Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 4, 2017

A. Galis
University College London
K. Makhijani
D. Yu
Huawei Technologies

October 31, 2016

Autonomic Slice Networking-Requirements and Reference Model
draft-galis-anima-autonomic-slice-networking-01

Abstract

This document describes the technical requirements and the related reference model for the intercommunication and coordination among devices in Autonomic Slicing Networking. The goal is to define how the various elements in a network slicing context work and orchestrate together, to describe their interfaces and relations. While the document is written as generally as possible, the initial solutions are limited to the chartered scope of the WG.

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 May 4, 2017.

Copyright Notice

Copyright (c) 2016 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

The document "Autonomic Networking - Definitions and Design Goals" [RFC7575] explains the fundamental concepts behind Autonomic Networking, and defines the relevant terms in this space, as well as a high level reference model. This document defines this reference model with more detail, to allow for functional and protocol specifications to be developed in an architecturally consistent, non-overlapping manner. While the document is written as generally as possible, the initial solutions are limited to the chartered scope of the WG.

Most networks will run with some autonomic functions for the full networks or for a group of nodes [RFC7576] or for a group of slice networks while the rest of the network is traditionally managed.

The goal of this document is to focus on the autonomic slicing networking. [RFC7575] is focusing on fully or partially autonomic nodes or networks.

The proposed revised ANIMA reference model allows for this hybrid approach across all such capabilities.

This is a living document and will evolve with the technical solutions developed in the ANIMA WG. Sections marked with (*) do not represent current charter items.

While this document must give a long term architectural view, not all functions will be standardized at the same time.

2. The Network Slicing Overall View

2.1. Context

Network Slicing is end-to-end concept covering the radio and non-radio networks inclusive of access, core and edge / enterprise networks. It enables the concurrent deployment of multiple logical, self-contained and independent shared or partitioned networks on a common infrastructure platform.

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

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

Examples of physical or virtual resources to be shared or partitioned would include: bandwidth on a network link, forwarding tables in a network element (switch, router), processing capacity of servers, processing capacity of network or network clouds elements. As such slice instances would contain:

(i) a combination/group of the above resources which can act as a network,

(ii) appropriate resource abstractions,

(iii) exposure of abstract resources towards service and management clients that are needed for the operation of slices

The establishment of slices is both business-driven (i.e. slices are in support for different types and service characteristics and business cases) and technology-driven as slice is a grouping of physical or virtual) resources (network, compute, storage) which can act as a sub network and/or a cloud. A slice can accommodate service components and network functions (physical or virtual) in all network segments: access, core and edge / enterprise networks.

A complete slice is composed of not only various network functions which are based on virtual machines at C-RAN and C-Core, but also transport network resources which can be assigned to the slice at radio access/transport network. Different future businesses require different throughput, delay and mobility, and some businesses need very high throughput or/and low delay.

2.2. High Level Requirements

Slice creation: management plane create virtual or physical network functions and connects them as appropriate and instantiate them in the slice.

The instance of slice management then takes over the management and operations of all the (virtualised) network functions and network programmability functions assigned to the slice, and (re-)configure them as appropriate to provide the end-to-end service.

A complete slice is composed of not only various network functions which are based on virtual machines at C-RAN and C-Core, but also transport network resources which can be assigned to the slice at radio access/transport network. Different future businesses require different throughput, delay and mobility, and some businesses need very high throughput or/and low delay. Transport network shall provide QoS isolation, flexible network operation and management, and improve network utilization among different business.

QoS Isolation: Although traditional VPN technology can provide physical network resource isolation across multiple network segments, it is deemed far less capable of supporting QoS hard isolation, Which means QoS isolation on forwarding plane requires better coordination with management plane.

Independent Management Plane: Like above, network isolation is not sufficient, a flexible and more importantly a management plane per instance is required to operate on a slice independently and autonomously within the constraints of resources allocated to the slice.

Another flexibility requirement is that an operator can deploy their new business application or a service in network slice with low cost and high speed, and ensure that it does not affect existing of business applications adversely.

Programmability: Operator not only can slice a common physical infrastructure into different logical networks to meet all kinds of new business requirements, but also can use SDN based technology to improve the overall network utilization. By providing a flexible programmable interface; the 3rd party can develop and deploy new network business rapidly. Further, if a network slicing can run with its own slice controller, this network slicing will get more granular control capability [I-D.ietf-anima-autonomic-control-plane] to retrieve slice status, and issuing slicing flow table, statistics fetch etc.

Life cycle self-management: It includes creation, operations, re-configuration, composition, decomposition, deletion of slices. It would be performed automatically, without human intervention and based on a governance configurable model of the operators. As such protocols for slice set-up /operations /(de)composition / deletion must also work completely automatically. Self-management (i.e. self-configuration, self-composition, self-monitoring, self-optimisation, self-elasticity) is carried as part of the slice protocol characterization.

Extensibility: Since the Autonomic Slice Networking Infrastructure is a relatively new concept, it is likely that changes in the way of operation will happen over time. As such new networking functions will be introduced later, which allow changes to the way the slices operate.

Transport network shall provide QoS isolation, flexible network operation and management, and improve network utilization among different business.

The flexibility behind the slice concept needs to address QoS guarantee on the transport network and enable network openness.

Other considerations and requirements: TBD.

2.3. Key Terms and Definitions

A number of slice definitions were used in the last 10 years in distributed and federated testbed research [GENI], future internet research [ChinaCom09] and more recently in the context of 5G research [NGMN], [ONF], [IMT2020], [NGS-3GPP].

A unified Slice definition usable in the context of Autonomic Networking consist of 4 components:

- Service Instance component,
- Network Slice Instance component,
- Resources component and
- Slice Capability exposure component

The Service Instance component represents the end-user service or business services which are to be supported. It is an instance of an end-user service or a business service that is realized within or by a Network Slice. Each service is represented by a Service Instance. Services and service instances would be provided by the network operator or by 3rd parties.

A Network Slice Instance component is represented by a set of network functions, and resources to run these network functions, forming a complete instantiated logical network to meet certain network characteristics required by the Service Instance(s). It provides the network characteristics which are required by a Service Instance. A Network Slice Instance may also be shared across multiple Service Instances provided by the network operator. The Network Slice Instance may be composed by none, one or more Sub-network Instances, which may be shared by another Network Slice Instance.

Slice Capability exposure component is allowing 3rd parties to access / use via APIs information regarding services provided by the slice (e.g. connectivity information, QoS, mobility, autonomicity, etc.) and to dynamically customize the network characteristics for different diverse use cases (e.g. ultra-low latency, ultra-reliability, value-added services for enterprises, etc.) within the limits set of functions by the operator. It includes a description

of the structure (and contained components) and configuration of the slice instance.

Logical resource - An independently manageable partition of a physical resource, which inherits the same characteristics as the physical resource and whose capability is bound to the capability of the physical resource. It is dedicated to a Network Function or shared between a set of Network Functions.

Virtual resource - An abstraction of a physical or logical resource, which may have different characteristics from that resource, and whose capability may not be bound to the capability of that resource.

Network Function - It refers to processing functions in a network. This includes but is not limited to telecom nodes functionality, as well as switching functions e.g. switching function, IP routing functions.

Virtual Network Function - One or more virtual machines running different software and processes on top of 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.).

3. Autonomic Slice Networking

This section describes the various elements in a network with autonomic functions, and how these entities work together, on a high level. Subsequent sections explain the detailed inside view for each of the autonomic network elements, as well as the network functions (or interfaces) between those elements.

Figure 1 shows the high level view of an Autonomic Slice Networking.

It consists of a number of autonomic nodes resources, which interact directly with each other. Those autonomic nodes resources provide a common set of capabilities across a network slice, called the "Autonomic Slice Networking Infrastructure" (ASNI). The ASNI provides functions like naming, addressing, negotiation, synchronization, discovery and messaging.

Autonomic network functions typically span several slices in the network. The atomic entities of an autonomic function are called the "Autonomic Service Agents" (ASA), which are instantiated on slices.

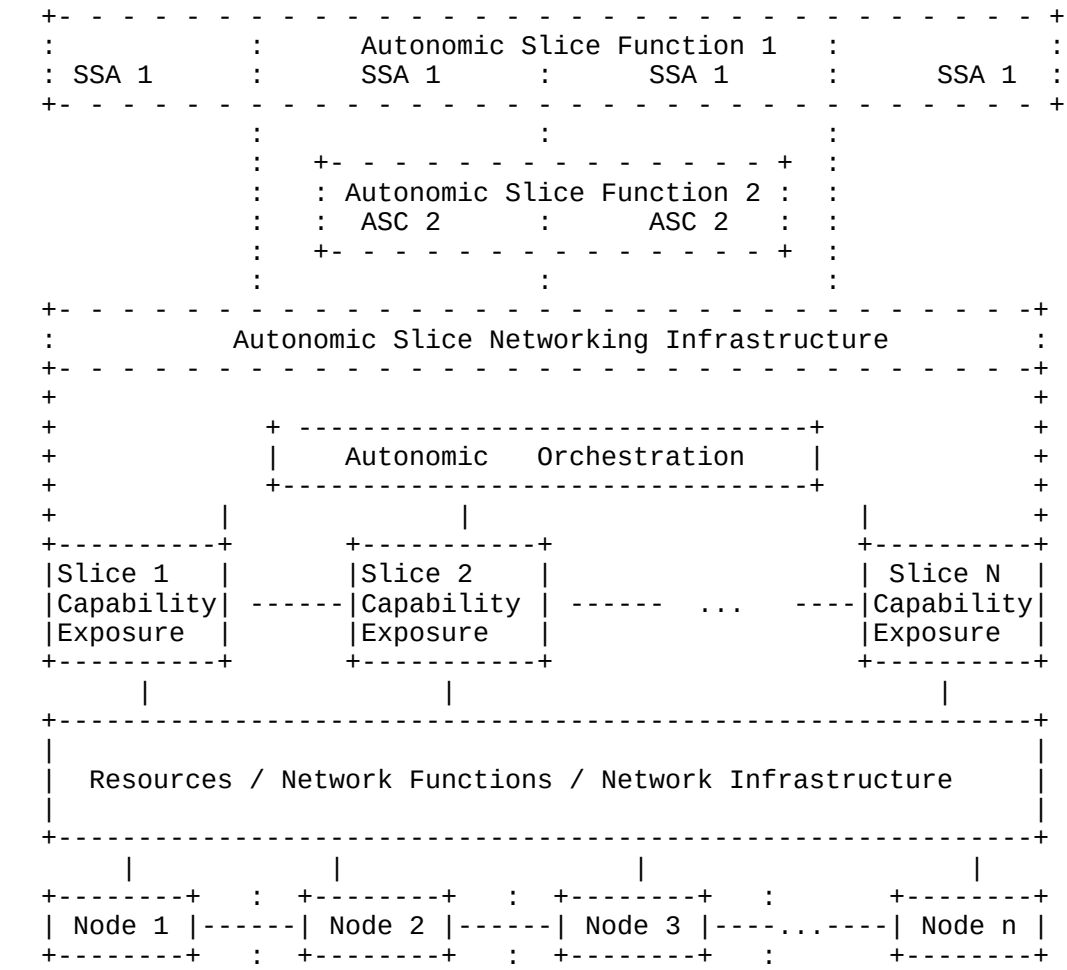


Figure 1: High level view of Autonomic Slice Networking

In a horizontal view, autonomic functions span across the network, as well as the Autonomic Slice Networking Infrastructure. In a vertical view, a slice always implements the ASNI, plus it may have one or several Autonomic Service Agents as part of slice capability exposure.

The Autonomic Networking Infrastructure (ASNI) therefore is the foundation for autonomic functions. The current charter of the ANIMA WG includes the specification of the ASNI, using a few autonomic functions as use cases. ASNI would represent a customized and an approach [I-D.ietf-anima-reference-model] for implementing a general purposed ASI.

Additionally, at least 2 autonomous functions are envisioned - Autonomous Slice control (ASC) and Slice Service agent (SSA). These are explained in sections below.

4. Autonomic Orchestration (*)

This section describes an autonomic orchestration and its functionality.

Orchestration refers to the functions that autonomically coordinate the slices lifecycle and all the components that are part of the slice (i.e. Service Instances, Network Slice Instances, Resources, Capabilities exposure) to ensure an optimized allocation of the necessary resources across the network. It is expected to coordinate a number of interrelated resources, often distributed across a number of subordinate domains, and to assure transactional integrity as part of the process [TETT1] .

It is also the continuing process of allocating resources to satisfy contending demands in an optimal manner [TETT2] . The idea of optimal would include at least prioritized SLA commitments, and factors such as customer endpoint location, geographic or topological proximity, delay, aggregate or fine-grained load, monetary cost, fate- sharing or affinity. The word continuing incorporates recognition that the environment and the service demands constantly change over the course of time, so that orchestration is a continuous, multi-dimensional optimization feedback loop.

It protects the infrastructure from instabilities and side effects due to the presence of many slice components running in parallel. It ensures the proper triggering sequence of slice functionality and their stable operation. It defines conditions/constraints under which service components will be activated, taking into account operator service and network requirements (inclusive of optimize the use of the available network & compute resources and avoid situations that can lead to sub-par performance and even unstable and oscillatory behaviors.

5. The Autonomic Network Slicing Element

This section describes an autonomic slice network element and its internal architecture. The reference model explained in the document "Autonomic Networking - Definitions and Design Goals" [RFC7575] shows the sources of information that an autonomic service agent can leverage: Self-knowledge, network knowledge (through discovery), Intent [I-D.du-anima-an-intent] , and feedback loops. Fundamentally, there are two levels inside an autonomic node: the level of Autonomic

Service Agents, and the level of the Autonomic Slice Networking Infrastructure, with the former using the services of the latter.

Figure 2 illustrates this concept.

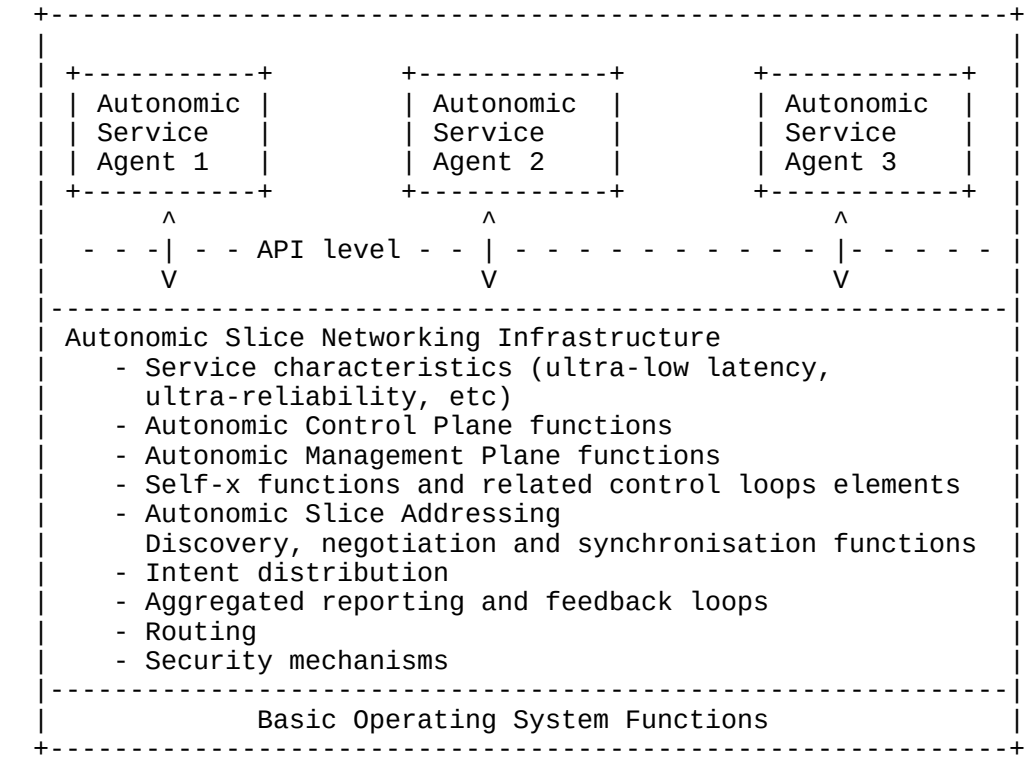


Figure 2: Model of an autonomic element

The Autonomic Slice Networking Infrastructure (lower part of Figure 2) contains slice specific data structures, for example trust information about itself and its peers, as well as a generic set of functions, independent of a particular usage. This infrastructure should be generic, and support a variety of Autonomic Service Agents (upper part of Figure 2). The Autonomic Control Plane is the summary of all interactions of the Autonomic Slice Networking Infrastructure with other services.

The use cases of "Autonomics" such as self-management, self-optimisation, etc, are implemented as Autonomic Service Agents. They use the services and data structures of the underlying autonomic networking infrastructure. The Autonomic Slice Networking Infrastructure should itself be self-managing.

The "Basic Operating System Functions" include the "normal OS", including the network stack, security functions, etc. Autonomic Network Slicing Element is a composition of autonomic slice service agents and autonomic slice control. Autonomic slice service agents obtain specific network resources and provide self-managing and self-controlling functions. An autonomic slice control is a higher-level autonomic function that takes the role of life-cycle management of a or many slice instances. There can be many slice control functions based on different types or attributes of slice.

6. The Autonomic Slice Networking Infrastructure

The Autonomic Networking Infrastructure provides a layer of common functionality across an Autonomic Network. It comprises "must implement" functions and services, as well as extensions. The Autonomic Slice Networking Infrastructure (ASNI) resides on top of an abstraction layer of resource, network function and network infrastructure as shown in figure 1. The document assumes abstraction layer enables different autonomous service agents to communicate with the underlying disaggregated and distributed network infrastructure, which itself maybe an autonomous networking (AN) domain or combination of multiple AN domain. The goal of ASNI is to provide autonomic life-cycle management of network slices.

6.1. Signaling Between Autonomic Slice Capability Exposures

The basic network capabilities are autonomically or through traditional techniques are learnt by slice agents. This depends on the fact that physical infrastructure is an autonomic network or not. The GASP signaling [I-D.ietf-anima-grasp] [I-D.liu-anima-grasp-distribution] [I-D.liu-anima-grasp-api] may be used to expose capabilities among SSAs or slice control. Optionally, SSA capabilities are more interesting to slice control autonomic functions for slice creation and install. The slice control must have the independent intelligence to process and filter capabilities to meet a network slice specification and have low level resources allocated for a slice through SSAs. 6.2 The Autonomic Control Plane.

6.2. The Autonomic Control Plane

TBD.

6.3. Naming & Addressing

A slice can be instantiated on demand, represents a logical network and therefore, must be assigned a unique identifier. A Slice Service Agent (SSA) may support functions of a single or multiple slices and communicate with each other, using the addressing of the Autonomic or

traditional (non-autonomic) Networking Infrastructure reside on. An SSA complies with ACP addressing mechanisms and in a domain, i.e., As part of the enrolment process the registrar assigns a number to the device, which is unique for slicing registrar and in ASNI domain.

6.4. Discovery

Slices themselves are not discovered but are instantiated through slice control autonomic function. However, both slice service agents and slice control functions must be discovered. Even though autonomic control plane will support discovery of all the SSAs and slice control, it may not be necessary.

6.5. Routing

Autonomic network slicing follows single routing protocol as described in [I-D.ietf-anima-autonomic-control-plane].

6.6. Intent

TBD.

7. Security and Trust Infrastructure

An Autonomic Slice Network is self-protecting. All protocols are secure by default, without the requirement for the administrator to explicitly configure security.

TBD.

7.1. Public Key Infrastructure

An autonomic domain uses a PKI model. The root of trust is a certification authority (CA). A registrar acts as a registration authority (RA).

A minimum implementation of an autonomic domain contains one CA, one Registrar, and network elements.

7.2. Domain Certificate

TBD.

8. Cross-Domain Functionality

TBD.

9. Autonomic Service Agents (ASA)

This section describes how autonomic services run on top of the Autonomic Slice Networking Infrastructure. There are at least two different types of autonomic functions are known:

1. Slice Service Agents are low level functions that learn capabilities of underlying infrastructure in terms of interfaces and available resources. They coordinate with Slice control to associate these resources with specific slice instances in effect performing full life cycle management of these resources.

2. Slice Control Autonomic Function: Slice control is responsible for high-level life-cycle management of a slice itself. This function will hold slice instances and their attributes related data structures in autonomic network slice infrastructure. As an example, a slice is defined for high bandwidth, highly secure transactional application. A slice control must be capable of negotiating resources required across different SSAs.

Out of scope are details of the mechanisms how the information is represented and exchanged between the two autonomic functions.

10. Management and Programmability

This section describes how an Autonomic Network is managed, and programmed.

10.1. How a Slice Network Is Managed

Slice network management is driven by Slice control, there are four categories operation:

1. Creating a network slice: Receive a network slice resource description request, upon successful negotiation with SSA allocate resource for it.

2. Shrink/Expand slice network: Dynamically alter resource requirements for a running slice network according service load.

3. (Re-)Configure slice network: The slice management user deploys a user level service into the slice. The slice control takes over the control of all the virtualized network functions and network programmability functions assigned to the slice, and (re-)configure them as appropriate to provide the end-to-end service.

4. Destroy slice network: Recycle all resource from the infrastructure.

10.2. Intent

TBD.

10.3. Control Loops

TBD.

10.4. APIs

The API model of for autonomic slicing semantically, is grouped into the following APIs to be defined.

10.4.1. Slice Control APIs

1. Create a slice network on user request. The request includes resource description. A unique identify a slice network, group all the resource.

2. Destroy a slice network identified by it’s id.

3. Query a slice network slicing state by it’s uuid.

4. Modify a slice network.

10.4.2. Service Agent - Device APIs

A service agent will interface with the physical infrastructure either through an autonomic network or traditional infrastructure. Depending upon which a device can either have autonomic or non-autonomic addressing. Service agents are required to perform life cycle management of network elements participating in a network slice and the following APIs are needed for addition, removal or update of a specific device. A device may be a logical or physical network element. Optionally, it may be a network function.

10.4.3. Service Agent - Port APIs

A port may be a physical or logical network port in a slice depending upon whether underlying infrastructure is an autonomic or traditional network. Service agents must be able to control the operational state of these ports. APIs are needed for addition, removal, update and operational state retrieval of a specific port.

10.4.4. Service Agent - Link APIs

A link connects two or more ports of devices described in above section. Service agents must be able to control the operational and connection status of these links through APIs for addition, removal, update and state retrieval for each link.

10.5. Relationship with MANO

Please refer to [MANO] for MANO introduction.

TBD.

11. Security Considerations

11.1. Threat Analysis

TBD.

11.2. Security Mechanisms

TBD.

12. IANA Considerations

This document requests no action by IANA.

13. Acknowledgements

Thanks Bing Liu for helping editing the draft.

This document was produced using the xml2rfc tool [RFC2629].

14. References

14.1. Normative References

[I-D.ietf-anima-grasp]
Bormann, C., Carpenter, B., and B. Liu, "A Generic Autonomic Signaling Protocol (GRASP)", draft-ietf-anima-grasp-07 (work in progress), September 2016.

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

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <http://www.rfc-editor.org/info/rfc2629>.

14.2. Informative References

[ChinaCom09]
"A. Galis et all - "Management and Service-aware Networking Architectures (MANA) for Future Internet" - Invited paper IEEE 2009 Fourth International Conference on Communications and Networking in China (ChinaCom09) 26-28 August 2009, Xi'an, China, <http://www.chinacom.org/2009/index.html>.".

[GENI] ""GENI Key Concepts" - Global Environment for Network Innovations (GENI) <http://groups.geni.net/geni/wiki/GENIConcepts>.".

[I-D.du-anima-an-intent]
Du, Z., Jiang, S., Nobre, J., Ciavaglia, L., and M. Behringer, "ANIMA Intent Policy and Format", draft-du-anima-an-intent-04 (work in progress), July 2016.

[I-D.ietf-anima-autonomic-control-plane]
Behringer, M., Eckert, T., and S. Bjarnason, "An Autonomic Control Plane", draft-ietf-anima-autonomic-control-plane-03 (work in progress), July 2016.

[I-D.ietf-anima-reference-model]
Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L., Pierre, P., Liu, B., Nobre, J., and J. Strassner, "A Reference Model for Autonomic Networking", draft-ietf-anima-reference-model-02 (work in progress), July 2016.

[I-D.liu-anima-grasp-api]
Carpenter, B., Liu, B., Wang, W., and X. Gong, "Generic Autonomic Signaling Protocol Application Program Interface (GRASP API)", draft-liu-anima-grasp-api-02 (work in progress), September 2016.

[I-D.liu-anima-grasp-distribution]
Liu, B. and S. Jiang, "Information Distribution over GRASP", draft-liu-anima-grasp-distribution-02 (work in progress), September 2016.

[I-D.strassner-anima-control-loops]
Strassner, J., Halpern, J., and M. Behringer, "The Use of Control Loops in Autonomic Networking", draft-strassner-anima-control-loops-01 (work in progress), April 2016.

[IMT2020] "ITU-T IMT2020 document "Report on Gap Analysis" - ITU-T IMT2020 ITU- Dec 2015 Published by ITU-T IMT2020. <http://www.itu.int/en/ITU-T/focusgroups/imt-2020/Pages/default.aspx>.".

[MANO] "ETSI European Telecommunications Standards Institute. Network Functions Virtualisation (NFV); Management and Orchestration v1.1.1. Website, December 2014. <http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_ nfv-man001v010101p.pdf>.".

[NGMN] "Hedmar,P., Mschner, K., et all - NGMN Alliance document "Description of Network Slicing Concept", January 2016. <https://www.ngmn.org/uploads/media/160113_Network_Slicing_v1_0.pdf>.".

[NGS-3GPP]
""Study on Architecture for Next Generation System" - latest version v1.0.2 September 2016 <http://www.3gpp.org/ftp/tsg_sa/WG2_Arch/Latest_SA2_Specs/Latest_draft_S2_Specs>.".

[ONF] "Paul, M, Schallen, S., Betts, M., Hood, D., Shirazipor, M., Lopes, D., Kaippallimalit, J., - Open Network Fundation document "Applying SDN Architecture to 5G Slicing", April 2016. <https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/Applying_SDN_Architecture_to_5G_Slicing_TR-526.pdf>.".

[RFC7575] Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals", RFC 7575, DOI 10.17487/RFC7575, June 2015, <http://www.rfc-editor.org/info/rfc7575>.

[RFC7576] Jiang, S., Carpenter, B., and M. Behringer, "General Gap Analysis for Autonomic Networking", RFC 7576, DOI 10.17487/RFC7576, June 2015, <http://www.rfc-editor.org/info/rfc7576>.

[TETT1] "Guerzoni,R., Vaishnavi,I.,Perez-Caparros, D., Galis,A., et all "Analysis of End-to-End Multi Domain Management and Orchestration Frameworks for Software Defined Infrastructures: an Architectural Survey", Transactions on Emerging Telecommunications Technologies, Wiley Online Library, DOI: 10.1002/ett.3103, June 2016, <onlinelibrary.wiley.com/doi/10.1002/ett.3103/pdf>.".

[TETT2] "Karl,H., Draexler,S., Peuster, M, Galis, A., et all "DevOps for Network Function Virtualization: An Architectural Approach", Transactions on Emerging Telecommunications Technologies, Wiley Online Library, DOI: 10.1002/ett.3084, July 2016, <http://onlinelibrary.wiley.com/doi/10.1002/ett.3084/full>.".

Authors' Addresses

Alex Galis
University College London
Department of Electronic and Electrical Engineering
Torrington Place
London WC1E 7JE
United Kingdom

Email: a.galis@ucl.ac.uk

Kiran Makhijani
Huawei Technologies
2890, Central Expressway
Santa Clara CA 95032
USA

Email: USA Email: kiran.makhijani@huawei.com

Delei Yu
Huawei Technologies
Q22, Huawei Campus
No.156 Beiqing Road
Hai-Dian District, Beijing 100095
P.R. China

Email: yudelei@huawei.com

No Working Group
Internet-Draft
Intended Status: Standards Track
Expires: March 30, 2019

A. Galis
University College London
K. Makhijani
D. Yu
B. Liu
Huawei Technologies

September 26, 2018

Autonomic Slice Networking
draft-galis-anima-autonomic-slice-networking-05

Abstract

This document describes the technical requirements and the related reference model for the intercommunication and coordination among devices in Autonomic Slicing Networking. The goal is to define how the various elements in a network slicing context work and orchestrate together, to describe their interfaces and relations. While the document is written as generally as possible, the initial solutions are limited to the chartered scope of the WG.

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

The list of current Internet-Drafts can be accessed at https://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at https://www.ietf.org/shadow.html

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, 2017.

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

The document "Autonomic Networking - Definitions and Design Goals" [RFC7575] explains the fundamental concepts behind Autonomic Networking, and defines the relevant terms in this space, as well as a high level reference model. This document defines this reference model with more detail, to allow for functional and protocol specifications to be developed in an architecturally consistent, non-overlapping manner. While the document is written as generally as possible, the initial solutions are limited to the chartered scope of the WG.

Most networks will run with some autonomic functions for the full networks or for a group of nodes [RFC7576] or for a group of slice networks while the rest of the network is traditionally managed.

The goal of this document is to focus on the autonomic slicing networking. [RFC7575] is focusing on fully or partially autonomic nodes or networks.

The proposed revised ANIMA reference model allows for this hybrid approach across all such capabilities. It enhances [ASN].

This is a living document and will evolve with the technical solutions developed in the ANIMA WG. Sections marked with (*) do not represent current charter items.

While this document must give a long term architectural view, not all functions will be standardized at the same time.

2. The Network Slicing Overall View

2.1. Key Terms and Context

A number of slice definitions were used in the last 10 years in distributed and federated testbed research [GENI], future internet research [ChinaCom09] and more recently in the context of 5G research [NGMN], [ONF], [IMT2020], [NGS-3GPP], [NS-ETSI]. Such definitions converge towards NS as group of components: Service Instance, Network Slice Instance, Resources and Slice Element Manager

In this draft we are using the following terms:

Logical resource - An independently manageable partition of a physical resource, which inherits the same characteristics as the physical resource and whose capability is bound to the capability of the physical resource. It is dedicated to a Network Function or shared between a set of Network Functions.

Virtual resource - An abstraction of a physical or logical resource, which may have different characteristics from that resource, and whose capability may not be bound to the capability of that resource

Network Function (NF) - A processing function in a network. It includes but is not limited to network nodes functionality, e.g. session management, mobility management, switching, routing functions, which has defined functional behaviour and interfaces. Network functions can be implemented as a network node on a dedicated hardware or as a virtualized software functions. Data, Control, Management, Orchestration planes functions are Network Functions.

Virtual Network Function (VNF) - 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 middle. boxes (e.g. router, NAT, firewall, load balancer, etc.) Network Slicing (NS) refers to a managed group of subsets of resources, network functions / network virtual functions at the data, control, management/orchestration planes and services at a given time. Network slice is programmable and has the ability to expose its capabilities. The behaviour of the network slice realized via network slice instance(s). Network resources include connectivity, compute, and storage resources.

Network Slicing is end-to-end concept covering the radio and non-radio networks inclusive of access, core and edge / enterprise networks. It enables the concurrent deployment of multiple logical, self-contained and independent shared or partitioned networks on a common infrastructure platform

Network slicing represents logically or physically isolated groups of network resources and network function/virtual network functions configurations separating its behavior from the underlying physical network.

Network Slice Instance - An activated network slice. It is created based on network template. A set of managed run-time network

functions, and resources to run these network functions, forming a complete instantiated logical network to meet certain network characteristics required by the service instance(s). It provides the network characteristics that are required by a service instance. A network slice instance may also be shared across multiple service instances provided by the network operator.

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

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

Examples of physical or virtual resources to be shared or partitioned would include: bandwidth on a network link, forwarding tables in a network element (switch, router), processing capacity of servers, processing capacity of network or network clouds elements [SLICING]. As such slice instances would contain:

(i) a combination/group of the above resources which can act as a network,
(ii) appropriate resource abstractions,
(iii) capability exposure of abstract resources towards service and management clients that are needed for the operation of slices

The capability exposure creates an abstraction of physical network devices that would provide information and information models allowing operators to manipulate the network resources. By utilizing open programmable network interfaces, it would enable access to control layer by customer interfaces and applications.

The establishment of slices is both business-driven (i.e. slices are in support for different types and service characteristics and business cases) and technology-driven as slice is a grouping of physical or virtual) resources (network, compute, storage) which can act as a sub network and/or a cloud. A slice can accommodate service components and network functions (physical or virtual) in all network segments: access, core and edge / enterprise networks.

A complete slice is composed of not only various network functions which are based on virtual machines at C-RAN and C-Core, but also transport network resources that can be assigned to the slice at radio access/transport network. Different future businesses require different throughput, delay and mobility, and some businesses need very high throughput or/and low delay.

2.2. High Level Requirements

Slice creation: management plane create virtual or physical network functions and connects them as appropriate and instantiate them in the slice, which is a subnetworks.

The instance of slice management then takes over the management and operations of all the (virtualised) network functions and network programmability functions assigned to the slice, and (re-)configure them as appropriate to provide the end-to-end service.

A complete slice is composed of not only various network functions which are based on virtual machines at C-RAN and C-Core, but also transport network resources that can be assigned to the slice at radio access/transport network. Different future businesses [5GNS], [PER-NS] require different throughput, delay and mobility, and some businesses need very high throughput or/and low delay. Transport network shall provide QoS isolation, flexible network operation and management, and improve network utilization among different business.

(1) Separation from partition of the physical network: Network slicing represents logically or physically isolated groups of network resources and network function/virtual network functions configurations separating its behavior from the underlying physical network.

(2) QoS Isolation: Although traditional VPN technology can provide physical network resource isolation across multiple network segments, it is deemed far less capable of supporting QoS hard isolation, Which means QoS isolation on forwarding plane requires better coordination with management plane.

(3) Independent Management Plane: Like above, network isolation is not sufficient, a flexible and more importantly a management plane per instance is required to operate on a slice independently and autonomously within the constraints of resources allocated to the slice.

(4) Another flexibility requirement is that an operator can deploy their new business application or a service in network slice with low cost and high speed, and ensure that it does not affect existing of business applications adversely.

(5) Stringent Resource Characteristics: A Network Slicing aware infrastructure allows operators to use part of the network resources to meet stringent resource characteristics.

(6) Type of resources: Network Slice instance is a dedicated network

that is build and activated on an infrastructure mainly composed of, but not limited to, connectivity, storage and computing.

(7) Programmability: Operator not only can slice a common physical infrastructure into different logical networks to meet all kinds of new business requirements, but also can use SDN based technology to improve the overall network utilization. By providing a flexible programmable interface; the 3rd party can develop and deploy new network business rapidly. Further, if a network slicing can run with its own slice controller, this network slicing will get more granular control capability [I-D.ietf-anima-autonomic-control-plane] to retrieve slice status, and issuing slicing flow table, statistics fetch etc.

(8) Life cycle self-management: It includes creation, operations, re- configuration, composition, decomposition, deletion of slices. It would be performed automatically, without human intervention and based on a governance configurable model of the operators. As such protocols for slice set-up /operations /(de)composition / deletion must also work completely automatically. Self-management (i.e. self- configuration, self-composition, self-monitoring, self-optimisation, self-elasticity) is carried as part of the slice protocol characterization.

(9) Network slice Self-management: Network slices will need to be self-managed by automated, autonomic and autonomous systems in order to cope with dynamic requirements, such as flexible scalability, extensibility, elasticity, residency and reliability of an infrastructure. Network slices will need to be self-managed by automated, autonomic and autonomous systems in order to cope with dynamic requirements, such as scalability or extensibility of an infrastructure. A common information model describing uniformly the NS in a single and/or multiple domain would support such self-managed.

(10) Extensibility: Since the Autonomic Slice Networking Infrastructure is a relatively new concept, it is likely that changes in the way of operation will happen over time. As such new networking functions will be introduced later, which allow changes to the way the slices operate.

(11) Network Slice elasticity: A Network Slice instance has the mechanisms and triggers for the growth/shrinkage of all resources, and/or network and service functions as enabled by a common information model that explicitly provides for elasticity policies for scaling up/down resources.

(12) Multiple domains activation: Network slice instances are concurrently activated as multiple logical, self-contained and independent, partitioned network functions and resources on a specific infrastructure domain.

(13) Resource Exposure: Each network slice has the ability to dynamically expose and possibly negotiate the parameters that characterize an NS as enabled by a common information model that explicitly provides monitoring policies for all model descriptors.

(14) Network Tenants: Network slicing support tenants that are strongly independent on infrastructure as enabled by a common information model that explicitly provides for a level of tenants management for the resources dedicated to an instance of network slice.

(15) End-to-end Orchestration of Network Slicing: Coordinating underlay network infrastructure and service function resources. In the process of orchestration of network slice, resource registration and templates for network slice repository are needed.

3. Autonomic Slice Networking

This section describes the various elements in a network with autonomic functions, and how these entities work together, on a high level. Subsequent sections explain the detailed inside view for each of the autonomic network elements, as well as the network functions (or interfaces) between those elements.

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 Slicing lifecycle includes the management plane selecting a group of network resources (whereby network resources can be physical, virtual or a combination thereof); it connects with the physical and virtual network and service functions as appropriate, and it instantiates all of the network and service functions assigned to the slice. For slice operations, the control plane takes over governing of all the network resources, network and service functions assigned to the slice. It (re-) configures them as appropriate and as per elasticity needs, in order to provide an end-to-end service.

One expected autonomic Slice Networking function is the capability and resource Usability for a slice. Applications or services requiring information of available slice capabilities and resources are satisfied by abstracted resource view and control. Usability of capabilities and resources can be enabled either by resource publishing or by discovery. In the latter case, the service performs resource collection directly from the provider of the slice by using discovery mechanisms to get total information about the available resources to be consumed. In the former, the network provider exposes available resources to services (e.g., through a resource catalog) reducing the amount of detail of the underlying network.

Slice Element Manager (SEM) is installed for each control domain. Control domain is defined according to geographic location and control functions. Each SEM converts requirements from orchestrator into virtual resources and manages virtual resources of a slice. SEM also exchanges information of virtual resources with other slice element managers via a dedicated resource interface. SEM provides also capability exposure facilities by allowing 3rd parties to access / use via APIs information regarding services provided by the slice (e.g. connectivity information, QoS, mobility, autonomicity, etc.) and to dynamically customize the network characteristics for different diverse use cases (e.g. ultra-low latency, ultra-reliability, value-added services for enterprises, etc.) within the limits set of functions by the operator.

Physical Element Manager (PEM) is installed for each control domain. Control domain is defined according to geographic location and control functions. PEM exchanges information of virtual resource with SEM via virtual resource interface and interconverts between virtual resource and physical resource. The PEM orders physical functions (ex. switches) to allocate physical resource via physical resource interface.

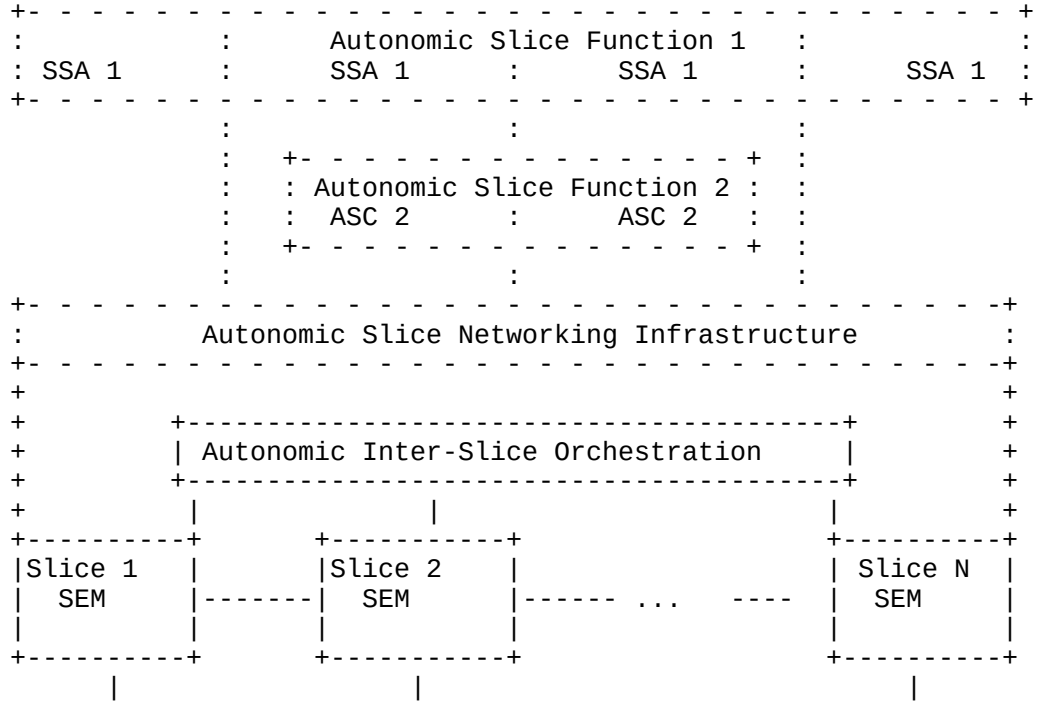
Figure 1 shows the high level view of an Autonomic Slice Networking.

It consists of a number of autonomic nodes resources, which interact directly with each other. Those autonomic nodes resources provide a common set of capabilities across a network slice, called the "Autonomic Slice Networking Infrastructure" (ASNI).

The ASN provides functions like naming, addressing, negotiation, synchronization, discovery and messaging.

Autonomic network functions typically span several slices in the network. The atomic entities of an autonomic function are called the "Autonomic Service Agents" (ASA), which are instantiated on slices.

In a horizontal view, autonomic functions span across the network, as well as the Autonomic Slice Networking Infrastructure. In a vertical view, a slice always implements the ASNI, plus it may have one or several Autonomic Service Agents as part of slice capability exposure. The Autonomic Networking Infrastructure (ASNI) therefore is the foundation for autonomic functions. The current charter of the ANIMA WG includes the specification of the ASNI, using a few autonomic functions as use cases. ASNI would represent a customized and an approach [I-D.ietf-anima-reference-model] for implementing a general purposed ASI.



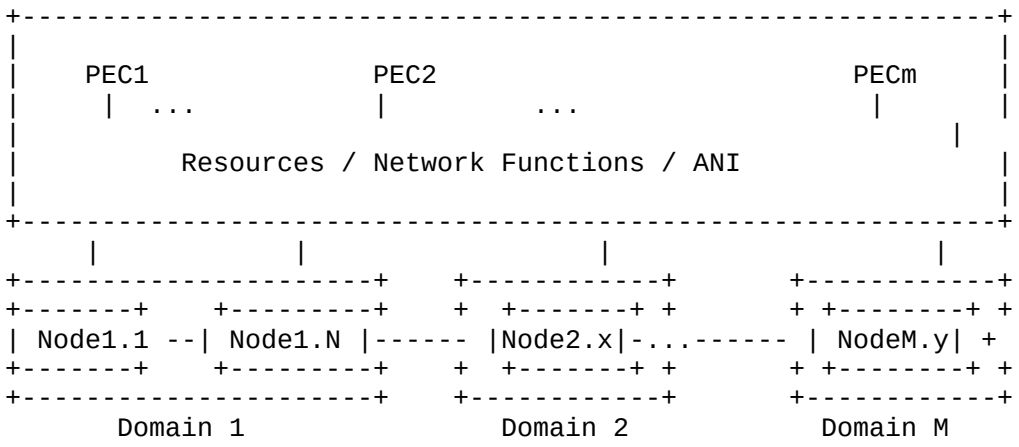


Figure 1: High level view of Autonomic Slice Networking

Additionally, at least 2 autonomous functions are envisioned - Autonomous Slice control (ASC) and Slice Service agent (SSA). These are explained in sections below.

4. Autonomic Inter-Slice Orchestration

This section describes an autonomic orchestration and its functionality.

Orchestration refers to the system functions that:

* automated and autonomically co-ordination of network functions in slices

* autonomically coordinate the slices lifecycle and all the components that are part of the slice (i.e. Service Instances, Network Slice Instances, Resources, Capabilities exposure) to ensure an optimized allocation of the necessary resources across the network.

* coordinate a number of interrelated resources, often distributed across a number of subordinate domains, and to assure transactional integrity as part of the process [TETT1].

* autonomically control of slice life cycle management, including concatenation of slices in each segment of the infrastructure including the data pane, the control plane, and the management plane.

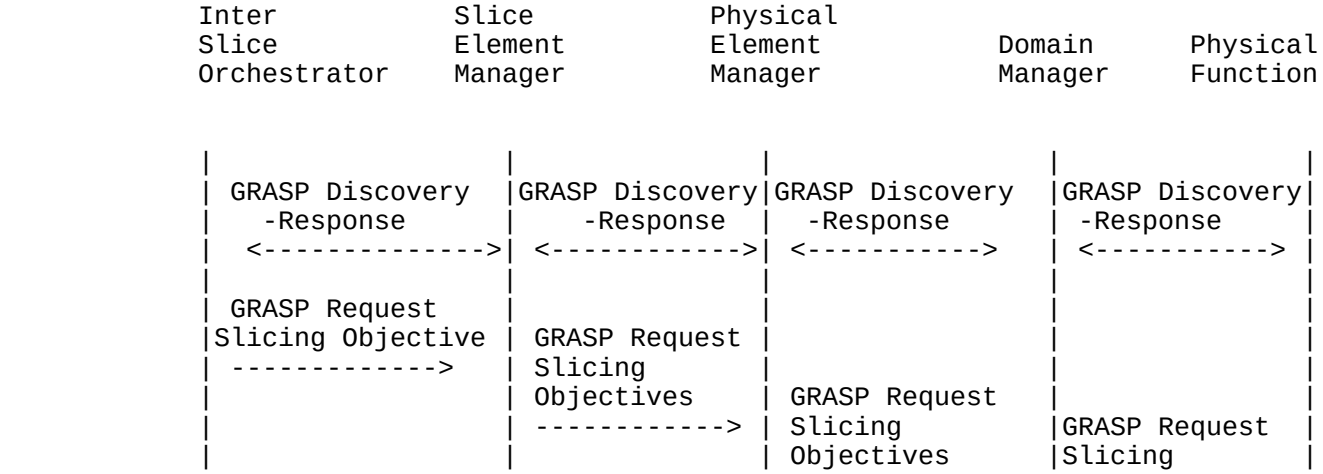
* autonomically coordinate and trigger of slice elasticity and placement of logical resources in slices.

* coordinates and (re)-configure logical resources in the slice by taking over the control of all the virtualized network functions assigned to the slice.

It is also the continuing process of allocating resources to satisfy contending demands in an optimal manner [TETT2]. The idea of optimal would include at least prioritized SLA commitments [SERMODEL], and factors such as customer endpoint location, geographic or topological proximity, delay, aggregate or fine-grained load, monetary cost, fate- sharing or affinity. The word continuing incorporates recognition that the environment and the service demands constantly change over the course of time, so that orchestration is a continuous, multi-dimensional optimization feedback loop [I-D.strassner-anima-control-loops].

It protects the infrastructure from instabilities and side effects due to the presence of many slice components running in parallel. It ensures the proper triggering sequence of slice functionality and their stable operation. It defines conditions/constraints under which service components will be activated, taking into account operator service and network requirements (inclusive of optimize the use of the available network & compute resources and avoid situations that can lead to sub-par performance and even unstable and oscillatory behaviors.

5. GRASP Resource Reservation / Release Messages flow



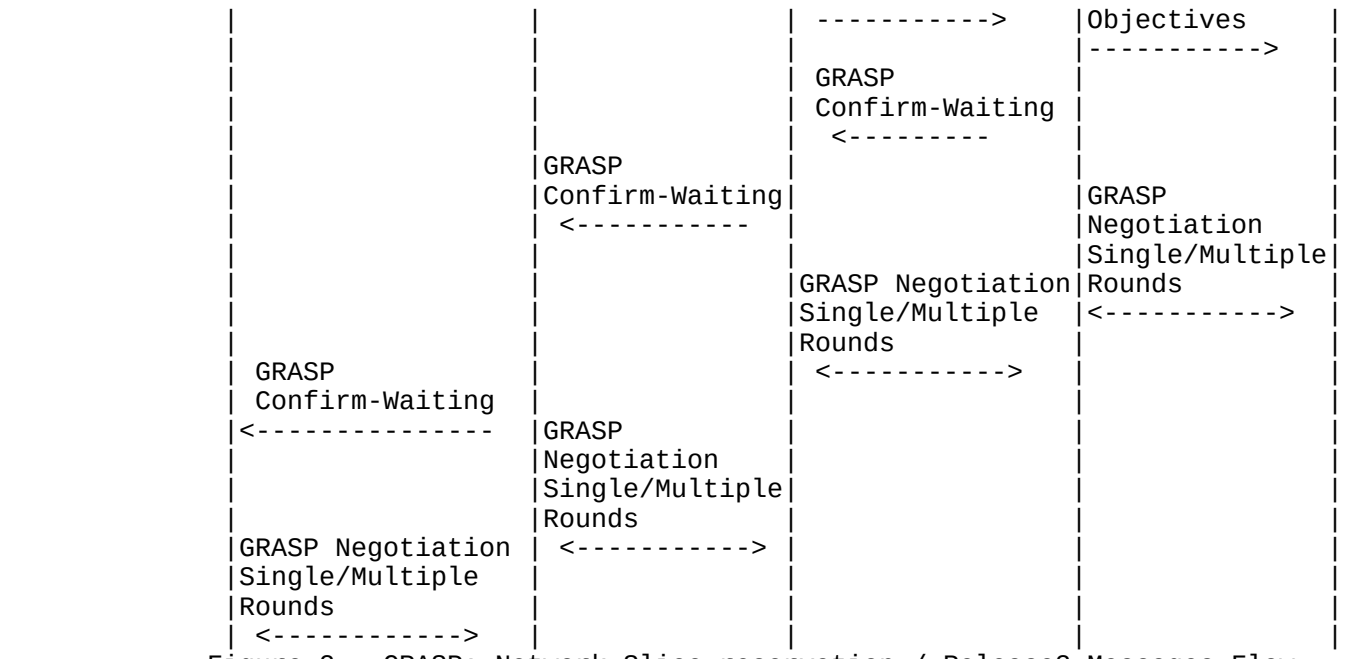


Figure 2 - GRASP: Network Slice reservation / Release3 Messages Flow

The above message sequence figure shows the message flows of the interactions between Inter-Slice Orchestrator, Slice Element Manager, Physical Element Manager, Domain Manager and Physical Network functions.

6. The Autonomic Network Slicing Element

This section describes an autonomic slice network element and its internal architecture. The reference model explained in the document "Autonomic Networking - Definitions and Design Goals" [RFC7575] shows the sources of information that an autonomic service agent can leverage: Self-management, Self-knowledge, network knowledge (through discovery), Intent [I-D.du-anima-an-intent], and feedback loops. Fundamentally, there are two levels inside an autonomic node: the level of Autonomic Service Agents, and the level of the Autonomic Slice Networking Infrastructure, with the former using the services of the latter. The self management functionality (self-configuration, self-optimisation, self- healing) could be implemented across the Inter Slice Orchestrator, Slice Element Manager and Physical Element Manager. Such functionality deals with dynamic

* coordination the life cycle of slices

* allocation of resources to slice instances in an efficient way that provides required slice instances performance,

* self-configuration, self-optimization and self-healing of slice instances during their lifecycle management including deployment and operations

* self-configuration, self-optimization and self-healing of services of each slice instance. Service lifecycle, that is typically different than slice instance lifecycle should also be managed in the autonomous way.

Figure 3 illustrates this concept.

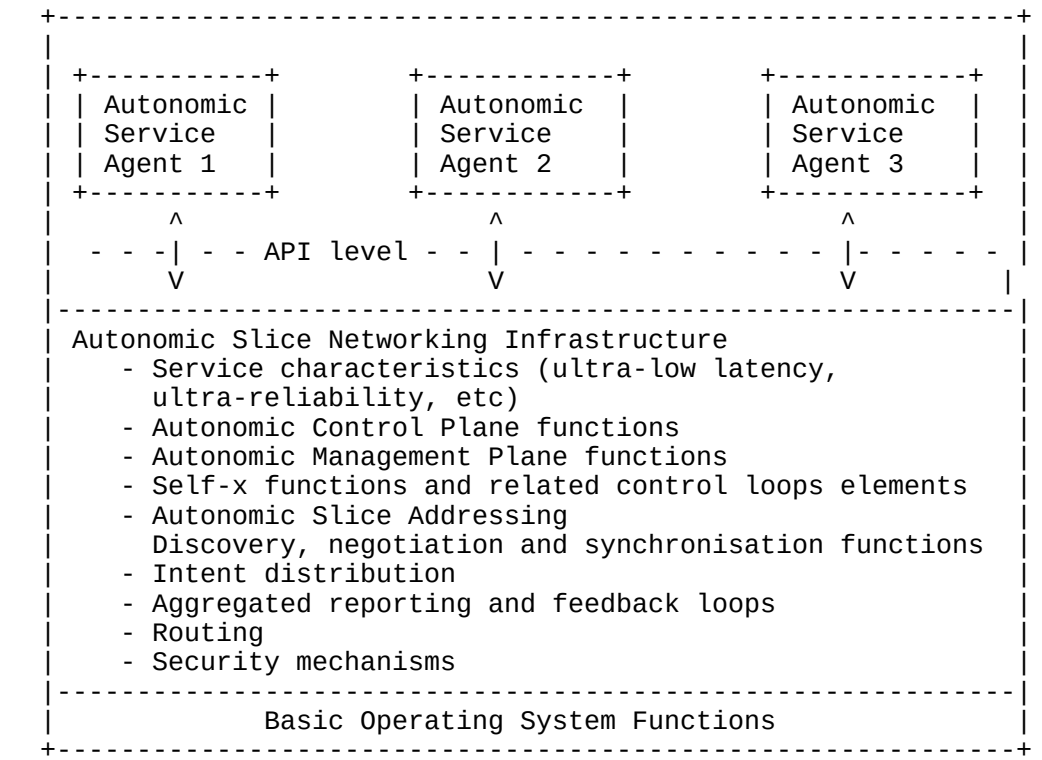


Figure 3: Model of an autonomic element

The Autonomic Slice Networking Infrastructure (lower part of Figure 2) contains slice specific data structures, for example trust information about itself and its peers, as well as a generic set of functions, independent of a particular usage. This infrastructure should be generic, and support a variety of Autonomic Service Agents

(upper part of Figure 2). The Autonomic Control Plane is the summary of all interactions of the Autonomic Slice Networking Infrastructure with other services.

The use cases of "Autonomics" such as self-management, self-optimisation, etc, are implemented as Autonomic Service Agents. They use the services and data structures of the underlying autonomic networking infrastructure. The Autonomic Slice Networking Infrastructure should itself be self-managing.

The "Basic Operating System Functions" include the "normal OS", including the network stack, security functions, etc. Autonomic Network Slicing Element is a composition of autonomic slice service agents and autonomic slice control. Autonomic slice service agents obtain specific network resources and provide self-managing and self-controlling functions. An autonomic slice control is a higher-level autonomic function that takes the role of life-cycle management of a or many slice instances. There can be many slice control functions based on different types or attributes of slice.

7. The Autonomic Slice Networking Ianfrastructure

The Autonomic Networking Infrastructure provides a layer of common functionality across an Autonomic Network. It comprises "must implement" functions and services, as well as extensions. The Autonomic Slice Networking Infrastructure (ASNI) resides on top of an abstraction layer of resource, network function and network infrastructure as shown in figure 1. The document assumes abstraction layer enables different autonomous service agents to communicate with the underlying disaggregated and distributed network infrastructure, which itself maybe an autonomous networking (AN) domain or combination of multiple AN domain. The goal of ASNI is to provide autonomic life-cycle management of network slices.

7.1. Signaling Between Autonomic Slice Element Managers

The basic network capabilities are autonomically or through traditional techniques are learnt by slice agents. This depends on the fact that physical infrastructure is an autonomic network or not. The GASP extensions signaling [I-D.liu-anima-grasp-distribution] [I-D.liu-anima-grasp-api] [I-D.ietf-anima-grasp] may be used for

* Discovery of SEMs - a process by which an one SEM discovers peers according to a specific discovery objective. The discovered SEMs peers may later be used as negotiation counterparts or as sources of other coordination activities.

* Negotiation between SEMs - a process by which two SEMs interact to agree on slice logical resource settings that best satisfy the objectives of both SEMs.

* The Synchronization between SEMs - a process by which Orchestrator and SEMs interact to receive the current state of capability exposure values used at a given time in other SEM. This is a special case of negotiation in which information is sent but the SEM or Orchestrator do not request their peers to change configuration settings.

* Self configuration of SEMs - a process by which Orchestrator and SEMs interact to receive the current state of capability exposure values used at a given time in other SEM. This is a special case of synchronization in which information is sent and the SEM is requesting their peers to change configuration settings.

* Self optimization of SEMs - a process by which Orchestrator and SEMs interact to receive the current state of capability exposure values used at a given time in other SEMs. This is a special case of configuration in which information is sent and the SEM is requesting their peers to change logical resource settings in a slice based on an optimisation criteria.

* Mediation for slice resources - a process by which two SEMs interact to agree to logically move resources between slices that best satisfy the objectives of both SEMs triggering of slice elasticity and placement of logical resources in slices. Th???is is a special case of negotiation in which information is sent Orchestrator do request SEMs to change logical resource configuration settings.

* Triggering and governing of elasticity ? a process for autonomic scaling intent configuration mechanisms and resources on the slice level; it allows rapid provisioning, automatic scaling out, or in, of resources. Scale in/out criteria might be used for network autonomics in order the controller to react to a certain set of variations in monitored slices.

* Providing on-demand a self-service network slicing.

Optionally, SSA capabilities are more interesting to slice control autonomic functions for slice creation and install. The slice control must have the independent intelligence to process and filter capabilities to meet a network slice specification and have low level resources allocated for a slice through SSAs.

7.2. The Autonomic Control Plane

TBD.

7.3. Naming & Addressing

A slice can be instantiated on demand, represents a logical network and therefore, must be assigned a unique identifier. A Slice Service Agent (SSA) may support functions of a single or multiple slices and communicate with each other, using the addressing of the Autonomic or traditional (non-autonomic) Networking Infrastructure reside on. An

SSA complies with ACP addressing mechanisms and in a domain, i.e., As part of the enrolment process the registrar assigns a number to the device, which is unique for slicing registrar and in ASNI domain.

7.4. Discovery

Slices themselves are not discovered but are instantiated through slice control autonomic function. However, both slice service agents and slice control functions must be discovered. Even though autonomic control plane will support discovery of all the SSAs and slice control, it may not be necessary.

7.5. Routing

Autonomic network slicing follows single routing protocol as described in [I-D.ietf-anima-autonomic-control-plane].

8. Security and Trust Infrastructure

An Autonomic Slice Network is self-protecting. All protocols are secure by default, without the requirement for the administrator to explicitly configure security.

TBD.

8.1. Public Key Infrastructure

An autonomic domain uses a PKI model. The root of trust is a certification authority (CA). A registrar acts as a registration authority (RA).

A minimum implementation of an autonomic domain contains one CA, one Registrar, and network elements.

8.2. Domain Certificate

TBD.

9. Cross-Domain Functionality

TBD.

10. Autonomic Service Agents (ASA)

This section describes how autonomic services run on top of the Autonomic Slice Networking Infrastructure. There are at least two different types of autonomic functions are known:

1. Slice Service Agents are low level functions that learn capabilities of underlying infrastructure in terms of interfaces and available resources. They coordinate with Slice control to associate these resources with specific slice instances in effect performing full life cycle management of these resources.
2. Slice Control Autonomic Function: Slice control is responsible for high-level life-cycle management of a slice itself. This function will hold slice instances and their attributes related data structures in autonomic network slice infrastructure. As an example, a slice is defined for high bandwidth, highly secure transactional application. A slice control must be capable of negotiating resources required across different SSAs.

Out of scope are details of the mechanisms how the information is represented and exchanged between the two autonomic functions.

11. Management and Programmability

This section describes how an Autonomic Network is managed, and programmed.

11.1. How a Slice Network Is Managed

Slice autonomic management is driven by Slice Element Managers, there are five categories operation:

1. Creating a network slice: Receive a network slice resource description request, upon successful negotiation with SSA allocate resource for it.
2. Shrink/Expand slice network: Dynamically alter resource requirements for a running slice network according service load.
3. (Re-)Configure slice network: The slice management user deploys a user level service into the slice. The slice control takes over the control of all the virtualized network functions and network programmability functions assigned to the slice, and

(re-)configure them as appropriate to provide the end-to-end service.

5. Self-X slice operation: namely self-configuration, self-composition, self-monitoring, self-optimisation, self-elasticity would be carried out as part of new slice protocols.

11.2. Autonomic Resource Information Model

TBD.

The proposed autonomic resource information model is presented as a tree structure of attributes including the following elements: connectivity resources, storage resources, compute resources, service instances, network slice level attributes, etc. The Yang language would be used to represent the autonomic resource information model.

11.3. Control Loops

TBD.

11.4. APIs

The API model of for autonomic slicing semantically, is grouped into the following APIs to be defined.

11.4.1. Slice Control APIs

1. Create a slice network on user request. The request includes resource description. A unique identify a slice network, group all the resource.
2. Destroy a slice network identified by it’s id.
3. Query a slice network slicing state by it’s uuid.
4. Modify a slice network.

11.4.2. Service Agent - Device APIs

A service agent will interface with the physical infrastructure either through an autonomic network or traditional infrastructure. Depending upon which a device can either have autonomic or non-autonomic addressing. Service agents are required to perform life cycle management of network elements participating in a network slice and the following APIs are needed for addition, removal or update of a specific device. A device may be a logical or physical network element. Optionally, it may be a network function.

11.4.3. Service Agent - Port APIs

A port may be a physical or logical network port in a slice depending upon whether underlying infrastructure is an autonomic or traditional network. Service agents must be able to control the operational state of these ports. APIs are needed for addition, removal, update and operational state retrieval of a specific port.

11.4.4. Service Agent - Link APIs

A link connects two or more ports of devices described in above section. Service agents must be able to control the operational and connection status of these links through APIs for addition, removal, update and state retrieval for each link.

11.5. Relationship with MANO

Please refer to [MANO] for MANO introduction.

12. Security Considerations

12.1. Threat Analysis

TBD.

12.2. Security Mechanisms

TBD.

13. IANA Considerations

This document requests no action by IANA.

14. Acknowledgements

This document was converted to nroff by Stuart Clayman (UCL) to comply with RFC format [RFC2629].

14. References

14.1. Normative References

[I-D.ietf-anima-grasp] Bormann, C., Carpenter, B., and B. Liu, "A Generic Autonomic Signaling Protocol (GRASP)", draft-ietf-anima- grasp-10 (work in progress), March 2017.

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

[RFC7665] Halpern, J., Pignataro, C., "Service Function Chaining (SFC) Architecture", October 2015 <https://tools.ietf.org/html/rfc7665>.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <http://www.rfc-editor.org/info/rfc2629>.

14.2. Informative References

[ChinaCom09] A. Galis et all - "Management and Service-aware Networking Architectures (MANA) for Future Internet" - Invited paper IEEE 2009 Fourth International Conference on Communications and Networking in China (ChinaCom09) 26-28 August 2009, Xi'an, China, <http://www.chinacom.org/2009/index.html>.

[GENI] "GENI Key Concepts - Global Environment for Network Innovations (GENI)" <http://groups.geni.net/geni/wiki/GENIConcepts>.

[I-D.du-anima-an-intent] Du, Z., Jiang, S., Nobre, J., Ciavaglia, L., and M. Behringer, "ANIMA Intent Policy and Format", draft-du- anima-an-intent-04 (work in progress), July 2016.

[I-D.ietf-anima-autonomic-control-plane] Behringer, M., Eckert, T., and S. Bjarnason, "An Autonomic Control Plane", draft-ietf-anima-autonomic-control- plane-03 (work in progress), July 2016.

[I-D.ietf-anima-reference-model] Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L., Pierre, P., Liu, B., Nobre, J., and J. Strassner, "A Reference Model for Autonomic Networking", draft-ietf- anima-reference-model-02 (work in progress), July 2016.

[I-D.liu-anima-grasp-api] Carpenter, B., Liu, B., Wang, W., and X. Gong, "Generic Autonomic Signaling Protocol Application Program Interface (GRASP API)", draft-liu-anima-grasp-api-02 (work in progress), September 2016.

[I-D.liu-anima-grasp-distribution] Liu, B. and S. Jiang, "Information Distribution over GRASP", draft-liu-anima-grasp-distribution-02 (work in progress), September 2016.

[I-D.strassner-anima-control-loops] Strassner, J., Halpern, J., and M. Behringer, "The Use of Control Loops in Autonomic Networking", draft-strassner- anima-control-loops-01 (work

in progress), April 2016.

[IMT2020] ITU-T IMT2020 document "Report on Gap Analysis" - ITU-T IMT2020 ITU- Dec 2015 Published by ITU-T IMT2020. <http://www.itu.int/en/ITU-T/focusgroups/imt-2020/Pages/default.aspx>.

[MANO] "ETSI European Telecommunications Standards Institute. Network Functions Virtualisation (NFV); Management and Orchestration v1.1.1." Website, December 2014. <http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_ nfv-man001v010101p.pdf>.

[NGMN] Hedmar,P., Mschner, K., et all - NGMN Alliance document "Description of Network Slicing Concept", January 2016. <https://www.ngmn.org/uploads/media/160113_Network_Slicing_v1_0.pdf>.

[NGS-3GPP] "Study on Architecture for Next Generation System" - latest version v1.0.2 September 2016 <http://www.3gpp.org/ftp/tsg_sa/WG2_Arch/Latest_SA2_Specs/Latest_draft_S2_Specs>.

[ONF] Paul, M, Schallen, S., Betts, M., Hood, D., Shirazipor, M., Lopes, D., Kaippallimalit, J., - Open Network Fundation document "Applying SDN Architecture to 5G Slicing", April 2016. <https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/Applying_SDN_Architecture_to_5G_Slicing_TR-526.pdf>.

[NS1] L. Geng, J. Dong, S. Bryant, K., Makhijani, A., Galis, X. de Foy, S. Kuklinski, - "Network Slicing Architecture", July 2017. <https://tools.ietf.org/html/draft-geng-netslices- architecture-02>.

[NS2] L. Geng, L. Wang, S. Kuklinski, L. Qiang, S. Matsushima, A., Galis, L. Contreras - "Problem Statement of Supervised Heterogeneous Network Slicing", October 2017 <https://datatracker.ietf.org/doc/draft-geng-coms-problem-statement/>.

[ASN] A., Galis, K., Makhijani, D. Yu, B. Liu - "Autonomic Slice Networking-Requirements and Reference Model" - May 2017 < https://datatracker.ietf.org/doc/draft-galis-anima-autonomic-slice-networking/>.

[RFC7575] Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A.,

Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals", RFC 7575, DOI 10.17487/RFC7575, June 2015, <http://www.rfc-editor.org/info/rfc7575>.

[RFC7576] Jiang, S., Carpenter, B., and M. Behringer, "General Gap Analysis for Autonomic Networking", RFC 7576, DOI 10.17487/RFC7576, July 2016, <http://www.rfc-editor.org/info/rfc7576>.

[TETT1] Guerzoni, R., Vaishnavi, I., Pares-Caparros, D., Galis, A., et al, "Analysis of End-to-End Multi Domain Management and Orchestration Frameworks for Software Defined Infrastructures: an Architectural Survey", Transactions on Emerging Telecommunications Technologies, Wiley Online Library, DOI: 10.1002/ett.3103, June 2016, <onlinelibrary.wiley.com/doi/10.1002/ett.3103/pdf>.

[TETT2] Karl, H., Draxler, S., Peuster, M, Galis, A., et all "DevOps for Network Function Virtualization: An Architectural Approach", Transactions on Emerging Telecommunications Technologies Wiley Online Library, DOI: 10.1002/ett.3084, July 2016, <http://onlinelibrary.wiley.com/doi/10.1002/ett.3084/full>.

[SERMODEL] C., Borman, B. Carpenter, B., Liu, "Service Models Explained " draft-wu-opsawg-service-model-explained-05 <https://datatracker.ietf.org/doc/draft-wu-opsawg-service-model- explained/>.

[5GNS] Galis, A. (UCL), Chih-Lin I (China Mobile) - "Towards 5G Network Slicing - Motivations and Challenges" March 2017, IEEE 5G Tech Focus, Volume 1, Number 1, March 2017- <http://5g.ieee.org/tech-focus/march-2017#networkslicing>.

[PER-NS] Galis, A. - " Perspectives on Network Slicing - Towards the New ’Bread and Butter’ of Networking and Servicing", IEEE SDN Initiative - January 2018 <https://sdn.ieee.org/newsletter/january-2018/perspectives-on- network-slicing-towards-the-new-bread-and-butter-of-networking-and-servicing>.

[NS-ETSI] "Network Functions Virtualisation (NFV) Release 3; Evolution and Ecosystem; Report on Network Slicing Support with ETSI NFV Architecture Framework- ETSI GR NFV-EVE 012 V3.1.1 (2017-12)" <http://www.etsi.org/deliver/etsi_gr/NFV-

EVE/001_099/012/03.01.01_60/gr_NFV-EVE012v030101p.pdf>

Authors' Addresses

Alex Galis (editor)
University College London
Department of Electronic and Electrical Engineering
Torrington Place
London WC1E 7JE
United Kingdom

Email: a.galis@ucl.ac.uk

Kiran Makhijani
Huawei Technologies
2890, Central Expressway
Santa Clara CA 95032
USA

Email: USA Email: kiran.makhijani@huawei.com

Delei Yu
Huawei Technologies
Q22, Huawei Campus
No.156 Beiqing Road
Hai-Dian District, Beijing 100095
P.R. China

Email: yudelei@huawei.com

Bing Liu
Huawei Technologies Co., Ltd
Q14, Huawei Campus
No.156 Beiqing Road
Hai-Dian District, Beijing 100095
P.R. China

Email: leo.liubing@huawei.com

SDN Research Group
Internet-Draft
Intended status: Standards Track
Expires: May 4, 2017

LM. Contreras
Telefonica
CJ. Bernardos
UC3M
D. Lopez
Telefonica
M. Boucadair
Orange
P. Iovanna
Ericsson

October 31, 2016

Cooperating Layered Architecture for SDN
draft-irtf-sdnrg-layered-sdn-01

Abstract

Software Defined Networking proposes the separation of the control plane from the data plane in the network nodes and its logical centralization on a control entity. Most of the network intelligence is moved to this functional entity. Typically, such entity is seen as a compendium of interacting control functions in a vertical, tight integrated fashion. The relocation of the control functions from a number of distributed network nodes to a logical central entity conceptually places together a number of control capabilities with different purposes. As a consequence, the existing solutions do not provide a clear separation between transport control and services that relies upon transport capabilities.

This document describes a proposal named Cooperating Layered Architecture for SDN. The idea behind that is to differentiate the control functions associated to transport from those related to services, in such a way that they can be provided and maintained independently, and can follow their own evolution path.

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 May 4, 2017.

Copyright Notice

Copyright (c) 2016 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

Software Defined Networking (SDN) proposes the separation of the control plane from the data plane in the network nodes and its logical centralization on a control entity. A programmatic interface is defined between such entity and the network nodes, which functionality is supposed to perform traffic forwarding (only). Through that interface, the control entity instructs the nodes involved in the forwarding plane and modifies their traffic forwarding behavior accordingly.

Most of the intelligence is moved to such functional entity. Typically, such entity is seen as a compendium of interacting control functions in a vertical, tight integrated fashion.

This approach presents a number of issues:

- Unclear responsibilities between actors involved in a service provision and delivery.
- Complex reuse of functions for the provision of services.
- Closed, monolithic control architectures.
- Difficult interoperability and interchangeability of functional components.
- Blurred business boundaries among providers.
- Complex service/network diagnosis and troubleshooting, particularly to determine which segment is responsible for a failure.

The relocation of the control functions from a number of distributed network nodes to another entity conceptually places together a number of control capabilities with different purposes. As a consequence, the existing solutions do not provide a clear separation between services and transport control.

This document describes a proposal named Cooperating Layered Architecture for SDN (CLAS). The idea behind that is to differentiate the control functions associated to transport from those related to services, in such a way that they can be provided and maintained independently, and can follow their own evolution path.

Despite such differentiation it is required a close cooperation between service and transport layers and associated components to provide an efficient usage of the resources.

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 [RFC2119].

This document makes use of the following terms:

- Transport: denotes the transfer capabilities offered by a networking infrastructure. The transfer capabilities can rely upon pure IP techniques, or other means such as MPLS or optics.

- Service: denote a logical construct that make use of transport capabilities. This document does not make any assumption on the functional perimeter of a service that can be built above a transport infrastructure. As such, a service can be an offering that is offered to customers or be invoked for the delivery of another (added-value) service.

- SDN intelligence: refers to the decision-making process that is hosted by a node or a set of nodes. The intelligence can be centralized or distributed. Both schemes are within the scope of this document. The SDN intelligence relies on inputs form various functional blocks such as: network topology discovery, service topology discovery, resource allocation, business guidelines, customer profiles, service profiles, etc. The exact decomposition of an SDN intelligence, apart from the layering discussed in this document, is out of scope.

Additionally, the following acronyms are used in this document.

CLAS: Cooperating Layered Architecture for SDN

FCAPS: Fault, Configuration, Accounting, Performance and Security

SDN: Software Defined Networking

SLA: Service Level Agreement

3. Architecture overview

Current operator networks support multiple services (e.g., VoIP, IPTV, mobile VoIP, critical mission applications, etc.) on a variety of transport technologies. The provision and delivery of a service independently of the underlying transport capabilities requires a separation of the service related functionalities and an abstraction of the transport network to hide the specificities of underlying transfer techniques while offering a common set of capabilities.

Such separation can provide configuration flexibility and adaptability from the point of view of either the services or the transport network. Multiple services can be provided on top of a common transport infrastructure, and similarly, different technologies can accommodate the connectivity requirements of a certain service. A close coordination among them is required for a consistent service delivery (inter-layer cooperation).

This document focuses particularly on:

- Means to expose transport capabilities to external services.
- Means to capture service requirements of services.
- Means to notify service intelligence with underlying transport events, for example to adjust service decision-making process with underlying transport events.
- Means to instruct the underlying transport capabilities to accommodate new requirements, etc.

An example is to guarantee some Quality of Service (QoS) levels. Different QoS-based offerings could be present at both service and transport layers. Vertical mechanisms for linking both service and transport QoS mechanisms should be in place to provide the quality guarantees to the end user.

CLAS architecture assumes that the logically centralized control functions are separated in two functional blocks or layers. One of the functional blocks comprises the service-related functions, whereas the other one contains the transport-related functions. The cooperation between the two layers is considered to be implemented through standard interfaces.

Figure 1 shows the CLAS architecture. It is based on functional separation in the NGN architecture defined by the ITU-T in [Y.2011].

Two strata of functionality are defined, namely the Service Stratum, comprising the service-related functions, and the Connectivity Stratum, covering the transport ones. The functions on each of these layers are further grouped on control, management and user (or data) planes.

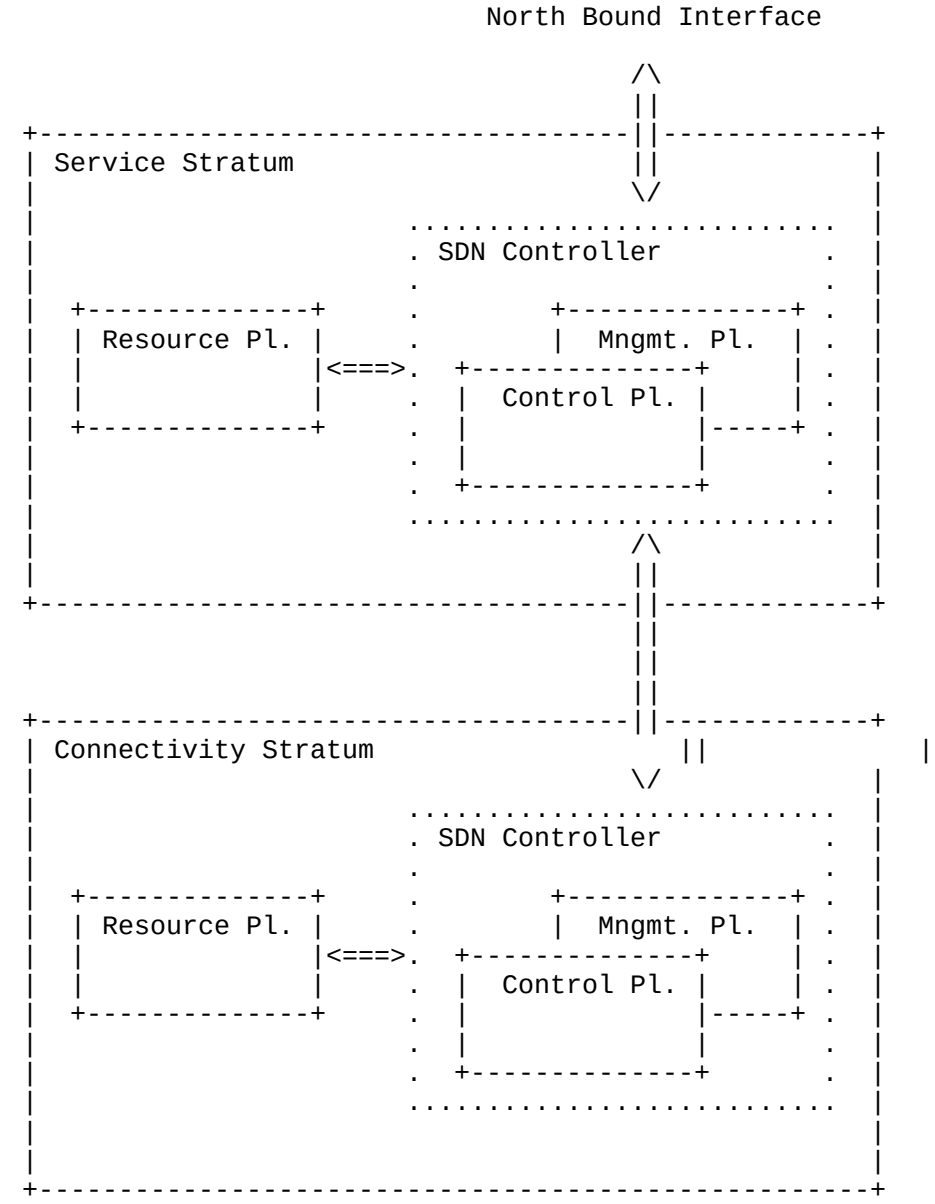


Figure 1: Cooperating Layered Architecture for SDN

In the CLAS architecture both the control and management functions are the ones logically centralized in one or a set of SDN controllers, in such a way that separated SDN controllers are present

in the Service and Connectivity strata. Furthermore, the generic user or data plane functions included in the NGN architecture are referred here as resource plane functions. The resource plane in each stratum is controlled by the corresponding SDN controller through a standard interface.

The SDN controllers cooperate for the provision and delivery of services. There is a hierarchy in which the Service SDN controller requests transport capabilities to the Transport SDN controller. Furthermore, the Transport SDN controller interacts with the Service SDN controller to inform it about events in the transport network that can motivate actions in the service layer.

The Service SDN controller acts as a client of the Transport SDN controller.

Despite it is not shown in the figure, the Resource planes of each stratum could be connected. This will depend on the kind of service provided. Furthermore, the Service stratum could offer an interface towards external applications to expose network service capabilities to those applications or customers.

This document does assume that SDN techniques can be enabled jointly with other distributed means (e.g., IGP).

3.1. Functional strata

As described before, the functional split separates transport-related functions from service-related functions. Both strata cooperate for a consistent service delivery.

Consistecy is determined and characterized by the service layer.

Communication between these two components could be implemented using a variety of means (such as [I-D.boucadair-connectivity-provisioning-protocol], Intermediate-Controller Plane Interface (I-CPI) [ONFArch], etc).

3.1.1. Connectivity stratum

The Connectivity stratum comprises the functions focused on the transfer of data between the communication end points (e.g., between end-user devices, between two service gateways, etc.). The data forwarding nodes are controlled and managed by the Transport SDN component. The Control plane in the SDN controller is in charge of instructing the forwarding devices to build the end to end data path for each communication or to make sure forwarding service is appropriately setup. Forwarding may not be rely on the sole pre-

configured entries; dynamic means can be enabled so that involved nodes can build dynamically routing and forwarding paths. Finally, the Management plane performs management functions (i.e., FCAPS) on those devices, like fault or performance management, as part of the Connectivity stratum capabilities.

3.1.2. Service stratum

The Service stratum contains the functions related to the provision of services and the capabilities offered to external applications. The Resource plane consists of the resources involved in the service delivery, such as computing resources, registries, databases, etc. The Control plane is in charge of controlling and configuring those resources, as well as interacting with the Control plane of the Transport stratum in client mode for requesting transport capabilities for a given service. In the same way, the Management plane implements management actions on the service-related resources and interacts with the Management plane in the Connectivity stratum for a cooperating management between layers.

3.1.3. Recursiveness

Recursive layering can happen in some usage scenarios in which the Connectivity Stratum is itself structured in Service and Connectivity Stratum. This could be the case of the provision of a transport services complemented with advanced capabilities additional to the pure data transport (e.g., maintenance of a given SLA [RFC7297]).

3.2. Plane separation

The CLAS architecture leverages on the SDN proposition of plane separation. As mentioned before, three different planes are considered for each stratum. The communication among these three planes (and with the corresponding plane in other strata) is based on open, standard interfaces.

3.2.1. Control Plane

The Control plane logically centralizes the control functions of each stratum and directly controls the corresponding resources. [RFC7426] introduces the role of the control plane in a SDN architecture. This plane is part of an SDN controller, and can interact with other control planes in the same or different strata for accomplishing control functions.

3.2.2. Management Plane

The Management plane logically centralizes the management functions for each stratum, including the management of the Control and Resource planes. [RFC7426] describes the functions of the management plane in a SDN environment. This plane is also part of the SDN controller, and can interact with the corresponding management planes residing in SDN controllers of the same or different strata.

3.2.3. Resource Plane

The Resource plane comprises the resources for either the transport or the service functions. In some cases the service resources can be connected to the transport ones (e.g., being the terminating points of a transport function) whereas in other cases it can be decoupled from the transport resources (e.g., one database keeping some register for the end user). Both forwarding and operational planes proposed in [RFC7426] would be part of the Resource plane in this architecture.

4. Required features

A number of features are required to be supported by the CLAS architecture.

- Abstraction: the mapping of physical resources into the corresponding abstracted resources.
- Service parameter translation: translation of service parameters (e.g., in the form of SLAs) to transport parameters (or capabilities) according to different policies.
- Monitoring: mechanisms (e.g. event notifications) available in order to dynamically update the (abstracted) resources' status taking in to account e.g. the traffic load.
- Resource computation: functions able to decide which resources will be used for a given service request. As an example, functions like PCE could be used to compute/select/decide a certain path.
- Orchestration: ability to combine diverse resources (e.g., IT and network resources) in an optimal way.
- Accounting: record of resource usage.
- Security: secure communication among components, preventing e.g. DoS attacks.

5. Communication between SDN Controllers

The SDN Controller residing respectively in the Service and the Connectivity Stratum need to establish a tight coordination. Mechanisms for transfer relevant information for each stratum should be defined.

From the Service perspective, the Service SDN controller needs to easily access transport resources through well defined APIs to access the capabilities offered by the Connectivity Stratum. There could be different ways of obtainign such transport-aware information, i.e., by discovering or publishing mechanisms. In the former case the Service SDN Controller could be able of handling complete information about the transport capabilities (including resources) offered by the Connectivity Stratum. In the latter case, the Connectivity Stratum exposes available capabilities e.g. through a catalog, reducing the amount of detail of the underlying network.

On the other hand, the Connectivity Stratum requires to properly capture Service requirements. These can include SLA requirements with specific metrics (such as delay), level of protection to be provided, max/min capacity, applicable resource constraints, etc.

The communication between controllers should be also secure, preventing denial of service.

6. Deployment scenarios

Different situations can be found depending on the characteristics of the networks involved in a given deployment.

6.1. Full SDN environments

This case considers the fact that the networks involved in the provision and delivery of a given service have SDN capabilities.

6.1.1. Multiple Service strata associated to a single Connectivity stratum

A single Connectivity stratum can provide transfer functions to more than one Service strata. The Connectivity stratum offers a standard interface to each of the Service strata. The Service strata are the clients of the Connectivity stratum. Some of the capabilities offered by the Connectivity stratum can be isolation of the transport resources (slicing), independent routing, etc.

6.1.2. Single service stratum associated to multiple Connectivity strata

A single Service stratum can make use of different Connectivity strata for the provision of a certain service. The Service stratum interfaces each of the Connectivity strata with standard protocols, and orchestrates the provided transfer capabilities for building the end to end transport needs.

6.2. Hybrid environments

This case considers scenarios where one of the strata is legacy totally or in part.

6.2.1. SDN Service stratum associated to a legacy Connectivity stratum

An SDN service stratum can interact with a legacy Connectivity stratum through some interworking function able to adapt SDN-based control and management service-related commands to legacy transport-related protocols, as expected by the legacy Connectivity stratum. The SDN controller in the Service stratum is not aware of the legacy nature of the underlying Connectivity stratum.

6.2.2. Legacy Service stratum associated to an SDN Connectivity stratum

A legacy Service stratum can work with an SDN-enabled Connectivity stratum through the mediation of and interworking function capable to interpret commands from the legacy service functions and translate them into SDN protocols for operating with the SDN-enabled Connectivity stratum.

6.3. Multi-domain scenarios in Connectivity Stratum

The Connectivity Stratum can be composed by transport resources being part of different administrative, topological or technological domains. The Service Stratum can yet interact with a single entity in the Connectivity Stratum in case some abstraction capabilities are provided in the transport part to emulate a single stratum.

Those abstraction capabilities constitute a service itself offered by the Connectivity Stratum to the services making use of it. This service is focused on the provision of transport capabilities, then different of the final communication service using such capabilities.

In this particular case this recursion allows multi-domain scenarios at transport level.

Multi-domain situations can happen in both single-operator and multi-operator scenarios. Multi-operator scenarios will be addressed in future versions of the document.

In single operator scenarios a multi-domain or end-to-end abstraction component can provide an homogeneous abstract view of the underlying heterogeneous transport capabilities for all the domains.

7. Use cases

This section presents a number of use cases as examples of applicability of this proposal

7.1. Network Function Virtualization

NFV environments offer two possible levels of SDN control [ETSI_NFV_EVE005]. One level is the need for controlling the NFVI to provide connectivity end-to- end among VNFs (Virtual Network Functions) or among VNFs and PNFs (Physical Network Functions). A second level is the control and configuration of the VNFs themselves (in other words, the configuration of the network service implemented by those VNFs), taking profit of the programmability brought by SDN. Both control concerns are separated in nature. However, interaction between both could be expected in order to optimize, scale or influence each other.

7.2. Abstraction and Control of Transport Networks

To be completed.

8. IANA Considerations

TBD.

9. Security Considerations

TBD. Security in the communication between strata to be addressed.

10. References

10.1. 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, <http://www.rfc-editor.org/info/rfc2119>.

[Y.2011] "General principles and general reference model for Next Generation Networks", ITU-T Recommendation Y.2011 , October 2004.

10.2. Informative References

[ETSI_NFV_EVE005] "Report on SDN Usage in NFV Architectural Framework", Dicember 2015.

[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-12 (work in progress), June 2016.

[ONFArch] Open Networking Foundation, "SDN Architecture, Issue 1", June 2014, <https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf>.

[RFC7297] Boucadair, M., Jacquenet, C., and N. Wang, "IP Connectivity Provisioning Profile (CPP)", RFC 7297, DOI 10.17487/RFC7297, July 2014, <http://www.rfc-editor.org/info/rfc7297>.

[RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, DOI 10.17487/RFC7426, January 2015, <http://www.rfc-editor.org/info/rfc7426>.

Authors' Addresses

Luis M. Contreras
Telefonica
Ronda de la Comunicacion, s/n
Sur-3 building, 3rd floor
Madrid 28050
Spain

Email: luismiguel.contrerasmurillo@telefonica.com
URI: http://lmcontreras.com

Carlos J. Bernardos
Universidad Carlos III de Madrid
Av. Universidad, 30
Leganes, Madrid 28911
Spain

Phone: +34 91624 6236
Email: cjbc@it.uc3m.es
URI: http://www.it.uc3m.es/cjbc/

Diego R. Lopez
Telefonica
Ronda de la Comunicacion, s/n
Sur-3 building, 3rd floor
Madrid 28050
Spain

Email: diego.r.lopez@telefonica.com

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Paola Iovanna
Ericsson
Pisa
Italy

Email: paola.iovanna@ericsson.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 9, 2017

A. Kumar
J. Kolhe
S. Ghemawat
L. Ryan
Google

July 8, 2016

gRPC Protocol
draft-kumar-rtgwg-grpc-protocol-00

Abstract

This document presents gRPC protocol specification.

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 January 9, 2017.

Copyright Notice

Copyright (c) 2016 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

This document serves as a detailed description for an implementation of gRPC carried over HTTP2 draft 17 framing. It assumes familiarity with the HTTP2 specification.

2. Outline

The following is the general sequence of message atoms in a GRPC request and response message stream.

- Request -> Request-Headers *Length-Prefixed-Message EOS
- Response -> (Response-Headers *Length-Prefixed-Message Trailers) / Trailers-Only

3. Protocol Requests

At a high level, the protocol has the following request and response fields.

- Request-Headers -> Call-Definition *Custom-Metadata
- Call-Definition -> Method Scheme Path TE [Authority] [Timeout] Content-Type [Message-Type] [Message-Encoding] [Message-Accept-Encoding] [User-Agent]
- Method -> ":method POST"
- Scheme -> ":scheme " ("http" / "https")
- Path -> ":path" {path identifying method within exposed API}
- Authority -> ":authority" {virtual host name of authority}
- TE -> "te" "trailers" # Used to detect incompatible proxies

- Timeout -> "grpc-timeout" TimeoutValue TimeoutUnit
- TimeoutValue -> {positive integer as ASCII string of at most 8 digits}
- TimeoutUnit -> Hour / Minute / Second / Millisecond / Microsecond / Nanosecond
- Hour -> "H"
- Minute -> "M"
- Second -> "S"
- Millisecond -> "m"
- Microsecond -> "u"
- Nanosecond -> "n"
- Content-Type -> "content-type" "application/grpc" [("+proto" / "+json" / {custom})]
- Content-Coding -> "identity" / "gzip" / "deflate" / "snappy" / {custom}
- Message-Encoding -> "grpc-encoding" Content-Coding
- Message-Accept-Encoding -> "grpc-accept-encoding" Content-Coding *("," Content-Coding)
- User-Agent -> "user-agent" {structured user-agent string}
- Message-Type -> "grpc-message-type" {type name for message schema}
- Custom-Metadata -> Binary-Header / ASCII-Header
- Binary-Header -> {Header-Name "-bin" } {base64 encoded value}
- ASCII-Header -> Header-Name ASCII-Value
- Header-Name -> 1*(%x30-39 / %x61-7A / "_" / "-" / ".") ; 0-9 a-z _ - .
- ASCII-Value -> 1*(%x20-%x7E) ; space and printable ASCII

HTTP2 requires that reserved headers, ones starting with ":" appear before all other headers. Additionally implementations should send

Timeout immediately after the reserved headers and they should send the Call-Definition headers before sending Custom-Metadata.

If Timeout is omitted a server should assume an infinite timeout. Client implementations are free to send a default minimum timeout based on their deployment requirements.

Custom-Metadata is an arbitrary set of key-value pairs defined by the application layer. Header names starting with "grpc-" but not listed here are reserved for future GRPC use and should not be used by applications as Custom-Metadata.

Note that HTTP2 does not allow arbitrary octet sequences for header values so binary header values must be encoded using Base64 as per https://tools.ietf.org/html/rfc4648#section-4. Implementations MUST accept padded and un-padded values and should emit un-padded values. Applications define binary headers by having their names end with "-bin". Runtime libraries use this suffix to detect binary headers and properly apply base64 encoding and decoding as headers are sent and received.

Custom-Metadata header order is not guaranteed to be preserved except for values with duplicate header names. Duplicate header names may have their values joined with "," as the delimiter and be considered semantically equivalent. Implementations must split Binary-Headers on "," before decoding the Base64-encoded values.

ASCII-Value should not have leading or trailing whitespace. If it contains leading or trailing whitespace, it may be stripped. The ASCII-Value character range defined is more strict than HTTP. Implementations must not error due to receiving an invalid ASCII-Value that’s a valid field-value in HTTP, but the precise behavior is not strictly defined: they may throw the value away or accept the value. If accepted, care must be taken to make sure that the application is permitted to echo the value back as metadata. For example, if the metadata is provided to the application as a list in a request, the application should not trigger an error by providing that same list as the metadata in the response. Servers may limit the size of Request-Headers, with a default of 8 KiB suggested. Implementations are encouraged to compute total header size like HTTP/2’s SETTINGS_MAX_HEADER_LIST_SIZE: the sum of all header fields, for each field the sum of the uncompressed field name and value lengths plus 32, with binary values’ lengths being post-Base64.

Servers may limit the size of Request-Headers, with a default of 8 KiB suggested. Implementations are encouraged to compute total header size like HTTP/2’s SETTINGS_MAX_HEADER_LIST_SIZE: the sum of all header fields, for each field the sum of the uncompressed field

name and value lengths plus 32, with binary values' lengths being post-Base64.

The repeated sequence of Length-Prefixed-Message items is delivered in DATA frames.

- Length-Prefixed-Message -> Compressed-Flag Message-Length Message
- Compressed-Flag -> 0 / 1 # encoded as 1 byte unsigned integer
- Message-Length -> {length of Message} # encoded as 4 byte unsigned integer
- Message \u002D-> *{binary octet}

A Compressed-Flag value of 1 indicates that the binary octet sequence of Message is compressed using the mechanism declared by the Message-Encoding header. A value of 0 indicates that no encoding of Message bytes has occurred. Compression contexts are NOT maintained over message boundaries, implementations must create a new context for each message in the stream. If the Message-Encoding header is omitted then the Compressed-Flag must be 0.

For requests, EOS (end-of-stream) is indicated by the presence of the END_STREAM flag on the last received DATA frame. In scenarios where the Request stream needs to be closed but no data remains to be sent implementations MUST send an empty DATA frame with this flag set.

4. Responses

- Response -> (Response-Headers *Length-Prefixed-Message Trailers) / Trailers-Only
- Response-Headers -> HTTP-Status [Message-Encoding] [Message-Accept-Encoding] Content-Type *Custom-Metadata
- Trailers-Only -> HTTP-Status Content-Type Trailers
- Trailers -> Status [Status-Message] *Custom-Metadata
- HTTP-Status -> ":status 200"
- Status -> "grpc-status"
- Status-Message -> "grpc-message"

Response-Headers and Trailers-Only are each delivered in a single HTTP2 HEADERS frame block. Most responses are expected to have both

headers and trailers but Trailers-Only is permitted for calls that produce an immediate error. Status must be sent in Trailers even if the status code is OK.

For responses end-of-stream is indicated by the presence of the END_STREAM flag on the last received HEADERS frame that carries Trailers.

Implementations should expect broken deployments to send non-200 HTTP status codes in responses as well as a variety of non-GRPC content-types and to omit Status and Status-Message. Implementations must synthesize a Status and Status-Message to propagate to the application layer when this occurs.

Clients may limit the size of Response-Headers, Trailers, and Trailers-Only, with a default of 8 KiB each suggested.

5. Examples

Sample unary-call showing HTTP2 framing sequence

Request

```
HEADERS (flags = END_HEADERS)
:method = POST
:scheme = http
:path = /google.pubsub.v2.PublisherService/CreateTopic
:authority = pubsub.googleapis.com
grpc-timeout = 1S
content-type = application/grpc+proto
grpc-encoding = gzip
authorization = Bearer y235.wef315yfh138vh31hv93hv8h3v

DATA (flags = END_STREAM)
<Length-Prefixed Message>
```

Response

```
HEADERS (flags = END_HEADERS)
:status = 200
grpc-encoding = gzip

DATA
<Length-Prefixed Message>

HEADERS (flags = END_STREAM, END_HEADERS)
grpc-status = 0 # OK
trace-proto-bin = jher831yy13JHy3hc
```

6. User Agents

While the protocol does not require a user-agent to function it is recommended that clients provide a structured user-agent string that provides a basic description of the calling library, version and platform to facilitate issue diagnosis in heterogeneous environments. The following structure is recommended to library developers

User-Agent \u002D-> "grpc-" Language ?("-" Variant) "/" Version ?(" (" *(AdditionalProperty ";") ")")

7. HTTP2 Transport Mapping

All GRPC calls need to specify an internal ID. We will use HTTP2 stream-ids as call identifiers in this scheme. NOTE: These id's are contextual to an open HTTP2 session and will not be unique within a given process that is handling more than one HTTP2 session nor can they be used as GUIDs.

DATA frame boundaries have no relation to Length-Prefixed-Message boundaries and implementations should make no assumptions about their alignment.

When an application or runtime error occurs during an RPC a Status and Status-Message are delivered in Trailers. In some cases it is possible that the framing of the message stream has become corrupt and the RPC runtime will choose to use an RST_STREAM frame to indicate this state to its peer. RPC runtime implementations should interpret RST_STREAM as immediate full-closure of the stream and should propagate an error up to the calling application layer.

HTTP2 Code#	GRPC Code
NO_ERROR(0)	INTERNAL - An explicit GRPC status of OK should have been sent but this might be used to aggressively lameduck in some scenarios.
PROTOCOL_ERROR(1)	INTERNAL
INTERNAL_ERROR(2)	INTERNAL
FLOW_CONTROL_ERROR(3)	INTERNAL
SETTINGS_TIMEOUT(4)	INTERNAL
STREAM_CLOSED	FRAME_SIZE_ERROR
STREAM_CLOSED	INTERNAL
REFUSED_STREAM	UNAVAILABLE - Indicates that no processing occurred and the request can be retried, possibly elsewhere.
CANCEL(8)	Mapped to call cancellation when sent by a client.Mapped to CANCELLED when sent by a server. Note that servers should only use this mechanism when they need to cancel a call but the payload byte sequence is incomplete.
COMPRESSION_ERROR	INTERNAL
CONNECT_ERROR	INTERNAL
ENHANCE_YOUR_CALM	RESOURCE_EXHAUSTED ...with additional error detail provided by runtime to indicate that the exhausted resource is bandwidth.
INADEQUATE_SECURITY	PERMISSION_DENIED ... with additional detail indicating that permission was denied as protocol is not secure enough for call.

Table 1: Error Code Mapping

The HTTP2 specification mandates the use of TLS 1.2 or higher when TLS is used with HTTP2. It also places some additional constraints on the allowed ciphers in deployments to avoid known-problems as well as requiring SNI support. It is also expected that HTTP2 will be used in conjunction with proprietary transport security mechanisms about which the specification can make no meaningful recommendations.

GOAWAY Frame Sent by servers to clients to indicate that they will no longer accept any new streams on the associated connections. This frame includes the id of the last successfully accepted stream by the server. Clients should consider any stream initiated after the last successfully accepted stream as UNAVAILABLE and retry the call elsewhere. Clients are free to continue working with the already accepted streams until they complete or the connection is terminated. Servers should send GOAWAY before terminating a connection to reliably inform clients which work has been accepted by the server and is being executed.

PING Frame Both clients and servers can send a PING frame that the peer must respond to by precisely echoing what they received. This is used to assert that the connection is still live as well as providing a means to estimate end-to-end latency. If a server initiated PING does not receive a response within the deadline expected by the runtime all outstanding calls on the server will be closed with a CANCELLED status. An expired client initiated PING will cause all calls to be closed with an UNAVAILABLE status. Note that the frequency of PINGs is highly dependent on the network environment, implementations are free to adjust PING frequency based on network and application requirements.

Connection failure If a detectable connection failure occurs on the client all calls will be closed with an UNAVAILABLE status. For servers open calls will be closed with a CANCELLED status.

8. Normative references

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <http://www.rfc-editor.org/info/rfc7540>.

Authors' Addresses

Abhishek Kumar
Google
1600 Amphitheatre Pkwy
Mountain View, CA 94043
US

Email: abhikumar@google.com

Jayant Kolhe
Google
1600 Amphitheatre Pkwy
Mountain View, CA 94043
US

Email: jkolhe@google.com

Sanjay Ghemawat
Google
1600 Amphitheatre Pkwy
Mountain View, CA 94043
US

Email: sanjay@google.com

Louis Ryan
Google
1600 Amphitheatre Pkwy
Mountain View, CA 94043
US

Email: lryan@google.com

Network Working Group   Z. Li
Internet-Draft   J. Zhang
Intended status: Informational   Huawei Technologies
Expires: May 4, 2017   October 31, 2016

An Architecture of Network Artificial Intelligence(NAI)
draft-li-rtgwg-network-ai-arch-00

Abstract

Artificial intelligence is an important technical trend in the industry. With the development of network, it is necessary to introduce artificial intelligence technology to achieve self-adjustment, self- optimization, self-recovery of the network through collection of huge data of network state and machine learning. This draft defines the architecture of Network Artificial Intelligence (NAI), including the key components and the key protocol extension requirements.

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 inRFC 2119 [RFC2119]

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 May 4, 2017.

Copyright Notice

Copyright (c) 2016 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

Artificial Intelligence is an important technical trend in the industry. The two key aspects of Artificial Intelligence are perception and cognition. Artificial Intelligence has evolved from an early non-learning expert system to a learning-capable machine learning era. In recent years, the rapid development of the deep learning branch based on the neural network and the maturity of the big data technology and software distributed architecture make the Artificial Intelligence in many fields (such as transportation, medical treatment, education, etc.) have been applied. With the development of network, it is necessary to introduce artificial intelligence technology to achieve self-adjustment, self-optimization, self-recovery of the network through collection of huge data of network state and machine learning. The areas of machine learning which are easier to be used in the network field may include: troubleshooting of network problems, network traffic prediction, traffic optimization adjustment, security defense, security auditing, etc., to implement network perception and cognition.

This draft defines the architecture of Network Artificial Intelligence (NAI), including the key components and the key protocol extension requirements.

2. Terminology

AI: Artificial Intelligence

NAI: Network Artificial Intelligence

3. Architecture

3.1. Reference Model

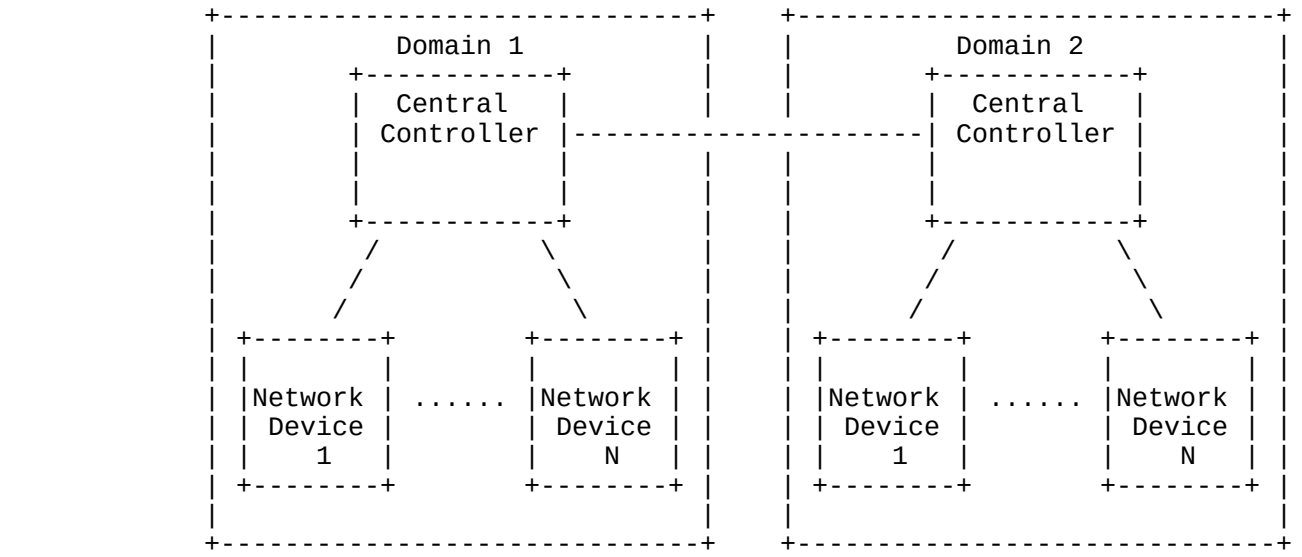


Figure 1: An Architecture of Network Artificial Intelligence(NAI)

The architecture of Network artificial intelligence includes following key component:

1. Central Controller: Centralized controller is the core component of Network Artificial Intelligence which can be called as 'Network Brain'. It man collect huge data of network states, store the data based on the big data platform, and carry on the machine learning, to achieve network perception and cognition, including network self-optimization, self- adjustment, self-recovery, intelligent fault location and a series of network artificial intelligence goals.

2. Network Device: IP network operation and maintenance are always a big challenge since the network can only provide limited state information. The network states includes but are not limited to topology, traffic engineering, operation and maintenance information, network failure information and related information to locate the

network failure.. In order to provide these information, the network must be able to support more OAM mechanisms to acquire more state information and report to the controller. Then the controller can get the complete state information of the network which is the base of Network Artificial Intelligence(NAI).

3. Southbound Protocol and Models of Controller: As network devices provide huge network state information, it proposes a number of new requirements for protocols and models between controllers and network devices. The traditional southbound protocol such as Netconf and SNMP can not meet the performance requirements. It is necessary to introduce some new high-performance protocols to collect network state data. At the same time, the models of network data should be completed. Moreover with the introduction of new OAM mechanisms of network devices, new models of network data should be introduced.

4. Northbound Model of Controller: The goal of the Network Artificial Intelligence is to reduce the technical requirements on the network administrators and release them from the heavy network management, control, maintenance work. The abstract northbound model of the controller for different network services should be simple and easy to be understood.

3.2. Requirement of Protocol Extensions

REQ 01: The new southbound protocol of the controller should be introduced to meet the performance requirements of collecting huge data of network states.

REQ 02: The models of network elements should be completed to collect the network states based on the new southbound protocol of the controller.

REQ 03: New OAM mechanisms should be introduced for the network devices in order to acquire more types of network state data.

REQ 04: New models of network elements should be introduced as the new OAM mechanisms are introduced.

REQ 05: The operation models of network elements should be completed based on the new southbound protocol to carry on the corresponding network operation as the result of Network Artificial Intelligence.

REQ 06: The abstract network-based service models should be provided by the controller as the northbound models to satisfy the requirements of different services.

4. IANA Considerations

This document makes no request of IANA.

5. Security Considerations

TBD.

6. 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, <http://www.rfc-editor.org/info/rfc2119>.

Authors' Addresses

Zhenbin Li
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: lizhenbin@huawei.com

Jinhui Zhang
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: jason.zhangjinhui@huawei.com