions.

In particular, many IETF discussions earlier in the summer of 2017 started from a top-down view of new virtualization technologies, but were often unable to explain the necessary delta to the wealth of existing IETF technology in this space. This document takes a different, bottom-up approach to the topic and attempts to document existing technology, and then identify areas of needed development.

In particular, whether one calls a particular piece of technology "virtualization", "slicing", "separation", or "network selection" does not matter at the level of a system. Any modern system will use several underlying technology components that may use different terms but provide some separation or management. So, for instance, in a given system you may use VLAN tags in an ethernet segment, MPLS or VPNs across the domain, NAIs to select the right AAA instance, and run all this top of virtualized operating system and software-based switches. As new needs are being recognised in the developing virtualization technology, what should drive the work is the need for specific capabilities rather than the need to distinghuish a particular term from another term.

2. Definitions

Network function virtualization is defined in Wikipedia as follows:

> "Network function virtualization or NFV is a network architecture concept that uses the technologies of IT virtualization to virtualize entire classes of network node functions into building blocks that may connect, or chain together, to create communication services.
>
> NFV relies upon, but differs from, traditional server-virtualization techniques, such as those used in enterprise IT. A virtualized network function, or VNF, may consist of one or more virtual machines running different software and processes, on top of standard high-volume servers, switches and storage devices, or even cloud computing infrastructure, instead of having custom hardware appliances for each network function."

We should not confuse NFV and network virtualization, the former, as the name suggests is about functions virtualization, and not the network.

The idea of network virtualization is almost as old as the networking technology itself. Network virtualization is hierarchical and multilayer in its nature, from layer 1 up to services on top. When talking about virtualization we usually define overlay to underlay relationship between different layers, bottom up. A VPN (Virtual Private Network) [RFC4026] is the most common form of network virtualization. The general benefits and desirability of VPNs have been described many times and in many places ([RFC4110] and [RFC4664]).

The only immutable infrastructure is the "physical" medium, that could be dedicated or "sliced" to provide services(VPNs) in a multi-tenant environment.

The term slicing has been used to describe a virtualization concept in planned 5G networks. The 3GPP architecture specification [TS-3GPP.23.501] defines network slices as having potentially different "supported features and network functions optimisations", and spanning functions from core network to radio access networks.

[I-D.king-teas-applicability-actn-slicing] defined slicing as "an approach to network operations that builds on the concept of network abstraction to provide programmability, flexibility, and modularity. It may use techniques such as Software Defined Networking (SDN) and Network Function Virtualization (NFV) to create multiple logical (virtual) networks, each tailored for a set of services that are sharing the same set of requirements, on top of a common network.

And, [I-D.geng-coms-problem-statement] defines slicing as a management mechanism that an service provider can use to allocate dedicated network resources from shared network infrastructures to a tenant.

3. General Observations

Software vs. Protocols

Many of the necessary tools for using virtualization are software, e.g., tools that enable running processes or entire machines in a virtual environment decoupled from physical machines and isolated from each other, virtual switches that connect systems together, management tools to set up virtual environments, and so on. From a communications perspective these tools operate largely in the same fashion as their real-world counterparts do, except that there may not be wires or other physical communication channels, and that connections can be made in the desired fashion.

In general, there is no reason for protocols to change just because a function or a connection exists on a virtual platform. However, sometimes there are useful underlying technologies that facilitiate connection to virtualized systems, or optimised or additional tools that are needed in the the virtualized environment.

For instance, many underlying technologies enable virtualization at hardware or physical networking level. For instance, Ethernet networks have Virtual LAN (VLAN) tags and mobile networks have a choice of Access Point Names (APNs). These techniques allow users and traffic to be put on specific networks, which in turn may comprise of virtual components.

Other examples of protocols providing helpful techniques include virtual private networking mechanisms or management mechanisms and data models that can assist in setting up and administering virtualized systems.

There may also be situations where scaling demands changes in protocols. An ability to replicate many instances may push the limits of protocol mechanisms that were designed primarily or originally for physical networks.

Selection vs. Creation and Orchestration

Two primary tasks in virtualization should be differentiated: selection of a particular virtual instance, and the tasks related to how that virtual instance was created and continues to be managed.

Selection involves choosing a particular virtual instance, or an entrypoint to a virtual network. In its simplest form, a customer could be hardwired by configuration to a particular virtual instance. In more complex cases, the connecting devices may have some settings that affect the choice. In the general case, both the connecting devices and the network they are connecting to it have a say in the choice.

The selection choice may even be dynamic in some cases. For instance, traffic pattern analysis may affect the selection.

Typically, however, connecting devices do not have a say in what the virtual instance does. This is directed by the network operator and its customers. An instance is specified, created, and needs to be continously managed and orchestrated. The creation can be manual and occur rarely, or be more dynamic, e.g., an instance can actually be instantiated automatically, and only when the first connecting device connects to it.

Protocols vs. Representations of Virtual Networks

Some of virtualization technology benefits from protocol support either in the data or control plane. But there are also management constructs, such as data models representing virtual services or networks and data models useful in the construction of such services.

There are also conceptual definitions that may be needed when constructing either protocols or data models or when discussing service agreements between providers and consumers.

4. Virtualization in 5G Networks

Goals for the support of virtualization in 5G relate to both the use of virtualized network functions to build the 5G network, and to enabling the separation of different user or traffic classes into separate network constructs called slices.

Slices enable a separation of concerns, allow the creation of dedicated services for special traffic types, allow faster evolution of the network mechanisms by easing gradual migration to new functionality, and enable faster time to market for new new functionality.

In 5G, slice selection happens as a combination of settings in the User Equipment (UE) and the network. Settings in the UE include, for instance, the Access Point Name (APN), Dedicated Core Network Indicator (DCN-ID) [TS-3GPP.23.401], and, with 5G, a slice indicator (Network Slice Selection Assistance Information or NSSAI) [TS-3GPP.23.501]. This information is combined with the information configured in the network for a given subscriber and the policies of the networks involved. Ultimately, a slice is selected.

A 5G access network carries a user’s connection attempt to the 5G core network and the Access Management Function (AMF) network function. This function collects information provided by the UE and the subscriber database from home network, and consults the Network Slice Selection Function (NSSF) to make a decision of the slice selected for the user. When the selection has been made, this may also mean that the connection is moved to a different AMF; enabling separate networks to have entirely different network-level service.

The creation and orchestration of slices does not happen at this signalling plane, but rather the slices are separately specified, created, and managed, typically with the help of an orchestrator function.

The exact mechanisms for doing this continue to evolve, but in any case involve multiple layers of technology, ranging from underlying virtualization software to network component configuration mechanisms and models (often in YANG) to higher abstraction level descriptions (often in TOSCA), to orchestrator software.

5. Overview of IETF Virtualization Technologies

General networking protocols are largely agnostic to virtualization. TCP/IP does not care whether it runs on a physical wire or on a computer-created connection between virtual devices.

As a result, virtualization generally does not affect TCP/IP itself or applications running on top. There are some exceptions, though, such as when the need to virtualize has caused previously held assumptions to break, and the Internet community has had to provide new solutions. For instance, early versions of the HTTP protocol assumed a single host served a single website. The advent of virtual hosting and pressure to not use large numbers of IPv4 addresses lead to HTTP 1.1 adopting virtual hosting, where the identified web host is indicated inside the HTTP protocol rather than inferred from the reception of a request at particular IP address [VirtualHosting] [RFC2616].

But where virtualization affects the Internet architecture and implementations is at lower layers, the physical and MAC layers, the systems that deal with the delivery of IP packets to the right destination, management frameworks controlling these systems, and data models designed to help the creation, monitoring, or management of virtualized services.

What follows is an overview of existing technologies and technologies currently under development that support virtualization in its various forms.

5.1. Selection of Virtual Instances

Some L2 technology allows the identification of traffic belonging to a particular virtual network or connection. For instance, Ethernet VLAN tags.

There are some IETF technologies that also allow similar identification of connections setup with the help of IETF protocols. For instance, Network Access Identifiers may identify a particular customer or virtual service within AAA, EAP or IKEv2 VPN connections.

5.2. Traffic Separation in VPNs

Technologies that assist separation and engineering of networks include both end-point and provider-based VPNs. End-point VPN tehchnologies include, for instance, IPsec-based VPNs [RFC4301].

For providing virtualized services, however, provider-based solutions are often the most relevant ones. L1VPN facilitates virtualization of the underlying L0 "physical" medium. L2[IEEE802.1Q] facilitates virtualization of the underlying Ethernet network Tunneling over IP (MPLS, GRE, VxLAN, IPinIP, L2TP, etc) facilitates virtualization of the underlying IP network - MPLS LSP’s - either traffic engineered or not belong here L2VPN facilitates virtualization of a L2 network L3VPN facilitates virtualization of a L3 network.

The IETF has defined a multiplicity of technologies that can be used for provider-based VPNs. The technologies choices available can be described along two axes, control mechanisms and dataplane encapsulation mechanisms. The two are not compeltely orthogonal.

In the data plane, for provider based VPNs, the first important observation is that the most obvious encapsulation is NOT used. While IPSec could be used for provider-based VPNs, it does not appear to be used in practice, and is not the focus for any of the available control mechanisms. Often, when end2end encryption is required it is used as an overlay over MPLS based L3VPN

The common encapsulation for provider-based VPNs is to use MPLS. This is particularly common for VPNs within one operator, and is sometimes supported across operators.

Keyed GRE can be used, particularly for cross-operator cases. However, it seems to be rare in practice.

The usage of MPLS for provider-based VPNs generally follows a pattern of using two (or more) MPLS labels, top (transport) label to represent the remote end point/egress provider-edge device, and bottom (service) label to signal the different VPNs on the remote end point. Using TE might result in a deeper label stack.

L2 VPNs could be signaled thru LDP[RFC4762] or MP-BGP[RFC4761], L3 VPN is signaled thru MP-BGP[RFC4364]

The LDP usage to control VPN establishment falls within the PALS working group, and is used to establish pseudo-wires to carry Ethernet (or lower layer) traffic. The Ethernet cases tend to be called VPLS (Virtual Private LAN Service) for multi-point connectivity and VPWS (Virtual Private Wire Service) for point-to-point connectivity. These mechanism do augment the data plane capabilites with control words that support additional features. In operation, LDP is used to signal the communicating end-points that are interested in communicating with each other in support of specific VPNs. Information about the MAC addresses used behind the provider edges is exchanged using classic Ethernet flooding technology. It has been proposed to use BGP to bootstrap the exchang eof information as to who the communicating endpoints are.

BGP can be used to establish Layer 2 or Layer 3 VPNs. Originally, the BGP based MPLS VPN technology was developed to support layer 3 VPNs. the BGP exchanges uses several different features in MP-BGP (specifically route distinguishers and route targets) to control the distribution of information about VPN end-points. The BGP information carries the VPN IP address prefixes, and the MPLS labels to be used to represent the VPN. This technolgoy combination is generally known as L3VPN.

This usage of BGP for VPNs has been extended to support Layer 2 VPNs. This is known as EVPN. The BGP exchanges are used to carry the MAC address reachability behind each provider edge router, providing an Ethernet multipoint service without a need to flood unkown-destination Ethernet packets.

In theory, the BGP mechanisms can also be used to support other tunnels such as keyed GRE. That is not widely practiced.

There are also hybrid variations, such as adding an ARP / ND proxy service so that an L3VPN can be used with an L2 Access, when the only desired service is IP.

5.3. Traffic Engineering and QoS

Traffic Engineering (TE) is the term used to refer to techniques that enable operators to control how specific traffic flows are treated within their networks.

The TEAS working group works on enhancements to traffic-engineering capabilities for MPLS and GMPLS networks:

> TE is applied to packet networks via MPLS TE tunnels and LSPs. The MPLS-TE control plane was generalized to additionally support non-packet technologies via GMPLS. RSVP-TE is the signaling protocol used for both MPLS-TE and GMPLS.
>
> The TEAS WG is responsible for:
>
> * Traffic-engineering architectures for generic applicability across packet and non-packet networks.
>
> * Definition of protocol-independent metrics and parameters.
>
> * Functional specification of extensions for routing (OSPF, ISIS), for path computation (PCE), and RSVP-TE to provide general enablers of traffic-engineering systems.
>
> * Definition of control plane mechanisms and extensions to allow the setup and maintenance of TE paths and TE tunnels that span multiple domains and/or switching technologies.

A good example of work that is currently considered in the TEAS WG is the set of models that detail earlier IETF-developed topology models with both traffic engineering information and connection to what services are running on top of the network [I-D.bryskin-teas-use-cases-sf-aware-topo-model] [I-D.bryskin-teas-sf-aware-topo-model]. These models enable reasoning about the state of the network with respect to those services, and to set up services with optimal network connectivity.

Traffic engineering is a common requirement for many routing systems, and also discussed, e.g., in the context of LISP.

5.4. Service Chaining

The SFC working group has defined the concept of Service Chaining:

> Today, common deployment models have service functions inserted on the data-forwarding path between communicating peers. Going forward, however, there is a need to move to a different model, where service functions, whether physical or virtualized, are not required to reside on the direct data path and traffic is instead steered through required service functions, wherever they are deployed.
>
> For a given service, the abstracted view of the required service functions and the order in which they are to be applied is called a Service Function Chain (SFC). An SFC is instantiated through selection of specific service function instances on specific network nodes to form a service graph: this is called a Service Function Path (SFP). The service functions may be applied at any layer within the network protocol stack (network layer, transport layer, application layer, etc.).

5.5. Management Frameworks and Data Models

There have been two working groups at the IETF, focusing on data models describing VPNs. The IETF and the industry in general is currently specifying a set of YANG models for network element and protocol configuration [RFC6020].

YANG is a powerful and versatile data modeling language that was designed from the requirements of network operators for an easy to use and robust mechanism for provisioning devices and services across networks. It was originally designed at the Internet Engineering Task Force (IETF) and has been so successful that it has been adopted as the standard for modeling design in many other standards bodies such as the Metro Ethernet Forum, OpenDaylight, OpenConfig, and others. The number of YANG modules being implemented for interfaces, devices, and service is growing rapidly.

(It should be noted that there are also other description formats, e.g., Topology and Orchestration Specification for Cloud Applications (TOSCA) [TOSCA-1.0] [TOSCA-Profile-1.1], common in many higher abstract level network service descriptions. The ONAP open source project plans to employ it for abstract mobile network slicing models, for instance.)

A service model is an abstract model, at a higher level than network element or protocol configuration. A service model for VPN service describes a VPN in a manner that a customer of the VPN service would see it.

It needs to be clearly understood that such a service model is not a configuration model. That is, it does not provide details for configuring network elements or protocols: that work is expected to be carried out in other protocol-specific working groups. Instead, service models contain the characteristics of the service as discussed between the operators and their customers. A separate process is responsible for mapping this customer service model onto the protocols and network elements depending on how the network operator chooses to realise the service.

The L2SM WG specifies a service model for L2-based VPNs:

> The Layer Two Virtual Private Network Service Model (L2SM) working group is a short-lived WG. It is tasked to create a YANG data model that describes a L2VPN service (a L2VPN customer service model). The model can be used for communication between customers and network operators, and to provide input to automated control and configuration applications.
>
> It is recognized that it would be beneficial to have a common base model that addresses multiple popular L2VPN service types. The working group derives a single data model that includes support for the following:
>
> * point-to-point Virtual Private Wire Services (VPWS),
>
> * multipoint Virtual Private LAN services (VPLS) that use LDP-signaled Pseudowires,
>
> * multipoint Virtual Private LAN services (VPLS) that use a Border Gateway Protocol (BGP) control plane as described in [RFC4761] and [RFC6624],
>
> * Ethernet VPNs specified in [RFC7432].

Other L2VPN service types may be included if there is consensus in the working group.

Similarly, the L3SM WG specified a sevice model for L3-based VPNs.

The Layer Three Virtual Private Network Service Model (L3SM) working group is a short-lived WG tasked to create a YANG data model that describes a L3VPN service (a L3VPN service model) that can be used for communication between customers and network operators, and to provide input to automated control and configuration applications.

It needs to be clearly understood that this L3VPN service model is not an L3VPN configuration model. That is, it does not provide details for configuring network elements or protocols. Instead it contains the characteristics of the service.

6. Architectural Observations

This section makes some observations about architectural trends and issues.

Role of Software

An obvious trend is that bigger and bigger parts of the functionality in a network is driven by software, e.g., orchestration or management tools that figure out how to control relatively simple network element functionality. The software components are where the intelligence is, and a smaller fraction of the intelligence resides in network elements, nor is the intelligence encoded in the behaviour rules of the protocols that the network elements use to communicate with each other.

Centralization of Functions

An interesting architectural trend is that virtualization and data /software driven networking technologies are driving network architectures where functionality moves towards central entities such as various controllers, path computation servers, and orchestration systems.

A natural consequence of this is the simplification (and perhaps commoditization) of network elements, while the "intelligent" or higher value functions migrate to the center.

The benefits are largely in the manageability, control, and speed of change. There are, however, potential pitfalls to be aware of as well. First off, networks need to continue to be operate even under partial connectivity situations and breakage, and it is key that designs can handle those situations as well.

And it is important that network users and peers continue to be able to operate and connect in the distributed, voluntary manner that we have today. Today’s virtualization technology is primarily used to manage single administrative domains and to offer specific service to others. One could imagine centralised models being taken too far as well, limiting the ability of other network owners to manage their own networks.

Tailored vs. general-purpose networking

The interest in building tailored solutions, tailored Quality-of-Service offerings vs. building general-purpose "low touch" networks seems to fluctuate over time.

It is important to find the right balance here. From an economics perspective, it may not be feasible to provide specialised service -- at least if it requires human effort -- for large fraction of use cases. Even if those are very useful in critical applications.

Need for descriptions

As networks deal more and more with virtual services, there arises a need to have generally understood, portable descriptions of these service. Hence the creation of YANG data models representing abstract VPN services, for instance.

We can also identify some potential architectural principles, such as:

Data model layering

Given the heterogenuity of networking technologies and the differing users that data models are being designed for, it seems difficult to provide a single-level model. It seems preferable to construct a layered set of models, for instance abstract, user-facing models that specify services that can then be mapped to concrete configuration model for networks. And these can in turn be mapped to individual network element configuration models.

Getting this layered design right is crucial for our ability to evolve a useful set of data models.

Ability to evolve modelling tools and mapping systems

The networks and their models are complex, and mapping from high abstraction level specifications to concrete network configurations is a hard problem.

It is important that each of the components can evolve on its own. It should be possible to plug in a new language that represents network models better. Or replace a software component that performs mapping between layers to one that works better.

While this should normally be possible, there's room to avoid too tight binding between the different aspects of a system. For instance, abstraction layers within software can shield the software from being too closely tied with a particular representation language.

Similarly, it would be an advantage to develop algorithms and mapping approaches separately from the software that actually does that, so that another piece of software could easily follow the same guidelines and provide an alternate implementation. Perhaps there's an opportunity for specification work to focus more on processing rules than protocol behaviours, for instance.

General over specific

In the quick pace of important developments, it is tempting to focus on specific concepts and service offerings such as 5G slicing.

But a preferrable approach seems to provide general-purpose tools that can be used by 5G and other networks, and whose longetivity exceeds that of a version of a specific offering. The quick development pace is likely driving the evolution of concepts in any case, and building IETF tools that provide the ability to deal with different technologies is most useful.

7. Further Work

There may be needs for further work in this area at the IETF. Before discussing the specific needs, it may be useful to classify the types of useful work that might come to question. And perhaps also outline some types of work that is not appropriate for the IETF.

The IETF works primarily on protocols, but in many cases also with data models that help manage systems, as well as operational guidance documents. But the IETF does not work on software, such as abstractions that only need to exist inside computers or ones that do not have an effect on protocols either on real or simulated "wires".

The IETF also does not generally work on system-level design. IETF is best at designing components, not putting those components together to achieve a particular purpose or build a specific application.

As a result, IETF's work on new systems employing virtualization techniques (such as 5G slicing concept) is more at the component improvement level than at the level of the concept. There needs to be a mapping between a vision of a system and how it utilizes various software, hardware, and protocol tools to achieve the particular virtualization capabilities it needs to. Developing a new concept does not necessarily mean that entirely new solutions are needed throughtout the stack. Indeed, systems and concepts are usually built on top of solid, well defined components such as the ones produced by the IETF.

That mapping work is necessarily something that those who want to achieve some new functionality need to do; it is difficult for others to take a position on what the new functionality is. But at the same time, IETF working groups and participants typically have a perspective on how their technology should develop and be extended. Those two viewpoints must meet.

The kinds of potential new work in this space falls generally in the following classes:

Virtualization selectors

Sometimes protocols need mechanisms that make it possible to use them as multiple instances. E.g., VLAN tags were added to Ethernet frames, NAIs were added to PPP and EAP, and so on. These cases are rare today, because most protocols and mechanisms have some kind of selector that can be used to run multiple instances or connect to multiple different networks.

Traffic engineering

A big reason for building specific networks for specific purposes is to provide an engineered service level on delay and other factors to the given customer. There are a number of different tools in the IETF to help manage and engineer networks, but it is also an area that continues to develop and will likely see new functionality.

Virtual service data models

Data models -- such as those described by L2SM or L3SM working groups can represent a "service" offered by a network, a setup built for a specific customer or purpose.

Some specific areas where work is likely needed include:

- The ability to manage heterogenous technologies, e.g., across SDN and traditionally built networks, or manage both general-purpose and very technology-specific parameters such as those associated with 5G radio.

- The ability to specify "statistical" rather than hard performance parameters. In some networks -- notably with wireless technology -- recent advances have made very high peak rates possible, but with increased bursty-ness of traffic and with potential bottlenecks on the aggregation parts of the networks. The ability to specify statistical performance in data models and in VPN configuration would be important, over different timescales and probabilities.

- Mapping from high abstraction level specifications to concrete network configurations.

  There is a lot of work on data models and templates at various levels and in different representations. There are also many systems built to manage these models and orchestrate network configuration. But the mapping of the abstract models to concrete network configurations remains a hard problem, and it certainly will need more work.

  There are even some questions about how to go about this. Is it enough that we specify models, and leave the mapping to "magic" of the software? Are the connections something that different vendors compete in producing good products in? Or are the mapping algorithms something that needs to be specified together, and their ability to work with different types of network equipment verified in some manner?

- Cross-domain: A big problem is that we have little tools for cross-domain management of virtualized networks and resources.

Finally, there is a question of where all this work should reside. There’s an argument that IETF-based virtualization technologies deserve proper management tools, including data models.

And there’s another argument that with the extensive use of virtualization technology, solutions that can manage many different networks should be general, and as such, potential IETF work material. Yet, the IETF is not and should not be in the space of replacing various tools and open source toolkits that have been created for managing virtualization. It seems though that work on commonly usable data models at several layers of abstraction would be good work at the IETF.

Nevertheless, the IETF should understand where the broader community is and what tools they use for what purpose, and try to help by building on those components. Virtualization and slicing are sometimes represented as issues needing a single solution. In reality, they are an interworking of a number of different tools.

8. Acknowledgements

The authors would like to thank Gonzalo Camarillo, Gabriel Montenegro, Alex Galis, Adrian Farrell, Liang Geng, Yi Zhao, Hannu Flinck, Yi Zhao, Barry Leiba, Georg Mayer, Benoit Claise, Daniele Ceccarelli, Warren Kumari, Ted Hardie, and many others for interesting discussions in this problem space.

9. Informative References

[CC2015] claffy, kc. and D. Clark, "Adding Enhanced Services to the Internet: Lessons from History", September 2015 (https://www.caida.org/publications/papers/2015/adding_enhanced_services_internet/adding_enhanced_services_internet.pdf).

[I-D.bryskin-teas-sf-aware-topo-model]
Bryskin, I. and X. Liu, "SF Aware TE Topology YANG Model", draft-bryskin-teas-sf-aware-topo-model-01 (work in progress), March 2018.

[I-D.bryskin-teas-use-cases-sf-aware-topo-model]
Bryskin, I., Liu, X., Guichard, J., Lee, Y., Contreras, L., and D. Ceccarelli, "Use Cases for SF Aware Topology Models", draft-bryskin-teas-use-cases-sf-aware-topo-model-02 (work in progress), March 2018.

[I-D.geng-coms-problem-statement]
67, 4., Slawomir, S., Qiang, L., Matsushima, S., Galis, A., and L. Contreras, "Problem Statement of Supervised Heterogeneous Network Slicing", draft-geng-coms-problem-statement-00 (work in progress), September 2017.

[I-D.ietf-sfc-nsh]

Quinn, P., Elzur, U., and C. Pignataro, "Network Service Header (NSH)", draft-ietf-sfc-nsh-28 (work in progress), November 2017.

[I-D.king-teas-applicability-actn-slicing]
King, D. and Y. Lee, "Applicability of Abstraction and Control of Traffic Engineered Networks (ACTN) to Network Slicing", draft-king-teas-applicability-actn-slicing-01 (work in progress), July 2017.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, DOI 10.17487/RFC2616, June 1999, <https://www.rfc-editor.org/info/rfc2616>.

[RFC4026] Andersson, L. and T. Madsen, "Provider Provisioned Virtual Private Network (VPN) Terminology", RFC 4026, DOI 10.17487/RFC4026, March 2005, <https://www.rfc-editor.org/info/rfc4026>.

[RFC4110] Callon, R. and M. Suzuki, "A Framework for Layer 3 Provider-Provisioned Virtual Private Networks (PPVPNs)", RFC 4110, DOI 10.17487/RFC4110, July 2005, <https://www.rfc-editor.org/info/rfc4110>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

[RFC4664] Andersson, L., Ed. and E. Rosen, Ed., "Framework for Layer 2 Virtual Private Networks (L2VPNs)", RFC 4664, DOI 10.17487/RFC4664, September 2006, <https://www.rfc-editor.org/info/rfc4664>.

[RFC4761] Kompella, K., Ed. and Y. Rekhter, Ed., "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", RFC 4761, DOI 10.17487/RFC4761, January 2007, <https://www.rfc-editor.org/info/rfc4761>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6624] Kompella, K., Kothari, B., and R. Cherukuri, "Layer 2 Virtual Private Networks Using BGP for Auto-Discovery and Signaling", RFC 6624, DOI 10.17487/RFC6624, May 2012, <https://www.rfc-editor.org/info/rfc6624>.

[RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <https://www.rfc-editor.org/info/rfc7432>.

[RFC8049] Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8049, DOI 10.17487/RFC8049, February 2017, <https://www.rfc-editor.org/info/rfc8049>.

[TOSCA-1.0] OASIS, "Topology and Orchestration Specification for Cloud Applications Version 1.0", OASIS OASIS Standard, http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html, November 2013.

[TOSCA-Profile-1.1] OASIS, "TOSCA Simple Profile in YAML Version 1.1", OASIS OASIS Standard, http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.1/TOSCA-Simple-Profile-YAML-v1.1.html, January 2018.

[TS-3GPP.23.401] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access; (Release 15)", 3GPP Technical Specification 23.401, December 2017.

[TS-3GPP.23.501] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security architecture and procedures for 5G System; (Release 15)", 3GPP Technical Specification 23.501, December 2017.

[VirtualHosting] Wikipedia, "Virtual Hosting", Wikipedia article https://en.wikipedia.org/wiki/Virtual_hosting, August 2017.

Authors’ Addresses

Jari Arkko
Ericsson
Kauniainen 02700
Finland

Email: jari.arkko@piuha.net

Jeff Tantsura
Nuagenetworks

Email: jefftant.ietf@gmail.com

Joel Halpern
Ericsson

Email: joel.halpern@ericsson.com

Balazs Varga
Ericsson
Budapest 1097
Hungary

Email: balazs.a.varga@ericsson.com

none X. de Foy
Internet-Draft A. Rahman
Intended status: Informational InterDigital Inc.
Expires: September 9, 2020 A. Galis
University College London
K. Makhijani
L. Qiang
Huawei Technologies
S. Homma
NTT
P. Martinez-Julia
NICT
March 8, 2020

Interconnecting (or Stitching) Network Slice Subnets
draft-defoy-coms-subnet-interconnection-04

Abstract

This document defines the network slice (NS) subnet as a general management plane concept that augments a baseline YANG network slice model with management attributes and operations enabling interconnections (or stitching) between network slices. The description of NS subnet interconnections is technology agnostic, and is not tied to a particular implementation of the interconnection in data plane.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 9, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

Network Slicing enables deployment and management of services with diverse requirements on end-to-end partitioned virtual networks over the same infrastructure, including networking, compute and storage resources. There were recent efforts in the IETF to define a transport slice ([I-D.nsdt-teas-transport-slice-definition]) and to define a north-bound interface for such a transport slice ([I-D.contreras-teas-slice-nbi]). The mapping of transport slices in 5G mobile systems is also studied in [I-D.clt-dmm-tn-aware-mobility] and [I-D.geng-teas-network-slice-mapping].

Network slices may be managed through usage of YANG data models. For example, [I-D.liu-teas-transport-network-slice-yang] describes how existing YANG models can be augmented with network slice attributes.

Nevertheless, defining and managing a network slice (NS) end-to-end does not always have to be done directly. It may be convenient to define and manage separately subsets of an end-to-end slice. The concept of network slice subnet is defined originally in [NGMN_Network_Slicing], though we only need to retain its definition in the most universal form: network slice subnets are similar to network slices in most ways but cannot be operated in isolation as a complete network slice (e.g., a NS subnet can be seen as a network slice with unconnected links). NS subnets are interconnected with other NS subnets to form a complete, end-to-end network slice (i.e. interconnection and/or stitching of NS subnets). In the present draft, we describe a data model for describing interconnections between NS subnets, that enables assembling them in a hierarchical fashion.

1.1. Motivation and Roles of NS Subnet

NS subnet is a management plane concept that facilitates interconnections (also known as stitching) of network slices. It augments the base slice information model, that can be used to represent an end-to-end network slice. The extensions described in this document can be used to represent a slice subnet instead, and can also be used to represent an interconnection inside an end-to-end slice, i.e. they aim to represent interconnection points both "before" and "after" the interconnection takes place. Operations such as stitching subnets are also described.

The description of NS subnet interconnections is technology agnostic following the approach of the slice information model. Some interconnections may be implemented using the interplay between management plane and gateways in the data plane. [I-D.homma-rtgwg-slice-gateway] describes the requirements on such data plane network elements, and will provide input for the management plane mechanisms described in the present document.

1.2. Usage of NS Subnets

Using NS subnets can help:

- Isolate management and maintenance of different portions of a network slice, over multiple infrastructure domains, or even within a single domain. For example, in Figure 1, NS orchestrator (NSO) 2 manages subnet A, in isolation from subnets B and C managed by NSO 3. NSO 1 can still manage the end-to-end slice as a whole, but it does not need to deal in detail with each subnet.

- Isolate mapping towards different infrastructure technologies, even within the same domain. This can simplify NS orchestrator implementation, since each NSO can specialize in managing a smaller set of technologies.

o Enable advanced functions such as sharing a slice subnet between several slices, or substituting one slice subnet for another, e.g. for coping with load.

```
                   +-----------+
             ******| NS Orch. 1|********
             *     +-----------+       *
             *                         *
             *                         *
        +-----------+              +-----------+
        | NS Orch. 2|              | NS Orch. 3|*****
        +-----------+              +-----------+    *
             *                         *            *
             *                         *            *
             *   A-B Inter-            * B-C Inter- *
             *   connection            * connection *
+-----------------+   .  +-----------------+  .  +-----------------+
|      +--+       |   .  |      +--+       |  .  |      +--+       |
|      |  +---------------------+  +--------------------+  |       |
|      ++-+       |   .  |      ++-+       |  .  |      ++-+       |
|       |         |   .  |       |         |  .  |       |         |
| +---+ |  +---+  |   .  | +---+ |  +---+  |  .  | +---+ |  +---+  |
| |   +-+--+   +-----------+   +-+--+   +----------+   +-+--+   |  |
| +---+    +---+  |   .  | +---+    +---+  |  .  | +---+    +---+  |
+-----------------+   .  +-----------------+  .  +-----------------+

<.. NS subnet A ..>      <.. NS subnet B ..>     <.. NS subnet C ..>

<....................... end-to-end slice .........................>
```

Figure 1: Overview of Network Slice Subnets Interconnection

Figure 1 illustrates how an end-to-end network slice may be composed of multiple slice subnets, each managed independently by a same or different NSO. In multi-administrative domain scenarios, using NS subnets can help limiting the information that needs to be shared between domains. At the infrastructure layer (i.e. in the data plane), the interconnection between NS subnets may involve:

o a gateway, that performs protocol and/or identifier/label translation as needed,

o two gateways, especially in cases where interconnected NS subnets are in different administrative domains, o nothing at all, in cases where the interconnection point can be abstracted away, e.g. when the NS subnets share a common infrastructure. In this case nodes from both NS subnets end up being directly interconnected between each other.

More detailed usage scenarios are described in Section 2.4.2.

1.3. Terminology

Network slicing terminology, especially focusing on transport slices, is defined in [I-D.nsdt-teas-transport-slice-definition].

Network Slice Subnet (NS subnet): a network slice designed to be interconnected with other network slices.

NS Stitching: a management operation consisting in creating an end-to-end NS or a larger NS subnet, by interconnecting a set of NS subnets together.

Interconnection Anchor: a management plane entity, part of a NS subnet model, representing an end point for use in future stitching operation.

Interconnection Instance (or Interconnect): a management plane entity, part of a NS subnet model, representing an interconnection realized by a stitching operation. It is distinct from a (data plane) gateway: an interconnect may be realized with or without using a gateway in the data plane.

2. Information Model

2.1. Base Information Model

The information model we use as base for network slicing is the network topology model ietf-network defined in [RFC8345], in which networks are composed of nodes and links, and in which termination points (TP), defined in nodes, are used to define source and destination of links.

A network slice data model instance, i.e. a YANG data model augmented using [I-D.liu-teas-transport-network-slice-yang]), represents a network slice. When such a data model instance includes at least an "interconnection anchor", as defined below, it represents a network slice subnet instance.

At high level, the extensions defined in this document will augment nodes and termination points:

```
module: ietf-network
+--rw networks
   +--rw network* [network-id]
      +--rw network-id
      +--rw network-types
      +--rw supporting-network* [network-ref]
      |  +--rw network-ref
      +--rw node* [node-id]
      |  +--... (augmented with attributes for
      |  |       anchor/interconnection nodes)
      |  +--rw nt:termination-point* [tp-id]
      |  |  ... (augmented with attributes for
      |  |       anchor/interconnection TP)
```

2.2. Interconnection Anchors

To represent an anchor point for future interconnections (i.e. an unconnected end of a link), a simple solution is to use an "interconnection anchor" termination point (or anchor TP). Within the data model describing a subnet, any link not entirely contained within the NS subnet must be terminated with such an anchor TP as source or destination. An anchor TP belongs to a "node" attribute, which we refer to as interconnection anchor node (or anchor node). Several anchor TPs can be grouped together in an anchor node, and such grouping may be used as a hint during a stitching operation (e.g. to place all interconnection points at a same location).

Figure 2 represents 2 interconnected network slice subnets.

```
                               Slice Provider
                                     |
   +---------------------------------v---------------------------------+
   |  Network Slice Orchestrator                                       |
   |                                                                   |
   | +---------------------------------------------------------------+ |
   | |   Data model: network slice composed of NS subnet 1 and 2     | |
   | |                                                               | |
   | |      Network Slice Subnet 1            Network Slice Subnet 2 | |
   | | +---------------------------+  +----------------------------+ | |
   | | |     cross-subnet link     |  |   cross-subnet             | | |
   | | |    +----------------+     |  |       link    +------+     | | |
   | | |    |                |     |  |      +--------o node |     | | |
   | | |    |                |Interconnection|        +---o--+     | | |
   | | |+---o--+     +-------|-----+--+------|------+     |        | | |
   | | || node |     |       |     |  |      |      |     |        | | |
   | | |+---o--+     | +-----|---+ |  | +----|----+ |     |        | | |
   | | |    |        | |     |   | |  | |    |    | |     |        | | |
   | | |    |        | |     O - - - - - - - O    | |     |        | | |
   | | |    |        | |         | |  | |         | |     |        | | |
   | | |    |        | | anchor  | |  | | anchor  | |     |        | | |
   | | |    |        | |  node   | |  | |  node   | |     |        | | |
   | | |    |        | |         | |  | |         | |     +---+    | | |
   | | |    |        | |     O - - - - - - - O    | |         |    | | |
   | | |    |        | |     |   | |  | |    |    | |         |    | | |
   | | |    |        | +-----|---+ |  | +----|----+ |     +---o--+ | | |
   | | |    |        |       |     |  |      |      |     | node | | | |
   | | |    |        +-------|-----+--+------|------+     +---o--+ | | |
   | | |    | +------+       |     |  |      |                |    | | |
   | | |    +-o node o-------+     |  |      +----------------+    | | |
   | | |      +------+ cross-subnet|  |         cross-subnet       | | |
   | | |                link       |  |           link             | | |
   | | +---------------------------+  +----------------------------+ | |
   | +---------------------------------------------------------------+ |
   +--------------------------------+----------------------------------+
                                    |
                                    v
                            Network Infrastructure
```

Legend: o = termination point, O = anchor termination point

Figure 2: Network Slice Subnets Interconnection

Attributes of interconnection anchor nodes and termination points include:

- o Information enabling NS orchestrators to match anchor nodes and TPs from both NS during a stitching operation. A label may be a simple way to enable this.

- o Information to help locate the interconnection. For example, it could be a (sub-)domain name or geo-location information, that indicates where the interconnection point should be located. This can help for example in cases where the subnet is instantiated before stitching.

- o Information to help select the type of interconnection establishment: for example, this can indicate a preference for using interconnection over a gateway, or for abstracting away the interconnection point in the infrastructure plane.

```
+--rw node* [node-id]
   +-- (...)
   +-- anchor_node_config
   |    +-- label (and/or other auto stitching help)
   |    +-- hint for location (domain, geolocation, etc.)
   |    +-- hint for type (1 gateway, 2 gateways, ...)
   +--rw nt:termination-point* [tp-id]
       +-- (...)
       +-- anchor_tp_config
           +-- label (and/or other auto stitching help)
           +-- location (domain, geolocation, etc.)
           +-- type (1 gateway, 2 gateways, ...)
```

2.3. Interconnection Instances

There are two options for representing post-stitching network slices (or subnets). They are not mutually exclusive:

- o Option 1: subnet data models are updated with information describing the interconnection (e.g. anchor TPs and nodes are updated with new attributes representing the existing connection, if necessary).

- o Option 2: a new data model is generated to represent the resulting network slice (or subnet). In this composite data model, the interconnection may or may not be represented, this can be a choice made by the operator.

Option 1 and 2 can be used concurrently in a network. For example, a parent NS orchestrator may manage stitched NS subnets through underlying NS orchestrators, and at the same time expose to the NS operator a composite data model representing the resulting end-to-end slice.

To represent an existing interconnection in option 1, a simple solution is to add attributes to existing anchor nodes and anchor TPs. Those attributes will be described below. They aim to describe state and configuration associated with an active interconnection.

To represent an existing interconnection in option 2, a simple solution is to create new interconnection instance nodes and termination point. The same attributes as in option 1 may be associated with these nodes and TPs.

Attributes of interconnection instance nodes and termination points include:

o State information (interconnection type, status, location...).

o Service assurance related information: besides measurements (on throughput, loss rate, etc.), triggers depending on throughput, latency, etc. can be linked with a management action or event. A NS operator can use such events to take the decision to disable a NS subnet, replace a NS subnet with another, etc. to maintain overall service performance.

```
+--rw node* [node-id]
   +-- (...)
   +-- interconnection_instance_node_state
   |    +-- status
   |    +-- location (domain, geolocation, etc.)
   |    +-- type (1 gateway, 2 gateways, ...)
   +-- interconnection_instance_node_service_assurance
   |    +-- events (including triggers and event IDs)
   |    +-- measurements
   +--rw nt:termination-point* [tp-id]
       +-- (...)
       +-- interconnection_instance_tp_state
       |    +-- status
       |    +-- location (domain, geolocation, etc.)
       |    +-- type (1 gateway, 2 gateways, ...)
       +-- interconnection_instance_node_service_assurance
           +-- events (including triggers and event IDs)
           +-- measurements
```

2.4. Stitching Operation

2.4.1. Operation Overview

Stitching is an operation that takes two or more NS subnets as input, and produces a single composite NS subnet or end-to-end slice. It may occur when the slice subnets are being instantiated, or later.

The first step in this operation is to identify the anchors that will be used in the interconnection. This may be done by an automated algorithm that matches the possible interconnection points and decides which one will be used, according to the policies established by the NS operator. The operation in this case will require the presence of semantically-rich attributes in the candidate anchors to enable automatic matching without human intervention.

Other attributes of slices and anchors will also influence the operation and the resulting stitched (composite) object. For instance, network links that are interconnected must have compatible QoS attributes. Moreover, available networking protocols must also match among the underlying network elements that are being stitched. Otherwise, the operation will fail unless the NS operator (based on policy and/or NS subnet attributes) enables it to search for, and use, some "bridge" element in the underlying infrastructure.

2.4.2. Stitching Scenarios

This section briefly describes examples of usage for subnet stitching.

Traversal through a transport network.

Let's consider a network slice composed of (NS) subnet-A, and subnet-C (Figure 3). Subnet-A and subnet-C are deployed in independent domains and are mapped into a slice information model; in order to stitch these two together a transport segment is needed. N1 and N2 are anchor nodes within NS subnets A and C. Segment-B could be a simple link between the two NS subnets but it may also be a TE-link made available by a transport network provider. Segment-B may be involved in the stitching operation in one of several ways:

Segment-B may be set up as part of the stitching operation between NS subnets A and C, as a form of "bridge" mentioned in Section 2.4. Segment-B will need to comply with service specific traffic constraints that are determined during the stitching operation, possibly using attributes from NS subnets A and C. In this case, the data plane implementation of N1 and N2 in the composite slice may be, for example, 2 distinct gateway functions terminating segment-B.

Segment-B may alternatively be represented as a distinct NS subnet, e.g. in cases where segment-B is complex and/or involves multiple network functions. In this case, the stitching operation may therefore involve 3 NS subnets A-B-C.

```
+-----------+                    +----------+
|   +--+    |    ______          |   +--+   |
|   |N1+==========(______)============|N2|   |
|   +--+    |  --transport--     |   +--+   |
+-----------+                    +----------+
--subnet-A---  --segment-B------  --subnet-C--
<---------------end to end slice ------------>
```

Figure 3: Example of NS subnets interconnection through transport network

Subnets in a single domain.

In this scenario multiple network slice subnets are defined as basic building blocks with specific service functions (or chains), topologies and traffic handling characteristics. These building blocks can be assembled through stitching to build end-to-end customized slices, but also to dynamically extend slices to adapt to traffic load. Additionally, stitching can also be used to share building blocks between multiple slices, e.g. to interconnect multiple slices with a shared function. In all these cases, interconnection instances may be entirely abstracted away, although they may also be implemented through one or multiple gateways, e.g. when stitched subnets belong to different sub-domains.

3. Security Considerations

Security aspects relative to network slices (e.g., for transport slices, in [I-D.liu-teas-transport-network-slice-yang]) are applicable to slice subnets, including transport security aspects, access control and protection of write operation on newly introduced nodes (e.g., termination-point).

4. IANA Considerations

This document has no actions for IANA.

5. Informative References

[I-D.clt-dmm-tn-aware-mobility]
Chunduri, U., Li, R., Bhaskaran, S., Kaippallimalil, J., Tantsura, J., Contreras, L., and P. Muley, "Transport Network aware Mobility for 5G", draft-clt-dmm-tn-aware-mobility-05 (work in progress), November 2019.

[I-D.contreras-teas-slice-nbi]
Contreras, L., Homma, S., and J. Ordonez-Lucena, "Considerations for defining a Transport Slice NBI", draft-contreras-teas-slice-nbi-00 (work in progress), November 2019.

[I-D.geng-teas-network-slice-mapping]
Geng, X., Dong, J., Niwa, T., and J. Jin, "5G End-to-end Network Slice Mapping from the view of Transport Network", draft-geng-teas-network-slice-mapping-00 (work in progress), February 2020.

[I-D.homma-rtgwg-slice-gateway]
Homma, S., Foy, X., Galis, A., and L. Contreras, "Gateway Function for Network Slicing", draft-homma-rtgwg-slice-gateway-01 (work in progress), November 2019.

[I-D.liu-teas-transport-network-slice-yang]
Liu, X., Tantsura, J., Bryskin, I., Contreras, L., and Q. WU, "Transport Network Slice YANG Data Model", draft-liu-teas-transport-network-slice-yang-00 (work in progress), November 2019.

[I-D.nsdt-teas-transport-slice-definition]
Rokui, R., Homma, S., and K. Makhijani, "IETF Definition of Transport Slice", draft-nsdt-teas-transport-slice-definition-00 (work in progress), November 2019.

[NGMN_Network_Slicing]
NGMN, "Description of Network Slicing Concept", 10 2016, <https://www.ngmn.org/uploads/media/161010_NGMN_Network_Slicing_framework_v1.0.8.pdf>.

[RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March 2018, <https://www.rfc-editor.org/info/rfc8345>.

Authors' Addresses

Xavier de Foy
InterDigital Inc.
1000 Sherbrooke West
Montreal
Canada

Email: Xavier.Defoy@InterDigital.com

Akbar Rahman
InterDigital Inc.
1000 Sherbrooke West
Montreal
Canada

Email: Akbar.Rahman@InterDigital.com

Alex Galis
University College London
Torrington Place
London WC1E 7JE
United Kingdom

Email: a.galis@ucl.ac.uk

Kiran Makhijani
Huawei Technologies
2890 Central Expressway
Santa Clara CA 95050
USA

Email: kiran.makhijani@huawei.com

Li Qiang
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: qiangli3@huawei.com

Shunsuke Homma
NTT, Corp.
3-9-11, Midori-cho
Musashino-shi, Tokyo 180-8585
Japan

Email: homma.shunsuke@lab.ntt.co.jp

Pedro Martinez-Julia
National Institute of Information and Communications Technology
Japan

Email: pedro@nict.go.jp

none
Internet-Draft
Intended status: Informational
Expires: September 6, 2018

L. Geng
China Mobile
L. Qiang
Huawei
J. Ordonez
O. Adamuz-Hinojosa
P. Ameigeiras
University of Granada
D. Lopez
Telefonica I+D
L. Contreras
Telefonica

March 5, 2018

COMS Architecture
draft-geng-coms-architecture-02

Abstract

This document defines the overall architecture of a COMS based network slicing system. COMS works on the top level network slice orchestrator which directly communicates with the network slice provider and enables the technology-independent network slice management.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 6, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

Network slicing itself is a new concept triggered by vertical industry, but that doesn't mean new forwarding technology is needed. As an example given by [draft-arkko-arch-virtualization] shows, there are multiple existing technologies could be used for network slicing - VLAN tags are used in an ethernet segment, MPLS or VPNs across the domain. If the storage and computing resources are considered, there will be more available technologies (e.g., SFC).

Let's follow IETF's routine and image what will happen from the bottom-up view. At first, existing technologies evolve toward network slicing at forwarding plane in their own scopes. Then slice management related functions will be patched at management/control planes. When a network slice is going to be deployed inside a domain, one of implementation technology will be selected, and the NS provider directly operates on the management plane of this selected technology. For example, If VPN is selected as the implementation technology, then a network slice is a VPN for the NS provider in this domain. While if SFC is selected in other domain, then a network slice is a SFC for NS provider. What will happen if a network slice across both VPN and SFC domains? There is no uniform management manner in this case.

Then try to consider from the top-down view. There is no doubt that slicing requirement is generated from NS tenant. When a NS tenant request for NS service, normally he will not specify which implementation technology should be used. Similarly, when the tenant operates/manages his purchased slice, he doesn't want to care about the technical details.

We can easily observe that bottom-up and top-down approaches will eventually converge on a technology-independent common management plane, that is exactly what COMS (Common Operation and Management on network Slices) doing.

This document will explain how COMS works, and define the architecture of COMS. Architecture discussed in this document is assumed to be used only inside Transport Network region, and the end-to-end network slice/slicing also just refers to the slice/slicing across multiple TN domains in this document.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Other network slicing related words used in this document are interpreted as description in [COMS-PS].

Notations used in this document are interpreted as follows:

T(x->y): end-to-end delay from x to y;

B(x->y): bandwidth from x to y;

S(x): storage space of x.

3. Overall Architecture

This section provides the overall architecture for a COMS based network slicing system as shown in Figure 1. If multiple such kind of systems deployed in different domains, these systems may stitches together through the method discussed in [Stitching-Management] and [Stitching-Data]. COMS works on the top network orchestrator inside Transport Network region, which directly receives the network slice service profile, operation and management requests for network slices. Based on received information, the network orchestrator will select the most appropriate implementation technologies, and map the technology independent requests into the technology specific configuration information that will be sent to the corresponding network slice controller/orchestrator downwards.

```
                                      | Network Slice
                                      | Service Profile
                                      |
           +--------------------------v----------------------------+
           |                                                       |
           |     Top Level Network Orchestrator based on COMS      |
           |                                                       |
           +--------------------------+----------------------------+
                                      |
  +--------+------------------+-------+-----------+----------------+
  |        |                  |                   |                |
  | +------v--------+  +------v--------+  +-------v-------+   +----v----+
  | |               |  |               |  |               |   |         |
  | | NS Ctrlr/Orch |  | NS Ctrlr/Orch |  | NS Ctrlr/Orch |   |   ...   |
  | |               |  |               |  |               |   |         |
  | +-------+-------+  +-------+-------+  +-------+-------+   +----+----+
  |         |                  |                  |                |
  |         |                  |                  |                |
==v=========v==================v==================v================v=====
=                                                                       =
= +------------+ +------------+ +------------+ +------------+ +-------+ =
= |Connectivity| | Computing  | |  Storage   | |Generalized | |  ...  | =
= |            | |            | |            | |Functions   | |       | =
= +------------+ +------------+ +------------+ +------------+ +-------+ =
=                                                                       =
=========================================================================
```

Figure 1: Overall Architecture of COMS

4. Advanced Architecture

This section discusses the detailed architecture of a COMS based network slicing system through an example shown in Figure 2. We do not intend to design the inner framework of the top level network slicing orchestrator but to explain how COMS works. Four components insides the top level network orchestrator are logical components that could be converged sometimes.

- o Common Information Model: can be understood as the template, according to which the received network slice service profile is translated.

- o Split Service Profile into Domains: the end-to-end service profile is split into the service profiles inside different domains.

o Select Specific Implementation Technologies: there may be multiple available implementation technologies inside a domain, select the most appropriate one according to the service profile. As Figure 2’s example shows, since the end-to-end delay in Domain 1 is very small, the Flex-E will be selected. While in Domain 2 two storage units are required, the NFV technology will be selected.

o Map to Selected Technologies: necessary mapping to the controller/ orchestrator of selected technologies.

```
                                   |Network Slice Service Profile
                                   |
**************************************************************************
*Top Level Network Orchestrator based on COMS                            *
*                                  |                                     *
*  +-------------------------------v ---------------------------------+  *
*  |Common Information Model                                          |  *
*  | ................................................................ |  *
*  | . ~~~~~~~   T(A->B)<=10ms; B(A->B)>=10M         ~~~~~~~        . |  *
*  | . ~  A  ~---------------------------------------~  B  ~ S(B)=1G. |  *
*  | . ~~~~~~~                                       ~~~~~~~        . |  *
*  | .     |     T(A->C)<=20ms; B(A->C)>=10M         ~~~~~~~        . |  *
*  | .     +-----------------------------------------~  C  ~ S(C)=2G. |  *
*  | .                                               ~~~~~~~        . |  *
*  | ................................................................ |  *
*  +--------------------------------+---------------------------------+  *
*                                   |                                    *
*  +--------------------------------v---------------------------------+  *
*  |Split Service Profile into Domains                                |  *
*  |.................................. ...............................|  *
*  |.    Domain 1                   . .Domain 2                      .|  *
*  |.               T(A->D)<=2ms   . .    T(D->B)<=8ms   S(B)=1G    .|  *
*  |.      ~~~~~~~  B(A->D)>=10M ~~~~~~~  B(D->B)>=10M   ~~~~~~~    .|  *
*  |.      ~  A  ~ --------------~  D  ~-----------------~  B  ~    .|  *
*  |.      ~~~~~~~               ~~~~~~~                 ~~~~~~~    .|  *
*  |.          |    T(A->E)<=2ms   . .    T(E->C)<=18ms  S(C)=2G    .|  *
*  |.          |    B(A->E)>=10M ~~~~~~~  B(E->C)>=10M   ~~~~~~~    .|  *
*  |.          +-----------------~  E  ~-----------------~  C  ~    .|  *
*  |.                            ~~~~~~~                 ~~~~~~~    .|  *
*  |..................................................................|  *
*  +---------------------------------+--------------------------------+  *
*                                    |                                   *
*  +---------------------------------v---------------------------------+ *
*  |Select Specific Implementation Technologies                        | *
*  |    .............................   ............................   | *
*  |    .Domain 1                   .   .Domain 2                  .   | *
*  |    .        Flex-E             .   .         VPN+NFV          .   | *
*  |    .............................   ............................   | *
```

```
*  +---------------+----------------------------------+----------------+ *
*                  |                                  |                  *
*  +---------------v----------------------------------v----------------+ *
*  |Map to Selected Technologies                                       | *
*  +---------+------------------------+--------------------+-----------+ *
**************************************************************************
             |                        |                    |
    *********v***********    ********v********  **********v*********
    * Flex-E Controller *    * VPN Controller *  * NFV Orchestrator *
    *********+***********    ********+********  **********+*********
             |                        |                    |
    *********v***********    *********v*******************v*********
    *  Physical/Logical *    *            Physical/Logical          *
    *  Resources inside *    *            Resources inside          *
    *  Domain 1         *    *            Domain 2                  *
    *********************    ****************************************
```

Figure 2: Advanced Architecture of COMS

5. Integration with NFV

This section details the integration of the NFV framework [NFV-MANO] in the COMS architecture.

Network slice providers aim to accommodate a myriad of use cases and application scenarios from multiple tenants over a common network infrastructure. To that end, network slice providers build up multiple network slice instances (NSIs), each customized to serve the specific service demands of a particular tenant. An NSI is a logical self-contained network instance that network slice providers offer to a tenant, and that a tenant can consume. Although NSIs may span across multiple network segments (e.g., RAN, transport, and core network), this document only considers the transport network domain.

NFV may play a key role in network slicing, enabling its realization in a cost-efficient manner. Using the flexibility and virtualization capabilities that the NFV framework brings, a network slice provider can create and operate multiple NSIs over a common shared network infrastructure with isolation guarantees in terms of performance, management, security, and privacy [Ordonez-Network-Slicing]. To provide the tenant with the required performance and functionality, an NSI includes one or more network services, each consisting of a chained set of compassable atomic units called virtualized network functions (VNFs). These VNFs are software-based implementations of network functions that rely on computing, storage, and connectivity resources for their execution and communication. To simultaneously serve the requirements of multiple NSIs, the network slice provider makes use of the resources that are at its disposal, and efficiently orchestrate them across NSIs. Although the network slice provider can own these resources, we consider it rents them from one or more infrastructure owners following the Infrastructure-as-a-Service (IaaS) paradigm. In this case, the network slice provider takes the role of a network infrastructure tenant. Note that each of the three actors presented here (network infrastructure owner, network slice provider, and network slice tenant) defines a different administrative domain.

The NSIs shown in Figure 3 run parallel on a common shared transport network infrastructure. The transport network infrastructure consists of connectivity resources that may span across multiple administrative domains (i.e., different network infrastructure owners). These resources include WAN nodes and links providing reachability across geographically remote data centers, where the VNFs from different NSIs run. In particular, they connect together the network connectivity endpoints (e.g., gateways) of those data centers.

To simultaneously serve the connectivity needs of the NSIs using resources within its administrative domain, each network infrastructure owner has a WAN Infrastructure Manager (WIM). The WIM is a NFV functional block that performs control-management actions over the underlying connectivity resources to deploy and operate a number of L2/L3 virtual topologies with different levels of abstractions. To enforces the connectivity required by an NSI, the WIM abstracts the resources under its management, and creates a customized virtual topology that logically connects the data centers hosting the NSI’s VNFs. The resources of each data center are managed with a Virtual Infrastructure Manager (VIM). This NFV functional block play a similar role to the WIM, but extending their management domain to computing and storage resources.

The transport network resources, managed by the underlying network infrastructure owners using their WIMs/VIMs are delivered to the network slice provider logically placed on top of them. The network slice provider makes use of these resources to deploy and operate the NSIs that are under its management. For this end, it may rely on the NFV Orchestrator (NFVO) functionality. According to the NFV framework, NFVO is a functional block with two well-defined functionalities: resource orchestration and network service orchestration. The former focuses on orchestrating network infrastructure resources across multiple VIMs/WIMs, while the latter performs lifecycle management operations (e.g., instantiation, scaling, updating, termination, etc.) over the network service(s) built using those resources. Due to the different scope of these two set of functions, the NFVO may be logically split into two functional blocks: Resource Orchestrator and Network Service Orchestrator.

```
       ^        +---------------------------------------------------------+
       |        |             Cross-Segment Slice Manager                 |
       |        +-------------------------^-------------------------------+
       |                                  |
       |              +-------------------v------------------------+
       |              |    Transport Network Slice Orchestrator    <------+
       |              +----------------------------------^-----^---+      |
       |                                                 |     |          |
       |           +------------------------------------ | ----+------+   |
       |           |                 NSI M               |            |   |
       |        +--------------------------------------- | --------+  |   |
       |        |                 NSI 1                  |         |  |   |
                | +---------------------------+      +---v---+     |  |   |
    Network     | |   Tenant SDN Controller   <------>  OSS  |     |  |   |
    Slice       | +--^-------^--------^----+--+      +---^---+     |  |   |
    Provider    |    |       |        |    |             |         |  |   |
    Domain      | +--v--+ +--v--+     | +--v--+   +------v-------+ |  |   |
                | | VNF | | VNF |  .. | | VNF |   |   Network    | |  |   |
       |        | +--^--+ +--^--+     | +--^--+   |   Service    | |  |   |
       |        |    |       |        |    |      | Orchestrator | |  |   |
       |        |    | +-----+--------v--+ |      +------^-------+ |  |   |
       |        |    +-> vSwitch/vRouter <-+             |         |-++   |
       |        |      +-----------------+               |         | |    |
       |        +--------------------------------------- | --------+ |    |
       |                                                 |           |    |
       |      +------------------------------------------v-----------v--+ |
       +      |                    Resource Orchestrator                <-+
              +--------^-------------------^-------------------^--------+
   +-------+           |                   |                   |
                 +-----v-----+       +-----v-----+       +-----v-----+
       +         |    WIM    |  ...  |    VIM    |  ...  |    WIM    |
       |         +-----+-----+       +-----+-----+       +-----+-----+
                       |                   |                   |
   Network     +-------+------+   +-----------------+   +------+-------+
Infrastructure | Connectivity |   | +-------------+ |   | Connectivity |
    Owner      |   Resources  |   | |  Computing/ | |   |   Resources  |
    Domain     +--------------+   | |   Storage/  | |   +--------------+
                                  | |Connectivity | |
       |                          | |  Resources  | |
       |                          | +-------------+ |
       |                          |   Data Center   |
       v                          +-----------------+
```

Figure 3: Integration of NFV framework in COMS architecture

To orchestrate the resources that are at its disposal (those provided by the underlying network infrastructure owners), the network slice provider has a single Resource Orchestrator. The main role of the

Resource Orchestrator is to dispatch this finite set of resources across the operative NSIs in an optimal way, with the aim of simultaneously satisfying their (potentially diverging) performance requirements. To bring multiplexing gains and cost savings in this task, the Resource Orchestrator may take advantage of resource sharing. Resource sharing introduces flexibility and efficiency in slice provisioning, as network slice provider’s resources can be dynamically allocated and released across NSIs according to the time-varying resource requirements that their tenants impose. This approach requires an adequate resource management framework for the Resource Orchestrator that carefully finds an optimal solution, enabling resource sharing among NSIs when necessary, while preserving their performance isolation.

As shown in Figure 3, each of the operative NSIs serving a network slice tenant comprises a tenant SDN controller, a Network Service Orchestrator, and an Operation Support System (OSS). On the one hand, the tenant SDN controller configures the VNFs at application level, and chains them to dynamically build up the network service(s) that are required in the NSI. For VNF configuration management, the tenant SDN controller uses southbound configuration protocols such as NETCONF. For VNF chaining management, it leverages the networking capabilities provided by virtual switches/routers, sending them appropriate forwarding instructions using southbound control protocols such as OpenFlow. On the other hand, the Network Service Orchestrator manages the lifecycle of the network service(s). Finally, the OSS performs the intra-NSI management, bridging the gap between the Network Service Orchestrator and the tenant SDN controller, and coordinating their operations and management data. The OSS is also the entry point of the NSI, providing management capability exposure to external blocks. By way of example, the network slice tenant can use the OSS to gain access to the NSI and operate it at its convenience.

The description given above focuses on run-time phase, assuming the NSIs are operative, and omitting the deployment steps referred in Section 1. To trigger the deployment of a network slice, the network slice provider needs other functional blocks. These functional blocks include a Cross-Segment Slice Manager, and one or more Network Slice Domain Orchestrators. The Cross-Segment Slice Manager receives a network slice service profile from the tenant. This profile contains the (end-to-end) slice requirements. The Cross-Segment Slice Manager decompose these requirements into one or more network slice domain slice requirements, and send them to the respective Network Slice Domain Orchestrators (e.g., RAN Slice Orchestrator, Transport Network Slice Orchestrator, Core Network Slice Orchestrator). Since the architecture discussed in this document is assumed to be inside the transport network domain, we only consider the Network Slice Transport Orchestrator. The Network Slice Transport Orchestrator uses the network slice transport requirements to determine which VNFs and network service(s) are required, and what are their resource requirements. Once checked the Resource Orchestrator can provision them, the steps for deploying the slice begin. First, the Resource Orchestrator creates the resource slice. Then, the OSS takes over the resource slice and configures it, resulting in a networking slice. Finally, the OSS (assisted by the Network Service Orchestrator and the Tenant SDN controller), instantiates one or more network services (and their constituent VNFs) over this networking slice to realize a service slice, making it usable for the network slice tenant.

6. Security Considerations

There is no security problems introduced by this document.

7. IANA Considerations

There is no IANA action required by this document.

8. Acknowledgements

TBD

9. Informative References

[COMS-PS] "Problem Statement of Supervised Heterogeneous Network Slicing", <https://datatracker.ietf.org/doc/draft-geng-coms-problem-statement/>.

[draft-arkko-arch-virtualization] "Considerations on Network Virtualization and Slicing", <https://tools.ietf.org/html/draft-arkko-arch-virtualization-00>.

[I-D.boucadair-connectivity-provisioning-protocol] Boucadair, M., Jacquenet, C., Zhang, D., and P. Georgatsos, "Connectivity Provisioning Negotiation Protocol (CPNP)", draft-boucadair-connectivity-provisioning-protocol-15 (work in progress), December 2017.

[NFV-MANO] ETSI GS NFV-MAN 001, "Network Functions Virtualisation (NFV); Virtual Network Functions Architecture", V1.1.1, December 2014.

[Ordonez-Network-Slicing]
Ordonez-Lucena, J., Ameigeiras, P., Lopez, D., Ramos-Munoz, J., Lorca, J., and J. Folgueira, "Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges", IEEE Communications Magazine, vol. 55, no. 5, pp. 80-87, May 2017.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, DOI 10.17487/RFC5440, March 2009, <https://www.rfc-editor.org/info/rfc5440>.

[Stitching-Data]
"Gateway Function for Network Slicing", <https://datatracker.ietf.org/doc/draft-homma-coms-slice-gateway/>.

[Stitching-Management]
"Interconnecting (or Stitching) Network Slice Subnets", <https://datatracker.ietf.org/doc/draft-defoy-coms-subnet-interconnection/>.

Authors' Addresses

Liang Geng
China Mobile

Email: gengliang@chinamobile.com

Li Qiang
Huawei

Email: qiangli3@huawei.com

Jose Ordonez Lucena
University of Granada

Email: jordonez@ugr.es

Oscar Adamuz Hinojosa
University of Granada

Email: oadamuz@ugr.es

Pablo Ameigeiras
University of Granada

Email: pameigeiras@ugr.es

Diego Lopez
Telefonica I+D

Email: diego.r.lopez@telefonica.com

Luis Miguel Contreras Murillo
Telefonica

Email: luismiguel.contrerasmurillo@telefonica.com

none S. Homma
Internet-Draft NTT
Intended status: Informational X. de Foy
Expires: September 6, 2018 InterDigital Inc.
A. Galis
University College London
March 5, 2018

Gateway Function for Network Slicing
draft-homma-coms-slice-gateway-01

Abstract

This document describes the roles and requirements for a slice gateway that is a data plane function or function group for connecting/disconnecting and compose/decompose network slice subnets and providing network slices from end to end. The interworkings between management and control elements at the management and control planes with the gateway function for controlling and orchestrating end-to-end network slices are also presented in this document.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 6, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

Network slicing is an approach to create separate virtual networks in support of service depending on several requirements on the same physical resources, and it enables networks to adapt to requirements, which is diverse more, inexpensively and flexibly. It's also expected to enhance usability of infrastructural networks for tenants and create new business opportunities. For example, by using network slices lent from infrastructure operators, other industrial companies can provide communication services including ensurance of network transport without having physical infrastructure.

From a business point of view, a slice includes a combination of all the relevant network resources, functions, and assets required to fulfill a specific business case or service, including OSS, BSS and DevOps processes.

From the network infrastructure point of view, network slice requires the partitioning and assignment of a set of resources that can be used in an isolated, disjunctive or non- disjunctive manner for that slice.

From the tenant point of view, network slice provides different capabilities, specifically in terms of their management and control capabilities, and how much of them the network service provider hands over to the slice tenant. As such there are two kinds of slices: (A) Inner slices, understood as the partitions used for internal services of the provider, retaining full control and management of them. (B) Outer slices, being those partitions hosting customer services, appearing to the customer as dedicated networks.

Network slices are established with combination of various technologies, such as software defined network (SDN), network function virtualization (NFV), or traffic engineering, and managed/ operated with automation technologies such as orchestrator.

Assumed use cases of network slices include establishment of virtual networks whose qualities are guaranteed from end to end under the supervision of multi-domain orchstrators. In such cases, a network slice subnet is created on each domain, such as access network and core network, and such network slices are composed of connected subnets.

Network slice subnets are built based on specification of the underlay network, and thus the used technologies might vary. Therefore, a gateway function, which enables to connect subnets while adapting the differentiations and forward data packets to/from the appropriate next subnet, is required. Defining a new data plane technology is not a goal of this draft. This draft aims to specify management-related requirements for an SLG, which may be implemented using existing data plane technologies.

In this document, the gateway function is called slice gateway or SLG, and the role and requirements are described.

2. Definition of Terms

Network Slicing: Network slicing is a technology or an approach to create separate virtual networks in support of services, depending on several requirements, on the same physical resources. This is possible by combinations of several network technologies.

Network Slice (NS): An NS is a virtual network established on network infrastructure. Some include additional network functions such as firewall or load-balancer in addition to basically forwarding functions such as switches or routers. It has an overlay architecture and is independent from the underlay network’s topology.

NS Subnet: An NS subnet is partially virtual network established within a single domain.

End-to-End Network Slice (E2E-NS): An E2E-NS is a virtual network connecting between end points. E2E slices are composed of a single NS subnet or multiple NS subnets.

Network Slice as a Service (NSaaS): An NSaaS is a NS distribution model in which a third-party provider hosts NSs and makes them available to customers.

Network Slice Tenant (NS Tenant): An NS tenant is a person or group that rents and occupies NSs from NS providers.

Domain: A domain is a group of a network and devices administrated as a unit with common rules and procedures.

Administrative Domain: An administrative domain is a group of networks and devices managed by an administrator.

Resource: A resource is element used to create virtual networks. There are several types of resources, i.e., connectivity, computing and storage.

Network Function Virtualization (NFV): NFV is the concept or technologies to provide dedicated network appliances as software.

Software Defined Network (SDN): SDN is the concept or technologies to separate network control plane from data plane, and control network devices dynamically and flexibly.

Virtual Network: A virtual network is a network running a number of virtual network functions.

Virtual Network Function (VNF): A virtual network function (VNF) is a network function whose functional software is decoupled from hardware. One or more virtual machines running different software and processes on top of industry-standard high-volume servers, switches and storage, or cloud computing infrastructure, and capable of implementing network functions traditionally implemented via custom hardware appliances and middleboxes (e.g., router, NAT, firewall, load balancer, etc.)

Slice Gateway Function (SLG): An SLG is a function or a group of functions to connect/disconnect NS subnets. The role is described in the following sections.

Business Support System and Operation Support System (BSS/OSS): BSS/OSS are systems to support service providing and operation of network devices.

Orchestrator: Orchestrator is an entity to operate network components automatically. There are several types of orchestrators including NFV Orchestrator (NFVO) or service orchestrator defined by ETSI NFV and Open Source MANO (OSM) ([NFV-Architectural-Framework] and [OSM-White-Paper]).

SLG Controller (SLG-Ctrl): An SLG-Ctrl is an entity that controls SLGs. An SLG-Ctrl is controlled by upper-level operation systems such as OSS/BSS or orchestrator.

3. Motivations and Roles of SLG

SLG main role is the enablement of interworkings between data plane with management and control elements for controlling and orchestrating end-to-end slices.

Use cases of network slices are discussed in several Standard Developing Organizations (SDOs). Some examples are described in use cases document ([I-D.netslices-usecases]).

In some proposed use cases, an NS is structured across multiple network domains. The capability of NS subnets might be different because the components are domain-specific. In particular, the differentiation in capability between different administrative domains is large.

For connecting some different NS subnets and providing a NS that guarantees the prescribed quality from end to end, SLGs are required to connect such NS subnets. SLGs enable to provide E2E-NSs independently of specifications of underlay networks by hiding the differentiations and connecting between NS subnets. An overview of this concept is shown in Figure 1. SLGs glue NS subnets established on each domain and provide an E2E-NS.

```
                                   E2E-NS
             ________________________A_________________________
            /                                                  \
              _____________ ____________      _________________
             /            //           /     /                 /
    end    +---+   NS   +---+   NS   +---+ +---+   NS   ,------.
    host==>|SLG| Subnet |SLG| Subnet |SLG+-+SLG| Subnet( Server )
           +---+   #1   +---+   #2   +---+ +---+   #3   `------'
           /____________//___________/     /________________/
           /____________//___________/     /________________/
                 :              :                  :
                 :              :                  :
               .--.           .--.               .--.
             (    )-.       (    )-.           (    )-.
           .' Access '    .' Core    '       .'  Data  '
           ( Network   )  ( Network   )      (  Center   )
            (        -'    (        -'        ( /Cloud -'
             '-(      )     '-(      )         '-(      )
                '---'          '---'              '---'

         \_______ _______/ \______ ______/    \_________ _________/
              V               V                    V
            Domain#1        Domain#2             Domain#3

         \_____________ _____________/    \_________ _________/
                        V                          V
           Domain of Administrator#A     Domain of Administrator#B
```

Figure 1: E2E-NS composed of multiple NS subnets

Moreover, identification of user service traffic and their allocation/disallocation to the appropriate NS subnet are required at the edges of E2E-NSs, as shown in Figure 2, and SLGs might take on these roles.

```
                +-----+    _______________
      end       |     |-->/_______________
      host ====>| SLG |      NS Subnet#1
                |@Edge|    _______________
                |     |-->/______________
                |     |      NS Subnet#2
                |     | :        :
                +-----+
```

Figure 2: NS subnet selection of SLG

Note that, this model has the assumption that transitions of data packets from one NS subnet to another are executed at only SLGs. Also, an SLG is not necessarily implemented as a single device or virtual machine (VM).

4. Architecture Overview of NS Management

The architecture overview of NS management system is shown in Figure 3. Orchestrators manage whole resources including network elements and server resources (i.e., routing, bandwidth, compute or storage). In this figure, the resources including network elements and server resources are managed by resource orchestrators installed in each domain, and the E2E-orchestrator and network service orchestrator handle resource orchestrators.

NSs are requested from NS tenants via the portal system and the order of creations of an NS is given to the E2E orchestrator from the portal system via BSS/OSS. When an NS across multiple administrative domains are requested, the portal system that received the request forwards the order to create NS subnets to the other infrastructure providers’ systems via Cross-Segment Slice Manager. The details of COMS architecture are described in the architecture document ([I-D.qiang-coms-architecture]).

SLGs are also controlled via orchestrators. An SLG basically belongs to a network element, and it might also belong to server resource if it runs as a VNF. (An example of position of SLG deployed as a VNF is shown in Appendix A.)

SLGs are located at the edges of each NS subnet. They translate data packets into the appropriate form and send them to the next NS subnet. SLGs located at the end of E2E-NSs additionally provide identification of data packets and select the assigned NS subnet based on the identification result.

The information model used in this architecture is described in information model document ([I-D.qiang-coms-netslicing-information-model]).

```
     NS Tenant
       |
    . .|. . . . . . . . . . . . . . . . . . . . .
   . +-v---------+                               .
   . |Portal/GUI +--+                            .
   . +-+---------+  |                            .   . . . . . . . . .
   .   |          +-v-------------------------+    .  . +------------+  .
   .   |          |CSS-Mngr./TNS-Orch.      |<------->|CSS-M/TNS-O |  .
   .   |          +-+-------+---------------+    .  . +-+--------+-+  .
   .   |            |       |                    .  .   |        |    .
   . +-v-----+      |       |                    .  . +-v-----+  |    .
   . |BSS/OSS|<-----+       |                    .  . |BSS/OSS|  |    .
   . +-+-----+              |                    .  . +-+-----+  |    .
   .   |                    |                    .  .   |        |    .
   . +-v-------------------v-----------+        .  . +-v--------v-+  .
   . |  E2E-Orch./Network Service-Orch. |        .  . |E2E-O/NS-O  |  .
   . +-+-------------------+-----------+        .  . +-+--------+-+  .
   .   |                   |                     .  .   |        |    .
   . +-v----------------+ +-v----------------+   .  .       :         .
   . | Resource Orch.#1 | | Resource Orch.#2 |.. .   . . . . . . . . .
   . +-+---------+------+ +-+---------+------+   .   Administrative
   .   |         |          |         |          .   Domain#2
   . +-v------++-v------+ +-v------++-v------+   .
   . |Network ||NFV     | |Network ||NFV     |   .
   . |Ctrl.   ||Ctrl.   | |Ctrl.   ||Ctrl.   |   .
   . +-+------++-+------+ +-+------++-+------+   .
   .   |         |          |         |          .
   . +-v------++-v------+ +-v------++-v------+   .
   . |Network ||Server  | |Network ||Server  |   .
   . |Elements||Resource| |Elements||Resource|.. .
   . |in      ||in      | |in      ||in      |   .
   . |Domain#1||Domain#1| |Domain#2||Domain#2|   .
   . +--------++--------+ +--------++--------+   .
    . . . . . . . . . . . . . . . . . . . . . . .
     Administrative Domain#1

                    CSS-Mngr./CSS-M:Cross-Segment Slice Manager
                    TNS-Orch./TNS-O:Transport Network Slice Orchestrator
```

Figure 3: Overview of NS Management Architecture

5. Requirements for SLG

An SLG is basically a component in the data plane and has the roles of data packet processing. Moreover, it is required to have functions for control/management processes such as connecting to underlay networks or managing NSs.

Furthermore, an SLG might be required to support handling services provided on NSs in addition to controlling of NS because an SLG is an edge node on an E2E-NS.

In this section, we describe the requirements for an SLG in terms of the following aspects and their interworkings.

1. Data plane for NSs as infrastructure

2. Control/management plane for NSs as infrastructure

3. Data plane for services on NSs

4. Control/management plane for services on NSs

5.1. Management of NS as Infrastructure

5.1.1. Data Plane Aspect

5.1.1.1. Identification/Classification

SLGs at the edge of E2E-NSs MUST have the capability to identify and classify data packets, and assign them to the appropriate E2E-NS. This requirement varies depending on the location.

Fixed Access: An SLG MUST identify and classify data packet with access point, including CPE or WiFi-AP, or subscriber ID such as VLAN-ID. Moreover, in some services, an SLG should identify and classify data packets based on user device or application used in the communication.

Mobile Access: An SLG MUST identify and classify data packet with subscriber-ID such as IMSI, radio-wave bandwidth, or identifier of tunnels. Moreover, in some services, an SLG should identify and classify data packets based on application used in the communication or location of the user equipment (UE).

Between NS subnets: An SLG MUST identify and classify data packet based on the tunnel-ID or virtual routing and forwarding (VRF) that received the packets. If specific slice identifier such as a value mapped in the metadata field of the IP header is used; an SLG should identify and classify data packets with the ID.

5.1.1.2. Transporting/Forwarding

SLGs MUST provide functions for transport data packets depending on the specifications of the underlay networks.

Encapsulation/Decapsulation/Tagging: In network slicing, duplication of IP addresses of user packets between NSs MUST be accepted, thus, using techniques that enable separation of a network logically is preferred. In short, some tunnel protocols or tagging approaches should be used as transport of NSs. For this reason, SLG MUST support encapsulation or tagging of data packets based on the specification of the underlay network. Also, SLG MUST support the packets' decapsulation or untagging. Examples of tunnel protocols and tags that can be used for creating NSs on L2/L3 segments are described below.

L2 Segment: VLAN, MPLS, Segment Routing MPLS (SR-MPLS), PPPoE, etc.

L3 Segment: GRE, L2TP, GTP-U, VxLAN, IPv6 Segment Routing (SRv6), etc.

VxLAN, SR-MPLS, and SRv6 are described in their specification documents ([RFC7348], [I-D.ietf-spring-segment-routing-mpls], and [I-D.ietf-6man-segment-routing-header]).

Translation of Encapsulation/Tagging Form: SLG MUST support to translate tunnel header or tag of received packets to the appropriate tunnel header or tag when it forwards data packets to the next NS subnet that has different transport capability.

Distribution of Traffic: Some NSs have multiple route between the same end points within the same NS subnet because of traffic engineering, switching to a redundant path, or other reasons, and SLG MAY forward data packets with the appropriate route based on some trigger information. An example of the overview of this requirement is shown in Figure 4. In this figure, there are two routes, main and sub, between SLGs, and an SLG switches forwarding route depending on the network situation such as congestion occurrence on the current route.

```
              ____________________________
             /        . . . . .          /
       +-----+     .           .   +-----+
       |     |. .               . .|     |
       | SLG |                     | SLG |
       |     |* *               * *|     |
       +-----+   *           *     +-----+
        /         * * * * *         /
       /___________________________/
         NS Subnet
                              *** : Main-route
                              ... : Sub-route
```

Figure 4: An example of traffic distribution by SLG

5.1.1.3. Isolation between NSs

In NSaaS, isolation control is required for avoiding an NS being affect by other NSs. Traffic engineering or QoS control is ones of the most fundamental approaches to prevent disturbances between NSs.

Traffic Shaping/Policing: An SLG MUST execute traffic shaping and policing at its egress and ingress ports to avoid an NS using excessive traffic bandwidth.

Quality of service (QoS) Control: If there is an order of priority between NSs on the same underlay infrastructure, an SLG should remark the appropriate QoS parameter of the outer-most header of each packet following the preconfigured setting and provide packet scheduling based on the QoS parameter for providing priority control. The field that SLG refers may vary depending on the specification of the underlay network. For example, COS value is remarked in L2 segments; on the other hand, DSCP value is remarked in L3 segments.

5.1.1.4. Service Chaining as Infrastructural Mechanism(*Optional)

If an SLG is composed of a combination of several components, a service chaining mechanism is required to make them work together and achieve SLG functionality.

Moreover, some NSs may traverse NFVs such as firewalls or cache servers for providing value-added services to their users. In such cases, SLG might be required to support service chaining mechanisms, such as handling of network service header (NSH) defined in [RFC8300]. If an NS includes the service chaining architecture defined in [RFC7665], some SLG would be required to support following functions; classifier(CF), service function forwarder (SFF), and inter boundary node(IBN). (Details of CF, SFF and IBN are described in SFC documents; [RFC7665], [I-D.ietf-sfc-hierarchical].)

5.1.2. Control/Management Planes Aspects

5.1.2.1. Interfaces to Controllers or Operation Systems

SLG MUST have interface to its controller or operation systems for set parameters related to the data plane functions described in Section 5.1.1. In addition, an SLG at the edges of E2E-NSs MUST have interfaces to authentication servers.

5.1.2.2. Address Resolution/Routing

An SLG MUST support address resolution or routing mechanisms to connect to underlay network elements including routers or L2 switches.

5.1.2.3. Authentication Authorization Accounting (AAA)

For preventing entry of irregular traffic to NSs, an SLG at the edge of E2E-NS MUST support AAA mechanism for incoming traffic. Also, when an SLG connects to another SLG in other administrative domain, SLGs should have a mechanism to confirm that the connection is established with the regular processes. For example, an SLG is required to support authentication of the opponent SLG with key information indicated from higher-level operation systems.

5.1.2.4. Operation Administration and Maintenance(OAM)

In management of NSs, OAM or monitoring mechanisms for both underlay and overlay networks is required for SLGs. For an underlay network, an SLG MUST have OAM functions to confirm connectivity to interconnect equipment. For an overlay network, an SLG MUST have OAM functions to confirm connectivity to the some node on the same NS, and measure the traffic amount of flowing packets on each NS.

5.2. Management of Services on NS (*Optional)

5.2.1. Data Plane Aspect

5.2.1.1. Identification/Classification

In NSaaS, some NS tenants may need delivery of an individual service to each user, device, or application on the same NS. For such service deliveries, an SLG might be required to identify and classify user traffic based on some information such as subscriber ID or payload of data packets. Also, an SLG should be controllable from the NS tenant.

5.2.1.2. QoS Control

An NS accommodates several communication devices and SLGs might be required to have fair queueing mechanisms for maintaining service quality of each user. Also, different types of service traffic that have different priorities might coexist on an NS. For example, some NS providers might provide telephone and internet access services to their users with an NS. In such cases, SLG might be required to provide QoS control mechanisms for enforcing priority control based on service priorities.

These QoS controls are executed depending on the information of inner packets and are independent of isolation mechanisms as infrastructure. An SLG might be required to have a hierarchical QoS control mechanism in case that both QoS controls for services over NSs and isolation between NSs are required.

5.2.1.3. Steering/Service Chaining(Cooperation with VNFs)

SLG might be required to support steering or service chaining function for conveying data packets to the appropriate network functions deployed on an NS based on the classification result and user’s contract information.

5.2.2. Control/Management Planes Aspects

5.2.2.1. Interfaces to Service Management Systems

An SLG might have interfaces to controllers for managing user policies on each NS. Some controllers might be deployed on the same NS. If some controllers are located at external networks, they might require SLGs to have APIs.

5.2.2.2. Collection of Telemetry information

In an NSaaS, collection of telemetry information of each NS might be required for understanding traffic usage. Thus, an SLG might be required to support to collect and report telemetry information of connected NSs.

6. Deployment of SLG

This section describes considerations related with deployment of SLGs.

6.1. Examples of Components Required to Maintain SLG Functions

For providing E2E-NSs on existing network infrastructures, some components located at boundaries of domains are required to have the same set of functionality as an SLG. Examples of such components in each domain type are described below.

Fixed Network: CPE/HGW, Service Edge, Gateway Router, etc.

Mobile Network: User Equipment, Radio-AP, eNodeB, S/P-GW ([LTE-Specs]), etc.

Data Center: Gateway Router, L2 switch, ToR switch, Server, etc.

6.2. SLG Types Depending on Locations on NS

There are mainly three types of SLG for creating E2E-NS across multiple administrative domains. The requirements of each SLG type are listed in Appendix B.

6.2.1. Edge SLG(E-SLG)

This is located at an edge of an E2E-NS, and supports identification, classification and authentication of user traffic in addition to fundamental SLG functions, such as transport and isolation. Also, it might be required to have capabilities for services delivered on an NS.

6.2.2. Inter-Subnet SLG(IS-SLG)

This is located between NS subnets within a single administrative domain and has only fundamental functions. It is not necessarily required if a common transport mechanism in all domains is used.

6.2.3. Inter-Domain SLG(ID-SLG)

This is located between NS subnets established on different domains. It supports authentication for connecting to the opponent SLG in addition to fundamental functions.

6.3. Horizontal Connection

The connection form of an SLG varies depending on which type it is. Examples of horizontal connection forms of each SLG type are described below.

E-SLG: An E-SLG accommodates several hosts and NS subnets. This has a forwarding table of end hosts and insert their packets to the appropriate NS subnet. An overview of this connection is shown in Figure 5.

```
   *Virtual Layer*

                 +-----+
     host#1 ====>|     |    _______________
                 |     |-->/_______________
     host#2 ====>|E-SLG|     NS Subnet#1
                 |     |    _______________
     host#3 ====>|     |-->/_______________
                 |     |     NS Subnet#2
      :     :    |     | :        :
                 +-----+

   ////////////////////////////////////////
   *Physical Layer*

                     ,--------------------
     [UE#1] -----\  /
     [UE#2] -----[Edge]    Domain#1
     [UE#3] -----/  \
      :       :      `-------------------

   Edge: Edge Node
```

Figure 5: Overview of horizontal connection of E-SLG

IS-SLG: An IS-SLG has the role of mediator between NS subnets and passes packets received from an NS subnet to the next one. If transport methods used in each domain are different, the IS-SLG translate packet form to the appropriate one. An overview of this connection is shown in Figure 6.

Virtual Layer

```
                   +------+
       _________   |      |     ___________
      _________/-->|IS-SLG|--> /__________
     NS Subnet#1   |      |    NS Subnet#2
                   +------+

   ///////////////////////////////////////
   *Physical Layer*

    --------------.        ,--------------
                   \      /
       Domain#1    [  GW  ]     Domain#2
                   /      \
    --------------'        '--------------
```

GW: Gateway Node

Figure 6: Overview of horizontal connection of IS-SLG

ID-SLG: An ID-SLG passes data packets to another ID-SLG located on a different administrative domain. Some tunnel established between them in advance may be used for the passing of packets. An overview of this connection is shown in Figure 7.

Virtual Layer

```
                   +------+        +------+
       _________   |      | ______ |      |    ___________
      _________/-->|ID-SLG|O______)|ID-SLG|-->/__________
     NS Subnet#1   |      | Tunnel |      |   NS Subnet#2
                   +------+        +------+

   ///////////////////////////////////////////////////////
   *Physical Layer*

    --------------------.          ,-------------------
        Administrative   \        /      Administrative
        Domain#1        [ GW ]---[ GW ]    Domain#2
                         /        \
    --------------------'          '-------------------
```

GW: Gateway Node

Figure 7: Overview of horizontal connection of ID-SLG

6.4. Vertical Connection

There are two patterns of vertical connection of SLGs in the middle of E2E-NSs. The first pattern is that the SLGs accommodate only a set of NS subnets, which are composition of the same E2E-NS. In this pattern, such SLGs are not required to support NS subnet selection, however, establishment of a new SLG is required when a new E2E-NS is created. This might causes extra overheads because of deploying many SLGs.

The other pattern is that such SLGs are acceptable to accommodate multiple NS subnets from each domain. In this pattern, SLGs are support NS subnet selection. On the other hand, this pattern can restrain the number of SLGs. Also, it is easy to provide transit of data packets from an NS subnet to other subnet on the same domain.

The overviews of these patterns are shown in Figure 8 and Figure 9.

```
                   +-----+
       _________   |     |   ___________
      _________/-->|SLG#1|-->/__________
     NS Subnet#1   |     |   NS Subnet#2
                   +-----+
                   +-----+
       _________   |     |   ___________
      _________/-->|SLG#2|-->/__________
     NS Subnet#3   |     |   NS Subnet#4
                   +-----+
          :           :             :
```

Figure 8: Overview of vertical connection of SLG: Separated Pattern

```
                   +-----+
       _________   |     |   ___________
      _________/-->|     |-->/__________
     NS Subnet#1   |SLG#1|   NS Subnet#2
       _________   |     |   ___________
      _________/-->|     |-->/__________
     NS Subnet#3   |     |   NS Subnet#4
                   |     |
          :        |     |         :
                   +-----+
```

Figure 9: Overview of vertical connection of SLG: Shared Pattern

6.5. Software vs. Hardware

An SLG can be created as either a software or hardware function. NSs are virtual networks created depending on requests from external NS tenants, and thus software would be more compatible with usage for NSs in terms of flexibility or manageability. Moreover, it enables to increase or decrease for each function if SLG is composed of combination of several components. However, it is difficult to provide high performance or sufficient throughput for carrier-grade networks with software function. In addition, it would be difficult to implement sufficient QoS control mechanisms with general servers, because they requires special hardware structures.

On the other hand, hardware appliances are able to provide high throughput compared with software. However, they are inflexible in terms of provisioning.

From the above considerations, operators should prepare SLG in appropriate ways depending on their usages or locations.

7. Interconnection between NS subnets

SLG provides interconnectivity between NS subnets. The concept and fundamental framework including the related NS information model are described in subnets interconnection document ([I-D.defoy-coms-subnet-interconnection]).

This section is focused on interconnection between NS subnets established on different administrative domains, and describes considerations related to this condition.

7.1. Pre-arrangement of transport protocols

For interconnection between different administrative NS subnets, pre-arrangement of the transport protocol, which is used to connect between SLGs is required. Orchestration systems indicate the protocol and configuration to each SLG.

7.2. Quality Assurance between SLGs

In addition to establishing connection, quality control of communication is important. SLGs of egress side should execute traffic shaping to prevent some NSs from excessively occupying the link between SLGs. Moreover, some SLGs are connected to several other SLGs that are deployed on the different locations. Therefore SLGs of the ingress side should execute traffic policing to avoid excessive inflow of traffic into some NSs. The parameters for these controls are pre-configured by orchestration systems.

The above approaches are ones of the simplest ways to provide quality assurance of inter-administrative subnets. If there is stricter isolation request, more considerations would be required.

7.3. Secure Interconnection

For connecting networks of different administrators, secure interconnection schemes are required. Especially, in an NSaaS, networks might be connected to several networks, and schemes for ensuring secure connectivity would be more important.

SLGs confirm whether the opponent SLG is regular when it requests to connect, and reject the request if the SLG is not regular. In some cases, SLGs might be confirm whether the inner packets received from the other SLGs are sent from regular users.

8. Security Considerations

Requirements and considerations for SLG related to security are described in Section 5 and Section 7.

9. IANA Considerations

This memo includes no request to IANA.

10. Acknowledgement

The authors would like to thank Li Qiang for her kind review and valuable feedback.

11. Informative References

[I-D.defoy-coms-subnet-interconnection]
Foy, X., Rahman, A., Galis, A., kiran.makhijani@huawei.com, k., and L. Qiang, "Interconnecting (or Stitching) Network Slice Subnets", draft-defoy-coms-subnet-interconnection-01 (work in progress), October 2017.

[I-D.ietf-6man-segment-routing-header]
Previdi, S., Filsfils, C., Raza, K., Dukes, D., Leddy, J., Field, B., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Matsushima, S., Leung, I., Linkova, J., Aries, E., Kosugi, T., Vyncke, E., Lebrun, D., Steinberg, D., and R. Raszuk, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-08 (work in progress), January 2018.

[I-D.ietf-sfc-hierarchical]
Dolson, D., Homma, S., Lopez, D., and M. Boucadair, "Hierarchical Service Function Chaining (hSFC)", draft-ietf-sfc-hierarchical-05 (work in progress), November 2017.

[I-D.ietf-spring-segment-routing-mpls]
Filsfils, C., Previdi, S., Bashandy, A., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-11 (work in progress), October 2017.

[I-D.netslices-usecases]
kiran.makhijani@huawei.com, k., Qin, J., Ravindran, R., 67, 4., Qiang, L., Peng, S., Foy, X., Rahman, A., Galis, A., and G. Fioccola, "Network Slicing Use Cases: Network Customization and Differentiated Services", draft-netslices-usecases-02 (work in progress), October 2017.

[I-D.qiang-coms-netslicing-information-model]
Qiang, L., Galis, A., 67, 4., kiran.makhijani@huawei.com, k., Martinez-Julia, P., Flinck, H., and X. Foy, "Technology Independent Information Model for Network Slicing", draft-qiang-coms-netslicing-information-model-01 (work in progress), October 2017.

[LTE-Specs]
3rd Generation Partnership Project (3GPP), "3GPP TS 36.300", December 2007, <http://www.qtc.jp/3GPP/Specs/36300-830.pdf>.

[NFV-Architectural-Framework]
Network Functions Virtualisation (NFV) ETSI Industry Specification Group (ISG), "Network Functions Virtualisation (NFV); Architectural Framework", December 2014, <http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf>.

[OSM-White-Paper]
ETSI, "OSM White Paper", October 2016, <https://osm.etsi.org/images/OSM-Whitepaper-TechContent-ReleaseONE-FINAL.pdf>.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <https://www.rfc-editor.org/info/rfc7348>.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <https://www.rfc-editor.org/info/rfc7665>.

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

Appendix A. Position of SLG on ETSI NFV MANO

The mapping of SLG as a VM into ETSI NFV MANO architecture is described in Figure 10.

```
+---------------------------+
| BSS/OSS, Orchestrator     |
+-+---------------+---------+
  |               |
+-v------+      +-v---------+
|SLG-Ctrl|      | NFV Orch. |
+-+------+      +-+---------+
  |               |
,-v------.        |
|  SLG   |        |
:========:      +-v---------+
|  VM    |<-----+ VNF Mngr. |
'--------'      +-+---------+
+--------+        |
|HostOS/ |      +-v---------+
|Server  |<-----| VIM       |
+--------+      +-----------+
```

Figure 10: Position of SLG as a VM on ETSI NFV MANO

Appendix B. Requirements for each SLG Type

The requirements for each SLG type are listed in Figure 11.

	E-SLG	IS-SLG	ID-SLG	Reference
*Data-Plane of NS as Infrastructure				
Identification/ Classification	M	O	O	Section 5.1.1.1.
Transport/ Forwarding	M	O	M	Section 5.1.1.2.
Isolation	M	M	M	Section 5.1.1.3.
Service Chain	O	O	O	Section 5.1.1.4.
*Control/Management-Plane of NS as Infrastructure				
IF to Ctrl/OpS	M	M	M	Section 5.1.2.1.
Addr Resolution /Routing	M	M	M	Section 5.1.2.2.
AAA	M	-	M	Section 5.1.2.3.
OAM	M	M	M	Section 5.1.2.4.
*Data-Plane for Service on NS				
Identification/ Classification	O	-	O	Section 5.2.1.1.
QoS Control	O	O	O	Section 5.2.1.2.
Steering/ Service Chain	O	-	O	Section 5.2.1.3.
*Control/Management-Plane for Service on NS				
IF to Service Manager	O	O	O	Section 5.2.2.1.
Telemetory	O	O	O	Section 5.2.2.2.

M: Mandatry, O: Optional, - : Not Required

Figure 11: List of Requirements for each SLG

Authors' Addresses

Shunsuke Homma
NTT, Corp.
3-9-11, Midori-cho
Musashino-shi, Tokyo 180-8585
Japan

Email: homma.shunsuke@lab.ntt.co.jp

Xavier de Foy
InterDigital Inc.
1000 Sherbrooke West
Montreal
Canada

Email: Xavier.Defoy@InterDigital.com

Alex Galis
University College London
Torrington Place
London WC1E 7JE
United Kingdom

Email: a.galis@ucl.ac.uk

none L. Qiang
Internet-Draft Huawei
Intended status: Informational A. Galis
Expires: July 30, 2018 University College London
L. Geng
China Mobile
K. Makhijani
Huawei
P. Martinez-Julia
NICT
H. Flinck
Nokia
X. de Foy
InterDigital Inc.
January 26, 2018

Technology Independent Information Model for Network Slicing
draft-qiang-coms-netslicing-information-model-02

Abstract

This document provides a technology independent information model for transport network slicing.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 30, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

Network slicing is a tool to share network resources and to offer customized network architectures for diverse use cases that share the same underlying infrastructure [NGMN-NS-Framework]. Customers may not be familiar with underlying networking technologies, and therefore may prefer to interface with network slices in a technology-agnostic way. On the other hand, service providers may have multiple candidate technologies for supporting network slicing. As shown in Figure 1, there is a gap between technology-agnostic network slicing service requirements and specific implementation technologies, that needs to be filled by a technology independent information model. Such a technology independent information model describes the entities that compose a network slice, their properties, attributes and operations, and the way they relate to each other of an end to end network slice that may span across multiple technology domains. It is independent of any specific repository, software usage, protocol, or platform, hence supports common operations and management of network slices.

```
     +-----------+               +-----------+
     |NS Tenant/ +---------------+NS Provider|
     |Customer   |   Service     |           |
     +-----------+   Model       +------+----+
                                        |
                                        |Service Delivery
                                        |Model
                                        |
                     +------------------+---------------+
                     |        NS Management System      |
                     |                                  |
                     |   ****************************   +----------+
                     |   *Technology Independent NS *   |          |
                     |   *    Information Model     *   |       Device
                     |   ****************************   |Configuration
                     |                                  |        Model
                     +--------+-------------------+-----+          |
                              |                   |                |
                           Network Configuration Model             |
     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|~~~~~~~~~~~~~~~~|~~~
      Available NS Activation |Technologies       |                |
                              |                   |                |
      +-----------------------+-+    +------------+------------+   |
      |Controller/Orchestrator/ |    |Controller/Orchestrator/ |   |
      |Manager of Implementation|    |Manager of Implementation|   |
      |Technology A             |    |Technology B             |  .|...
      +---------------+---------+    +-------------+-----------+   |
                      |                            |               |
                      | Device Configuration Model |               |
                      |                            |               |
      +---------------+----------------------------+---------------+-+
      |                                                              |
      |             Underlying Network Resources/Functions           |
      |                                                              |
      +--------------------------------------------------------------+
```

Figure 1: Technology Independent NS Information Model

mapping to specific technology is out of scope.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Other network slicing related terminology used in this document are interpreted as description in [COMS-PS].

3. Network Slice Tree Structure

The YANG data modeling language [RFC7950] will be used to represent the transport network slicedata model. Moreover,the data model for network topologies developed in [draft-ietf-i2rs-yang-network-topo] will be used as a base.

The proposed NS information model includes the following elements: connectivity resources, storage resources, compute resources, service instance based on predefined function blocks, network slice level attributes, etc. It is presented as a tree structure of attributes. The Yang language is used to represent the network slice information model. The following tree shows an overview of the tree structure. New attributes proposed in this draft are in the "netslice:" namespace, while other attributes are defined in [draft-ietf-i2rs-yang-network-topo].

```
module: ietf-network
+--rw networks
   +--rw network* [network-id]
      +--rw network-id                         network-id
      +--rw network-types
      +--rw supporting-network* [network-ref]
      |  +--rw network-ref
      +--rw node* [node-id]
      |  +--...
      |  +--rw netslice:compute-unit* [compute-unit-ref]
      |  |  +--rw netslice:compute-unit-ref    compute-unit-ref
      |  +--rw netslice:storage-unit* [storage-unit-ref]
      |  |  +--rw netslice:storage-unit-ref    storage-unit-ref
      |  +--rw netslice:service-instance* [service-instance-ref]
      |     +--rw netslice:service-instance-ref    service-instance-ref
      +--rw nt:link* [link-id]
      |  +--...
      |  +--rw netslice:link-qos
      |     +--...
      +--rw netslice:compute-unit* [compute-unit-id]
      |  +--...
      +--rw netslice:storage-unit* [storage-unit-id]
      |  +--...
      +--rw netslice:service-instance* [service-instance-id]
      |  +--...
      +--rw netslice:slice-level-attributes
         +--...
```

3.1. resources

Basic resources are used to construct a network slice. Resources comprise: nodes, links, compute units and storage units.

Different resources can exist independently, they can also be bound together when necessary. For example, bind a storage unit to a connectivity node.

A whole network attribute can represent a network slice instance. The network slice instance "supporting network" list can include underlying networks which are used to implement the network slice.

In this model, nodes and links will represent virtual nodes and links exposed to the slice user.

The network-id attribute will represent a network slice instance ID.

3.1.1. nodes

```
     +--rw node* [node-id]
     |  +--rw node-id                       node-id
     |  +--rw supporting-node* [network-ref node-ref]
     |  |  +--rw network-ref
     |  |  +--rw node-ref
     |  +--rw nt:termination-point* [tp-id]
     |  |  +--rw nt:tp-id                            tp-id
     |  |  +--rw nt:supporting-termination-point*
     |  |                             [network-ref node-ref tp-ref]
     |  |  |  +--rw nt:network-ref
     |  |  |  +--rw nt:node-ref
     |  |  |  +--rw nt:tp-ref
     |  |  +--rw netslice:packet-rate?               int64
     |  |  +--rw netslice:packet-loss-probability?   int64
     |  |  +--rw netslice:packet-loss-threshold?     int64
     |  |  +--rw netslice:received-packets?          int64
     |  |  +--rw netslice:sent-packets?              int64
     |  +--rw netslice:compute-unit* [compute-unit-ref]
     |  |  +--rw netslice:compute-unit-ref    compute-unit-ref
     |  +--rw netslice:storage-unit* [storage-unit-ref]
     |  |  +--rw netslice:storage-unit-ref    storage-unit-ref
     |  +--rw netslice:service-instance* [service-instance-ref]
     |     +--rw netslice:service-instance-ref    service-instance-ref
```

Nodes are defined in [draft-ietf-i2rs-yang-network-topo].

Nodes are augmented with the following attributes, that used to represent requirements, configuration and statistics associated with a termination point:

packet-rate: the packet forwarding capability of a port for this node in the unit of pps (packet per second).

packet-loss-probability: a statistical value which reflects the probability of packet loss.

packet-loss-threshold: a threshold of the packet loss probability. If the value of packet-loss-probability is larger than packet-loss-threshold, should actively notify the management system.

received-packets: a statistical value which reflects the number of received packets in a period of time.

sent-packets: a statistical value which reflects the number of sent packets in a period of time.

3.1.2. links

```
 +--rw nt:link* [link-id]
 |  +--rw nt:link-id            link-id
 |  +--rw nt:source
 |  |  +--rw nt:source-node?
 |  |  +--rw nt:source-tp?
 |  +--rw nt:destination
 |  |  +--rw nt:dest-node?
 |  |  +--rw nt:dest-tp?
 |  +--rw nt:supporting-link* [network-ref link-ref]
 |  |  +--rw nt:network-ref
 |  |  +--rw nt:link-ref
 |  +--rw netslice:link-qos
 |     +--rw netslice:link-bandwidth-agreement?    int64
 |     +--rw netslice:link-throughput?             int64
 |     +--rw netslice:link-throughput-threshold?   int64
 |     +--rw netslice:link-latency-agreement?      int64
 |     +--rw netslice:link-latency?                int64
 |     +--rw netslice:link-jitter-agreement?       int64
 |     +--rw netslice:link-jitter?                 int64
 |     +--rw netslice:link-jitter-threshold?       int64
 |     +--rw netslice:mandatory-node* [node-ref]
 |     |  +--rw netslice:node-ref    node-ref
 |     +--rw netslice:mandatory-link* [link-ref]
 |     |  +--rw netslice:link-ref    link-ref
 |     +--rw netslice:excluded-node* [node-ref]
 |     |  +--rw netslice:node-ref    node-ref
 |     +--rw netslice:excluded-link* [link-ref]
 |        +--rw netslice:link-ref    link-ref
```

Links are defined in [draft-ietf-i2rs-yang-network-topo].

Links are associated with nodes through termination points placed under nodes. Links are augmented with QoS information as follows:

link-bandwidth-agreement: specify the bandwidth requirement for this link. If this parameter does not be set specifically, then the link will be constructed according to the default bandwidth value provided by management plane.

link-throughput: the current throughput of this link.

link-throughput-threshold: a threshold for link throughput. If the value of link-throughput is smaller than link-throughput-threshold, should actively notify the management system.

link-latency-agreement: specify the latency requirement for this link. If this parameter does not be set specifically, then the link will be constructed according to the default latency agreement provided by management plane.

link-latency: the current latency of this link.

link-jitter-agreement: specify the jitter requirement for this link. If this parameter does not be set specifically, then the link will be constructed according to the default jitter agreement provided by management plane.

link-jitter: the current jitter of this link.

link-jitter-threshold: a threshold for link jitter. If the value of link-jitter is larger than link-jitter-threshold, should actively notify the management system.

mandatory-node/link: a list of underlying nodes/links that must be passed by the mapped physical path of this link.

exclusive-node/link: a list of underlying nodes/links that cannot be traversed by the mapped physical path of this link.

3.1.3. storage-units

```
+--rw netslice:storage-unit* [storage-unit-id]
|  +--rw netslice:storage-unit-id         inet:uri
|  +--rw netslice:size?                   int64
|  +--rw netslice:access-rate             int32
|  +--rw netslice:access-mode?       access-qualifier
|  +--rw netslice:read-write-mode-type?   read-write-mode-type
|  +--rw netslice:redundancy-type?        redundancy-type
|  +--rw netslice:location?               string
```

size: size of the storage unit in MB.

access-rate: the minimum rate to write/read 8KB files into/from the storage unit.

access-mode: there are two options include public or dedicated.

read-write-mode: there are two options include read only, and read & write.

redundancy-type: there are four options include best efforts (i.e, no redundancy), n+1 (n storage units with one extra backup), 2n (each storage unit has one backup), 2n+1 (n storage units with n+1 extra backup).

location: a string describing the location of the storage unit.

3.1.4. compute-units

```
+--rw netslice:compute-unit* [compute-unit-id]
|  +--rw netslice:compute-unit-id    inet:uri
|  +--rw netslice:num-cores?         int8
|  +--rw netslice:ram?               int64
|  +--rw netslice:access-mode?       access-mode-type
|  +--rw netslice:location?          string
|  +--rw netslice:unit-type          compute-unit-type
```

num-cores: the number of arithmetic logic unit.

ram: RAM in bytes.

access-mode: there are two options include shared or dedicated.

location: a string describing the location of the compute unit.

unit-type: two types of compute unit include GPU or CPU

3.2. generalized-function-block

```
      +--rw netslice:service-instance* [service-instance-id]
      |  +--rw netslice:service-instance-id           inet:uri
      |  +--rw netslice:domain-agent
      |  |  +--rw netslice:agent-name?                string
      |  |  +--rw netslice:sb-ip-address?             string
      |  |  +--rw netslice:sb-port?                   string
      |  |  +--rw netslice:nb-ip-address              string
      |  |  +--rw netslice:nb-port?                   string
      |  +--rw netslice:load-balancer [element-id]
      |  |  +--rw element-id                          inet:uri
      |  |  +--rw nt:termination-point* [tp-id]
      |  |  |  +--rw nt:tp-id                         tp-id
      |  |  |  +--rw nt:supporting-termination-point*
      |  |  |                     [network-ref node-ref tp-ref]
      |  |  |  |  +--rw nt:network-ref
      |  |  |  |  +--rw nt:node-ref
      |  |  |  |  +--rw nt:tp-ref
      |  |  |  +--rw netslice:packet-rate?               int64
      |  |  |  +--rw netslice:packet-loss-probability?   int64
      |  |  |  +--rw netslice:packet-loss-threshold?     int64
      |  |  |  +--rw netslice:received-packets?          int64
      |  |  |  +--rw netslice:sent-packets?              int64
      |  |  +--rw netslice:lb-name?      string
      |  |  +--rw netslice:ip-address?   string
      |  |  +--rw netslice:port?         string
```

Some general features could be packaged into function blocks in advance, such as agent, firewall, load balancer, etc.

3.3. slice-level-attributes

```
      +--rw netslice:slice-level-attributes
         +--rw netslice:service-time-start? yang:date-and-time
         +--rw netslice:service-time-end?   yang:date-and-time
         +--rw netslice:lifecycle-status?   lifecycle-status-type
         +--rw netslice:access-control
         |  +--rw netslice:match?      string
         |  +--rw netslice:action?     string
         |  +--rw netslice:priority?   string
         |  +--rw netslice:counter?    int64
         +--rw netslice:reliability-level?  reliability-level-type
         +--rw netslice:resource-reservation-level?
                            resource-reservation-level-type
         +--rw netslice:availability?                 int64
         +--rw netslice:availability-threshold?       string
```

The slice-level-attributes refers to a set of attributes applicable to a network slice. Some explanations are provided as follows for easy going:

service-time-start/end: specify the time during which the network slice service exists (e.g., three months, one year).

lifecycle-status: specify the status of the network slice, there are four enumeration values: construction, modification, activation and deletion.

access-control: illustrates each role can take what kind of operations on the network slice.

reliability-level: the ability of a network slice to be in a stable state. In this document, the main method to achieve reliability is "backup". If necessary, other methods also can be extended based on the current definition. The detailed definition of Reliability_Level is provided in Table 1.

resource-reservation-level: classify different resource reservation levels of a network slice. This attribute is related to the slice isolation but is not strictly bound. The detailed definition is provided in Table 2.

availability: a statistical value which reflects the probability for a network slice instance to work with expected SLA in a period of time (e.g., 99.999% of time).

availability-threshold: a threshold of the availability. If the value of Availability is smaller than Availability_Threshold, should actively notify the management system.

Value	Explanation	Note
none	No specific reliability requirement	The lowest reliability level
path-backup	Each path has a backup path	Path reliability
logical-backup	Each node/link has a backup node/link	Logical resource reliability
physical-backup	Each node/link has a backup node/link, and the primary and backup nodes/links must be mapped to different physical devices/paths (the mapped two physical paths couldn't have any shared device)	Physical resource reliability

Table 1: Explanation of reliability-level

Value	Explanation	Note
none	No specific resource reservation requirement	The lowest resource reservation level, the network slice instance will share and compete for resource with other network slice instances
shared-non-preemptive	A certain of resource reservation, the free reserved resources could be used by other slice instances, and uable to be retrieved if other slice instances are usring them	Shared and non-preemptive
shared-preemptive	More stringent resouce reservation, the free reserved resources could be used by other slice instances, and will be retrieved if the network slice needs them	Shared and preemptive
exclusive	The reserved resources couldn't be used by other slice instances, even if these resources are free	The highest resource reservartion level, exclusive

Table 2: Explanation of resource-reservation-level

4. Operations

The defined information model should be able to support the following operations on network slices. Except for support the operations on a complete network slice, each element insides a network slice also should be able to be operated specifically.

o construct: construct a network slice

- o delete: delete a network slice

- o modify: modify a constructed network slice

- o set_element_value: set the value of an indicated element in a network slice

- o get_element_value: get the value of an indicated element in a network slice

- o monitor: monitor the status of a network slice

- o enable_report: enable the active report to the subscribes/ management system when the monitored status changes beyond expectation

5. Yang Module

```
 <CODE BEGINS> module ietf-coms-core {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-coms-core";
  prefix netslice;

  import ietf-yang-types { prefix "yang"; }
  import ietf-inet-types { prefix inet; }
  import ietf-network { prefix nd;  }
  import ietf-network-topology { prefix lnk;  }

  organization
    "IETF";

  contact
    "Editors:    X. de Foy, Cristina QIANG
               <mailto:>";

  description
    "This module contains a collection of YANG definitions for COMS.

    Copyright (c) 2016 IETF Trust and the persons identified as
    authors of the code.  All rights reserved.

    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject
    to the license terms contained in, the Simplified BSD License
    set forth in Section 4.c of the IETF Trust's Legal Provisions
    Relating to IETF Documents
    (http://trustee.ietf.org/license-info).
```

```
    This version of this YANG module is part of
    draft-...;
    see the RFC itself for full legal notices.";

  revision "2018-01-26" {
    description
      "Initial revision of COMS topology.";
    reference
      "draft-qiang-coms-netslicing-information-model-02";
  }

  /*
    Types
  */

  typedef read-write-mode-type {
    type enumeration {
      enum read-write {
        description "R/W";
      }
      enum read-only {
        description "R/O";
      }
    }
    description "Indicates if entity is read-write,
    read-only, etc.";
  }

  typedef access-mode {
    type enumeration {
      enum access-mode-public {
        description "Underlying storage can be
        shared with other instances";
      }
      enum access-mode-dedicated {
        description "Underlying storage is not
        shared with other instances";
      }
    }
    description "access-mode";
  }

  typedef compute-unit-type {
    type enumeration {
      enum compute-unit-cpu {
        description "Underlying compute unit is CPU based";
      }
      enum compute-unit-gpu {
```

```
        description "Underlying compute unit is GPU based";
      }
    }
    description "compute-unit-type";
  }

  typedef lifecycle-status-type {
    type enumeration {
      enum construction {
        description "construction";
      }
      enum modification {
        description "modification";
      }
      enum activation  {
        description "activation";
      }
      enum deletion {
        description "deletion";
      }
    }
    description "Lifecycle status";
  }

  typedef resource-reservation-level-type {
    type enumeration {
      enum none {
        description "No specific reliability requirement";
      }
      enum shared-non-preemptive {
        description "Each path has a backup path";
      }
      enum shared-preemptive  {
        description "Each node/link has a backup node/link";
      }
      enum exclusive {
        description "Each node/link has a backup node/link,
        mapped to different physical devices/paths";
      }
    }
    description "Resource reservation level";
  }

  typedef reliability-level-type {
    type enumeration {
      enum none {
        description "No specific reliability requirement";
      }
```

```
      enum path-backup {
        description "Each path has a backup path";
      }
      enum logical-backup  {
        description "Each node/link has a backup node/link";
      }
      enum physical-backup {
        description "Each node/link has a backup node/link,
        mapped to different physical devices/paths";
      }
    }
    description "Reliability level";
  }

  typedef redundancy-type {
    type enumeration {
      enum none {
        description "no redundancy";
      }
      enum n+1 {
        description "n storage units with one extra backup";
      }
      enum 2n {
        description "each storage unit has one backup";
      }
      enum 2n+1 {
        description "n storage units with n+1 extra backup";
      }
    }
    description "Redundancy type";
  }

  typedef node-ref {
    type instance-identifier;
    description "A reference to a node";
  }

  typedef link-ref {
    type instance-identifier;
    description "A reference to a link";
  }

  typedef compute-unit-ref {
    type instance-identifier;
    description "A reference to a compute unit";
  }

  typedef storage-unit-ref {
```

```
    type instance-identifier;
    description "A reference to a storage unit";
  }

  typedef service-instance-ref {
    type instance-identifier;
    description "A reference to a service instance";
  }

  grouping rule {
    description "Access Control Rule";
    leaf match{
      type string;
      description "Match";
    }
    leaf action{
      type string;
      description "Action";
    }
    leaf priority{
      type string;
      description "Priority";
    }
    leaf counter{
      type int64;
      description "Counter";
    }
  }

  grouping port-config {
    description "Configuration of a port/connection point";
    leaf packet-rate {
      type int64;
      description "Data rate in packets per seconds";
    }
    leaf packet-loss-probability {
      type int64;
      description "Packet loss probability (actual type is TBD)";
    }
    leaf packet-loss-threshold {
      type int64;
      description "Packet loss probability threshold to alert
      management system (actual type is TBD)";
    }
  }

  grouping port-stats {
    description "Statistics of a port/connection point";
```

```
    leaf received-packets {
      type int64;
      description "Total number of packets received";
    }
    leaf sent-packets {
      type int64;
      description "Total number of packets sent";
    }
  }

  grouping storage-unit-specs {
    description "Storage unit specs";
    leaf size {
      type int64;
      description "storage size in MB";
    }
    leaf access-rate {
      type int32;
      description "lower limit of storage access rate";
    }
    leaf access-mode {
      type access-mode;
      description "access-mode";
    }
    leaf read-write-mode-type {
      type read-write-mode-type;
      description "Read and write mode";
    }
    leaf redundancy-type {
      type redundancy-type;
      description "Redundancy type";
    }
  }

  grouping storage-unit-desc {
    description "Storage unit description";
    leaf storage-unit-id {
      type inet:uri;
      description "storage-unit ID";
    }
    uses storage-unit-specs;
    leaf location {
      type string;
      description "Location hint";
    }
  }

  grouping compute-unit-specs {
```

```
    description "Compute unit specs";
    leaf num-cores {
      type int8;
      description "Number of CPU Cores";
    }
    leaf ram {
      type int64;
      description "RAM in bytes";
    }
    leaf access-mode {
      type access-mode-type;
      description "access mode";
    }
  }

  grouping compute-unit-desc {
    description "Compute unit description";
    leaf compute-unit-id {
      type inet:uri;
      description "storage-unit ID";
    }
    uses compute-unit-specs;
    leaf location {
      type string;
      description "Location hint";
    }
    leaf unit-type {
      type compute-unit-type;
      description "specify the category of compute unit";
    }
  }

  grouping path-restrictions {
    description "Physical path restriction type: nodes and
    links of underlying networks
    that must or must not be traversed by a link";
    list mandatory-node {
      key "node-ref";
      description "List of mandatory nodes";
      leaf node-ref {
        type node-ref;
        description "Node";
      }
    }
    list mandatory-link {
      key "link-ref";
      description "List of mandatory links";
      leaf link-ref {
```

```
        type link-ref;
        description "Link";
      }
    }
    list excluded-node {
      key "node-ref";
      description "List of excluded nodes";
      leaf node-ref {
        type node-ref;
        description "Node";
      }
    }
    list excluded-link {
      key "link-ref";
      description "List of excluded links";
      leaf link-ref {
        type link-ref;
        description "Link";
      }
    }
  }

  grouping link-qos-desc {
    description "QoS associated with a link";
    leaf link-bandwidth-agreement {
      type int64;
      description "Link bandwidth agreement";
    }
    leaf link-throughput {
      type int64;
      description "Link throughput";
    }
    leaf link-throughput-threshold {
      type int64;
      description "Link throughput threshold";
    }
    leaf link-latency-agreement {
      type int64;
      description "Link latency agreement";
    }
    leaf link-latency {
      type int64;
      description "Link latency";
    }
    leaf link-jitter-agreement {
      type int64;
      description "Link jitter agreement";
    }
```

```
    leaf link-jitter {
      type int64;
      description "Link jitter";
    }
    leaf link-jitter-threshold {
      type int64;
      description "Link jitter threshold";
    }
    uses path-restrictions;
  }

  grouping slice-level-attributes {
    description "network slice level attributes";
    leaf service-time-start {
      type yang:date-and-time;
      description "Start of service";
    }
    leaf service-time-end {
      type yang:date-and-time;
      description "End of service";
    }
    leaf lifecycle-status {
      type lifecycle-status-type;
      description "Step in the slice lifecycle";
    }
    container access-control {
      uses rule;
      description "Control of access to operations per role";
    }
    leaf reliability-level {
      type reliability-level-type;
      description "Reliability level";
    }
    leaf resource-reservation-level {
      type resource-reservation-level-type;
      description "Resource reservation level";
    }
    leaf availability {
      type int64;
      description "Measure of probability to work with
      expected SLA (TBD: type should be expanded)";
    }
    leaf availability-threshold {
      type string;
      description "Availability threshold to actively
      notify the management system";
    }
  }
```

```
  grouping generalized-function-block {
    description "generalized function blocks that can be
    used to create an instance (more funcution blocks TBD)";

    container domain-agent {
      description "a network slice agent to receive manager request";
      leaf agent-name {
        type string;
        description "agent name";
      }
      leaf sb-ip-address {
        type string;
        description "IP Address of the server which for southbound protocols";
      }
      leaf sb-port {
        type string;
        description "Port of the server which for southbound protocols";
      }
      leaf nb-ip-address {
        type string;
        description "IP Address of the server which for northbound protocols";
      }
      leaf nb-port {
        type string;
        description "Port of the server which for northbound protocols";
      }
    }

    container load-balancer {
      description "load balancer (type TBD)";
      leaf element-id {
          type inet:uri;
          description "load balancer element id";
      }
      list termination-point {
        use  termination-point-desc;
      }

      leaf LB-name {
        type string;
        description "load balancer name";
      }
      leaf ip-address {
        type string;
        description "IP Address of the load balancer (type TBD)";
      }
      leaf port {
        type string;
```

```
        description "Port of the load balancer (type TBD)";
      }
    }
  }

  grouping termination-point-desc {
    description "Augment network nodes termination points with
    port information.";

    leaf tp-id {
      type tp-id;
      description
            "Termination point identifier.";
    }

    list supporting-termination-point {
      key "network-ref node-ref tp-ref";
      description
            "This list identifies any termination points that
           the termination point is dependent on, or maps onto.
           Those termination points will themselves be contained
           in a supporting node.
           This dependency information can be inferred from
           the dependencies between links.  For this reason,
           this item is not separately configurable.  Hence no
           corresponding constraint needs to be articulated.
           The corresponding information is simply provided by the
           implementing system.";
      leaf network-ref {
        type leafref {
          path
            "../../../nw:supporting-node/nw:network-ref";
        }
        description
             "This leaf identifies in which topology the
            supporting termination point is present.";
      }

      leaf node-ref {
        type leafref {
          path
            "../../../nw:supporting-node/nw:node-ref";
        }
        description
             "This leaf identifies in which node the supporting
            termination point is present.";
      }
```

```
      leaf tp-ref {
        type leafref {
          path
            "/nw:networks/nw:network[nw:network-id=current()/"
                  + "../network-ref]/nw:node[nw:node-id=current()/../"
                  + "node-ref]/termination-point/tp-id";
        }
        description
          "Reference to the underlay node, must be in a
             different topology";
      }
    }  // list supporting-termination-point
    uses port-config;
    uses port-stats;
  }

  grouping service-instance-desc {
    description "Service instance description. An instance
    is based on a predefined function block";
    leaf service-instance-id {
      type inet:uri;
      description "service instance ID";
    }
    uses generalized-function-block;
  }

  /*
    Model
  */

  augment "/nd:networks/nd:network" {
    description "Augment network nodes with slice information.";
    list compute-unit {
      key "compute-unit-id";
      description "Compute units";
      uses compute-unit-desc;
    }
    list storage-unit {
      key "storage-unit-id";
      description "Storage units";
      uses storage-unit-desc;
    }
    list service-instance {
      key "service-instance-id";
      description "Service instance";
      uses service-instance-desc;
    }
    container slice-level-attributes {
```

```
      description "Attributes that apply to a whole network slice";
      uses slice-level-attributes;
    }
  }

  augment "/nd:networks/nd:network/nd:node" {
    description "Augment network nodes with slice information.";
    list compute-unit {
      key "compute-unit-ref";
      description "List of compute units present in node";
      leaf compute-unit-ref {
        type compute-unit-ref;
        description "Compute unit present in node";
      }
    }
    list storage-unit {
      key "storage-unit-ref";
      description "List of storage units present in node";
      leaf storage-unit-ref {
        type storage-unit-ref;
        description "Storage unit present in node";
      }
    }
    list service-instance {
      key "service-instance-ref";
      description "an instance of a service provided by the node";
      leaf service-instance-ref {
        type service-instance-ref;
        description "Service instance present in node";
      }
    }
  }

  augment "/nd:networks/nd:network/nd:node/lnk:termination-point" {
    description "Augment network nodes termination points with
    port information.";
    uses port-config;
    uses port-stats;
  }

  augment "/nd:networks/nd:network/lnk:link" {
    description "Augment network links with slice information.";
    container link-qos {
      description "QoS specifications for this link";
      uses link-qos-desc;
    }
  }
}
```

<CODE ENDS>

6. Security Considerations

Each component of the network slice has its own security requirements.

7. IANA Considerations

There is no IANA action required by this document.

8. Acknowledgements

Authors would like to acknowledge Guangpeng Li for help coding.

9. References

9.1. Normative References

[draft-ietf-i2rs-yang-network-topo]
"i2rs-yang-network-topo", <https://www.ietf.org/id/draft-ietf-i2rs-yang-network-topo-17.txt>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

9.2. Informative References

[COMS-PS] "COMS Problem Statement", <https://www.ietf.org/id/draft-geng-coms-problem-statement-00.txt>.

[NGMN-NS-Framework]
"NGMN Network Slicing Framework", <https://www.ngmn.org/uploads/media/161010_NGMN_Network_Slicing_framework_v1.0.8.pdf>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

Authors' Addresses

Li Qiang
Huawei

Email: qiangli3@huawei.com

Alex Galis
University College London

Email: a.galis@ucl.ac.uk

Liang Geng
China Mobile

Email: gengliang@chinamobile.com

Kiran Makhijani
Huawei

Email: Kiran.Makhijani@huawei.com

Pedro Martinez-Julia
NICT

Email: pedro@nict.go.jp

Hannu Flinck
Nokia

Email: hannu.flinck@nokia.com

Xavier de Foy
InterDigital Inc.

Email: Xavier.DeFoy@InterDigital.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 6, 2018

L. Qiang
Huawei Technologies
L. Geng
China Mobile
K. Makhijani, ed
Huawei Technologies
X. de Foy
InterDigital Inc.
A. Galis
University College London

February 02, 2018

The Use Cases of Common Operation and Management of Network Slicing
draft-qiang-coms-use-cases-00

Abstract

The Common Operation and Management of network Slicing (COMS) intends to provide a comprehensive approach for the overall operation and management of network slicing in the scope if IETF. The system is designed in a hierarchical and inter-operative manner. COMS is capable of recursive adaption in a hierarchical network management system. It is also independent of data plane technologies used in different administrative domains. Both network slice operator and network slice tenant may benefit for COMS for the purpose of slice management and maitenance.The purpose of this document is to discuss the use cases of COMS in different views.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 6, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

Network Slicing is a mechanism which a network slice provider can use to allocate dedicated infrastructures and services from shared systems to a network slice tenant. COMS acts as a technology-independent and resource-centric approach according to which the operation and management of network slice can be performed.

This document lists the use cases of COMS from various OAM aspects of network slicing. It provides a general reference of how COMS may be used from both network slice provider and network slice tenant viewpoint. The COMS community (the proposed WG) will consider these use cases and decide which related technology is going to be investigated under the problem scope of COMS.

All of the use cases are introduced in this document followed by a brief analysis regarding the relationship with COMS. As the document is being continuously worked on, the list of use cases is as follows:

- Heterogeneous Resource Management for Network Slicing
- Interoperation between Multiple Slice-aware Administrative Domain
- End-to-end Orchestration of Network Slicing
- Customized OAM for Network Slice Tenant
- Interaction with 3GPP Network Slicing
- Network Slice FCAPS - to be specified in -01 version
- Network Slice Stiching and Recursion - to be specified in -01 version

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Heterogeneous Resource Management for Network Slicing

2.1. Use Case Introduction

Network slice is a specific partition of resources. The resources are deliberately associated together for the purpose of fulfilling both functional and performance requirements of various applications. Heterogeneity is the nature of those underlay resources based-on which network slices are created. In order to provide end-to-end orchestration of network slices, it is required that all resources are manageable by a network slice provider. COMS will be used as the fundamental technology for the purpose of heterogeneous resource management.

2.1.1. Combination of Networking and Computing

Networking used to be the absolute major asset and resources of a telecommunication service provider. As the rapid development of cloud computing and NFV technology in recent years, computing infrastructures such as data center, distributed edge cloud, CDN and cache facilities start to play more and more important roles. Nowadays, not only is the amount of data centers dramatically growing in service providers’ network, but also network/service functions are migrating to NFV deployment, which depends heavily on common computing and storage resources. An obvious trend of more interactive relationship between networking and computing resources (computing resource referred in this section also includes storage resources) deployment is seen worldwide.

The goal of network slicing is to provide a "turn-key" solution for vertical application provider, where certain performance and functional demands can be met according to specific SLAs. This is achieved by providing infrastructure and functional dedication to vertical application providers. It is expected that a vertical application provider, as a network slice tenant, will purchase a network slice which is equipped with both preferred connectivity topology and associated computing/storage resources. Hence, the vertical application provider is able to deploy whatever applications according to its preference.

Relying on the underlay network infrastructure, computing resource has become an inevitable part of the network slice. In general, it may comes in various forms in a manner of IaaS as follows.

- o Bare metal equipment with required specifications

- o Hypervisor-based virtual machine

- o Container-based infrastructures
- o Other customized type of presentation of computing resources

Under the regime of network slicing, computing (including storage) resources provided in any form above need to be specified with geographical or logical location information. These computing resources may distributed among the network slice topology as terminal or intermediate network nodes. This location information is essential for the purpose of associating these resources with connectivity components within a network slice.

It is not always easy to jointly manage both computing resources and the underlay networking. Connectivity is normally supervised by using traditional EMS of the connected devices, or by using more advanced SDN approaches for more advanced systems. In contrast, computing resources are typically managed by VIMs. A manager who understands both EMS/SDN controller and VIMs is requirement for overall orchestration of an end-to-end network slice.

2.1.2. Technology Diversity of Network Infrastructure

Due to architectural and commercial reasons, the underlay technology choices for different administrative domains are unlikely to be the same. For example, regional administrative domains may be favor of choosing single-vendor solutions for its backbone network. This minimize the complexity of intra-domain OAM. However, adjacent regional administrative domains may use equipments from different vendors. This makes the overall backbone network infrastructure resources heterogeneous. The technology diversity of the resource consisting a network slice mainly results from the following reasons.

- o Various technology choices for access, aggregation and backbone networks
- o Legacy equipments due to deployment iteration
- o Administrative concerns caused by geographical reasons
- o Vendor-specific technology for customized deployment

It is common for an end-to-end network slice asking for resources from various administrative domains with distinctive technology options. This include data plane, control plane and management plane technologies. COMS, as an management tool, can be used for operation and management of such systems.

2.1.3. Network and Service Functions Variety

A complete network slice may consist of many types of network ans service functions. This functions are likely to be deployed either in NFV or physical forms. In practice, virtualized network functions are managed by VNFM under MANO system, whilst physical network functions are managed by resource management system (RMS). Meanwhile, the management plane of service functions is even more diversified which may associated with extremely customized service management platforms.

In order to make network and service function usable and manageable as a part of network slice, it is necessary to have an overall management system on which the orchestration of such functions can rely. Existing technology such as SFC already provides a comprehensive solution for the purpose of service-level integration. It would be interesting to investigate how underlay network infrastructure can better serve and map with requirements of particular SFC or interconnection between SFCs under network slice regime. Such system should be capable of associating service function resources to required network infrastructure, making the formation of an end-to-end network slice possible.

2.2. Use Case Analysis

It is always preferred to have more diversified resources on which network slices can be built. Heterogeneity becomes an inevitable issue caused by this nature of variety. At present, countless management systems are being used by service providers for different types of resource domains. COMS may help to aggregate and coordinate the management plane of such systems and provide unified slice-level OAM.

3. Interoperation between Multiple Slice-aware Administrative Domain

3.1. Use Case Introduction

As mentioned in section 2, the slice orchestrator need to supervise heterogeneous resources in various administrative domains in response to diversified demand from the network slice tenants. For example, the network slice orchestrator needs to supervise some heterogeneous technology domains, which obviously have separated administrative systems. Examples include optical transport network, IP routing network in terms of network infrastructure and SFCs in terms of service function. Administrative domain may also isolated for technology-evolution reasons. For instance, the slice orchestrator is necessary to be compatible with either controller-based networks or EMS-based networks. Furthermore, as computing plays more and more significant role as infrastructure resource, the requirement of coordinating between networking and computing in management plane is obvious.

3.2. Use Case Analysis

Either it is a green field implementation or not, given the heterogeneity property of resources, the administrative domains can only be more diversified. Meshed interoperation between these administrative domains is infeasible. Hence, a higher level management entity is one of the most cost effective and straight forward solution.

4. End-to-end Orchestration of Network Slicing

4.1. Use Case Introduction

When a network slice tenant purchases a network slice service, it does not necessarily know the what underlay resources exactly are allocated for the purpose of the network slice creation. It is the network slice orchestrator who takes care of this process. As the network slice orchestrator receives network slice service delivery model from service provider’s OSS/BSS, it executes slice-level operation and management accordingly. End-to-end orchestration is an essential part of this process.

The main functionality of end-to-end network slice orchestration may include the following aspects:

1. Coordinating underlay network infrastructure and service function resources

2. Life-cycle management, which includes the common operation of network slice creation, activation/de-activation, modification, deletion and status monitoring.

3. Pre-defining templates of common types of network slices and provide repository for network slice instances created by templates or full customization

4.1.1. Resource Registration

In the process of end-to-end orchestration of network slice, resource registration is one of the fundamental prerequisite. The network slice orchestrator needs to know exactly what resources are available under the overall management. The information for resource registration may include the the following aspects:

- o The type of resources (whether it is a connectivity, computing, storage or pre-defined network/sevice function)
- o The physical/logical location of the resources
- o Data plane and control plane technology capabilities
- o Performance capabilities
- o Availability information
- o Domain topology information

The network slice orchestrator can only use registered resources in the process of network slice creation. Any change of resource information caused by equipment upgrading, new deployment or abolishing of legacy system need to be reported to the network slice orchestrator.

4.1.2. Life-cycle Management

It is important that the network slice orchestrator can continuously manage the creation, activation/de-activation, modification, deletion and status monitoring processes of the network slice for a complete life cycle. In general, a network slice profile can be created in several ways:

- o A network slice profile can be created according to the network slice templates. In this way, the network slice profile is create by direct configuration of the parameters in a pre-defined network slice template according to exciting index.
- o A network slice profile can be created by customized parameter index and value.

In both cases, the value of parameters come from the service delivery interface of the network slice orchestrator. Particularly for the latter case, a complete network slice profile is needed from the service delivery interface.

Additionally, the operation of life cycle management also comes from the OSS/BSS service delivery model. After receive such operation request, the orchestrator need to map certain them to different administrative domains respectively.

4.1.3. Network Slice Template and Repository

As mentioned in section 3.1.2, network slice orchestrator can use templates to create network slice profiles. Templates are extremely useful in cases where multiple network slice tenants require exact same type of network slices. For example, URLLC is regarded as one of the most popular scenario in 5G application. It would be useful to pre-define a URLLC network slice template, to which the network slice orchestrator can refer, upon request of network slice tenants.

A network slice repository make it handy to manage the templates of different types. It also helps to categorize different network slice profiles created under given templates. A category of "Customized network slice" might also be useful for the cases where network slice is created from scratch.

4.2. Use Case Analysis

End-to-end orchestration is the most essential functionality of network slicing management. COMS information model will act as a significant reference for resource registration, network slice template definition and and the creation of network slice profile. At the same time, life-cycle management will be enabled by the COMS service delivery model.

5. Customized OAM for Network Slice Tenant

5.1. Use Case Introduction

As a network slice instance is activated, the network slice tenant is able to access the network slice and apply intra-slice configuration under network slice provider’s policies. This include operation and management functionalities, which are likely to be a subset of the overall network slice management. Typical functionalities a network slice tenant may prefer to have include the following aspects:

1. Network slice life-cycle status monitoring

2. Performance dash board of individual/set of resource components in a network slice

3. Slice-level parameter adjustments under network slice providers’ policies, strictly avoiding conflicts with other network slices.

4. Slice subset operation and management based on COMS at network slice provider’s permission

5.2. Use Case Analysis

The network slice orchestrator has two NBI interfaces respectively. One of them is designed for the purpose of customized OAM. A network slice tenant may use this interface to perform the actions listed in section 5.1. COMS is in the position of defining the NBI interface

6. Interaction with 3GPP Network Slicing

6.1. Use Case Introduction

3GPP is the born-place of the concept of 5G network slicing. However in 3GPP, only radio access network and core network are considered as the resource pool for network slices. The transport network is modelled as a link between them. Technically in 3GPP language, network slicing does not include transport network.

In 5G, the requirements of network slicing focus on the guaranteed end-to-end quality in terms of Bandwidth (eMBBs), Latency (URLLC) and connections (eMTC). For the purpose of end-to-end network slicing is to provide guaranteed service for vertical user. Transport network will also play an important role in this scenario. One of the most straight forward solution for service-guaranteed mapping to the sliced 3GPP network is to make the TN also slice-aware.

As 3GPP SA5 delivers the performance requirements from 3GPP slice manager to IETF network slice orchestrator, the orchestrator will treat the requirements similarly to a general service delivery model received from OSS/BSS. It is not 3GPP’s concern weather IETF is using slice or not toe fulfill this requirements

6.2. Use Case Analysis

Network slicing is one of the key technology in 5G network. It is important that transport network can provide certain quality guarantee, so that the end-to-end network slice run over can fulfill the overall requirements. COMS provides NBI for the purpose of gathering transport network requirements. These requirements will be further broken down into underlay systems requirements accordingly, where COMS can help the mapping by providing the general information model.

7. Network Slice FCAPS

7.1. Use Case Introduction

This is a place holder for slice-level FCAPS use cases for COMS. It is due to be updated in 01 version of this document

7.2. Use Case Analysis

8. Network Slice Stiching and Recursion

8.1. Use Case Introduction

This is a place holder for inter-slice operation use cases for COMS. It is due to be updated in 01 version of this document

8.2. Use Case Analysis

9. IANA Considerations

This document makes no request of IANA.

10. Security Considerations

There is no security consideration in this draft.

11. Acknowledgements

The authors would like to acknowledge Benoit Claise, Warren Kumari and Adrian Farrel for valuable discussions.

12. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

Authors' Addresses

Li Qiang
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: qiangli3@huawei.com

Liang Geng
China Mobile
Beijing 100053
China

Email: gengliang@chinamobile.com

Kiran Makhijani
Huawei Technologies
2890 Central Expressway
Santa Clara CA 95050
USA

Email: kiran.makhijani@huawei.com

Xavier de Foy
InterDigital Inc.
1000 Sherbrooke West
Montreal
Canada

Email: Xavier.Defoy@InterDigital.com

Alex Galis
University College London
London
U.K.

Email: a.galis@ucl.ac.uk