

NMRG  
Internet-Draft  
Intended status: Informational  
Expires: July 24, 2022

LM. Contreras  
Telefonica  
P. Lucente  
NTT  
January 20, 2022

Interconnection Intent  
draft-contreras-nmr-connection-intents-02

Abstract

This memo introduces the use case of the usage of intents for expressing advance interconnection features, further than traditional IP peering.

Status of This Memo

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

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

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

This Internet-Draft will expire on July 24, 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	2
2. Evolution of Network interconnection . . . . .	3
2.1. Potential interconnection intent types . . . . .	3
2.2. Interconnection intent lifecycle . . . . .	4
2.3. Protocol aspects . . . . .	4
3. Interconnection intent structure . . . . .	5
4. Security Considerations . . . . .	5
5. IANA Considerations . . . . .	5
6. References . . . . .	5
Acknowledgments . . . . .	5
Authors' Addresses . . . . .	5

## 1. Introduction

The success of Internet-based services has been built on top of the global reachability of content accessed by the end-users, which is facilitated by the interconnection of individual networks owned by distinct service providers constituting independent administrative domains.

Such interconnection services have been initially based simply on delivery of IP traffic between the interconnected parties leveraging on BGP. This peer model enables full connectivity. However, the traditional interconnection model shows some limitations when additional information to that related to routing is needed.

New network capabilities based on programmability and virtualization are producing service situations where a connectivity-only approach is not sufficient. The increasing availability of computing capabilities internal to the networks, or attached to them, enable new scenarios where those capabilities can be consumed through the advertisement or exposure of these execution environments (i.e., in terms of compute, storage and associated networking resources). Such information from an interconnected provider can be obtained from e.g. [I-D.llc-teas-dc-aware-topo-model].

In addition or complementary to that, even services or network functions could be advertised in order to make them available for interconnection. For example, as service we could consider the advertisement of CDN capabilities as in CDNi approach [RFC7336], while as network function we could consider functions like firewall, CGNAT, etc, present in the network [I-D.ietf-teas-sf-aware-topo-model].

All these scenarios present clear evolutions of the interconnection model which can not be simply expressed through existing mechanisms,

or at least, cannot be expressed in a simple (and comprehensive) way with such existing mechanisms. Here is where an advanced interconnection intent can assist on declaring the goal of the interconnection transcending pure IP traffic exchange and including more advance capabilities as the ones mentioned before.

## 2. Evolution of Network interconnection

It becomes clear the trend to increasingly rely on multi-domain scenarios for the provision of services. For instance, the access today to an on-demand OTT video on Internet implies the interaction of more than one single administrative domain. Thus, end-to-end service delivery over multiple providers or domains is becoming the norm.

Complex network services leveraging on virtualization solutions and different infrastructure environments pertaining to distinct administrative domains (i.e., operated and managed by distinct providers) can be easily foreseen.

It is then necessary to explore mechanisms for interconnecting that multiple domain environments in a common, portable way independently of the owner of such infrastructure.

### 2.1. Potential interconnection intent types

The interconnection intent should provide enough abstractions to express a variety of interconnection options.

The purpose of the interconnection intent can be multiple:

- o to enable multi-domain network service programming, by soliciting interconnection of service / network functions in different domains
- o to enable multi-domain deployment of virtualized network functions, by advertising the availability of compute and storage resources in different domains
- o to facilitate multi-domain network function or service charging, by advertising (cumulative) costs in the different domains
- o to enable traffic interchange, ie. IP as in traditional peering or optical
- o to put in place the right collection of policies to implement and operate the interconnection

- o to facilitate whatever combination of all of them

## 2.2. Interconnection intent lifecycle

[I-D.irtf-nmrg-ibn-concepts-definitions] defines an intent lifecycle composed of two phases, namely fulfillment and assurance. Figure 1 captures the intent procedure for the fulfillment phase (assurance phase will be detailed in future versions of this draft).

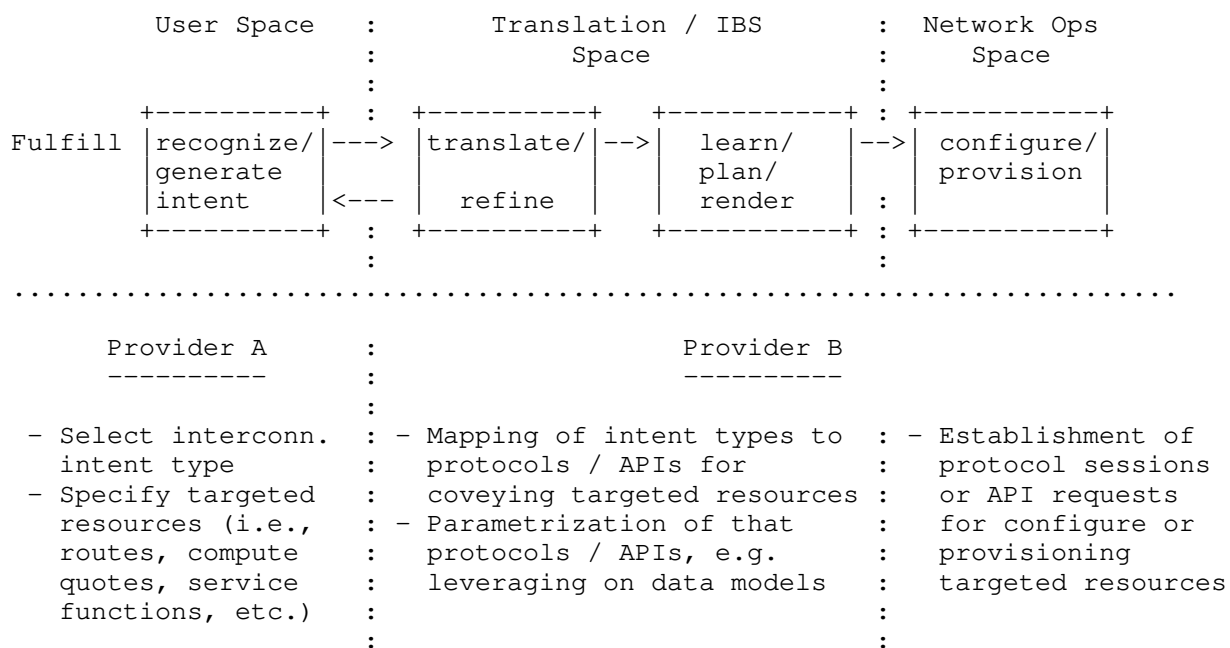


Figure 1: Fulfillment phase of the Interconnection Intent

## 2.3. Protocol aspects

Ultimately the ideas and notions elaborated in this document will need to find room in a framework made of one or multiple protocols (ie. BGP, LISP, etc.) and/or API definitions. While the exact definition of such framework is left as future work, in this document we intend to perform some seminal work in this sense (ie. identify existing protocols that could fit, determine gaps of such protocols, etc.).

### 3. Interconnection intent structure

To be done.

### 4. Security Considerations

To be done.

### 5. IANA Considerations

This draft does not include any IANA considerations

### 6. References

[I-D.ietf-teas-sf-aware-topo-model]

Bryskin, I., Liu, X., Lee, Y., Guichard, J., Contreras, L. M., Ceccarelli, D., Tantsura, J., and D. Shyti, "SF Aware TE Topology YANG Model", draft-ietf-teas-sf-aware-topo-model-08 (work in progress), July 2021.

[I-D.irtf-nmrg-ibn-concepts-definitions]

Clemm, A., Ciavaglia, L., Granville, L. Z., and J. Tantsura, "Intent-Based Networking - Concepts and Definitions", draft-irtf-nmrg-ibn-concepts-definitions-06 (work in progress), December 2021.

[I-D.llc-teas-dc-aware-topo-model]

Lee, Y., Liu, X., and L. M. Contreras, "DC aware TE topology model", draft-llc-teas-dc-aware-topo-model-01 (work in progress), July 2021.

[RFC7336] Peterson, L., Davie, B., and R. van Brandenburg, Ed., "Framework for Content Distribution Network Interconnection (CDNI)", RFC 7336, DOI 10.17487/RFC7336, August 2014, <<https://www.rfc-editor.org/info/rfc7336>>.

### Acknowledgments

This work has been partly funded by the European Commission through the H2020 project 5GROWTH (Grant Agreement no. 856709).

### 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](mailto:luismiguel.contrerasmurillo@telefonica.com)  
URI: <http://lmcontreras.com/>

Paolo Lucente  
NTT  
Siriusdreef 70-72  
Hoofddorp, WT 2132  
Netherlands

Email: [paolo@ntt.net](mailto:paolo@ntt.net)

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: 22 September 2022

A. Clemm  
Futurewei  
L. Ciavaglia  
Rakuten Mobile  
L. Z. Granville  
Federal University of Rio Grande do Sul (UFRGS)  
J. Tantsura  
Microsoft  
March 2022

Intent-Based Networking - Concepts and Definitions  
draft-irtf-nmrg-ibn-concepts-definitions-09

Abstract

Intent and Intent-Based Networking (IBN) are taking the industry by storm. At the same time, IBN-related terms are often used loosely and inconsistently, in many cases overlapping and confused with other concepts such as "Policy." This document clarifies the concept of "Intent" and provides an overview of the functionality that is associated with it. The goal is to contribute towards a common and shared understanding of terms, concepts, and functionality that can be used as the foundation to guide further definition of associated research and engineering problems and their solutions.

This document is a product of the IRTF Network Management Research Group (NMRG). It reflects the consensus of the research group, having received many detailed and positive reviews by RG participants. It is published for informational purposes.

Status of This Memo

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

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

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

This Internet-Draft will expire on 2 September 2022.

## Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	3
2. Definitions and Acronyms . . . . .	6
3. Introduction of Concepts . . . . .	7
3.1. Intent and Intent-Based Management . . . . .	7
3.2. Related Concepts . . . . .	11
3.2.1. Service Models . . . . .	11
3.2.2. Policy and Policy-Based Network Management . . . . .	13
3.2.3. Distinguishing between Intent, Policy, and Service Models . . . . .	15
4. Principles . . . . .	16
5. Intent-Based Networking - Functionality . . . . .	19
5.1. Intent Fulfillment . . . . .	19
5.1.1. Intent Ingestion and Interaction with Users . . . . .	19
5.1.2. Intent Translation . . . . .	20
5.1.3. Intent Orchestration . . . . .	20
5.2. Intent Assurance . . . . .	21
5.2.1. Monitoring . . . . .	21
5.2.2. Intent Compliance Assessment . . . . .	21
5.2.3. Intent Compliance Actions . . . . .	21
5.2.4. Abstraction, Aggregation, Reporting . . . . .	22
6. Lifecycle . . . . .	22
7. Additional Considerations . . . . .	24
8. IANA Considerations . . . . .	25
9. Security Considerations . . . . .	25
10. Acknowledgments . . . . .	26
11. Informative References . . . . .	26
Authors' Addresses . . . . .	29



## 1. Introduction

This document is a product of the IRTF Network Management Research Group (NMRG). It reflects the consensus of the RG, receiving reviews and explicit support from many participants. It is published for informational purposes.

In the past, interest regarding management and operations in the IETF has focused on individual network and device features. Standardization emphasis has generally been put on management instrumentation that needed to be provided to a networking device. A prime example of this is SNMP-based management [RFC3411] and the 200+ MIBs that have been defined by the IETF over the years. More recent examples include YANG data model definitions [RFC7950] for aspects such as interface configuration, ACL configuration, and Syslog configuration.

There is a clear sense and reality that managing networks by configuring myriads of "nerd knobs" on a device-by-device basis is no longer an option in modern network environments. Significant challenges arise with keeping device configurations not only consistent across a network but also consistent with the needs of services and service features they are supposed to enable. Additional challenges arise with regard to being able to rapidly adapt the network as needed and to be able to do so at scale. At the same time, operations need to be streamlined and automated wherever possible to not only lower operational expenses but also allow for rapid reconfiguration of networks at sub-second time scales and to ensure that networks are delivering their functionality as expected. Among other things, this requires the ability to consume operational data, perform analytics, and dynamically take actions in a way that is aware of context as well as intended outcomes at near real-time speeds.

Accordingly, the IETF has begun to address end-to-end management aspects that go beyond the realm of individual devices in isolation. Examples include the definition of YANG models for network topology [RFC8345] or the introduction of service models used by service orchestration systems and controllers [RFC8309]. Much of the interest has been fueled by the discussion about how to manage autonomic networks, as discussed in the ANIMA working group. Autonomic networks are driven by the desire to lower operational expenses and make the management of the network as a whole more straightforward, putting it at odds with the need to manage the network one device and one feature at a time. However, while autonomic networks are intended to exhibit "self-management" properties, they still require input from an operator or outside system to provide operational guidance and information about the goals, purposes, and service instances that the network is to serve.

This input and operational guidance are commonly referred to as "intent," and networks that allow network operators to provide their input using intent as "Intent-Based Networks" (IBN) and the systems that help implement intent as "Intent-Based Systems" (IBS). Those systems can manifest themselves in a number of ways, for example as a controller or management system that are implemented as an application that runs on a server or set of servers, or as a set of functions that are distributed across a network and that collectively perform their intent-based functionality.

However, intent is about more than just enabling a form of operator interaction with the network that involves higher-layer abstractions. It is also about the ability to let operators focus on what they want their desired outcomes to be while leaving details about how those outcomes would, in fact, be achieved to the IBN (respectively IBS). This, in turn, enables much greater operational efficiency at greater scale, in shorter time scales, with less dependency on human activities (and possibility for mistakes), and an ideal candidate, e.g., for artificial intelligence techniques that can bring about the next level of network automation [Clemm20].

This vision has since caught on with the industry, leading to a significant number of solutions that offer Intent-Based Management that promise network providers to manage networks holistically at a higher level of abstraction and as a system that happens to consist of interconnected components, as opposed to a set of independent devices (that happen to be interconnected). Those offerings include IBN and IBS (offering a full lifecycle of intent), SDN controllers (offering a single point of control and administration for a network), and network management and Operations Support Systems (OSS).

It has been recognized for a long time that comprehensive management solutions cannot operate only at the level of individual devices and low-level configurations. In this sense, the vision of intent is not entirely new. In the past, ITU-T's model of a Telecommunications Management Network (TMN) introduced a set of management layers that defined a management hierarchy, consisting of network element, network, service, and business management [M3010]. High-level operational objectives would propagate in a top-down fashion from upper to lower layers. The associated abstraction hierarchy was crucial to decompose management complexity into separate areas of concern. This abstraction hierarchy was accompanied by an information hierarchy that concerned itself at the lowest level with device-specific information, but that would, at higher layers, include, for example, end-to-end service instances. Similarly, the concept of Policy-Based Network Management (PBNM) has, for a long time, touted the ability to allow users to manage networks by specifying high-level management policies, with policy systems automatically "rendering" those policies, i.e., breaking them down into low-level configurations and control logic. (As a note, in the context of this document, "users" generally refers to operators and administrators who are responsible for the management and operation of communication services and networking infrastructures, not to "end users" of communication services.)

What has been missing, however, is putting these concepts into a more current context and updating them to account for current technology trends. This document clarifies the concepts behind intent. It differentiates intent from related concepts. It also provides an overview of first-order principles of IBN as well as the associated functionality. The goal is to contribute to a common and shared understanding that can be used as a foundation to articulate research and engineering problems in the area of IBN.

It should be noted that the articulation of IBN-related research problems is beyond the scope of this document. However, it should be recognized that IBN has become an important topic in the research community. Per IEEE Xplore [IEEEXPLORE], as of December 2021, in the past decade since 2012, there have been 1138 papers with the index term "intent", of which 411 specifically mention networking. The time period since 2020 alone accounts for 316 papers on intent and 153 for intent networking, indicating accelerating interest. In addition, workshops dedicated to this theme are beginning to appear, such as the IEEE International Workshop on Intent-Based Networking [WIN21], as well as various special journal issues [IEEE-TITS21] [MDPI22]. A survey of current intent-driven networking research has been published in [Pang20], listing among the most pressing current research challenges aspects such as intent translation and understanding, intent interfaces, and security.

## 2. Definitions and Acronyms

ACL: Access Control List

API: Application Programming Interface

Intent: A set of operational goals (that a network should meet) and outcomes (that a network is supposed to deliver), defined in a declarative manner without specifying how to achieve or implement them.

IBA: Intent-Based Analytics - Analytics that are defined and derived from users' intent and used to validate the intended state.

Intent-Based Management - The concept of performing management based on the concept of intent.

IBN: Intent-Based Network - A network that can be managed using intent.

IBS: Intent-Based System - A system that supports management functions that can be guided using intent.

PBNM: Policy-Based Network Management

Policy: A set of rules that governs the choices in behavior of a system.

PDP: Policy Decision Point

PEP: Policy Enforcement Point

Service Model: A model that represents a service that is provided by a network to a user.

SSoT: Single Source of Truth - A functional block in an IBN system that normalizes users' intent and serves as the single source of data for the lower layers.

SVoT: Single Version of Truth

User: In the context of this document, an operator and/or administrator responsible for the management and operation of communication services and networking infrastructure (as opposed to an end user of a communication service)

### 3. Introduction of Concepts

The following section provides an overview of the concept of intent and Intent-Based Management. It also provides an overview of the related concepts of service models and policies (and Policy-Based Network Management), and explains how they relate to intent and Intent-Based Management.

#### 3.1. Intent and Intent-Based Management

In this document, intent is defined as a set of operational goals (that a network is supposed to meet) and outcomes (that a network is supposed to deliver), defined in a declarative manner without specifying how to achieve or implement them.

The term "intent" was first introduced in the context of Autonomic Networks, where it is defined as "an abstract, high-level policy used to operate a network" [RFC7575]. According to this definition, an intent is a specific type of policy provided by a user to provide guidance to the Autonomic Network that would otherwise operate without human intervention. However, to avoid using intent simply as a synonym for policy, a distinction that differentiates intent clearly from other types of policies needs to be introduced.

Intent-Based Management aims to lead towards networks that are fundamentally simpler to manage and operate, requiring only minimal outside intervention. Networks, even when they are autonomic, are not clairvoyant and have no way of automatically knowing particular operational goals nor which instances of networking services to support. In other words, they do not know what the intent of the network provider is that gives the network the purpose of its being. This still needs to be communicated to the network by what informally constitutes intent. That being said, the concept of intent is not limited just to autonomic networks, such as networks that feature an Autonomic Control Plane [RFC8994], but applies to any network.

Intent defines goals and outcomes in a manner that is purely declarative, specifying what to accomplish, not how to achieve it. Intent thus applies several important concepts simultaneously:

- \* It provides data abstraction: users do not need to be concerned with low-level device configuration and nerd knobs.

- \* It provides functional abstraction from particular management and control logic: users do not need to be concerned even with how to achieve a given intent. What is specified is the desired outcome, with the IBS automatically figuring out a course of action (e.g., using an algorithm or applying a set of rules derived from the intent) for how to achieve the outcome.

The following are some examples of intent, expressed in natural language for the sake of clarity (actual interfaces used to convey intent may differ):

- \* "Steer networking traffic originating from endpoints in one geography away from a second geography, unless the destination lies in that second geography." (States what to achieve, not how.)
- \* "Avoid routing networking traffic originating from a given set of endpoints (or associated with a given customer) through a particular vendor's equipment, even if this occurs at the expense of reduced service levels." (States what to achieve, not how, providing additional guidance for how to trade off between different goals when necessary)
- \* "Maximize network utilization even if it means trading off service levels (such as latency, loss) unless service levels have deteriorated 20% or more from their historical mean." (A desired outcome, with a set of constraints for additional guidance, which does not specify how to achieve this.)
- \* "VPN service must have path protection at all times for all paths." (A desired outcome of which it may not be clear how it can be precisely accommodated.)
- \* "Generate in-situ OAM data and network telemetry for later offline analysis whenever significant fluctuations in latency across a path are observed." (Goes beyond traditional event-condition-action by not being specific about what constitutes "significant" and what specific data to collect)
- \* "Route traffic in a Space Information Network in a way that minimizes dependency on stratospheric balloons unless the intended destination is an aircraft." (Does not specify how to precisely achieve this; extrapolates on scenarios mentioned in [Pang20])

- \* "For a smart city service, ensure traffic signal control traffic uses dedicated and redundant slices that avoid fate sharing." (A desired outcome with a set of constraints and additional guidance without specifying how to precisely achieve this; extrapolates on scenarios from [Gharbaoui21]).

In contrast, the following are examples of what would not constitute intent (again, expressed in natural language for the sake of clarity):

- \* "Configure a given interface with an IP address." This would be considered device configuration and fiddling with configuration knobs, not intent.
- \* "When interface utilization exceeds a specific threshold, emit an alert." This would be a rule that can help support network automation, but a simple rule is not an intent.
- \* "Configure a VPN with a tunnel from A to B over path P." This would be considered as a configuration of a service.
- \* "Deny traffic to prefix P1 unless it is traffic from prefix P2." This would be an example of an access policy or a firewall rule, not intent.

In networks, in particular in networks that are deemed autonomic, intent should ideally be rendered by the network itself, i.e., translated into device-specific rules and courses of action. Ideally, intent would not need to be orchestrated or broken down by a higher-level, centralized system but by the network devices themselves using a combination of distributed algorithms and local device abstractions. In this idealized vision, because intent holds for the network as a whole, intent should ideally be automatically disseminated across all devices in the network, which can themselves decide whether they need to act on it.

However, such decentralization will not be practical in all cases. Certain functions will need to be at least conceptually centralized. For example, users may require a single conceptual point of interaction with the network. The system providing this point acts as the operational front end for the network through which users can direct requests at the network and from which they can receive updates about the network. It may appear to users as a single system, even if it is implemented in a distributed manner. In turn, it interacts with and manages other systems in the network as needed in order to realize (i.e., to fulfill and to assure) the desired intent. Likewise, the vast majority of network devices may be intent-agnostic and focus only (for example) on the actual forwarding

of packets. Many devices may also be constrained in terms of their processing resources. This means that not every device may be able to act on intent on its own. Again, intent in those cases can be achieved by a separate system which performs the required actions.

Another reason to provide intent functionality from a conceptually centralized point is in cases where the the realization of a certain type of intent benefits from global knowledge of a network and its state. In many cases, such a global view may be impractical to maintain by individual devices, for example due to the volume of data and time lags that are involved. It may even be impractical for devices to simply access such a view from another remote system if such were available.

All of this implies that in many cases, certain intent functionality needs to be provided by functions that are specialized for that purpose and that may be provided by dedicated systems (which in some cases could also co-host other networking functions). For example, the translation of specific types of intent into corresponding courses of action and algorithms to achieve the desired outcomes may need to be provided by such specialized functions. Of course, to avoid single points of failure, the implementation and hosting of such functions may still be distributed, even if conceptually centralized.

Regardless of its particular implementation in centralized or decentralized manner, an IBN is a network that can be managed using intent. This means that it is able to recognize and ingest intent of an operator or user and configure and adapt itself according to the user intent, achieving an intended outcome (i.e., a desired state or behavior) without requiring the user to specify the detailed technical steps for how to achieve the outcome. Instead, the IBN will be able to figure out on its own how to achieve the outcome. Similarly, an IBS is a system that allows users to manage a network using intent. Such a system will serve as a point of interaction with users and implement the functionality that is necessary to achieve the intended outcomes, interacting for that purpose with the network as required.

Other definitions of intent exist, such as [TR523]. Intent there is simply defined as a declarative interface that is typically provided by a controller. It implies the presence of a centralized function that renders the intent into lower-level policies or instructions and orchestrates them across the network. While this is certainly one way of implementation, the definition that is presented here is more encompassing and ambitious, as it emphasizes the importance of managing the network by specifying desired outcomes without the specific steps to be taken in order to achieve the outcome. A



controller API that simply provides a network-level of abstraction is more limited and would not necessarily qualify as intent. Likewise, ingestion and recognition of intent may not necessarily occur via a traditional API but may involve other types of human-machine interactions.

### 3.2. Related Concepts

#### 3.2.1. Service Models

A service model is a model that represents a service that is provided by a network to a user. Per [RFC8309], a service model describes a service and its parameters in a portable/implementation-agnostic way that can be used independently of the equipment and operating environment on which the service is realized. Two subcategories are distinguished: a "Customer Service Model" describes an instance of a service as provided to a customer, possibly associated with a service order. A "Service Delivery Model" describes how a service is instantiated over existing networking infrastructure.

An example of a service could be a Layer 3 VPN service [RFC8299], a Network Slice [I-D.ietf-teas-ietf-network-slice-definition], or residential Internet access. Service models represent service instances as entities in their own right. Services have their own parameters, actions, and lifecycles. Typically, service instances can be bound to end users of communication services, who might be billed for the services provided.

Instantiating a service typically involves multiple aspects:

- \* A user (or northbound system) needs to define and/or request a service to be instantiated.
- \* Resources, such as IP addresses, AS numbers, VLAN or VxLAN pools, interfaces, bandwidth, or memory need to be allocated.
- \* How to map services to the resources needs to be defined. Multiple mappings are often possible, which to select may depend on context (such as which type of access is available to connect the end user of a communication service with the service).
- \* Bindings between upper and lower-level objects need to be maintained.
- \* Once instantiated, the service operational state needs to be validated and assured to ensure that the network indeed delivers the service as requested.

The realization of service models involves a system, such as a controller, that provides provisioning logic. This includes breaking down high-level service abstractions into lower-level device abstractions, identifying and allocating system resources, and orchestrating individual provisioning steps. Orchestration operations are generally conducted using a "push" model in which the controller/manager initiates the operations as required, then pushes down the specific configurations to the device and validates whether the new changes have been accepted and the new operational/derived states are achieved and in sync with the intent/desired state. In addition to instantiating and creating new instances of a service, updating, modifying, and decommissioning services need to be also supported. The device itself typically remains agnostic to the service or the fact that its resources or configurations are part of a service/concept at a higher layer.

Instantiated service models map to instantiated lower-layer network and device models. Examples include instances of paths or instances of specific port configurations. The service model typically also models dependencies and layering of services over lower-layer networking resources that are used to provide services. This facilitates management by allowing to follow dependencies for troubleshooting activities, to perform impact analysis in which events in the network are assessed regarding their impact on services and customers. Services are typically orchestrated and provisioned top-to-bottom, which also facilitates keeping track of the assignment of network resources (composition), while troubleshooted bottom-up (decomposition). Service models might also be associated with other data that does not concern the network but provides business context. This includes things such as customer data (such as billing information), service orders and service catalogs, tariffs, service contracts, and Service Level Agreements (SLAs), including contractual agreements regarding remediation actions.

[I-D.ietf-teas-te-service-mapping-yang] is an example of a data model that provides a mapping for customer service models (e.g., the L3VPN Service Model) to Traffic Engineering (TE) models (e.g., the TE Tunnel or the Abstraction and Control of Traffic Engineered Networks Virtual Network model)

Like intent, service models provide higher layers of abstraction. Service models are often also complemented with mappings that capture dependencies between service and device or network configurations. Unlike intent, service models do not allow to define a desired "outcome" that would be automatically maintained by an IBS. Instead, the management of service models requires the development of sophisticated algorithms and control logic by network providers or system integrators.

### 3.2.2. Policy and Policy-Based Network Management

Policy-Based Network Management (PBNM) is a management paradigm that separates the rules that govern the behavior of a system from the functionality of the system. It promises to reduce maintenance costs of information and communication systems while improving flexibility and runtime adaptability. It is present today at the heart of a multitude of management architectures and paradigms, including SLA-driven, Business-driven, autonomous, adaptive, and self-\* management [Boutaba07]. The interested reader is asked to refer to the rich set of existing literature, which includes this and many other references. In the following, we will only provide a much-abridged and distilled overview.

At the heart of policy-based management is the concept of a policy. Multiple definitions of policy exist: "Policies are rules governing the choices in the behavior of a system" [Sloman94]. "Policy is a set of rules that are used to manage and control the changing and/or maintaining of the state of one or more managed objects" [Strassner03]. Common to most definitions is the definition of a policy as a "rule." Typically, the definition of a rule consists of an event (whose occurrence triggers a rule), a set of conditions (which get assessed and which must be true before any actions are actually "fired"), and, finally, a set of one or more actions that are carried out when the condition holds.

Policy-based management can be considered an imperative management paradigm: Policies precisely specified what needs to be done when and in which circumstance. By using policies, management can, in effect, be defined as a set of simple control loops. This makes policy-based management a suitable technology to implement autonomic behavior that can exhibit self-\* management properties, including self-configuration, self-healing, self-optimization, and self-protection. This is notwithstanding the fact that policy-based management may make use of the concept of abstractions (such as, "Bob gets gold service") that hide from the user the specifics of how that abstraction is rendered in a particular deployment.

Policies typically involve a certain degree of abstraction in order to cope with the heterogeneity of networking devices. Rather than having a device-specific policy that defines events, conditions, and actions in terms of device-specific commands, parameters, and data models, a policy is defined at a higher level of abstraction involving a canonical model of systems and devices to which the policy is to be applied. A policy agent on a controller or the device subsequently "renders" the policy, i.e., translates the canonical model into a device-specific representation. This concept allows applying the same policy across a wide range of devices

without needing to define multiple variants. In other words - policy definition is de-coupled from policy instantiation and policy enforcement. This enables operational scale and allows network operators and authors of policies to think in higher terms of abstraction than device specifics and be able to reuse the same, high-level definition across different networking domains, WAN, DC, or public cloud.

PBNM is typically "push-based": Policies are pushed onto devices where they are rendered and enforced. The push operations are conducted by a manager or controller, which is responsible for deploying policies across the network and monitoring their proper operation. That being said, other policy architectures are possible. For example, policy-based management can also include a pull-component in which the decision regarding which action to take is delegated to a so-called Policy Decision Point (PDP). This PDP can reside outside the managed device itself and has typically global visibility and context with which to make policy decisions. Whenever a network device observes an event that is associated with a policy but lacks the full definition of the policy or the ability to reach a conclusion regarding the expected action, it reaches out to the PDP for a decision (reached, for example, by deciding on an action based on various conditions). Subsequently, the device carries out the decision as returned by the PDP - the device "enforces" the policy and hence acts as a PEP (Policy Enforcement Point). Either way, PBNM architectures typically involve a central component from which policies are deployed across the network and/or policy decisions served.

Like Intent, policies provide a higher layer of abstraction. Policy systems are also able to capture dynamic aspects of the system under management through the specification of rules that allow defining various triggers for specific courses of action. Unlike intent, the definition of those rules (and courses of action) still needs to be articulated by users. Since the intent is unknown, conflict resolution within or between policies requires interactions with a user or some kind of logic that resides outside of PBNM. In that sense, policy constitutes a lower level of abstraction than intent, and it is conceivable for Intent-Based Systems to generate policies that are subsequently deployed by a PBNM system, allowing PBNM to support Intent-Based Networking.

### 3.2.3. Distinguishing between Intent, Policy, and Service Models

What Intent, Policy, and Service Models all have in common is the fact that they involve a higher-layer of abstraction of a network that does not involve device-specifics, that generally transcends individual devices, and that makes the network easier to manage for applications and human users compared to having to manage the network one device at a time. Beyond that, differences emerge.

Summarized differences:

- \* A service model is a data model that is used to describe instances of services that are provided to customers. A service model has dependencies on lower-level models (device and network models) when describing how the service is mapped onto the underlying network and IT infrastructure. Instantiating a service model requires orchestration by a system; the logic for how to orchestrate/manage/provide the service model and how to map it onto underlying resources is not included as part of the model itself.
- \* Policy is a set of rules, typically modeled around a variation of events/conditions/actions, used to express simple control loops that can be rendered by devices without requiring intervention by the outside system. Policies let users define what to do under what circumstances, but they do not specify the desired outcome.
- \* Intent is a high-level, declarative goal that operates at the level of a network and services it provides, not individual devices. It is used to define outcomes and high-level operational goals, without specifying how those outcomes should be achieved or how goals should specifically be satisfied, and without the need to enumerate specific events, conditions, and actions. Which algorithm or rules to apply can be automatically "learned/derived from intent" by the IBS. In the context of autonomic networking, intent is ideally rendered by the network itself; also, the dissemination of intent across the network and any required coordination between nodes is resolved by the network without the need for external systems.

One analogy to capture the difference between policy and Intent-Based Systems is that of Expert Systems and Learning Systems in the field of Artificial Intelligence. Expert Systems operate on knowledge bases with rules that are supplied by users, analogous to policy systems whose policies are supplied by users. They are able to make automatic inferences based on those rules but are not able to "learn" new rules on their own. Learning Systems (popularized by deep learning and neural networks), on the other hand, are able to learn

without depending on user programming or articulation of rules. However, they do require a learning or training phase requiring large data sets; explanations of actions that the system actually takes provide a different set of challenges. Analogous to intent-based systems, users focus on what they would like the learning system to accomplish but not how to do it.

#### 4. Principles

The following main operating principles allow characterizing the intent-based/-driven/-defined nature of a system.

1. Single Source of Truth (SSoT) and Single Version/View of Truth (SVoT). The SSoT is an essential component of an intent-based system as it enables several important operations. The set of validated intent expressions is the system's SSoT. SSoT and the records of the operational states enable comparing the intended/desired state and actual/operational states of the system and determining drift between them. SSoT and the drift information provide the basis for corrective actions. If the IBS is equipped with the means to predict states, it can further develop strategies to anticipate, plan, and pro-actively act on any diverging trends with the aim to minimize their impact. Beyond providing a means for consistent system operation, SSoT also allows for better traceability to validate if/how the initial intent and associated business goals have been properly met, to evaluate the impacts of changes in the intent parameters and impacts and effects of the events occurring in the system.

Single Version (or View) of Truth derives from the SSoT and can be used to perform other operations, such as querying, polling, or filtering measured and correlated information in order to create so-called "views." These views can serve the users of the intent-based system. In order to create intents as single sources of truth, the IBS must follow well-specified and well-documented processes and models. In other contexts, SSoT is also referred to as the invariance of the intent [Lenrow15].

2. One-touch but not one-shot. In an ideal intent-based system, the user expresses intent in one form or another, and then the system takes over all subsequent operations (one-touch). A zero-touch approach could also be imagined in the case where the intent-based system has the capabilities or means to recognize intentions in any form of data. However, the zero- or one-touch approach should not distract from the fact that reaching the state of a well-formed and valid intent expression is not a one-shot process. On the contrary, the interfacing between the user and the intent-based system could be designed as an interactive

and iterative process. Depending on the level of abstraction, the intent expressions may initially contain more or less implicit parts and unprecise or unknown parameters and constraints. The role of the intent-based system is to parse, understand, and refine the intent expression to reach a well-formed and valid intent expression that can be further used by the system for the fulfillment and assurance operations. An intent refinement process could use a combination of iterative steps involving the user to validate the proposed refined intent and to ask the user for clarifications in case some parameters or variables could not be deduced or learned by means of the system itself. In addition, the Intent-Based System will need to moderate between conflicting intent, helping users to properly choose between intent alternatives that may have different ramifications.

3. **Autonomy and Supervision.** A desirable goal for an intent-based system is to offer a high degree of flexibility and freedom on both the user side and system side, e.g., by giving the user the ability to express intents using the user's own terms, by supporting different forms of expression of intents and being capable of refining the intent expressions to well-formed and exploitable expressions. The dual principle of autonomy and supervision allows operating a system that will have the necessary levels of autonomy to conduct its tasks and operations without requiring the intervention of the user and taking its own decisions (within its areas of concern and span of control) as how to perform and meet the user expectations in terms of performance and quality, while at the same time providing the proper level of supervision to satisfy the user requirements for reporting and escalation of relevant information.
4. **Learning.** An intent-based system is a learning system. In contrast to an imperative-type of system, such as Event-Condition-Action policy rules, where the user defines beforehand the expected behavior of the system to various events and conditions, in an intent-based system, the user only declares what the system is supposed to achieve and not how to achieve these goals. There is thus a transfer of reasoning/rationality from the human (domain knowledge) to the system. This transfer of cognitive capability also implies the availability in the intent-based system of capabilities or means for learning, reasoning, and knowledge representation and management. The learning abilities of an IBS can apply to different tasks such as optimization of the intent rendering or intent refinement processes. The fact that an intent-based system is a continuously evolving system creates the condition for continuous learning and optimization. Other cognitive capabilities such as

planning can also be leveraged in an intent-based system to anticipate or forecast future system state and response to changes in intents or network conditions and thus elaboration of plans to accommodate the changes while preserving system stability and efficiency in a trade-off with cost and robustness of operations.

5. Capability exposure. Capability exposure consists in the need for expressive network capabilities, requirements, and constraints to be able to compose/decompose intents and map the user's expectations to the system capabilities.
6. Abstract and outcome-driven. Users do not need to be concerned with how intent is achieved and are empowered to think in terms of outcomes. In addition, they can refer to concepts at a higher level of abstractions, independent e.g. of vendor-specific renderings.

The described principles are perhaps the most prominent, but they are not an exhaustive list. There are additional aspects to consider, such as:

- \* Intent targets are not individual devices but typically aggregations (such as groups of devices adhering to a common criteria, such as devices of a particular role) or abstractions (such as service types, service instances, topologies).
- \* Abstraction and inherent virtualization: agnostic to implementation details.
- \* Human-tailored network interaction: IBN should speak the language of the user as opposed to requiring the user to speak the language of the device/network.
- \* Explainability as an important IBN function, detection and IBN-aided resolution of conflicting intent, reconciliation of what the user wants and what the network can actually do.
- \* Inherent support, verification, and assurance of trust.

All of these principles and considerations have implications on the design of intent-based systems and their supporting architecture. Accordingly, they need to be considered when deriving functional and operational requirements.



## 5. Intent-Based Networking - Functionality

Intent-Based Networking involves a wide variety of functions that can be roughly divided into two categories:

- \* Intent Fulfillment provides functions and interfaces that allow users to communicate intent to the network and that perform the necessary actions to ensure that intent is achieved. This includes algorithms to determine proper courses of action and functions that learn to optimize outcomes over time. In addition, it also includes more traditional functions such as any required orchestration of coordinated configuration operations across the network and rendering of higher-level abstractions into lower-level parameters and control knobs.
- \* Intent Assurance provides functions and interfaces that allow users to validate and monitor that the network is indeed adhering to and complying with intent. This is necessary to assess the effectiveness of actions taken as part of fulfillment, providing important feedback that allows those functions to be trained or tuned over time to optimize outcomes. In addition, Intent Assurance is necessary to address "intent drift." Intent is not meant to be transactional, i.e. "set and forget", but expected to be remain in effect over time (unless explicitly stated otherwise). Intent drift occurs when a system originally meets the intent, but over time gradually allows its behavior to change or be affected until it no longer does or does so in a less effective manner.

The following sections provide a more comprehensive overview of those functions.

### 5.1. Intent Fulfillment

Intent fulfillment is concerned with the functions that take intent from its origination by a user (generally, an administrator of the responsible organization) to its realization in the network.

#### 5.1.1. Intent Ingestion and Interaction with Users

The first set of functions is concerned with "ingesting" intent, i.e., obtaining intent through interactions with users. They provide functions that recognize intent from interaction with the user as well as functions that allow users to refine their intent and articulate it in such ways so that it becomes actionable by an Intent-Based System. Typically, those functions go beyond those provided by a traditional API, although APIs may still be provided (and needed for interactions beyond human users, i.e., with other

machines). Many cases would also involve a set of intuitive and easy-to-navigate workflows that guide users through the intent ingestion phase, making sure that all inputs that are necessary for intent modeling and consecutive translation have been gathered. They may support unconventional human-machine interactions, in which a human will not simply give simple commands but which may involve a human-machine dialog to provide clarifications, to explain ramifications and trade-offs, and to facilitate refinements.

The goal is ultimately to make IBSs as easy and natural to use and interact with as possible, in particular allowing human users to interact with the IBS in ways that do not involve a steep learning curve that forces the user to learn the "language" of the system. Ideally, it will be the Intent-Based Systems that is increasingly be able to learn how to understand the user as opposed to the other way round. Of course, further research will be required to make this a reality.

#### 5.1.2. Intent Translation

A second set of functions needs to translate user intent into courses of action and requests to take against the network, which will be meaningful to network configuration and provisioning systems. These functions lie at the core of IBS, bridging the gap between interaction with users on the one hand and the traditional management and operations side that will need to orchestrate provisioning and configuration across the network.

Beyond merely breaking down a higher layer of abstraction (intent) into a lower layer of abstraction (policies, device configuration), Intent Translation functions can be complemented with functions and algorithms that perform optimizations and that are able to learn and improve over time in order to result in the best outcomes, specifically in cases where multiple ways of achieving those outcomes are conceivable. For example, satisfying an intent may involve computation of paths and other parameters that will need to be configured across the network. Heuristics and algorithms to do so may evolve over time to optimize outcomes that may depend on a myriad of dynamic network conditions and context.

#### 5.1.3. Intent Orchestration

A third set of functions deals with the actual configuration and provisioning steps that need to be orchestrated across the network and that were determined by the previous intent translation step.

## 5.2. Intent Assurance

Intent assurance is concerned with the functions that are necessary to ensure that the network indeed complies with the desired intent once it has been fulfilled.

### 5.2.1. Monitoring

A first set of assurance functions monitors and observes the network and its exhibited behavior. This includes all the usual assurance functions such as monitoring the network for events and performance outliers, performing measurements to assess service levels that are being delivered, generating and collecting telemetry data. Monitoring and observation are required as the basis for the next set of functions that assess whether the observed behavior is in fact in compliance with the behavior that is expected based on the intent.

### 5.2.2. Intent Compliance Assessment

At the core of Intent Assurance are functions that compare the actual network behavior that is being monitored and observed with the intended behavior that is expected per the intent and is held by SSoT. These functions continuously assess and validate whether the observation indicates compliance with intent. This includes assessing the effectiveness of intent fulfillment actions, including verifying that the actions had the desired effect and assessing the magnitude of the effect as applicable. It can also include functions that perform analysis and aggregation of raw observation data. The results of the assessment can be fed back to facilitate learning functions that optimize outcomes.

Intent compliance assessment also includes assessing whether intent drift occurs over time. Intent drift can be caused by a control plane or lower-level management operations that inadvertently cause behavior changes which conflict with intent that was orchestrated earlier. Intent-Based Systems and Networks need to be able to detect when such drift occurs or is about to occur as well as assess the severity of the drift.

### 5.2.3. Intent Compliance Actions

When intent drift occurs or network behavior is inconsistent with desired intent, functions that are able to trigger corrective actions are needed. This includes actions needed to resolve intent drift and bring the network back into compliance. Alternatively, and where necessary, reporting functions need to be triggered that alert operators and provide them with the necessary information and tools to react appropriately, e.g., by helping them articulate

modifications to the original intent to moderate between conflicting concerns.

#### 5.2.4. Abstraction, Aggregation, Reporting

The outcome of Intent Assurance needs to be reported back to the user in ways that allow the user to relate the outcomes to their intent. This requires a set of functions that are able to analyze, aggregate, and abstract the results of the observations accordingly. In many cases, lower-level concepts such as detailed performance statistics and observations related to low-level settings need to be "up-leveled" to concepts the user can relate to and take action on.

The required aggregation and analysis functionality needs to be complemented with functions that report intent compliance status and provide adequate summarization and visualization to human users.

### 6. Lifecycle

Intent is subject to a lifecycle: it comes into being, may undergo changes over the course of time, and may at some point be retracted. This lifecycle is closely tied to various interconnection functions that are associated with the intent concept.

Figure 1 depicts an intent lifecycle and its main functions. The functions were introduced in Section 5 and are divided into two functional (horizontal) planes, reflecting the distinction between fulfillment and assurance. In addition, they are divided into three (vertical) spaces.

The spaces indicate the different perspectives and interactions with different roles that are involved in addressing the functions:

- \* The User Space involves the functions that interface the network and intent-based system with the human user. It involves the functions that allow users to articulate and the intent-based system to recognize that intent. It also involves the functions that report back the status of the network relative to the intent and that allow users to assess outcomes and whether their intent has the desired effect.
- \* The Translation, or Intent-Based System (IBS) Space involves the functions that bridge the gap between intent users and the network operations infrastructure. This includes the functions used to translate an intent into a course of action as well as the algorithms that are used to plan and optimize those courses of action also in consideration of feedback and observations from the network. It also includes the functions to analyze and aggregate

observations from the network in order to validate compliance with the intent and to take corrective actions as necessary. In addition, it includes functions that abstract observations from the network in ways that relate them to the intent as communicated by users. This facilitates the reporting functions in the user space.

- \* The Network Operations Space, finally, involves the traditional orchestration, configuration, monitoring, and measurement functions, which are used to effectuate the rendered intent and observe its effects on the network.

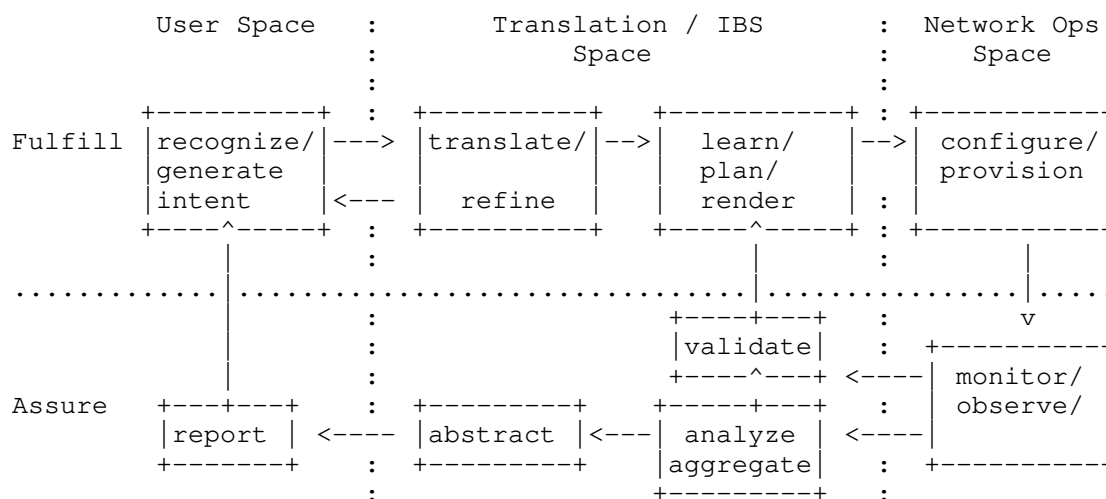


Figure 1: Intent Lifecycle

When carefully inspecting the diagram, it becomes apparent that the intent lifecycle, in fact, involves two cycles, or loops:

- \* The "inner" intent control loop between IBS and Network Operations space is completely autonomic and does not involve any human in the loop. It represents closed-loop automation that involves automatic analysis and validation of intent based on observations from the network operations space. Those observations are fed into the function that plans the rendering of networking intent in order to make adjustments as needed in the configuration of the network. The loop addresses and counteracts any intent drift that may be occurring, using observations to assess the degree of the network's intent compliance and automatically prompting actions to address any discrepancies. Likewise, the loop allows to assess the effectiveness of any actions that are taken in order to continuously learn and improve how intent needs to be rendered in order to achieve the desired outcomes.
- \* The "outer" intent control loop extends to the user space. It includes the user taking action and adjusting their intent based on observations and feedback from the IBS. Intent is thus subjected to a lifecycle: It comes into being, may undergo refinements, modifications, and changes of time, and may at some point in time even get retracted.

## 7. Additional Considerations

Given the popularity of the term "intent," it is tempting to broaden its use to encompass also other related concepts, resulting in "intent-washing" that paints those concepts in a new light by simply applying new intent terminology to them. A common example concerns referring to the northbound interface of SDN controllers as "intent interface". However, in some cases, this actually makes sense not just as a marketing ploy but as a way to better relate previously existing and new concepts.

In that sense and regards to intent, it makes sense to distinguish various subcategories of intent as follows:

- \* Operational Intent: defines intent related to operational goals of an operator; corresponds to the original "intent" term and the concepts defined in this document.
- \* Rule Intent: a synonym for policy rules regarding what to do when certain events occur.
- \* Service intent: a synonym for customer service model [RFC8309].
- \* Flow Intent: A synonym for a Service Level Objective for a given flow.

A comprehensive set of classifications of different concepts and categories of intent will be described in a separate document.

#### 8. IANA Considerations

Not applicable

#### 9. Security Considerations

This document describes concepts and definitions of Intent-Based Networking. As such, the below security considerations remain high level, i.e. in the form of principles, guidelines or requirements. More detailed security considerations will be described in the documents that specify the architecture and functionality.

Security in Intent-Based Networking can apply to different facets:

- \* Securing the intent-based system itself.
- \* Mitigating the effects of erroneous, harmful or compromised intents.
- \* Expressing security policies or security-related parameters with intents.

Securing the intent-based system aims at making the intent-based system operationally secure by implementing security mechanisms and applying security best practices. In the context of Intent-based Networking, such mechanisms and practices may consist in intent verification and validation; operations on intents by authenticated and authorized users only; protection against or detection of tampered intents. Such mechanisms may also include the introduction of multiple levels of intent. For example, intent related to securing the network should occur at a "deeper" level that overrides other levels of intent if necessary, and that is not subject to modification through regular operations but through ones that are specifically secured. Use of additional mechanisms such as explanation components that describe the security ramifications and trade-off should be considered as well.

Mitigating the effects of erroneous or compromised intents aims at making the intent-based system operationally safe by providing checkpoint and safeguard mechanisms and operating principles. In the context of Intent-based Networking, such mechanisms and principles may consist in the ability to automatically detect unintended, detrimental or abnormal behavior; the ability to automatically (and gracefully) rollback or fallback to a previous "safe" state; the ability to prevent or contain error amplification (due to the

combination of a higher degree of automation and the intrinsic higher degree of freedom, ambiguity, and implicitly conveyed by intents); dynamic levels of supervision and reporting to make the user aware of the right information, at the right time with the right level of context. Erroneous or harmful intents may inadvertently propagate and compromise security. In addition, compromised intents, for example, intent forged by an inside attacker, may sabotage or harm the network resources and make them vulnerable to further, larger attacks, e.g., by defeating certain security mechanisms.

Expressing security policies or security-related parameters with intents consists of using the intent formalism (a high-level, declarative abstraction), or part(s) of an intent statement to define security-related aspects such as data governance, level(s) of confidentiality in data exchange, level(s) of availability of system resources, of protection in forwarding paths, of isolation in processing functions, level(s) of encryption, authorized entities for given operations, etc.

The development and introduction of Intent-Based Networking in operational environments will certainly create new security concerns. Such security concerns have to be anticipated at the design and specification time. However, Intent-Based Networking may also be used as an enabler for better security. For instance, security and privacy rules could be expressed in a more human-friendly and generic way and be less technology-specific and less complex, leading to fewer low-level configuration mistakes. The detection of threats or attacks could also be made more simple and comprehensive thanks to conflict detection at higher-level or at coarser granularity

More thorough security analyses should be conducted as our understanding of Intent-Based Networking technology matures.

## 10. Acknowledgments

We would like to thank the members of the IRTF Network Management Research Group (NMRG) for many valuable discussions and feedback. In particular, we would like to acknowledge the feedback and support from Remi Badonnel, Walter Cerroni, Marinos Charalambides, Luis Contreras, Jerome Francois, Molka Gharbaoui, Olga Havel, Chen Li, William Liu, Barbara Martini, Stephen Mwanje, Jeferson Nobre, Haoyu Song, Peter Szilagyi, and Csaba Vulkan. Of those, we would like to thank the following persons who went one step further and also provided reviews of the document: Remi Badonnel, Walter Cerroni, Jerome Francois, Molka Gharbaoui, Barbara Martini, Stephen Mwanje, Peter Szilagyi, and Csaba Vulkan.

## 11. Informative References



- [Boutaba07] Boutaba, R. and I. Aib, "Policy-Based Management: A Historical perspective. Journal of Network and Systems Management (JNSM), Springer, Vol. 15 (4).", December 2007.
- [Clemm20] Clemm, A., Faten Zhani, M., and R. Boutaba, "Network Management 2030: Operations and Control of Network 2030 Services. Journal of Network and Systems Management (JNSM), Springer, Vol. 28 (4).", October 2020.
- [Gharbaoui21] Gharbaoui, M., Martini, B., and P. Castoldi, "Implementation of an Intent Layer for SDN-enabled and QoS-aware Network Slicing", in IEEE 7th Int. Conference on Network Softwarization (NetSoft), pp 17-23, doi: 10.1109/NetSoft51509.2021.9492643", June 2021.
- [I-D.ietf-teas-ietf-network-slice-definition] Rokui, R., Homma, S., Makhijani, K., Contreras, L. M., and J. Tantsura, "Definition of IETF Network Slices", Work in Progress, Internet-Draft, draft-ietf-teas-ietf-network-slice-definition-01, 22 February 2021, <<https://www.ietf.org/archive/id/draft-ietf-teas-ietf-network-slice-definition-01.txt>>.
- [I-D.ietf-teas-te-service-mapping-yang] Lee, Y., Dhody, D., Fioccola, G., Wu, Q., Ceccarelli, D., and J. Tantsura, "Traffic Engineering (TE) and Service Mapping YANG Model", Work in Progress, Internet-Draft, draft-ietf-teas-te-service-mapping-yang-10, 7 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-teas-te-service-mapping-yang-10.txt>>.
- [IEEE-TITS21] Garg, S., "Special Issue on Intent-Based Networking for 5G-Envisioned Internet of Connected Vehicles, in IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 8, DOI: 10.1109/TITS.2021.3101259", August 2021.
- [IEEEEXPLORE] IEEE, "IEEE eXplore, <https://ieeexplore.ieee.org/>", 2021.
- [Lenrow15] Lenrow, D., "Intent As The Common Interface to Network Resources, Intent Based Network Summit 2015 ONF Boulder: IntentNBI", February 2015.
- [M3010] ITU-T, "M.3010 : Principles for a telecommunications management network.", February 2000.

- [MDPI22] Gharbaoui, M., "Special Issue "Intent-Based Software-Defined Networks", in Computers (MDPI) (to appear)", 2022.
- [Pang20] Pang, L., Yang, C., Chen, D., Song, Y., and M. Guizan, "A Survey on Intent-Driven Networks", in IEEE Access Vol 8 pp. 22862-22873, DOI: 10.1101/ACCESS.2020.2969208", 2020.
- [RFC3411] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, DOI 10.17487/RFC3411, December 2002, <<https://www.rfc-editor.org/info/rfc3411>>.
- [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, <<https://www.rfc-editor.org/info/rfc7575>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8299] Wu, Q., Ed., Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8299, DOI 10.17487/RFC8299, January 2018, <<https://www.rfc-editor.org/info/rfc8299>>.
- [RFC8309] Wu, Q., Liu, W., and A. Farrel, "Service Models Explained", RFC 8309, DOI 10.17487/RFC8309, January 2018, <<https://www.rfc-editor.org/info/rfc8309>>.
- [RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March 2018, <<https://www.rfc-editor.org/info/rfc8345>>.
- [RFC8994] Eckert, T., Ed., Behringer, M., Ed., and S. Bjarnason, "An Autonomic Control Plane (ACP)", RFC 8994, DOI 10.17487/RFC8994, May 2021, <<https://www.rfc-editor.org/info/rfc8994>>.

- [Sloman94] Sloman, M., "Policy Driven Management for Distributed Systems. Journal of Network and Systems Management (JNSM), Springer, Vol. 2 (4).", December 1994.
- [Strassner03] Strassner, J., "Policy-Based Network Management. Elsevier.", 2003.
- [TR523] Foundation, O. N., "Intent NBI - Definition and Principles. ONF TR-523.", October 2016.
- [WIN21] Ciavaglia, L. and E. Renault, "1st IEEE International Workshop on Intent-Based Networking, <https://netsoft2021.ieee-netsoft.org/program/workshops/win-2021/>", June 2021.

## Authors' Addresses

Alexander Clemm  
Futurewei  
2330 Central Expressway  
Santa Clara,, CA 95050  
United States of America  
Email: ludwig@clemm.org

Laurent Ciavaglia  
Rakuten Mobile  
Email: laurent.ciavaglia@rakuten.com

Lisandro Zambenedetti Granville  
Federal University of Rio Grande do Sul (UFRGS)  
Av. Bento Gonçalves  
Porto Alegre-  
9500  
Brazil  
Email: granville@inf.ufrgs.br

Jeff Tantsura  
Microsoft  
Email: jefftant.ietf@gmail.com

Network Working Group  
Internet Draft  
Intended status: Informational  
Expires: August 2022

C. Li  
China Telecom  
O. Havel  
A. Olariu  
Huawei Technologies  
P. Martinez-Julia  
NICT  
J. Nobre  
UFRGS  
D. Lopez  
Telefonica, I+D  
February 22, 2022

Intent Classification  
draft-irtf-nmrg-ibn-intent-classification-06

Abstract

Intent is an abstract, high-level policy used to operate the network. Intent-based management system includes an interface for users to input requests and an engine to translate the intents into the network configuration and manage their life-cycle.

This document discusses mostly the concept of network intents, but other types of intents are also being considered. Specifically, it highlights stakeholder perspectives of intent, methods to classify and encode intent, the associated intent taxonomy, and defines relevant intent terms where necessary. This document provides a foundation for intent related research and facilitates solution development.

This document is a product of the IRTF Network Management Research Group (NMRG) and is not issued by the IETF and is not an IETF standard.

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 August 22, 2022.

#### Copyright Notice

Copyright (c) 2022 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.....	4
1.1. Research activities.....	4
1.2. Standards and open source activities.....	5
1.3. Scope.....	6
2. Acronyms.....	7
3. Definitions.....	8
4. Abstract Intent Requirements.....	8
4.1. What is Intent?.....	8
4.2. Intent Solutions and Intent Users.....	9
4.3. Benefits of Intents for Different Stakeholders.....	11
4.4. Intent Types that need to be supported.....	12
5. Functional Characteristics and Behaviour.....	13
5.1. Abstracting Intent Operation.....	13
5.2. Intent User Types.....	14
5.3. Intent Scope.....	15
5.4. Intent Network Scope.....	15
5.5. Intent Abstraction.....	16
5.6. Intent Life-cycle.....	16
5.7. Autonomous Driving Levels.....	16
6. Intent Classification.....	17
6.1. Intent Classification Methodology.....	18
6.2. Intent Taxonomy.....	21
6.3. Intent Classification for Carrier Solution.....	23
6.3.1. Intent Users and Intent Types.....	23
6.3.2. Intent Categories.....	27
6.3.3. Intent Classification Example.....	27
6.4. Intent Classification for Data Center Network Solutions.....	31
6.4.1. Intent Users and Intent Types.....	31
6.4.2. Intent Categories.....	35
6.4.3. Intent Classification Example.....	35
6.5. Intent Classification for Enterprise Solution.....	39
6.5.1. Intent Users and Intent Types.....	39
6.5.2. Intent Categories.....	41
7. Conclusions.....	43
8. Security Considerations.....	43
9. IANA Considerations.....	43
10. Contributors.....	44
11. Acknowledgments.....	44
12. Informative References.....	44

## 1. Introduction

The vision of intent-based networks has attracted a lot of attention, as it promises to simplify the management of networks by human operators. This is done by simply specifying what should happen on the network, without giving any instructions on how to do it. This promise led many researcher-led activities and telecom companies to start researching this new vision, and many Standards Development Organization (SDOs) to propose different intent frameworks.

This draft proposes an intent classification methodology and an intent taxonomy. The scope of these proposals is to ensure a common understanding in the research community in terms of what are the intent users, intent types, or intent solutions, etc. for specific scenarios that are being considered.

The document represents the consensus of the RG. During the document's lifecycle it received many positive expressions of support and detailed reviews beyond the authors. Only in the last call period it received more than 12 positive expressions of support and more than 5 detailed reviews. It is published for informational purposes.

### 1.1. Research activities

Intent-based networking is an active research topic which spans across different areas that could benefit from an intent classification and taxonomy.

One such area is intent expression and recognition ([Bezahaf21], [Bezahaf19]), NILE [Jacobs18]). The use of a common classification can provide consistency in the understanding of the various forms of intent expressions being proposed and investigated.

Another area where this intent classification could contribute is the orchestration of cognitive autonomous RANs [Banerjee21] where intents are classified based on their content.

The work carried in intent network verification [Tian19] where the authors are proposing new intent language is another candidate where intent classification could be used advantageously.

Furthermore, this draft is proving itself already extremely relevant to the research community as it has been used as the basis for proposing self-generated Intent-based systems [Bezahaf19], for advancing IBN-based VNF placement solutions that rely on defining user intent profiles corresponding to abstract network services [Leivadeas21], for improving existing solutions in provisioning

intent-based networks, and proposing new approaches to service management [Davoli21], or even for defining grammars for users to specify the high-level requirements for blockchain selection in the form of intent [Padovan20]. As well, the draft has been mentioned in surveys addressing the topic of intelligent intent-based autonomous networks [Mehmood21], [Szilagyi21].

This document describes as well an example on how this proposal has been successfully applied in an academic environment [IBN-POC] by researchers in the area of SDN/NFV for defining the scope of their project. The specific problem addressed by researches is how to apply intent concepts at different levels that correspond to different stakeholders.

IEEE Communications Society Technical Committee on Network Operation and Management (IEEE-CNOM), IRTF-NMRG and IFIP WG6.6 have developed a taxonomy for network and service management [IFIP-NSM] that is used by the research community in network management and operations to structure the research area through a well-defined set of keywords and to improve quality of reviews in submissions to journals, conferences and workshops. The proposed intent taxonomy may be contributed as an extension to this taxonomy for intent driven management.

## 1.2. Standards and open source activities

Several SDOs and open source projects, such as Internet Research Task Force (IRTF)/ Network Management Research Group (NMRG), Open Networking Foundation (ONF) [ONF] /Open Network Operating System (ONOS) [ONOS], European Telecommunications Standards Institute (ETSI)/Experiential Networked Intelligence (ENI), TMF with its Autonomous Networks, have proposed intents for defining a set of network operations to execute in a declarative manner.

More recently, the IRTF NMRG is working on the Intent-based Networking - Concepts and Definitions document, [CLEMM]. This document clarifies the concept of "Intent" and provides an overview of the functionality that is associated with it. The goal is to contribute towards a common and shared understanding of terms, concepts, and functionality that can be used as the foundation to guide further definition of associated research and engineering problems and their solutions.

The present document, together with [CLEMM], aims to become the foundation for future intent-related topic discussions regarding the NMRG.



The SDOs usually came up with their own way of specifying an intent, and with their own understanding of what an intent is. Besides that, each SDO defines a set of terms and level of abstraction, its intended intent users, and the applications and usage scenarios.

However, most intent approaches proposed by SDOs share the same following features:

- o It must be declarative in nature, meaning that an intent user specifies the goal on the network without specifying how to achieve that goal.
- o It must be vendor agnostic, in the sense that it abstracts the network capabilities, or the network infrastructure from the intent user, and it can be ported across different platforms.
- o It must provide an easy-to-use interface, which simplifies the intent users' interaction with the intent system through the usage of familiar terminology or concepts.

It should be able to detect and resolve intent conflicts, which include, for example, static (compile-time) conflicts and dynamic (run-time) conflicts.

### 1.3. Scope

This document mostly addresses intents in the context of network intents, however other types of intents are not excluded, as presented in section 4.4. and section 6.2. .

It is impossible to fully differentiate intents only by the common characteristics followed by concepts, terms and intentions. This document clarifies what an intent represents for different stakeholders through a classification on various dimensions, such as solutions, intent users, and intent types. This classification ensures common understanding among all participants and is used to determine the scope and priority of individual projects, proof-of-concept (PoCs), research initiatives, or open source projects.

The scope of intent classification in this document includes solutions, intent users and intent types, and the initial classification table is made according to this scope. The methodology presented can be used to update the classification tables by adding or removing different solutions, intent users, or intent types to cater for future scenarios, applications or domains.

## 2. Acronyms

AI: Artificial Intelligence

CE: Customer Equipment

CFS: Customer Facing Service

CLI: Command Line Interface

DB: Database

DC: Data Center

ECA: Event-Condition-Action

GBP: Group-Based Policy

GPU: Graphics Processing Unit

IBN: Intent Based Network

NFV: Network Function Virtualization

O&M: Operations & Maintenance

ONF: Open Networking Foundation

ONOS: Open Network Operating System

PNF: Physical Network Function

QoE: Quality of Experience

RFS: Resource Facing Service

SDO: Standards Development Organization

SD-WAN: Software-Defined Wide-Area Network

SLA: Service Level Agreement

SUPA: Simplified Use of Policy Abstractions

VM: Virtual Machine

VNF: Virtual Network Function

### 3. Definitions

A common and shared understanding of terms and definitions related to IBN is provided in [CLEMM], as follows:

- o Intent: A set of operational goals (that a network should meet) and outcomes (that a network is supposed to deliver), defined in a declarative manner without specifying how to achieve or implement them.
- o Intent-Based Network: A network that can be managed using intent.
- o Policy: A set of rules that governs the choices in behaviour of a system.
- o Intent User: A user that defines and issues the intent request to the intent-based management system.

Other definitions relevant to this draft, such as intent scope, intent network scope, intent abstraction, intent abstraction, and intent lifecycle are available in section 5.

### 4. Abstract Intent Requirements

In order to understand the different intent requirements that would drive intent classification, we first need to understand what intent means for different intent users.

#### 4.1. What is Intent?

The term Intent has become very widely used in the industry for different purposes, sometimes it is not even in agreement with SDO shared principles mentioned in the Introduction section.[CLEMM] draft brings clarification with relation to what an intent is and how it differentiates from policies and services.

Different stakeholders have different perspective of the network and therefore have different intent requirements. Their intent is sometimes technical, non-technical, abstract or technology specific. Therefore, it is important to start a discussion in the industry and academia communities about what intent is for different solutions and intent users. It is also imperative to try to propose some intent categories/ classifications that could be understood by a wider audience. This would help us define intent interfaces, domain-specific languages, and models.

#### 4.2. Intent Solutions and Intent Users

Intent types are defined by all aspects that are required to profile different requirements to easily distinguish among them. However, in order to facilitate a clustered classification, we can focus on two aspects, the solution and intent user. They can be considered as the main keys to classify intents, as we can easily group requirements by solution and intent user.

On the one hand, different solutions and intent users have different requirements, expectations and priorities for intent-based networking. Therefore, intent users require different intent types, depending on their context, since they participate in different use cases. For instance, some intent users are more technical and require intents that expose more technical information. Other intent users do not have knowledge of the network infrastructure and require intents that shield them from different networking concepts and technologies.

The following are the solutions and intent users that intent-based networking needs to support:

Solutions	Intent Users
Carrier Networks	Network Operator Service Designers/App Developer Service Operators Customers/Subscribers
DC Networks	Cloud Administrator Underlay Network Administrator Application Developers Customer/Tenants
Enterprise Networks	Enterprise Administrator Application Developers End-Users

Table 1 - Intent Solutions and Intent Users

These intent solutions and intent users represent a starting point for the classification and are expendable through the methodology presented in section 6.1. .

- o For carrier networks scenario, for example, if a customer/subscriber wants to watch high-definition video, then the intent is to convert the video image to 1080p rate.
- o For DC networks scenario, administrators have their own clear network intent such as load balancing. For all traffic flows that need NFV service chaining, restrict the maximum load of any VNF node/container below 50% and the maximum load of any network link below 70%.
- o For enterprise networks scenario, when hosting a video conference multiple remote accesses are required. An example of the intent from the network administrator is: for any end-user of this application, the arrival time of hologram objects of all the remote tele-presenters should be synchronised within 50ms to reach the destination viewer for each conversation session.

#### 4.3. Benefits of Intents for Different Stakeholders

Current network APIs and CLIs are too complex because they are highly integrated with the low level concepts exposed by networks. Customers, application developers and end-users must not be required to set IP addresses, VLANs, subnets, ports, while operators may still want to have more technical and network visibility. All stakeholders would benefit from the simpler interfaces, like:

- o Request gold VPN service between my sites A, B and C
- o Provide CE redundancy for the customer sites
- o Add access rules to the network service

Operators and administrators manually troubleshoot and fix their networks and services. They instead want to:

- o simplify and automate network operations
- o simplify definitions of network services
- o provide simple customer APIs for value added services (operators)
- o be informed if the network or service is not behaving as requested
- o enable automatic optimization and correction for selected scenarios
- o have systems that learn from historic information and behaviour

Currently, intent users cannot build their own services and policies without becoming technical experts and performing manual maintenance actions. They instead want to be able to:

- o build their own network services with their own policies via simple interfaces, without becoming networking experts
- o have their network services up and running based on intent and automation only, without any manual actions or maintenance

#### 4.4. Intent Types that need to be supported

Next to the intent solutions and intent users, another way to categorize the intent is through the intent types. The following intent types and subtypes need to be supported, in order to address the requirements from different solutions and intent users:

- o Customer service intent
  - o for customer self-service with SLA
  - o for service operator orders
- o Network and underlay network service intent
  - o for service operator orders
  - o for intent driven network configuration, verification, correction and optimization
  - o for intent created and provided by the underlay network administrator
- o Network and underlay network intent
  - o for network configuration
  - o for automated lifecycle management of network configurations
  - o for network resources (switches, routers, routing, policies, underlay)
- o Cloud management intent
  - o for DC configuration, VMs, DB servers, APP servers
  - o for communication between VMs
- o Cloud resource management intent
  - o for cloud resource life-cycle management (policy driven self-configuration and auto-scaling and recovery/optimization)
- o Strategy intent
  - o for security, QoS, application policies, traffic steering, etc.

- o for configuring and monitoring policies, alarms generation for non-compliance, auto-recovery
  - o for design models and policies for network and network service design
  - o for design workflows, models and policies for operational task intents
- o Operational task intents
  - o for network migration
  - o for device replacements
  - o for network software upgrades
  - o for automating any other tasks that operators/administrator often perform

It is important to mention there all of the previously mentioned types and subtypes may affect other intents. For example, operational task intent can modify many other intents. The task itself is short-lived, but the modification of other intents has an impact on their life-cycle, so those changes must continue to be continuously monitored and self-corrected/self-optimized.

## 5. Functional Characteristics and Behaviour

Intent can be used to operate immediately on a target (much like issuing a command), or whenever it is appropriate (e.g., in response to an event). In either case, intent has a number of behaviours that serve to further organize its purpose, as described by the following subsections.

### 5.1. Abstracting Intent Operation

The modelling of intents can be abstracted using the following three-tuple:

{Context, Capabilities, Constraints}

- o Context grounds the intent, and determines if it is relevant or not for the current situation. Thus, context selects intents based on applicability.



- o Capabilities describe the functionality that the intent can perform. Capabilities take different forms, depending on the expressivity of the intent as well as the programming paradigm(s) used.
- o Constraints define any restrictions on the capabilities to be used for that particular context.

Metadata can be attached via strategy templates to each of the elements of the three-tuple, and may be used to describe how the intent should be used and how it operates, as well as prescribe any operational dependencies that must be taken into account.

Although different intent categories share the same abstracted intent model, each category will have its own specific context, capabilities and constraints.

## 5.2. Intent User Types

Expanding on the introduction in section 4.2. , intent user types represent the intent users that define and issue the intent request. Depending on the intent solutions, there are specific intent users. Examples of intent users are customers, network operators, service operators, enterprise administrators, cloud administrators, and underlay network administrators, or application developers.

- o Customers and end-users do not necessarily know the functional and operational details of the network that they are using. Furthermore, they lack skills to understand such details; in fact, such knowledge is typically not relevant to their job. In addition, the network may not expose these details to its intent users. This class of intent users focuses on the applications that they run, and uses services offered by the network. Hence, they want to specify policies that provide consistent behaviour according to their business needs. They do not have to worry about how the intents are deployed onto the underlying network, and especially, whether the intents need to be translated to different forms to enable network elements to understand them.

- o Application developers work in a set of abstractions defined by their application and programming environment(s). For example, many application developers think in terms of objects (e.g., a VPN). While this makes sense to the application developer, most network devices do not have a VPN object per se; rather, the VPN is formed through a set of configuration statements for that device in concert with configuration statements for the other devices that together make up the VPN. Hence, the view of application developers matches the services provided by the network, but may not directly correspond to other views of other intent users.
- o Network operators may have the knowledge of the underlying network. However, they may not understand the details of the applications and services of customers.

### 5.3. Intent Scope

Intents are used to manage the behaviour of the networks they are applied to and all intents are applied within a specific scope, such as:

- o Connectivity scope, if the intent creates or modifies a connection.
- o Security/privacy scope, if the intent specifies the security characteristics of the network, customers, or end-users.
- o Application scope, when the intent specifies the applications to be affected by the intent request.
- o QoS scope, when the intent specifies the QoS characteristics of the network.

These intent scopes are expendable through the methodology presented in section 6.1. .

### 5.4. Intent Network Scope

Regardless on the intent user type, their intent request is affecting the network, or network components, which are representing the intent targets.

Thus, intent network scope, or policy target as known in the area of declarative policy, can represent VNFs or PNFs, physical network elements, campus networks, SD-WAN networks, radio access networks, cloud edge, cloud core, branch, etc.

### 5.5. Intent Abstraction

Intent can be classified by whether it is necessary to feedback technical network information or non-technical information to the intent user after the intent is executed. As well, intent abstraction covers the level of technical details in the intent itself.

- o For non-technical intent users, they do not care how the intent is executed, or the details of the network. As a result, they do not need to know the configuration information of the underlying network. They only focus on whether the intent execution result achieves the goal, and the execution effect such as the quality of completion and the length of execution. In this scenario, we refer to an abstraction without technical feedback.
- o For administrators, such as network administrators, they perform intents, such as allocating network resources, selecting transmission paths, handling network failures, etc. They require multiple feedback indicators for network resource conditions, congestion conditions, fault conditions, etc. after execution. In this case, we refer to an abstraction with technical feedback.

As per intent definition provided in [CLEMM], lower-level intents are not considered to qualify as intents. However, we kept this classification to identify any PoCs/Demos/Use Cases that still either require or implement lower level of abstraction for intents.

### 5.6. Intent Life-cycle

Intents can be classified into transient and persistent intents:

- o If the intent is transient, it has no life-cycle management. As soon as the specified operation is successfully carried out, the intent is finished, and can no longer affect the target object.
- o If the intent is persistent, it has life-cycle management. Once the intent is successfully activated and deployed, the system will keep all relevant intents active until they are deactivated or removed.

### 5.7. Autonomous Driving Levels

In different phases of the autonomous driving network [TMF-auto], the intents are different. Depending on the Autonomous Network Level of the overall solution, we may have different intent requirements and

types. For example, at lower level the customer intent is automatically converted to configuration policies only, while at the higher levels the customer intent is covering the full life cycle, it is converted to both configuration and monitoring policies and is self-assured using AI.

A typical example of autonomous driving network level 0 to 5 are listed as below.

- o Level 0 - Traditional manual network: O&M personnel manually control the network and obtain network alarms and logs. - No intent
- o Level 1 - Partially automated network: Automated scripts are used to automate service provisioning, network deployment, and maintenance. Shallow perception of network status and decision making suggestions of machine; - No intent
- o Level 2 - Automated network: Automation of most service provisioning, network deployment, and maintenance of a comprehensive perception of network status and local machine decision making; - simple intent on service provisioning
- o Level 3 - Self-optimization network: Deep awareness of network status and automatic network control, meeting requirements of intent users of the network. - Intent based on network status cognition
- o Level 4 - Partial autonomous network: In a limited environment, people do not need to participate in decision-making and networks can adjust itself. - Intent based on limited AI
- o Level 5 - Autonomous network: In different network environments and network conditions, the network can automatically adapt to and adjust to meet people's intentions. - Intent based on AI

## 6. Intent Classification

This section proposes an intent classification approach that may help to classify mainstream intent related demos/tools.

The three classifications in this document have been proposed from scratch, following the methodology presented, through three iterations: one for carrier network intent solution, one for DC intent solution, and one for enterprise intent solution. For each intent solution, we identified the specific intent users and intent types. Then, we further identified intent scope, network scope, abstractions, and life-cycle requirements.

These classifications and the generated tables can be easily extended. For example, for the DC intent solution, a new category is identified, i.e. resource scope, and the classification table has been extended accordingly.

In the future, as new scenarios, applications, and domains are emerging, new classifications and taxonomies can be identified, following the proposed methodology.

The intent classifications have been documented to the best of our knowledge at this point in time. Additional classifications will most probably see the light in the future.

The output of the intent classification is the intent taxonomy introduced in the next sections.

Thus, this section first introduces the proposed intent classification methodology, followed by consolidated intent taxonomy for three intent solutions, and then by concrete examples of intent classifications for three different intent solutions (e.g. carrier network, data center, and enterprise) that were derived using the proposed methodology and then can be filled in for PoCs, demos, research projects or future drafts.

#### 6.1. Intent Classification Methodology

This section describes the methodology used to derive the initial classification proposed in the draft. The proposed methodology can be used to create new intent classifications from scratch, by analysing the solution knowledge. As well, the methodology can be used to update existing classification tables by adding or removing different solutions, intent users or intent types in order to cater for future scenarios, applications or domains.

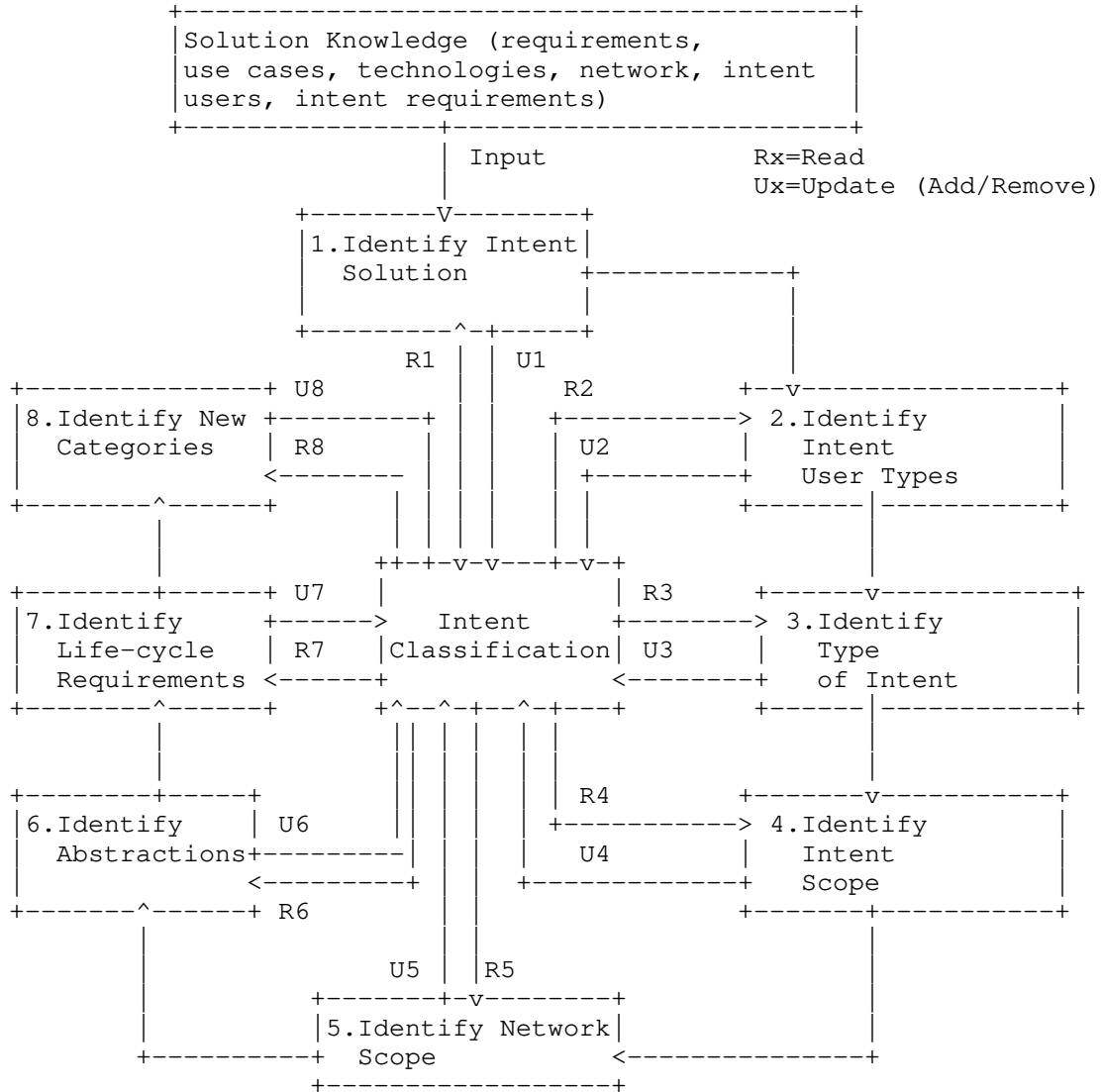


Figure 1 - Intent Classification Methodology

The intent classification workflow starts from the solution knowledge, which can provide information on requirements, use cases, technologies used, network properties, intent users that define and issue the intent request, and requirements. The following, defines the steps to classify an intent:

1. The information provided in the solution knowledge is given as input for identifying the intent solution (e.g. carrier, enterprise, and data center). Intent solutions are reviewed against the existing classification and they can either be used if present or added if not there or removed if not needed, from the classification. (R1-U1).
2. Identify the intent user types (e.g. customer, network operators, service operators, etc.), review existing intent classification and use the intent user type if present, add if it is not there or remove it if not needed (R2-U2).
3. Identify the types of intent (e.g. network intent, customer service intent) and then review existing classification and use/add/remove the intent type (R3-U3).
4. Identify the intent scopes (e.g. connectivity, application) based on the solution knowledge and then review existing classification and use/add/remove the identified intent scope (R4-U4).
5. Identify the network scopes (e.g. campus, radio access) and then review existing classification and either use it or add/remove the identified network scope (R5-U5).
6. Identify the abstractions (e.g. technical, non-technical) and then review existing classification and use/add/remove the abstractions (R6-U6).
7. Identify the life-cycle requirements (e.g. persistent, transient) and then review existing classification and use/add/remove the life-cycle requirements (R7-U7).
8. Identify any new categories and use/add the newly identified categories. New categories can be identified as new domains or applications are emerging, or new areas of concern (e.g. privacy, compliance) might arise, which are not listed in the current methodology.

## 6.2. Intent Taxonomy

The following taxonomy describes the various intent solutions, intent user types, intent types, intent scopes, network scopes, abstractions and life-cycle and represents the output of the intent classification tables for each of the solutions addressed (i.e. carrier, data center, and enterprise solutions).

The intent scope categories in Figure 2 are shared among the carrier, DC, and enterprise solutions. The abbreviations (Cx) in sections 6.3.2. 6.4.2. are introduced with the scope of fitting as column title in the following tables.



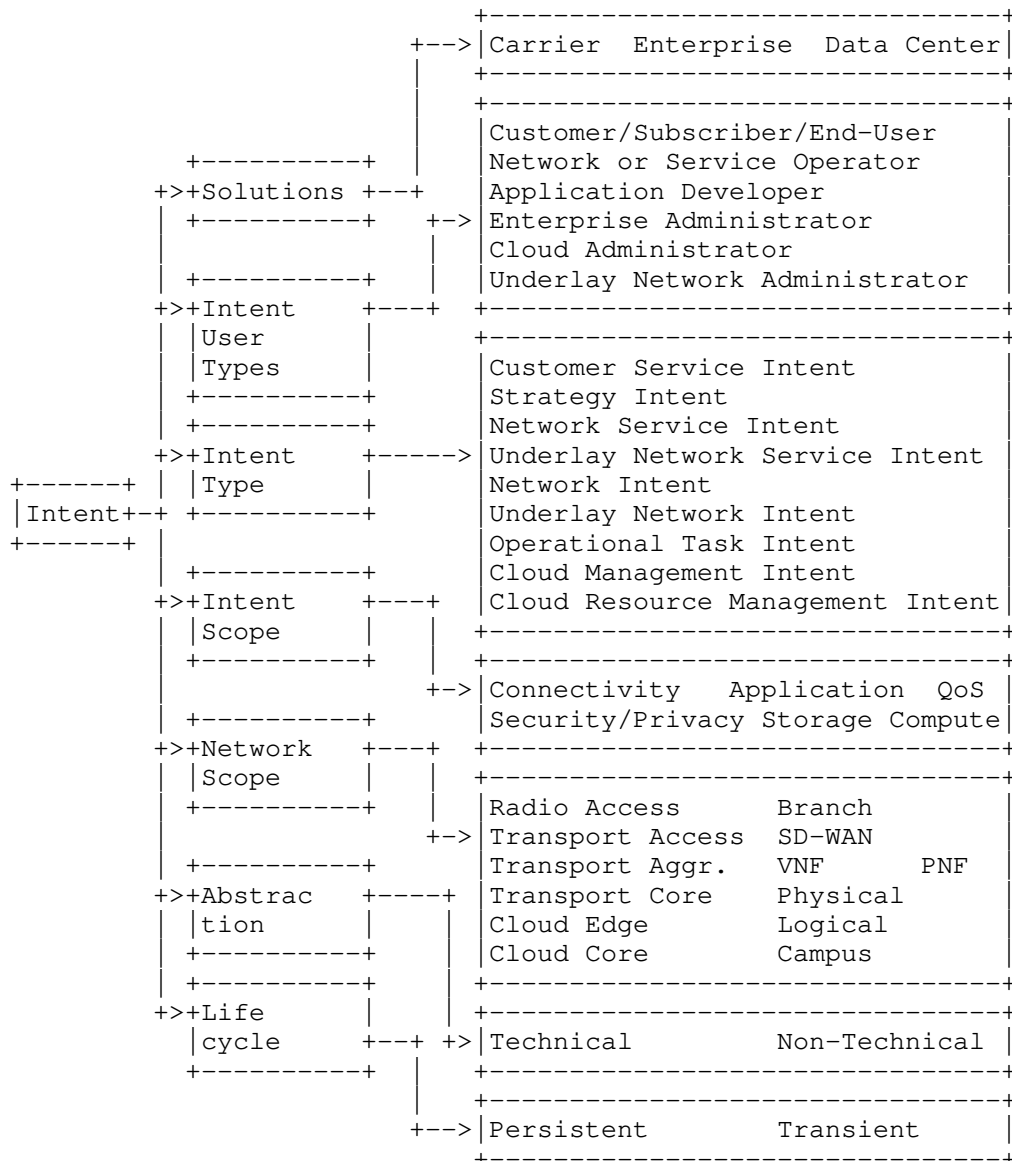


Figure 2 - Intent Taxonomy

### 6.3. Intent Classification for Carrier Solution

#### 6.3.1. Intent Users and Intent Types

This section addresses step 1, 2, and 3 from Figure 1 and the following table describes the intent users in carrier solutions and intent types with their descriptions for different intent users.

Intent User	Intent Type	Intent Type Description
Customer/ Subscriber	Customer Service Intent	Customer self-service with SLA and value added service Example: Always maintain high quality of service and high bandwidth for gold level subscribers. Operational statement: Measure the network congestion status, give different adaptive parameters to stations of different priority, thus in heavy load situation, make the bandwidth of the high-priority customers guaranteed. At the same time ensure the overall utilization of system, improve the overall throughput of the system.
	Strategy Intent	Customer designs models and policy intents to be used by customer service intents. Example: Request reliable service during peak traffic periods for apps of type video.
Network Operator	Network Service Intent	Service provided by network service operator to the customer (e.g. the service operator) Example: Request network service with delay guarantee for access customer A.
	Network Intent	Network operator requests network-wide (service underlay or other network-wide

		configuration) or network resource configurations (switches, routers, routing, policies). Includes connectivity, routing, QoS, security, application policies, traffic steering policies, configuration policies, monitoring policies, alarm generation for non-compliance, auto-recovery, etc. Example: Request high priority queueing for traffic of class A.
	Operational Task Intent	Network operator requests execution of any automated task other than network service intent and network intent (e.g. network migration, server replacements, device replacements, network software upgrades). Example: Request migration of all services in network N to backup path P.
	Strategy Intent	Network operator designs models, policy intents and workflows to be used by network service Intents, network intents and operational task intents. Workflows can automate any tasks that network operator often performed in addition to network service intents and network intents. Example: Ensure the load on any link in the network is not higher than 50%.

Service Operator	Customer Service Intent	Service operator's customer orders, customer service / SLA Example: Provide service S with guaranteed bandwidth for customer A.
	Network Service Intent	Service operator's network orders / network SLA Example: Provide network guarantees in terms of security, low latency and high bandwidth
	Operational Task Intent	Service operator requests execution of any automated task other than customer service intent and network service intent Example: Update service operator portal platforms and their software regularly. Move services from network operator 1 to network operator 2.
	Strategy Intent	Service operator designs models, policy intents and workflows to be used by customer service intents, network service intents and operational task intents. Workflows can automate any tasks that service operator often performed in addition to network service intents and network intents. Example: Request network service guarantee to avoid network congestion during special periods such as black Friday, and Christmas.
Application Developer	Customer Service Intent	Customer service intent API provided to the application developers Example: API to request network to watch HD video 4K/8K.

	Network Service Intent	Network service intent API provided to the application developers Example: API to request network service , monitoring and traffic grooming.
	Network Intent	Network intent API provided to the application developers Example: API to request network resources configuration.
	Operational Task Intent	Operational task intent API provided to the application developers. This is for the trusted internal operator / service providers / customer DevOps Example: API to request server migrations.
	Strategy Intent	Application developer designs models, policy and workflows to be used by customer service intents, network service intents and operational task intents. This is for the trusted internal operator/service provider/customer DevOps Example: API to design network load balancing strategies during peak times

Table 2 - Intent Classification for Carrier Solution

### 6.3.2. Intent Categories

This subsection addresses step 4 to 7 from Figure 1, and the following are the proposed categories:

- o Intent Scope: C1=Connectivity, C2=Security/Privacy, C3=Application, C4=QoS
- o Network Scope:
  - o Network Domain: C1=Radio Access, C2=Transport Access, C3=Transport Aggregation, C4=Transport Core, C5=Cloud Edge, C6=Cloud Core)
  - o Network Function (NF) Scope: C1=VNFs, C2=PNFs
- o Abstraction (ABS): C1=Technical (with technical feedback), C2=Non-technical (without technical feedback) see section 5.2. .
- o Life-cycle (L-C): C1=Persistent (full life-cycle), C2=Transient (short lived)

### 6.3.3. Intent Classification Example

This section depicts an example on how the methodology described in section 6.1. can be used in order to classify intents introduced in the 'A Multi-Level Approach to IBN' PoC demonstration [POC-IBN]. This PoC is led by academics carrying research in the area of SDN/NFV and the specific problem they are addressing is to apply the intent concept at different levels that correspond to different stakeholders. For this research work, they considered two types of intents: slice intents and service chain intents.

In this PoC [POC-IBN], a slice intent expresses a request for a network slice with two types of components: a set of top layer virtual functions, and a set of virtual switches and/or routers of L2/L3 VNFs. A service chain intent expressed a request for a service operated through a chain of service components running in L4-L7 virtual functions.

Following the intent classification methodology described step-by-step in section 6.1. , the following can be derived:

1. The intent solution for both intents is carrier network.
2. The intent user type is network operator for the slice intent, and service operator for the service chain intent.
3. The type of intent, is a network service intent for the slice intent, and a customer service intent for the service chain intent.

4. The intent scopes are connectivity and application.
5. The network scope is VNF, cloud edge, and cloud core.
6. The abstractions are with technical feedback for the slice intent, and without technical feedback for the service chain intent
7. The life-cycle is persistent.

The following table shows how to represent this information in a tabular form. The 'X' in the table refers to the slice intent, and the 'Y' in the table refers to the service chain intent.

Intent User	Intent Type	Intent Scope				NF Scope		Network Scope						ABS		L-C	
		C1	C2	C3	C4	C1	C2	C1	C2	C3	C4	C5	C6	C1	C2	C1	C2
Customer / Sub-scriber	Customer Service Intent																
	Strategy Intent																
Network Operator	Network Service Intent	X		X		X						X		X		X	
	Network Intent																
	Operational Task Intent																
	Strategy Intent																
Service Operator	Customer Service Intent	Y		Y		Y						Y	Y		Y	Y	
	Network Service Intent																
	Op Task Intent																
	Strategy Intent																



App Developer	Customer Intent																		
	Network Service Intent																		
	Network Intent																		
	Op Task Intent																		
	Strategy Intent																		

Table 3 - Intent Classification Example for Carrier Solution

#### 6.4. Intent Classification for Data Center Network Solutions

##### 6.4.1. Intent Users and Intent Types

The following table describes the intent users in DC network solutions and intent types with their descriptions for different intent users.

Intent User	Intent Type	Intent Type Description
Customer / Tenants	Customer Service	Customer self-service via tenant portal. Example: Request GPU computing and storage resources to meet 10k video surveillance services.
	Strategy Intent	This includes models and policy intents designed by customers/tenants to be reused later during instantiation. Example: Request dynamic computing and storage resources of the service in special and daily times.
Cloud Administrator	Cloud Management Intent	Configuration of VMs, DB Servers, app servers, connectivity, communication between VMs. Example: Request connectivity between VMs A,B,and C in network N1.
	Cloud Resource Management Intent	Policy-driven self-configuration and recovery / optimization Example: Request automatic life-cycle management of VM cloud resources.
	Operational Task Intent	Cloud administrator requests execution of any automated task other than cloud management intents and cloud resource management intents. Example: Request upgrade operating system to version X on all VMs in network N1.

		Operational statement: an intent to update a system might reconfigure the system topology (connect to a service and to peers), exchange data (update the content), and uphold a certain QoE level (allocate sufficient network resources). The network, thus, carries out the necessary configuration to best serve such an intent; e.g. setting up direct connections between terminals, and allocating fair shares of router queues considering other network services.
	Strategy Intent	Cloud administrator designs models, policy intents and workflows to be used by other intents. Automate any tasks that administrator often performs, in addition to life-cycle of cloud management intents and cloud management resource intents. Example: In case of emergency, automatically migrate all cloud resources to DC2.
Underlay Network Administrator	Underlay Network Service Intent	Service created and provided by the underlay network administrator. Example: Request underlay service between DC1 and DC2 with bandwidth B.
	Underlay Network Intent	Underlay network administrator requests some DCN-wide underlay network configuration or network resource configurations. Example: Establish and allocate DHCP address pool.
	Operational Task Intent	Underlay network administrator requests execution of the any automated task other than underlay network service and resource

Application Developer		intent. Example: Request automatic rapid detection of device failures and pre-alarm correlation.
	Strategy Intent	Underlay network administrator designs models, policy intents & workflows to be used by other intents. Automate any tasks that administrator often performs. Example: For all traffic flows that need NFV service chaining, restrict the maximum load of any VNF node/container below 50% and the maximum load of any network link below 70%.
	Cloud Management Intent	Cloud management intent API provided to the application developers. Example: API to request configuration of VMs, or DB Servers.
	Cloud Resource Management Intent	Cloud resource management intent API provided to the application developers. Example: API to request automatic life-cycle management of cloud resources.
	Underlay Network Service Intent	Underlay network service API provided to the application developers. Example: API to request real-time monitoring of device condition.
	Underlay Network Intent	Underlay network resource API provided to the application developers. Example: API to request dynamic management of IPv4 address pool resources.

	Operational Task Intent	Operational task intent API provided to the trusted application developer (internal DevOps). Example: API to request automatic rapid detection of device failures and pre-alarm correlation
	Strategy Intent	Application developer designs models, policy intents and building blocks to be used by other intents. This is for the trusted internal DCN DevOps. Example: API to request load balancing thresholds.

Table 4 - Intent Classification for Data Center Network Solutions

#### 6.4.2. Intent Categories

The following are the proposed categories:

- o Intent Scope: C1=Connectivity, C2=Security/Privacy, C3=Application, C4=QoS C5=Storage C6=Compute
- o Network Scope
  - o Network Domain: DC Network
  - o DCN Network (DCN Net) Scope: C1=Logical, C2=Physical
  - o DCN Resource (DCN Res) Scope: C1=Virtual, C2=Physical
- o Abstraction (ABS): C1=Technical (with technical feedback), C2=Non-technical (without technical feedback), see section 5.2.
- o Life-cycle (L-C): C1=Persistent (full life-cycle), C2=Transient (short lived)

#### 6.4.3. Intent Classification Example

This section depicts an example on how the methodology described in section 6.1. can be used by the research community to classify intents. As mentioned in 6.3.3. a successful use of the classification proposed in this draft is introduced in the 'A Multi-Level Approach to IBN' PoC demonstration [POC-IBN]. The PoC is led by academics carrying research in the area of SDN/NFV and the specific problem they are addressing is to apply the intent concept at different levels that correspond to different stakeholders.

For their research work, they considered two types of intents: slice intents and service chain intents. For the data center solution, only the slice intent is relevant.

As already mentioned in section 6.3.3. , a slice intent expresses a request for a network slice with two types of components: a set of top layer virtual functions, and a set of virtual switches and/or routers of L2/L3 VNFs.

Following the intent classification methodology described step-by-step in section 6.1. , we identify the following:

1. The intent solution is for the data center.
2. The intent user type is the cloud administrator for the slice intent and service chain intent.
3. The type of intent, is a cloud management intent, for the slice intent.

4. The intent scopes are connectivity and application.
5. The network scope is logical, and the resource scope is virtual.
6. The abstractions are with technical feedback for the slice intent.
7. The life-cycle is persistent.

The following table shows how to represent this information in a tabular form, where the 'X' in the table refers to the slice intent.

Intent User	Intent Type	Intent Scope						DCN Res		DCN Net		ABS		L-C	
		C1	C2	C3	C4	C5	C6	C1	C2	C1	C2	C1	C2	C1	C2
Customer /Tenants	Customer Service Intent														
	Strategy Intent														
Cloud Admin	Cloud Management Intent	X		X				X		X		X		X	
	Cloud Resource Management Intent														
	Operational Task Intent														
	Strategy Intent														
Underlay Network Admin	Underlay Network Intent														
	Underlay Network Resource Intent														
	Operational Task Intent														
	Strategy														



	Intent																		
App Developer	Cloud Management Intent																		
	Cloud Resource Management Intent																		
	Underlay Network Intent																		
	Underlay Network Resource Intent																		
	Operational Task Intent																		
	Strategy Intent																		

Table 5 - Intent Classification Example for Data Center Network Solutions

## 6.5. Intent Classification for Enterprise Solution

### 6.5.1. Intent Users and Intent Types

The following table describes the intent users in enterprise solutions and their intent types.

Intent User	Intent Type	Intent Type Description
End-User	Customer Service Intent	Enterprise end-user self-service or applications, enterprise may have multiple types of end-users. Example: Request access to VPN service. Request video conference between end-user A and B.
	Strategy Intent	This includes models and policy intents designed by end-users to be used by end-user intents and their applications. Example: Create a video conference type for a weekly meeting.
Enterprise Administrator (internal or MSP)	Network Service Intent	Service provided by the administrator to the end-users and their applications. Example: For any end-user of application X, the arrival of hologram objects of all the remote tele-presenters should be synchronised within 50ms to reach the destination viewer for each conversation session. Create management VPN connectivity for type of service A. Operational statement: The job of the network layer is to ensure that the delay is between 50-70ms through

		the routing algorithm. At the same time, the node resources need to meet the bandwidth requirements of 4K video conferences.
	Network Intent	Administrator requires network wide configuration (e.g. underlay, campus) or resource configuration (switches, routers, policies). Example: Configure switches in campus network 1 to prioritise traffic of type A. Configure YouTube as business non-relevant.
	Operational Task Intent	Administrator requests execution of any automated task other than network service intents and network intents. Example: Request network security automated tasks such as web filtering and DDOS cloud protection.
	Strategy Intent	Administrator designs models, policy intents and workflows to be used by other intents. Automate any tasks that administrator often performs. Example: In case of emergency, automatically shift all traffic of type A through network N.
Application Developer	End-User Intent	End-user service / application intent API provided to the application developers. Example: API for request to open a VPN service.
	Network Service Intent	Network service API provided to application developers. Example: API for request network

		bandwidth and latency for hosting video conference.
	Network Intent	Network API provided to application developers. Example: API for request of network devices configuration.
	Operational Task Intent	Operational task intent API provided to the trusted application developer (internal DevOps). Example: API for requesting automatic monitoring and interception for network security
	Strategy Intent	Application developer designs models, policy intents and building blocks to be used by other intents. This is for the trusted internal DevOps. Example: API for strategy intent in case of emergencies.

Table 6 - Intent Classification for Enterprise Solution

### 6.5.2. Intent Categories

The following are the proposed categories:

- o Intent Scope: C1=Connectivity, C2=Security/Privacy, C3=Application, C4=QoS
- o Network (Net) Scope: C1=Campus, C2=Branch, C3=SD-WAN
- o Abstraction (ABS): C1=Technical (with technical feedback), C2=Non-technical (without technical feedback), see section 5.2.
- o Life-cycle (L-C): C1=Persistent (full life-cycle), C2=Transient (short lived)

The following is the intent classification table example for enterprise solutions.

Intent User	Intent Type	Intent Scope				Net			ABS		L-C	
		C1	C2	C3	C4	C1	C2	C3	C1	C2	C1	C2
End-User	Customer Service Intent											
	Strategy Intent											
Enterprise Administrator	Network Service Intent											
	Network Intent											
	Operational Task Intent											
	Strategy Intent											
Application Developer	End-User Intent											
	Network Service Intent											
	Network Intent											



## 10. Contributors

The following people all contributed to creating this document:

Xueyuan Sun, China Telecom  
Will (Shucheng) Liu, Huawei  
Ying Chen, China Unicom  
John Strassner, Huawei  
Weiping Xu, Huawei  
Richard Meade, Huawei

## 11. Acknowledgments

Special thanks to Xueyuan Sun from China Telecom for significant contributions to this document, and to Will (Shucheng) Liu from Huawei for contributions and guidance.

This document has benefited from reviews, suggestions, comments and proposed text provided by the following members, listed in alphabetical order: Mehdi Bezhaf, Brian E Carpenter, Laurent Ciavaglia, Benoit Claise, Alexander Clemm, Yehia Elkhatib, Jerome Francois, Pedro Andres Aranda Gutierrez, Daniel King, Branislav Meandzija, Bob Natale, Juergen Schoenwaelder, Xiaolin Song, Jeff Tantsura.

We thank to Barbara Martini, Walter Cerroni, Molka Gharbaoui, Davide Borsatti, for contributing with their 'A multi-level approach to IBN' PoC demonstration a first attempt to adopt the intent classification methodology.

## 12. Informative References

- [Bezahaf21] Bezhaf, M., Davies, E., Rotsos, C. and Race, N., "To All Intents and Purposes: Towards Flexible Intent Expression," 2021 IEEE 7th International Conference on Network Softwarization (NetSoft), 2021.
- [Bezhafl9] Bezhaf, M., Hernandez, MP, Bardwell, L., Davies, E., Broadbent, M., King, D. and Hutchison, D. , "Self-Generated Intent-Based System," 2019 10th International Conference on Networks of the Future (NoF), 2019.

- [Jacobs18] Jacobs, A.S., Pfitscher, R.J., Ferreira, R.A., and Granville, L.Z., "Refining Network Intents for Self-Driving Networks", Proceedings of the Afternoon Workshop on Self-Driving Networks (SelfDN 2018), 2018.
- [Banerjee21] Banerjee, A., Mwanje, S. and Carle, G., "Contradiction Management in Intent-driven Cognitive Autonomous RAN", 2021.
- [Tian19] Tian, B., Zhang, X., Zhai, E., Liu, H. H., Ye, Q., Wang, C., and Zhao, B. Y., "Safely and automatically updating in-network ACL configurations with intent language", SIGCOMM '19, 2019.
- [Leivadeas21] Leivadeas, A. and Falkner, M., "VNF Placement Problem: A Multi-Tenant Intent-Based Networking Approach," 24th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), 2021.
- [Davoli21] Davoli, G., "Programmability and Management of Software-defined Network Infrastructures", 2021.
- [Padovan20] Padovan, S., "Design and Implementation of a Blockchain Intent Management System", 2020.
- [Mehmood21] Mehmood, K., Kravlevska, K., and Palma, D., "Intent-driven Autonomous Network and Service Management in Future Networks: A Structured Literature Review", 2021.
- [Szilagyi21] Szilagyi, P., "I2BN: Intelligent Intent Based Networks", Journal of ICT Standardization, 2021.
- [POC-IBN] Barbara Martini, Walter Cerroni, Molka Gharbaoui, Davide Borsatti, "A multi-level approach to IBN", July 2020, <https://www.ietf.org/proceedings/108/slides/slides-108-nmrg-ietf-108-hackathon-report-a-multi-level-approach-to-ibn-02>
- [IFIP-NSM] IFIP - Network and Service Management Taxonomy, <https://www.simpleweb.org/ifip/taxonomy.html>
- [ONF] ONF, "Intent Definition Principles", 2017, <[https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR-523\\_Intent\\_Definition\\_Principles.pdf](https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR-523_Intent_Definition_Principles.pdf)>.



- [ONOS] ONOS, "ONOS Intent Framework", 2017,  
<<https://wiki.onosproject.org/display/ONOS/Intent+Framework>  
>.
- [CLEMM] A. Clemm, L. Ciavaglia, L. Granville, J. Tantsura, "Intent-  
Based Networking - Concepts and Overview", Work in  
Progress, draft-irtf-nmrg-ibn-concepts-definitions-05,  
February 2021, [https://tools.ietf.org/html/draft-irtf-nmrg-  
ibn-concepts-definitions-05](https://tools.ietf.org/html/draft-irtf-nmrg-ibn-concepts-definitions-05)
- [TMF-auto] Aaron Richard Earl Boasman-Patel, et, A whitepaper of  
Autonomous Networks: Empowering Digital Transformation For  
the Telecoms Industry, [inform.tmforum.org](http://inform.tmforum.org), 15 May, 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate  
Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC7575] Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A.,  
Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic  
Networking: Definitions and Design Goals", RFC 7575, June  
2015.
- [RFC8328] Liu, W., Xie, C., Strassner, J., Karagiannis, G., Klyus,  
M., Bi, J., Cheng, Y., and D. Zhang, "Policy-Based  
Management Framework for the Simplified Use of Policy  
Abstractions (SUPA)", March 2018.
- [RFC3198] Westerinen, A., Schnizlein, J., Strassner, J.,  
Scherling, M., Quinn, B., Herzog, S., Huynh, A., Carlson,  
M., Perry, J., Waldbusser, S., "Terminology for Intent-  
driven Management", RFC 3198, November 2001.
- [RFC6020] Bjorlund, M., "YANG - A Data Modelling Language for Network  
Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC7285] R. Alimi, R. Penno, Y. Yang, S. Kiesel, S. Previdi, W.  
Roome, S. Shalunov, R. Woundy "Application-Layer Traffic  
Optimization (ALTO) Protocol", September 2014.
- [ANIMA] Du, Z., "ANIMA Intent Policy and Format", 2017,  
<[https://datatracker.ietf.org/doc/draft-du-anima-an-  
intent/](https://datatracker.ietf.org/doc/draft-du-anima-an-intent/)>.
- [SUPA] Strassner, J., "Simplified Use of Policy Abstractions",  
2017, <[https://datatracker.ietf.org/doc/draft-ietf-sup-  
generic-policy-info-model/?include\\_text=1](https://datatracker.ietf.org/doc/draft-ietf-sup-generic-policy-info-model/?include_text=1)>.

[ANIMA-Prefix] Jiang, S., Du, Z., Carpenter, B., and Q. Sun,  
"Autonomic IPv6 Edge Prefix Management in Large-scale  
Networks", draft-ietf-anima-prefix-management-07 (work in  
progress), December 2017.

#### Authors' Addresses

Chen Li  
China Telecom  
No.118 Xizhimennei street, Xicheng District  
Beijing 100035  
P.R. China  
Email: lichen.bri@chinatelecom.cn

Olga Havel  
Huawei Technologies  
Ireland  
Email: olga.havel@huawei.com

Adriana Olariu  
Huawei Technologies  
Ireland  
Email: adriana.olariu@huawei.com

Pedro Martinez-Julia  
NICT  
Japan  
Email: pedro@nict.go.jp

Jeferson Campos Nobre  
Federal University of Rio Grande do Sul  
Porto Alegre  
Brazil  
Email: jcnobre@inf.ufrgs.br

Diego R. Lopez  
Telefonica I+D  
Don Ramon de la Cruz, 82  
Madrid 28006  
Spain  
Email: diego.r.lopez@telefonica.com



NMRG	P. Martinez-Julia, Ed.
Internet-Draft	NICT
Updates: draft-pedro-nmrg-intelligent-	S. Homma
reasoning-01 (if approved)	NTT
Intended status: Informational	D. R. Lopez
Expires: 23 April 2022	TID
	20 October 2021

Artificial Intelligence Framework for Network Management  
draft-pedro-nmrg-ai-framework-00

Abstract

The adoption of artificial intelligence (AI) in network management (NM) solutions is becoming a reality. It is mainly supported by the need to resolve complex problems arisen from the acceptance of SDN/NFV technologies as well as network slicing. Thus, the AINEMA framework, as discussed in this document, is required to keep focus and organize the efforts on applying AI to NM. This is enlarged by the inclusion of external events in NM operations as well as the consideration of a full intelligence process instead of simple AI-based guesses. Such process will be highly based in reasoning and formal and target-based intelligence analysis and decision. This will allow computer and network system infrastructures to grow in complexity. The same applies to user demands. The construction and maintenance of AINEMA requires a comprehensive inclusion of several mechanisms. For instance, although there has been a lot of effort to make Machine Learning (ML) solutions reliable and acceptable, other mechanisms have been forgotten. It is the particular case of reasoning, which is the key within AINEMA. It will provide enormous benefits to NM solutions by, for example, inferring new knowledge and applying different kind of rules (e.g. logical) to choose from several actions. While ML solutions work with data, so their only requirement from the network infrastructure is data retrieval, AINEMA will work in collaboration to the network it is managing. This makes the challenges arisen from intelligent reasoning essential for the evolution of NM. They will be addressed within the context of AINEMA.

Status of This Memo

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

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

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

This Internet-Draft will expire on 23 April 2022.

#### Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

#### Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	4
3. Background . . . . .	4
3.1. Virtual Computer and Network Systems . . . . .	4
3.2. SDN and NFV . . . . .	5
3.3. Management and Control . . . . .	5
3.4. Slice Gateway (SLG) . . . . .	6
3.5. Artificial Intelligence and Machine Learning . . . . .	6
3.6. Briefing Artificial Intelligence . . . . .	6
4. AI Framework for NM . . . . .	7
4.1. AINEMA Operation . . . . .	8
4.2. Closed Loop . . . . .	9
4.3. Network Intelligence: From Data to Wisdom . . . . .	10
4.4. External Event Detectors . . . . .	11
4.5. Anticipation of Network Requirements . . . . .	11
4.6. Intelligent Reasoning . . . . .	13
4.7. Gaps and Standardization Issues . . . . .	14
5. Relation to Other IETF/IRTF Initiatives . . . . .	14
6. IANA Considerations . . . . .	14
7. Security Considerations . . . . .	15

8. Acknowledgements . . . . .	15
9. References . . . . .	15
9.1. Normative References . . . . .	15
9.2. Informative References . . . . .	15
Appendix A. AINEMA Information Model . . . . .	17
A.1. Tree Structure . . . . .	17
A.1.1. event-payloads . . . . .	17
A.1.1.1. basic . . . . .	18
A.1.1.2. seismometer . . . . .	18
A.1.1.3. bigdata . . . . .	18
A.1.2. external-events . . . . .	19
A.1.3. notifications/event . . . . .	19
A.2. YANG Module . . . . .	19
Appendix B. The Autonomic Resource Control Architecture (ARCA) . . . . .	21
Appendix C. ARCA Integration With ETSI-NFV-MANO . . . . .	22
C.1. Functional Integration . . . . .	23
C.2. Target Experiment and Scenario . . . . .	26
C.3. OpenStack Platform . . . . .	27
C.4. Initial Results . . . . .	29
Authors' Addresses . . . . .	31

## 1. Introduction

The current network ecosystem is quickly evolving from an almost fixed network to a highly flexible, powerful, and somehow hybrid system. Network slicing, Software Defined Networking (SDN), and Network Function Virtualization (NFV) provide the basis for such evolution. The need to automate the management and control of such systems has motivated the move towards autonomic networking (ANIMA) and the inclusion of AI solutions alongside the management plane of the network, enough justified by the increasing size and complexity of the network, which exposes complex problems that must be resolved in scales that escape human possibilities. Therefore, in order to allow current computer and network system infrastructures to constantly grow in complexity, in parallel to the demands of users, the AI solutions must work together with other network management solutions.

However, exploiting the possibilities of AI is not an easy task. There has been a lot of effort to make Machine Learning (ML) solutions reliable and acceptable but, at the same time, other mechanisms have been forgotten. It is the particular case of reasoning. Although it can provide enormous benefits to management solutions by, for example, inferring new knowledge and applying different kind of rules (e.g. logical) to choose from several actions, it has received little attention. While ML solutions work with data, so their only requirement from the network infrastructure

is data retrieval, reasoning solutions work in collaboration to the network they are managing. This makes the challenges arisen from intelligent reasoning to be a key for the evolution of network management towards the full adoption of AI.

The present document aims to gather the necessary information for getting the most benefits from the application of intelligent reasoning to network management, including, but not limited to, defining the gaps that must be covered for reasoning to be correctly integrated into network management solutions.

## 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 RFC 2119 [RFC2119].

## 3. Background

### 3.1. Virtual Computer and Network Systems

The continuous search for efficiency and cost reduction to get the most optimum exploitation of available resources (e.g. CPU power and electricity) has conducted current physical infrastructures to move towards virtualization infrastructures. Also, this trend enables end systems to be centralized and/or distributed, so that they are deployed to best accomplish customer requirements in terms of resources and qualities.

One of the key functional requirements imposed to computer and network virtualization is a high degree of flexibility and reliability. Both qualities are subject to the underlying technologies but, while the latter has been always enforced to computer and network systems, flexibility is a relatively new requirement, which would not have been imposed without the backing of virtualization and cloud technologies.

### 3.2. SDN and NFV

SDN and NFV are conceived to bring high degree of flexibility and conceptual centralization qualities to the network. On the one hand, with SDN, the network can be programmed to implement a dynamic behavior that changes its topology and overall qualities. Moreover, with NFV the functions that are typically provided by physical network equipment are now implemented as virtual appliances that can be deployed and linked together to provide customized network services. SDN and NFV complements to each other to actually implement the network aspect of the aforementioned virtual computer and network systems.

Although centralization can lead us to think on the single-point-of-failure concept, it is not the case for these technologies. Conceptual centralization highly differs from centralized deployment. It brings all benefits from having a single point of decision but retaining the benefits from distributed systems. For instance, control decisions in SDN can be centralized while the mechanisms that enforce such decisions into the network (SDN controllers) can be implemented as highly distributed systems. The same approach can be applied to NFV. Network functions can be implemented in a central computing facility, but they can also take advantage of several replication and distribution techniques to achieve the properties of distributed systems. Nevertheless, NFV also allows the deployment of functions on top of distributed systems, so they benefit from both distribution alternatives at the same time.

### 3.3. Management and Control

The introduction of virtualization into the computer and network system landscape has increased the complexity of both underlying and overlying systems. On the one hand, virtualizing underlying systems adds extra functions that must be managed properly to ensure the correct operation of the whole system, which not just encompasses underlying elements but also the virtual elements running on top of them. Such functions are used to actually host the overlying virtual elements, so there is an indirect management operation that involves virtual systems. Moreover, such complexities are inherited by final systems that get virtualized and deployed on top of those virtualization infrastructures.

In parallel, virtual systems are empowered with additional, and widely exploited, functionality that must be managed correctly. It is the case of the dynamic adaptation of virtual resources to the specific needs of their operation environments, or even the composition of distributed elements across heterogeneous underlying infrastructures, and probably providers.



Taking both complex functions into account, either separately or jointly, makes clear that management requirements have greatly surpassed the limits of humans, so automation has become essential to accomplish most common tasks.

#### 3.4. Slice Gateway (SLG)

A slice gateway (SLG) (see [I-D.homma-rtgwg-slice-gateway]) is basically a component in the data plane and has the roles of data packet processing. Moreover, it provides an interface to export its functions for interacting with control and management components, so that it is quite relevant for implementing the requirements described above within the network slicing domain.

Furthermore, an SLG might be required to support handling services provided on network slices in addition to controlling them because an SLG is the edge node on an end-to-end network slice (E2E-NS).

Therefore, the SLG exposes the following requirements:

- \* Data plane for NSs as infrastructure.
- \* Control/management plane for NSs as infrastructure.
- \* Data plane for services on NSs.
- \* Control/management plane for services on NSs.

In summary, SLG provides the required functions for the enforcement of AI decisions in multi-domain (and federated) network slices, so it will play a key role in general network management.

#### 3.5. Artificial Intelligence and Machine Learning

ML is not AI. AI has a broader spectrum of methods, some of them are already exploited in the network for a long time. Perception, reasoning, and planning are still not fully exploited in the network.

#### 3.6. Briefing Artificial Intelligence

Intelligence does not directly imply intelligent. On the one hand, intelligence emphasizes data gathering and management, which can be processed by systematic methods or intelligent methods. On the other hand, intelligent emphasizes the reasoning and understanding of data to actually "posses" the intelligence.

The justification of applying AI in network (and) management is sometimes overseen. First, management decisions are more and more complex. We have moved from asking simple questions ("Is there a problem in my system?") to much more complex ones ("Where should I migrate this VM to accomplish my goals?"). Moreover, operation environments are more and more dynamic. On the one hand, softwarization and programmability elevate flexibility and allow networks to be totally adapted to their static and/or dynamic requirements. On the other hand, network virtualization highly enables network automation.

The new functions and possibilities allow network devices to become autonomic. However, they must take complex decisions by themselves, without human intervention, realizing the "dream" of Zero-Touch Networks, which exploit fully programmable elements and advanced automation methods (ETSI ZSM). Nevertheless, we have to remember that AI methods are just resources, not solutions. They will not replace the human decisions, just complement and "automate" them.

#### 4. AI Framework for NM

The basic concept is to develop a generic, scalable, deployable and trustworthy AI framework for network management (AINEMA). The framework will be used to design, deploy, instantiate, scale and validate AI solutions for network operation, administration and management (OAM). The framework will particularly target E2E network management. The framework is underpinned by the principle that a generic and scalable AI framework will allow for more general-purpose AI solutions to network OAM, which can be scaled from one network domain to multiple network domains, and to multi-site and multi-tenant scenarios.

The key components of such a framework are:

- \* The data framework. It is responsible for acquiring, modelling, storing, and distributing data, both historical, collected off-line, and real-time, on-line, from different parts of a network in a unified and efficient manner. It also provides the internal communication layer to the AI framework and serves as the communication path between the AI framework and network orchestration entities.
- \* The AI modules. They contain the AI functions that individually or collectively accomplish local, E2E or global intelligent tasks for network OAM.

- \* The AI hub. It receives data, knowledge, and localized decisions from AI modules and outputs desired actions as recommendations to network management entities. The AI hub is also in charge of the life cycle management of the AI modules.

#### 4.1. AINEMA Operation

The correct and pertinent operation of AINEMA provides enormous benefits, mainly in terms of making complex management operations feasible and improving the performance of typically expensive tasks. By taking advantage of these benefits, the amount of data that can be analyzed to make decisions on the network can be hugely increased.

As a result, AINEMA makes possible to leverage intelligence for network management operations. Instead of just being focused on the analysis of performance measurements retrieved from the managed network and the subsequent decision (proaction or reaction), the extension of management operation enabled by AINEMA encompasses different sub-processes.

First, AINEMA has a sub-process for retrieving the performance measurements from the managed network. This is the same found in typical management processes. Moreover, AINEMA encourages the application of the same ML techniques to obtain some insight of the situation of the managed network.

Second, AINEMA incorporates a reasoning sub-process. It receives both the output of the previous sub-process and additional context information, which can be provided by an external event detector, as described below. Then, this sub-process finds out and particularizes the rules that are governing the situation described above. Such rules are semantically constructed and will abstract the situation of the network in terms of logical and other semantic concepts, together with actions and transformations that can be applied to those rules. All such constructions will be stored in the Intelligent Network Management Knowledge Base (INMKB), which will follow a pre-determined ontology and will also extend the knowledge by applying basic and atomic logic inference statements.

Third, AINEMA defines the solving sub-process. It works as follows. Once obtained the abstracted situation of the managed network and the rules to it, the solving subprocess builds a graph with all semantic constructions. It reflects the managed network, since all network elements have their semantic counterpart, but it also has all situations, rules, actions, and even the measurements. The solving sub-process applies ontology transformations to find a graph that is acceptable in terms of the associated situation and its adherence to administrative goals.

Fourth, AINEMA incorporates the planning sub-process. It receives the solution graph obtained by the previous sub-process and makes a linear plan of actions to execute in order to enforce the required changes into the network. The actions used by this planning sub-process are the building blocks of the plan. Each block will be defined with a precondition, invariant, and postcondition. A planning algorithm should be used to obtain such plan of actions by linking the building blocks so they can be enforced to finally adapt the managed network to get the desired situation.

All these processes must be executed in parallel, using strong inter-process communication and synchronization constraints. Moreover, the requests to the underlying infrastructure for the adaptation of the managed network will be sent to the corresponding controllers without waiting for finishing the deliberation cycle. This way, the time required by the whole cycle is highly reduced. This can be possible because of the assumptions and anticipations tied to AINEMA and the intelligence it denotes.

#### 4.2. Closed Loop

AINEMA follows the closed loop methodology to achieve and assess the accomplishment of network management goals. It ensures that the state of the network is continuously within the working parameters desired by its administrators. This is enforced among all management cycles along the full life-cycle of the network.

To obtain the benefits from integrating AI within the closed loop, AINEMA processes must be re-wired to connect their outputs to their inputs, so obtaining feedback analysis. Moreover, an additional process must be defined for ensuring that the objectives defined in the last steps of AINEMA are actually present in the near future situation of the managed network.

In addition, the data plane elements, such as the SLG described above, must provide some capabilities to make them coherent to the closed control loop. Particularly, they must provide symmetric enforcement and telemetry interfaces, so that the elements composing the managed network can be modified and monitored using the same identifiers and having the same assumptions about their topology and context. For instance, SLG must be able to provide the needed functionality to enable AINEMA to request SLG to set up and connect the necessary structures for telemetry collection and request slice switching.

#### 4.3. Network Intelligence: From Data to Wisdom

Enabling AINEMA with full intelligence process extends the analytics and decision power beyond current AI and ML solutions. Instead of just analyzing observations, the incorporation of intelligence processes to AINEMA makes hypotheses on the current and projected situation of the managed system and finds evidences to justify it. This process is much faster and much more effective than relying on data. This is because the hypotheses will be formally formulated within the scope and policies established by administrators.

Several hypotheses can be formulated in parallel. After gathering evidences for all of them, the one that has the strongest evidences can be taken as real and the potential effects can be countermeasured or prevented (anticipated) as dicussed below. As AI methods gain access to a huge amount of intelligence data from the systems they manage, they become more and more able to take strategic decisions, mainly deriving such data to knowledge towards wisdom. This supports the well known DIKW process (Data, Information, Knowledge, Wisdom) that enables elements to operate autonomously, subject to the goals established by administrators.

In such way, AI methods can be guided by the events or situations found in underlying networks in a constantly evolving model. We can call it the Knowledge (and Intelligence) Driven Network. In this new network architecture, the structure itself of the network results from reasoning on intelligence data. The network adapts to new situations without requiring human involvement but administrative policies are still enforced to decisions. Nevertheless, intelligence data must be managed properly to exploit all its potential. Data with high accuracy and high frequency will be processed in real-time. Meanwhile, fast and scalable methods for information retrieval and decision enfrocement become essential to the objectives of the network.

To achieve such goals, AI algorithms must be adapted to work on network problems. Joint physical and virtual network elements can form a multi-agent system focused on achieving such system goals. It can be applied to several use-cases. For instance, it can be used for predicting traffic behaviour, iterative network optimization, and assessment of administrative policies.

#### 4.4. External Event Detectors

Current mechanisms for automated management and control rely only on the continuous monitoring of the resources they control or the underlying infrastructure that host them. However, there are several other sources of information that can be exploited to make the systems more robust and efficient. It is the case of the notifications that can be provided by physical or virtual elements or devices that are watching for specific events, hence called external event detectors.

The external event detectors are a huge source of intelligence data that can be used as evidence to demonstrate the hypotheses formulated by AINEMA. More specifically, although the notifications provided by these external event detectors are related to successes that occur outside the boundaries of the controlled system, such successes can affect the typical operation of controlled systems. For instance, a heavy rainfall or snowfall can be detected and correlated to a huge increase in the amount of requests experienced by some emergency support service. Therefore, the evidence they provide can be even stronger than performance measurements obtained by the managed system and, in general, they will be used for anticipating requirements much more effectively.

#### 4.5. Anticipation of Network Requirements

One of the main goals of the MANO mechanisms is to ensure the virtual computer and network system they manage meets the requirements established by their owners and administrators. It is currently achieved by observing and analyzing the performance measurements obtained either by directly asking the resources forming the managed system or by asking the controllers of the underlying infrastructure that hosts such resources. Thus, under changing or eventual situations, the managed system must be adapted to cope with the new requirements, increasing the amount of resources assigned to it, or to make efficient use of available infrastructures, reducing the amount of resources assigned to it.

However, the time required by the infrastructure to make effective the adaptations requested by the MANO mechanisms is longer than the time required by client requests to overload the system and make it discard further client requests. This situation is generally undesired but particularly dangerous for some systems, such as the emergency support system mentioned above. Therefore, in order to avoid the disruption of the service, the change in requirements must be anticipated to ensure that any adaptation has finished as soon as possible, preferably before the target system gets overloaded or underloaded.

AINEMA exploits ARCA (Appendix B). ARCA is integrated to NFV-MANO to implement a closed loop network management that integrates external events. ARCA provides to AINEMA the ability to correlate previous external events (causes) with current performance measurements (effects). It also is able to find the preventive actions (anticipated countermeasures) required to avoid the effects once the causes have been detected. Thus, AINEMA is able to enforce the necessary adaptations to avoid failure of the managed system beforehand, particularly before the system performance metrics have actually changed.

The following abstract algorithm formalizes the workflow expected to be followed by the different implementations of the operation proposed here.

```
while TRUE do
  event = GetExternalEventInformation()
  if event != NONE then
    anticipated_resource_amount = Anticipator.Get(event)
    if IsPolicyCompliant(anticipated_resource_amount) then
      current_resource_amount = anticipated_resource_amount
      anticipation_time = NOW
    end if
  end if
  anticipated_event = event
  if anticipated_event != NONE and
    (NOW - anticipation_time) > EXPIRATION_TIME then
    current_resource_amount = DEFAULT_RESOURCE_AMOUNT
    anticipated_event = NONE
  end if
  state = GetSystemState()
  if not IsAcceptable(state, current_resource_amount) then
    current_resource_amount = GetResourceAmountForState(state)
    if anticipated_event is not NONE then
      Anticipator.Set
        (anticipated_event, current_resource_amount)
      anticipated_event = NONE
    end if
  end if
end while
```

This algorithm considers both internal and external events to determine the necessary control and management actions to achieve the proper anticipation of resources assigned to the target system. We propose the different implementations to follow the same approach so they can guess what to expect when they interact. For instance, a consumer, such as an Application Service Provider (ASP), can expect some specific behavior of the Virtual Network Operator (VNO) from which it is consuming resources. This helps both the ASP and VNO to properly address resource fluctuations.

#### 4.6. Intelligent Reasoning

It is trivial for anybody to understand that the behavior or the network results from user activity. For instance, more users means more traffic. However, it is not commonly considered that user activity has a direct dependency on events that occur outside the boundaries of the networks they use. For example, if a video becomes trendy, the load of the network that hosts the video increases, but also the load of any network with users watching the video. In the same way, if a natural incident occurs (e.g. heavy rainfall, earthquake), people try to contact their relatives and the load of a telephony network increases. From this we can easily find out that there is a clear causality relation between events occurring in the real and digital world and the behaviour of the network (aka. The Internet).

Network management outcomes, in terms of system stability, performance, reliability, etc., would greatly improve by exploiting such causality relation. An easy and straightforward way to do so is to apply AI reasoning methods. These methods can be used to "guess" the effect for a given cause. Moreover, reasoning can be used to choose the specific events that can impact the system, so being the cause for some effect.

Meanwhile, reasoning on network behavior from performance measurements and external events places some challenges. First, external event information must cross the administrative domain of the network to which it is relevant. This means that there must be interfaces and security policies that regulate how information is exchanged between the external event detector, which can be some sensor deployed in some "smart" place (e.g. smart city, smart building), and the management solution, which resides inside the administrative domain of the managed network. This function must be highly conformed and regulated, and the protocols used to achieve it must be widely accepted and tested, in order for it to exploit the overall potential of external events.



Second, enough meta-data must be associated to performance measurements to clearly identify all aspects of the effects, so that they can be traced back to the causes (events). Such meta-data must follow an ontology (information model) that is somewhat common and widely accepted or, at leaset, to be able to easily transform it among the different formats and models used by different vendors and software.

Third, the management ontology must be extended by all concepts from the boundaries of the managed network, its external environment (surroundings), and any entity that, albeit being far away, can impact on the function of the managed network.

#### 4.7. Gaps and Standardization Issues

Several gaps and standardization issues arise from applying AI and reasoning to network management solutions:

- \* Methods from different providers/vendors must be able to coexist and work together, either directly or by means of a translator. They must, however, use the same concepts, albeit using different naming, so they actually share a common ontology.
- \* Information retrieval must be assessed for quality so that the outputs from AI reasoning, and thus management solutions, can be reliable.
- \* Ontological concepts must be consistent so that the types and qualities of information that is retrieved from a system or object are as expected.
- \* The protocols used to communicate (or disseminate, or publish) the information must respond to the constraints of their target usage.

#### 5. Relation to Other IETF/IRTF Initiatives

TBD

#### 6. IANA Considerations

This memo includes no request to IANA.

## 7. Security Considerations

As with other AI mechanisms, the major security concern for the adoption of intelligent reasoning on external events to manage network slices and SDN/NFV systems is that the boundaries of the control and management planes are crossed to introduce information from outside. Such communications must be highly and heavily secured since some malfunction or explicit attacks might compromise the integrity and execution of the controlled system. However, it is up to implementers to deploy the necessary countermeasures to avoid such situations. From the design point of view, since all operations are performed within the control and/or management planes, the security level of reasoning solutions is inherited and thus determined by the security measures established by the systems conforming such planes.

## 8. Acknowledgements

TBD

## 9. References

### 9.1. 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, <<https://www.rfc-editor.org/info/rfc2119>>.

### 9.2. Informative References

[ETSI-NFV-IFA-004]  
ETSI NFV GS NFV-IFA 004, "Network Functions Virtualisation (NFV); Acceleration Technologies; Management Aspects Specification", 2016.

[ETSI-NFV-IFA-005]  
ETSI NFV GS NFV-IFA 005, "Network Functions Virtualisation (NFV); Management and Orchestration; Or-Vi reference point - Interface and Information Model Specification", 2016.

[ETSI-NFV-IFA-006]  
ETSI NFV GS NFV-IFA 006, "Network Functions Virtualisation (NFV); Management and Orchestration; Vi-Vnfm reference point - Interface and Information Model Specification", 2016.

[ETSI-NFV-IFA-019]

ETSI NFV GS NFV-IFA 019, "Network Functions Virtualisation (NFV); Acceleration Technologies; Management Aspects Specification; Release 3", 2017.

[ETSI-NFV-MANO]

ETSI NFV GS NFV-MAN 001, "Network Functions Virtualisation (NFV); Management and Orchestration", 2014.

[I-D.geng-coms-architecture]

Geng, L., Qiang, L., Ordonez, J., Ameigeiras, P., Lopez, D., and L. M. Contreras, "COMS Architecture", Work in Progress, Internet-Draft, draft-geng-coms-architecture-02, 5 March 2018, <<https://www.ietf.org/archive/id/draft-geng-coms-architecture-02.txt>>.

[I-D.homma-rtgwg-slice-gateway]

Homma, S., Nakamura, T., Foy, X. D., Galis, A., Contreras, L. M., Rokui, R., and P. Martinez-Julia, "Gateway Function for Network Slicing", Work in Progress, Internet-Draft, draft-homma-rtgwg-slice-gateway-02, 8 March 2020, <<https://www.ietf.org/archive/id/draft-homma-rtgwg-slice-gateway-02.txt>>.

[I-D.ietf-opsawg-ntf]

Song, H., Qin, F., Martinez-Julia, P., Ciavaglia, L., and A. Wang, "Network Telemetry Framework", Work in Progress, Internet-Draft, draft-ietf-opsawg-ntf-09, 13 October 2021, <<https://www.ietf.org/archive/id/draft-ietf-opsawg-ntf-09.txt>>.

[I-D.qiang-coms-netslicing-information-model]

Qiang, L., Galis, A., Geng, L., Makhijani, K., Martinez-Julia, P., Flinck, H., and X. D. Foy, "Technology Independent Information Model for Network Slicing", Work in Progress, Internet-Draft, draft-qiang-coms-netslicing-information-model-02, 25 January 2018, <<https://www.ietf.org/archive/id/draft-qiang-coms-netslicing-information-model-02.txt>>.

[ICIN-2017]

P. Martinez-Julia, V. P. Kafle, and H. Harai, "Achieving the autonomic adaptation of resources in virtualized network environments, in Proceedings of the 20th ICIN Conference (Innovations in Clouds, Internet and Networks, ICIN 2017). Washington, DC, USA: IEEE, 2018, pp. 1--8", 2017.

[ICIN-2018]

P. Martinez-Julia, V. P. Kafle, and H. Harai,  
"Anticipating minimum resources needed to avoid service  
disruption of emergency support systems, in Proceedings of  
the 21th ICIN Conference (Innovations in Clouds, Internet  
and Networks, ICIN 2018). Washington, DC, USA: IEEE, 2018,  
pp. 1--8", 2018.

[OPENSTACK]

The OpenStack Project, "<http://www.openstack.org/>", 2018.

## Appendix A. AINEMA Information Model

In this section we introduce the basic information model needed by AINEMA to support reasoning on external events. It basically includes the concepts and structures used to describe external events and notify (communicate) them to the interested sink, the network controller/manager, through the control and management plane, depending on the specific instantiation of the system.

### A.1. Tree Structure

```
module: ietf-nmrg-nict-ainema
  +--rw events
    +--rw event-payloads
    +--rw external-events

  notifications:
    +---n event
```

The main models included in the tree structure of the module are the events and notifications. On the one hand, events are structured in payloads and the content of events itself (external-events). On the other hand, there is only one notification, which is the event itself.

#### A.1.1. event-payloads

```
+--rw event-payloads
  +--rw event-payloads-basic
  +--rw event-payloads-seismometer
  +--rw event-payloads-bigdata
```

The event payloads are, for the time being, composed of three types. First, we have defined the basic payload, which is intended to carry any arbitrary data. Second, we have defined the seismometer payload to carry information about seisms. Third, we have defined the bigdata payload that carries notifications coming from BigData sources.

#### A.1.1.1. basic

```
+--rw event-payloads-basic* [plid]
+--rw plid      string
+--rw data?     union
```

The basic payload is able to hold any data type, so it has a union of several types. It is intended to be used by any source of events that is (still) not covered by other model. In general, any source of telemetry information (e.g. OpenStack [OPENSTACK] controllers) can use this model as such sources can encode on it their information, which typically is very simple and plain. Therefore, the current model is tightly interrelated to a framework to retrieve network telemetry (see [I-D.ietf-opsawg-ntf]).

#### A.1.1.2. seismometer

```
+--rw event-payloads-seismometer* [plid]
+--rw plid      string
+--rw location?  string
+--rw magnitude? uint8
```

The seismometer model includes the main information related to a seism, such as the location of the incident and its magnitude. Additional fields can be defined in the future by extending this model.

#### A.1.1.3. bigdata

```
+--rw event-payloads-bigdata* [plid]
+--rw plid      string
+--rw description? string
+--rw severity?  uint8
```

The bigdata model includes a description of an event (or incident) and its estimated general severity, unrelated to the system. The description is an arbitrary string of characters that would normally carry information that describes the event using some higher level format, such as Turtle or N3 for carrying RDF knowlege items.

## A.1.2. external-events

```
+---rw external-events* [id]
+---rw id                string
+---rw source?           string
+---rw context?          string
+---rw sequence?         int64
+---rw timestamp?        yang:date-and-time
+---rw payload?          binary
```

The model defined to encode external events, which encapsulates the payloads introduced above, is completed with an identifier of the message, a string describing the source of the event, a sequence number and a timestamp. Additionally it includes a string describing the context of the event. It is intended to communicate the required information about the system that detected the event, its location, etc. As the description of the BigData payload, this field can be formatted with a high level format, such as RDF.

## A.1.3. notifications/event

```
notifications:
+---n event
+---ro id?                string
+---ro source?            string
+---ro context?           string
+---ro sequence?          int64
+---ro timestamp?         yang:date-and-time
+---ro payload?           binary
```

The event notification inherits all the fields from the model of external events defined above. It is intended to allow software and hardware elements to send, receive, and interpret not just the events that have been detected and notified by, for instance, a sensor, but also the notifications issued by the underlying infrastructure controllers, such as the OpenStack Controller.

## A.2. YANG Module

```
.

module ietf-nmrg-nict-ainema {
  namespace "urn:ietf:params:xml:ns:yang:ietf-nmrg-nict-ainema";
  prefix rant;
  import ietf-yang-types { prefix yang; }

  grouping external-event-information {
    leaf id { type string; }
```

```
    leaf source { type string; }
    leaf context { type string; }
    leaf sequence { type int64; }
    leaf timestamp { type yang:date-and-time; }
    leaf payload { type binary; }
  }

  grouping event-payload-basic {
    leaf plid { type string; }
    leaf data { type union { type string; type binary; } }
  }

  grouping event-payload-seismometer {
    leaf plid { type string; }
    leaf location { type string; }
    leaf magnitude { type uint8; }
  }

  grouping event-payload-bigdata {
    leaf plid { type string; }
    leaf description { type string; }
    leaf severity { type uint8; }
  }

  notification event {
    uses external-event-information;
  }

  container events {
    container event-payloads {
      list event-payloads-basic {
        key "plid";
        uses event-payload-basic;
      }
      list event-payloads-seismometer {
        key "plid";
        uses event-payload-seismometer;
      }
      list event-payloads-bigdata {
        key "plid";
        uses event-payload-bigdata;
      }
    }
    list external-events {
      key "id";
      uses external-event-information;
    }
  }
}
```

}

.

## Appendix B. The Autonomic Resource Control Architecture (ARCA)

As deeply discussed in ICIN 2018 [ICIN-2018], ARCA leverages the elastic adaptation of resources assigned to virtual computer and network systems by calculating or estimating their requirements from the analysis of load measurements and the detection of external events. These events can be notified by physical elements (things, sensors) that detect changes on the environment, as well as software elements that analyze digital information, such as connectors to sources or analyzers of Big Data. For instance, ARCA is able to consider the detection of an earthquake or a heavy rainfall to overcome the damages it can make to the controlled system.

The policies that ARCA must enforce will be specified by administrators during the configuration of the control/management engine. Then, ARCA continues running autonomously, with no more human involvement unless some parameter must be changed. ARCA will adopt the required control and management operations to adapt the controlled system to the new situation or requirements. The main goal of ARCA is thus to reduce the time required for resource adaptation from hours/minutes to seconds/milliseconds. With the aforementioned statements, system administrators are able to specify the general operational boundaries in terms of lower and upper system load thresholds, as well as the minimum and maximum amount of resources that can be allocated to the controlled system to overcome any eventual situation, including the natural crossing of such thresholds.

ARCA functional goal is to run autonomously while the performance goal is to keep the resources assigned to the controlled resources as close as possible to the optimum (e.g. 5 % from the optimum) while avoiding service disruption as much as possible, keeping client request discard rate as low as possible (e.g. below 1 %). To achieve both goals, ARCA relies on the Autonomic Computing (AC) paradigm, in the form of interconnected micro-services. Therefore, ARCA includes the four main elements and activities defined by AC, incarnated as:

Collector Is responsible of gathering and formatting the heterogeneous observations that will be used in the control cycle.

Analyzer Correlates the observations to each other in order to find



the situation of the controlled system, especially the current load of the resources allocated to the system and the occurrence of an incident that can affect to the normal operation of the system, such as an earthquake that increases the traffic in an emergency-support system, which is the main target scenario studied in this paper.

**Decider** Determines the necessary actions to adjust the resources to the load of the controlled system.

**Enforcer** Requests the underlying and overlying infrastructure, such as OpenStack, to make the necessary changes to reflect the effects of the decided actions into the system.

Being a micro-service architecture means that the different components are executed in parallel. This allows such components to operate in two ways. First, their operation can be dispatched by receiving a message from the previous service or an external service. Second, the services can be self-dispatched, so they can activate some action or send some message without being previously stimulated by any message. The overall control process loops indefinitely and it is closed by checking that the expected effects of an action are actually taking place. The coherence among the distributed services involved in the ARCA control process is ensured by enforcing a common semantic representation and ontology to the messages they exchange.

ARCA semantics are built with the Resource Description Framework (RDF) and the Web Ontology Language (OWL), which are well known and widely used standards for the semantic representation and management of knowledge. They provide the ability to represent new concepts without requiring to change the software, just plugin extensions to the ontology. ARCA stores all its knowledge in the Knowledge Base (KB), which is queried and kept up-to-date by the analyzer and decider micro-services. It is implemented by Apache Jena Fuseki, which is a high-performance RDF data store that supports SPARQL through an HTTP/REST interface. Being de-facto standards, both technologies enable ARCA to be easily integrated to virtualization platforms like OpenStack.

## Appendix C. ARCA Integration With ETSI-NFV-MANO

In this section we describe how to fit ARCA on a general SDN/NFV underlying infrastructure and introduce a showcase experiment that demonstrates its operation on an OpenStack-based experimentation platform. We first describe the integration of ARCA with the NFV-MANO reference architecture. We contextualize the significance of this integration by describing an emergency support scenario that clearly benefits from it. Then we proceed to detail the elements

forming the OpenStack platform and finally we discuss some initial results obtained from them.

### C.1. Functional Integration

The most important functional blocks of the NFV reference architecture promoted by ETSI (see ETSI-NFV-MANO [ETSI-NFV-MANO]) are the system support functions for operations and business (OSS/BSS), the element management (EM) and, obviously, the Virtual Network Functions (VNFs). But these functions cannot exist without being instantiated on a specific infrastructure, the NFV infrastructure (NFVI), and all of them must be coordinated, orchestrated, and managed by the general NFV-MANO functions.

Both the NFVI and the NFV-MANO elements are subdivided into several sub-components. The NFVI has the underlying physical computing, storage, and network resources, which are sliced (see[I-D.qiang-coms-netslicing-information-model] and [I-D.geng-coms-architecture]) and virtualized to conform the virtual computing, storage, and network resources that will host the VNFs. In addition, the NFV-MANO is subdivided in the NFV Orchestrator (NFVO), the VNF manager (VNFM) and the Virtual Infrastructure Manager (VIM). As their name indicates, all high-level elements and sub-components have their own and very specific objective in the NFV architecture.

During the design of ARCA we enforced both operational and interfacing aspects to its main objectives. From the operational point of view, ARCA processes observations to manage virtual resources, so it plays the role of the VIM mentioned above. Therefore, ARCA has been designed with appropriate interfaces to fit in the place of the VIM. This way, ARCA provides the NFV reference architecture with the ability to react to external events to adapt virtual computer and network systems, even anticipating such adaptations as performed by ARCA itself. However, some interfaces must be extended to fully enable ARCA to perform its work within the NFV architecture.

Once ARCA is placed in the position of the VIM, it enhances the general NFV architecture with its autonomic management capabilities. In particular, it discharges some responsibilities from the VNFM and NFVO, so they can focus on their own business while the virtual resources are behaving as they expect (and request). Moreover, ARCA improves the scalability and reliability of the managed system in case of disconnection from the orchestration layer due to some failure, network split, etc. It is also achieved by the autonomic capabilities, which, as described above, are guided by the rules and policies specified by the administrators and, here, communicated to

ARCA through the NFVO. However, ARCA will not be limited to such operation so, more generally, it will accomplish the requirements established by the Virtual Network Operators (VNOs), which are the owners of the slice of virtual resources that is managed by a particular instance of NFV-MANO, and therefore ARCA.

In addition to the operational functions, ARCA incorporates the necessary mechanisms to engage the interfaces that enable it to interact with other elements of the NFV-MANO reference architecture. More specifically, ARCA is bound to the Or-Vi (see ETSI-NFV-IFA-005 [ETSI-NFV-IFA-005]) and the Nf-Vi (see ETSI-NFV-IFA-004 [ETSI-NFV-IFA-004] and ETSI-NFV-IFA-019 [ETSI-NFV-IFA-019]). The former is the point of attachment between the NFVO and the VIM while the latter is the point of attachment between the NFVI and the VIM. In our current design we decided to avoid the support for the point of attachment between the VNFM and the VIM, called Vi-Vnfm (see ETSI-NFV-IFA-006 [ETSI-NFV-IFA-006]). We leave it for future evolutions of the proposed integration, that will be enabled by a possible solution that provides the functions of the VNFM required by ARCA.

Through the Or-Vi, ARCA receives the instructions it will enforce to the virtual computer and network system it is controlling. As mentioned above, these are specified in the form of rules and policies, which are in turn formatted as several statements and embedded into the Or-Vi messages. In general, these will be high-level objectives, so ARCA will use its reasoning capabilities to translate them into more specific, low-level objectives. For instance, the Or-Vi can specify some high-level statement to avoid CPU overloading and ARCA will use its innate and acquired knowledge to translate it to specific statements that specify which parameters it has to measure (CPU load from assigned servers) and which are their desired boundaries, in the form of high threshold and low threshold. Moreover, the Or-Vi will be used by the NFVO to specify which actions can be used by ARCA to overcome the violation of the mentioned policies.

All information flowing the Or-Vi interface is encoded and formatted by following a simple but highly extensible ontology and exploiting the aforementioned semantic formats. This ensures that the interconnected system is able to evolve, including the replacement of components, updating (addition or removal) the supported concepts to understand new scenarios, and connecting external tools to further enhance the management process. The only requirement to ensure this feature is to ensure that all elements support the mentioned ontology and semantic formats. Although it is not a finished task, the development of semantic technologies allows the easy adaptation and translation of existing information formats, so it is expected that more and more software pieces become easily integrable with the ETSI-NFV-MANO [ETSI-NFV-MANO] architecture.

In contrast to the Or-Vi interface, the Nf-Vi interface exposes more precise and low-level operations. Although this makes it easier to be integrated to ARCA, it also makes it to be tied to specific implementations. In other words, building a proxy that enforces the aforementioned ontology to different interface instances to homogenize them adds undesirable complexity. Therefore, new components have been specifically developed for ARCA to be able to interact with different NFVIs. Nevertheless, this specialization is limited to the collector and enforcer. Moreover, it allows ARCA to have optimized low-level operations, with high improvement of the overall performance. This is the case of the specific implementations of the collector and enforcer used with Mininet and Docker, which are used as underlying infrastructures in previous experiments described in ICIN 2017 [ICIN-2017]. Moreover, as discussed in the following section, this is also the case of the implementations of the collector and enforcer tied to OpenStack telemetry and compute interfaces, respectively. Hence it is important to ensure that telemetry is properly addressed, so we insist in the need to adopt a common framework in such endpoint (see [I-D.ietf-opsawg-ntf]).

Although OpenStack still lacks some functionality regarding the construction of specific virtual networks, we use it as the NFVI functional block in the integrated approach. Therefore, OpenStack is the provider of the underlying SDN/NFV infrastructure and we exploited its APIs and SDK to achieve the integration. More specifically, in our showcase we use the APIs provided by Ceilometer, Gnocchi, and Compute services as well as the SDK provided for Python. All of them are gathered within the Nf-Vi interface. Moreover, we have extended the Or-Vi interface to connect external elements, such as the physical or environmental event detectors and Big Data connectors, which is becoming a mandatory requirement of the current virtualization ecosystem and it conforms our main extension to the NFV architecture.

## C.2. Target Experiment and Scenario

From the beginning of our work on the design of ARCA we are targeting real-world scenarios, so we get better suited requirements. In particular we work with a scenario that represents an emergency support service that is hosted on a virtual computer and network system, which is in turn hosted on the distributed virtualization infrastructure of a medium-sized organization. The objective is to clearly represent an application that requires high dynamicity and high degree of reliability. The emergency support service accomplishes this by being barely used when there is no incident but also being heavily loaded when there is an incident.

Both the underlying infrastructure and virtual network share the same topology. They have four independent but interconnected network domains that form part of the same administrative domain (organization). The first domain hosts the systems of the headquarters (HQ) of the owner organization, so the VNFs it hosts (servants) implement the emergency support service. We defined them as ``servants'' because they are Virtual Machine (VM) instances that work together to provide a single service by means of backing the Load Balancer (LB) instances deployed in the separate domains. The amount of resources (servants) assigned to the service will be adjusted by ARCA, attaching or detaching servants to meet the load boundaries specified by administrators.

The other domains represent different buildings of the organization and will host the clients that access to the service when an incident occurs. They also host the necessary LB instances, which are also VNFs that are controlled by ARCA to regulate the access of clients to servants. All domains will have physical detectors to provide external information that can (and will) be correlated to the load of the controlled virtual computer and network system and thus will affect to the amount of servants assigned to it. Although the underlying infrastructure, the servants, and the ARCA instance are the same as those those used in the real world, both clients and detectors will be emulated. Anyway, this does not reduce the transferability of the results obtained from our experiments as it allows to expand the amount of clients beyond the limits of most physical infrastructures.

Each underlying OpenStack domain will be able to host a maximum of 100 clients, as they will be deployed on a low profile virtual machine (flavor in OpenStack). In general, clients will be performing requests at a rate of one request every ten seconds, so there would be a maximum of 30 requests per second. However, under the simulated incident, the clients will raise their load to reach a common maximum of 1200 requests per second. This mimics the shape

and size of a real medium-size organization of about 300 users that perform a maximum of four requests per second when they need some support.

The topology of the underlying network is simplified by connecting the four domains to the same, high-performance switch. However, the topology of the virtual network is built by using direct links between the HQ domain and the other three domains. These are complemented by links between domains 2 and 3, and between domains 3 and 4. This way, the three domains have three paths to reach the HQ domain: a direct path with just one hop, and two indirect paths with two and three hops, respectively.

During the execution of the experiment, the detectors notify the incident to the controller as soon as it happens. However, although the clients are stimulated at the same time, there is some delay between the occurrence of the incident and the moment the network service receives the increase in the load. One of the main targets of our experiment is to study such delay and take advantage of it to anticipate the amount of servants required by the system. We discuss it below.

In summary, this scenario highlights the main benefits of ARCA to play the role of VIM and interacting with the underlying OpenStack platform. This means the advancement towards an efficient use of resources and thus reducing the CAPEX of the system. Moreover, as the operation of the system is autonomic, the involvement of human administrators is reduced and, therefore, the OPEX is also reduced.

### C.3. OpenStack Platform

The implementation of the scenario described above reflects the requirements of any edge/branch networking infrastructure, which are composed of several distributed micro-data-centers deployed on the wiring centers of the buildings and/or storeys. We chose to use OpenStack to meet such requirements because it is being widely used in production infrastructures and the resulting infrastructure will have the necessary robustness to accomplish our objectives, at the time it reflects the typical underlying platform found in any SDN/NFV environment.

We have deployed four separate network domains, each one with its own OpenStack instantiation. All domains are totally capable of running regular OpenStack workload, i.e. executing VMs and networks, but, as mentioned above, we designate the domain 1 to be the headquarters of the organization. The different underlying networks required by this (quite complex) deployment are provided by several VLANs within a high-end L2 switch. This switch represents the distributed network

of the organization. Four separated VLANs are used to isolate the traffic within each domain, by connecting an interface of OpenStack's controller and compute nodes. These VLANs therefore form the distributed data plane. Moreover, other VLAN is used to carry the control plane as well as the management plane, which are used by the NFV-MANO, and thus ARCA. It is instantiated in the physical machine called ARCA Node, to exchange control and management operations in relation to the collector and enforcer defined in ARCA. This VLAN is shared among all OpenStack domains to implement the global control of the virtualization environment pertaining to the organization. Finally, other VLAN is used by the infrastructure to interconnect the data planes of the separated domains and also to allow all elements of the infrastructure to access the Internet to perform software installation and updates.

Installation of OpenStack is provided by the Red Hat OpenStack Platform, which is tightly dependent on the Linux operating system and closely related to the software developed by the OpenStack Open Source project. It provides a comprehensive way to install the whole platform while being easily customized to meet our specific requirements, while it is also backed by operational quality support.

The ARCA node is also based on Linux but, since it is not directly related to the OpenStack deployment, it is not based on the same distribution. It is just configured to be able to access the control and management interfaces offered by OpenStack, and therefore it is connected to the VLAN that hosts the control and management planes. On this node we deploy the NFV-MANO components, including the micro-services that form an ARCA instance.

In summary, we dedicate nine physical computers to the OpenStack deployment, all are Dell PowerEdge R610 with 2 x Xeon 5670 2.96 GHz (6 core / 12 thread) CPU, 48 GiB RAM, 6 x 146 GiB HD at 10 kRPM, and 4 x 1 GE NIC. Moreover, we dedicate an additional computer with the same specification to the ARCA Node. We dedicate a less powerful computer to implement the physical router because it will not be involved in the general execution of OpenStack nor in the specific experiments carried out with it. Finally, as detailed above, we dedicate a high-end physical switch, an HP ProCurve 1810G-24, to build the interconnection networks.

#### C.4. Initial Results

Using the platform described above we execute an initial but long-lasting experiment based on the target scenario introduced at the beginning of this section. The objective of this experiment is twofold. First, we aim to demonstrate how ARCA behaves in a real environment. Second, we aim to stress the coupling points between ARCA and OpenStack, which will raise the limitations of the existing interfaces.

With such objectives in mind, we define a timeline that will be followed by both clients and external event detectors. It forces the virtualized system to experience different situations, including incidents of many severities. When an incident is found in the timeline, the detectors notify it to the ARCA-based VIM and the clients change their request rates, which will depend on the severity of the incident. This behavior is widely discussed in ICIN 2018 [ICIN-2018], remarking how users behave after occurring a disaster or another similar incident.

The ARCA-based VIM will know the occurrence of the incident from two sources. First, it will receive the notification from the event detectors. Second, it will notice the change of the CPU load of the servants assigned to the target service. In this situation, ARCA has different opportunities to overcome the possible overload (or underload) of the system. We explore the anticipation approach deeply discussed in ICIN 2018 [ICIN-2018]. Its operation is enclosed in the analyzer and decider and it is based on an algorithm that is divided in two sub-algorithms.

The first sub-algorithm reacts to the detection of the incident and ulterior correlation of its severity to the amount of servants required by the system. This sub-algorithm hosts the regression of the learner, which is based on the SVM/SVR technique, and predicts the necessary resources from two features: the severity of the incident and the time elapsed from the moment it happened. The resulting amount of servants is established as the minimum amount that the VIM can use.

The second sub-algorithm is fed with the CPU load measurements of the servants assigned to the service, as reported by the OpenStack platform. With this information it checks whether the system is within the operating parameters established by the NFVO. If not, it adjusts the resources assigned to the system. It also uses the minimum amount established by the other sub-algorithm as the basis for the assignation. After every correction, this algorithm learns the behavior by adding new correlation vectors to the SVM/SVR structure.



When the experiment is running, the collector component of the ARCA-based VIM is attached to the telemetry interface of OpenStack by using the SDK to access the measurement data generated by Ceilometer and stored by Gnocchi. In addition, it is attached to the external event detectors in order to receive their notifications. On the other hand, the enforcer component is attached to the Compute interface of OpenStack by also using its SDK to request the infrastructure to create, destroy, query, or change the status of a VM that hosts a servant of the controlled system. Finally, the enforcer also updates the lists of servers used by the load balancers to distribute the clients among the available resources.

During the execution of the experiment we make the ARCA-based VIM to report the severity of the last incident, if any, the time elapsed since it occurred, the amount of servants assigned to the controlled system, the minimum amount of servants to be assigned, as determined by the anticipation algorithm, and the average load of all servants. In this instance, the severities are spread between 0 (no incident) and 4 (strongest incident), the elapsed times are less than 35 seconds, and the minimum server assignment (MSA) is below 10, although the hard maximum is 15.

With such measurements we illustrate how the learned correlation of the three features (dimensions) mentioned above is achieved. Thus, when there is no incident (severity = 0), the MSA is kept to the minimum. In parallel, regardless of the severity level, the algorithm learned that there is no need to increase the MSA for the first 5 or 10 seconds. This shows the behavior discussed in this paper, that there is a delay between the occurrence of an event and the actual need for updated amount of resources, and it forms one fundamental aspect of our research.

By inspecting the results, we know that there is a burst of client demands that is centered (peak) around 15 seconds after the occurrence of an incident or any other change in the accounted severity. We also know that the burst lasts longer for higher severities, and it fluctuates a bit for the highest severities. Finally, we can also notice that for the majority of severities, the increased MSA is no longer required after 25 seconds from the time the severity change was notified.

All that information becomes part of the knowledge of ARCA and it is stored both by the internal structures of the SVM/SVR and, once represented semantically, in the semantic database that manages the knowledge base of ARCA. Thus, it is used to predict any future behavior. For instance, if an incident of severity 3 has occurred 10 seconds ago, ARCA knows that it will need to set the MSA to 6 servants. In fact, this information has been used during the

experiment, so we can also know the accuracy of the algorithm by comparing the anticipated MSA value with the required value (or even the best value). However, the analysis of such information is left for the future.

While preparing and executing the experiment we found several limitation intrinsic to the current OpenStack platform. First, regardless of the CPU and memory resources assigned to the underlying controller nodes, the platform is unable to record and deliver performance measurements at a lower interval than every 10 seconds, so it is currently not suitable for real time operations, which is important for our long-term research objectives. Moreover, we found that the time required by the infrastructure to create a server that hosts a somewhat heavy servant is around 10 seconds, which is too far from our targets. Although these limitations can be improved in the future, they clearly justify that our anticipation approach is essential for the proper working of a virtual system and, thus, the integration of external information becomes mandatory for future system management technologies, especially considering the virtualization environments.

Finally, we found it difficult for the required measurements to be pushed to external components, so we had to poll for them. Otherwise, some component of ARCA must be instantiated along the main OpenStack components and services so it has first-hand and prompt access to such features. This way, ARCA could receive push notifications with the measurements, as it is for the external detectors. This is a key aspect that affects the placement of the NFV-VIM, or some subpart of it, on the general architecture. Therefore, for future iterations of the NFV reference architecture, an integrated view between the VIM and the NFVI could be required to reflect the future reality.

#### Authors' Addresses

Pedro Martinez-Julia (editor)  
NICT  
4-2-1, Nukui-Kitamachi, Koganei, Tokyo  
184-8795  
Japan

Phone: +81 42 327 7293  
Email: pedro@nict.go.jp

Shunsuke Homma  
NTT  
Japan

Email: shunsuke.homma.fp@hco.ntt.co.jp

Diego R. Lopez  
Telefonica I+D  
Don Ramon de la Cruz, 82  
28006 Madrid  
Spain

Email: diego.r.lopez@telefonica.com

Internet Research Task Force  
Internet-Draft  
Intended status: Informational  
Expires: September 6, 2022

D. Chen  
H. Yang  
K. Yao  
China Mobile  
G. Fioccola  
Huawei Technologies  
March 05, 2022

Network measurement intent - one of IBN use cases  
draft-yang-nmrg-network-measurement-intent-04

Abstract

As an important technical means to detect network state, network measurement has attracted more and more attention in the development of network. However, the current network measurement technology has the problem that the measurement method and the measurement purpose cannot match well. To solve this problem, this memo introduces network measurement intent, namely the process of realizing user or network operator to allocate network states as needed. And it can be as a specified user case of intent based network.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 6, 2022.

## Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	2
2. Definitions and Acronyms . . . . .	3
3. Connections to Existing Documents . . . . .	3
4. Overview . . . . .	5
5. Concrete Examples . . . . .	7
5.1. SLA measurement intent . . . . .	8
5.2. Clustered performance measurement intent . . . . .	10
6. Classification of NMI . . . . .	11
6.1. Static NMI . . . . .	12
6.2. Dynamic NMI . . . . .	12
7. Summary . . . . .	12
8. Security Considerations . . . . .	12
9. IANA Considerations . . . . .	13
10. References . . . . .	13
10.1. Normative References . . . . .	13
10.2. Informative References . . . . .	13
Authors' Addresses . . . . .	13

## 1. Introduction

With the rapid development of the current network, the scale of the network is getting larger and larger, while users' requirements for the network are getting higher and higher. At the same time, network resources are increasingly restrained. In order to realize the efficient allocation of network resources, it is necessary to understand the running state of the network, and network measurement, as a technical means to detect the network, has been paid of more and more attention. The continuous development of network measurement technology has also satisfied the higher and higher precision of network perception. However, both the traditional network measurement technology and the network telemetry technology, which

has emerged with the development of software-defined network in recent years, need to occupy the network resources when detecting the network state and feeding back the detection results. Therefore, to some extent, the choice of network measurement methods, in addition to different accuracy of measurement results, will also cause different degrees of burden to the network.

In order to balance the accuracy of network measurement results with the network load, it is very important to choose the appropriate network measurement method according to the different requirements of network measurement. As a result, accurate on-demand network measurement technology is becoming more and more important. At the same time, the development of Intent based Network (IBN) enables the network to be configured according to users' or network administrators' intent. Therefore, we can combine network measurement with IBN, that is, the users' or network administrators' perceived demand for network state is regarded as network measurement intent.

We want to use the network measurement intent to achieve network performance acquisition based on user/network administrator intent-based, verify whether network measurement results meet the measurement intent, and further improve the accuracy of the configuration in IBN.

## 2. Definitions and Acronyms

CLI: Command-line Interface.

IBN: Intent based Network.

Policy: A set of rules that governs the choices in behavior of a system.

NMI: Network Measurement Intent, refers to based on user/network operator's demand for network status, and automatically collect network status information on demand.

SLA: Service Level Agreement.

## 3. Connections to Existing Documents

As the rise of IBN, different groups have different definitions of intent. For example, ONF [ONOS] defines intent is represented as a list of CLI modes that allows users to pass low-level details on the network; and there are two active RG drafts in the NMRG right now, Intent-Based Networking - Concepts and Definitions, [I-D.irtf-nmr-ibn-concepts-definitions] solves the problem that

"What is an intent?"

and[I-D.irtf-nmrg-ibn-intent-classification] solves the problem "Given a specific intent, how to parse/disassemble it from different angles?".

Naturally, the question that needs to be solved after concept definition should be "How to realize an specific intent?". The classification draft can be considered as the first step of realization of a given intent, however, it is not enough. Some other issues should be clarified, like "whether the input intent is valid or not?" , "What would the IBN system do when the result is not acceptable?", "If the result is not acceptable, does human/operator interference required?"... We should take a specific IBN use case for illustration of the realization procedure, so we will take the network measurement intent as an example.

Referring to the taxonomy of intent proposed in[I-D.irtf-nmrg-ibn-concepts-definitions], the network measurement intent can be classified into different subgroups.

Solution: the intent could cover carrier and data center.

Intent user type: customer.

Intent type: customer service intent.

Intent scope: Application, QoS.

Network scope: Radio Access, Transport, Edge, Core.

Abstraction: Non-technical.

Lifecycle Requirements: transient.

In order to combine the NMI with the existing drafts of IBN, in this document we define the components of the NMI processing process as follows:

- o NMI Recognition and Acquisition
- o NMI Translation
- o NMI Policy
- o NMI Orchestration and pre-Verification
- o Data Collection and Analytics

- o NMI Compliance Assessment

#### 4. Overview

As mentioned above, NMI refers to the on-demand measurement of the network state based on the user/network operators' perceived intent of the network state. We will present the detailed process of it within each part and take the measurement of busy network performances as a simple example.

- o NMI Recognition and Acquisition.

- \* In this function, NMI will be recognized by "ingesting" users' or network operators' measurement intent. They have the ability to identify the NMI of a certain network performance that users want to measure, such as delay, jitter, etc., and at the same time allow users to express the NMI of network performance in a variety of interactive ways to ensure the accuracy of the identification of the NMI. To achieve this functionality, such an interaction requires the use of the intent-northbound interface defined in the IBN.

- o NMI Translation.

- \* In this function, NMI needs to be translated into corresponding measurement policy, which includes but is not limited to network performance parameters to be measured (such as delay, jitter, and packet loss), time period to be measured, and measurement precision. For a simple example, in the measurement of busy network performances, due to dynamic changes such as daily network bandwidth occupancy rate, the period of network busy time is not fixed. As a result, NMI Policy generated by NMI Translation can determine the threshold when the network state is busy on the same day based on the historical data learned by AI.

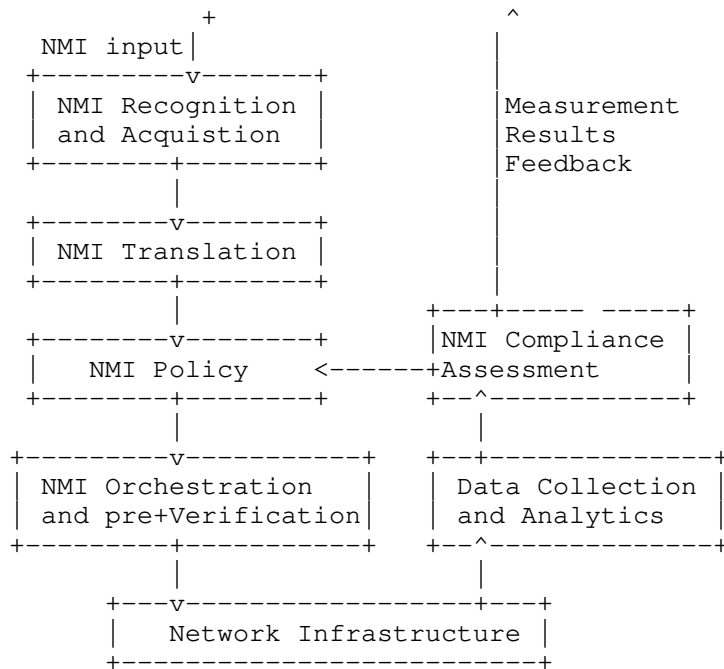
- o NMI Policy

- \* In this function, NMI policy needs to be translated into actions and requests taken against the specified network element. Therefore, NMI policy generated by NMI Translation must be executable, that is, corresponding underlying network devices must be able to support policy execution.. If the generated policy cannot be executed by the underlying device, the policy needs to be adjusted. And if the measurement results cannot meet the requirements, the policy also needs to be adjusted.



- o NMI Orchestration and pre-Verification.
  - \* In this function, according to the previous NMI Translation and NMI Policy step, NMI Orchestration and pre-Verification determines the measurement scheme according to the measurement policy generated by NMI Policy, and pre-verifies whether the measurement scheme is feasible.
  - \* Take busy time network measurement as an example, except for choosing of measurement schemes and contents, it also needs to determine whether the network is busy according to the current network state. In addition, this function performs automatic network deployment, such as in CLI mode.
- o Data Collection and Analytics.
  - \* In NMI, data collection and analysis should be based on the selected measurement scheme and the content to be measured that determined in previous steps, automatically realize the collection on demand, and generate corresponding data analysis results.
- o NMI Compliance Assessment.
  - \* At the end, this function verifies whether the results meets the requirement and whether the NMI is satisfied. If either of the two conditions is not satisfied, the NMI should be modified and re-enter the NMI Policy.

And the measurement flow diagram is shown as the following figure:



## 5. Concrete Examples

In this section, we will take SLA measurement intent as an example to illustrate each step of the process.

With the development of measurement technology in recent years, network measurement can be divided into active measurement, passive measurement and a combination of active and passive measurement. As mentioned above, no matter which measurement technology will occupy network resources. For example, if the transmission frequency of active measurement message is too fast, it will occupy too much bandwidth resources and affect the normal operation of actual business. While if the transmission frequency is too slow, some instantaneous network anomalies will be missed and the network status cannot be accurately reflected. Passive measurement requires real-time collection of actual business data. If the sampling rate is too high, a large amount of data will be accumulated in a short time. The analysis system for real-time analysis of these data needs strong processing capacity; if the sampling rate is too low, some network anomalies will also be omitted.

How to balance and accurately measure the network state, especially the abnormal network affecting the service, while occupying as little network bandwidth as possible, and the processing capacity of the

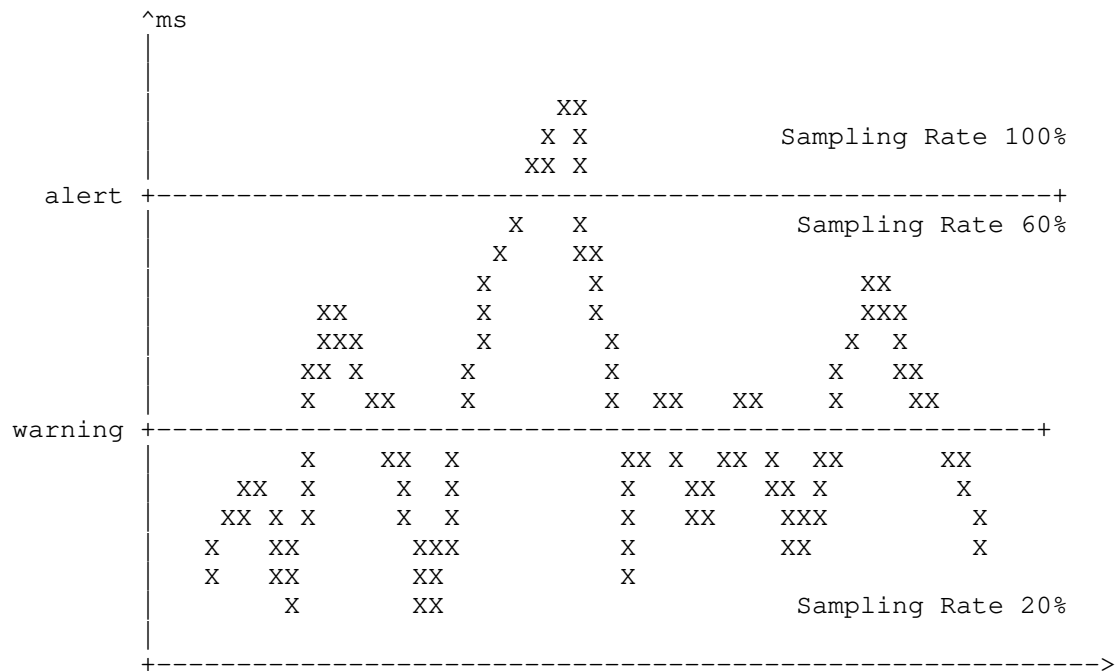
data analysis system is not high, this is the function that the NMI scheme based on IBN should realize.

In this section, we will consider two examples to illustrate each step of the process.

#### 5.1. SLA measurement intent

Taking network SLA performance index -- time delay measurement as an example, the simple schematic diagram is as follows, different thresholds, warning value and alert value should be set for network delay in advance. When the delay value is below warning, the network is normal and the business is normal. When the delay is between warning value and alert value, the network fluctuation is abnormal, but the business is normal. When the delay exceeds the alert value, both the network and business are abnormal. For delay in different thresholds, different measurement strategies should be adopted:

- o When the network delay exceeds the alert value, or when the historical data predict that the delay will exceed the alert value, passive measurement requires 100% sampling of business data, and the transmission frequency of active measurement is modulated to the maximum. At the same time, the log and alarm data of the whole network equipment are collected to realize the most fine-grained measurement of the network, locate the root cause of the problem and repair the network in time.
- o When the network delay exceeds warning value but is lower than alert value, passive measurement samples 60% of business data, and the transmission message frequency of the active measurement is adjusted to the median value, and the running state data of some key devices in the network is collected synchronously.
- o When the network delay is less than warning value, passive measurement data is sampled at 20%, and active measurement message frequency is adjusted to the lowest, and the network equipment running state of key nodes can be collected as needed.



Based on the above SLA time delay index measurement, different thresholds adopt different measurement strategies, the concrete steps of SLA measurement intent are as follows:

- o In NMI Recognition and Acquisition, SLA measurement intent is recognized, and business requirements and performance metrics are identified by interacting with users. Then the NMI Recognition and Acquisition module inputs the SLA measurement intent into the NMI Translation module.
- o The NMI Translation module combines the SLA measurement intent with the measurement policy in NMI Policy, and outputs the executable measurement policy, such as the message transmission frequency of active measurement, the sampling rate of passive measurement, the collection range of equipment running state, etc.
- o The NMI Orchestration and pre-Verification module arranges the measurement policy into the specific configuration and execution time of each device in the tested network. The NMI Orchestration and pre-Verification module verifies the implementation of the policy in the equipment and preanalyzes the measurement results.
- o The Data Collection and Analysis module will collect the measurement data according to the requirements of the previous

step, make a simple analysis of the collected data, and then send the collected measurement data to the NMI Compliance Assessment module. After that, it feedback the measurement results to the user to complete the closed loop of the measurement task.

- o According to the change of delay data in the measured data, the NMI Compliance Assessment module notifies the NMI Orchestration and pre-Verification module to modify the execution time of the policy in time, and at the same time updates the measured results to the delay history database to improve the accuracy of delay prediction. The NMI Compliance Assessment module evaluates whether the actual measurement results are in line with the user's intent. If they are, the results will be fed back. If they are not, the NMI Policy module will be informed to adjust the policy, and then the measurement will be restarted.

## 5.2. Clustered performance measurement intent

The desired approach is to accurately measure the network state, especially when there are some issues affecting the service, but at the same time, reduce the resources to be employed to achieve the desired accuracy.

In this regard, the Clustered Alternate-Marking framework [RFC8889] adds flexibility to Performance Management (PM), because it can reduce the order of magnitude of the packet counters. This allows the NMI Orchestration and pre-Verification module to supervise, control, and manage PM in large networks.

RFC 8889 [RFC8889] introduces the concept of cluster partition of a network. The monitoring network can be considered as a whole or split into clusters that are the smallest subnetworks (group-to-group segments), maintaining the packet loss property for each subnetwork. The clusters can be combined in new connected subnetworks at different levels, forming new clusters, depending on the level of detail to achieve.

The clustered performance measurement intent represents the spatial accuracy, that is the size of the subnetworks to consider for the monitoring. It is possible to start without examining in depth and, in case of necessity, the "network zooming" approach can be used.

This approach called "network zooming" and can be performed in two different ways:

1. change the traffic filter and select more detailed flows;

2. activate new measurement points by defining more specified clusters.

The network-zooming approach implies that some filters, rules or flow identifiers are changed. But these changes must be done in a way that do not affect the performance. Therefore there could be a transient time to wait once the new network configuration takes effect. Anyway, if the performance issue is relevant, it is likely to last for a time much longer than the transient time.

The concrete steps of the clustered performance measurement intent are as follows:

- o In NMI Recognition and Acquisition, the clustered performance measurement intent is recognized. Then the NMI Recognition and Acquisition module inputs the clustered performance measurement intent into the NMI Translation module.
- o The NMI Translation module analyzes the clustered performance measurement intent and outputs the executable measurement policy, such as network partition and the spatial accuracy for the monitoring.
- o The NMI Orchestration and pre-Verification module arranges and calibrates the measurement with the specific configuration to split the whole network into clusters at different levels.
- o The Data Collection and Analysis module collects the measurement data from the different clusters, and then send these data to the NMI Compliance Assessment module. It verifies the performance for each cluster and send the measurement results to the user.
- o The NMI Compliance Assessment module, in case a cluster is experiencing a packet loss or the delay is high, notifies the NMI Orchestration and pre-Verification module to modify the cluster partition of the network for further investigation. The network configuration can be immediately modified in order to perform a new partition of the network but only for the cluster with bad performance. In this way, the problem can be localized with successive approximation up to a flow detailed analysis. This is the so-called "closed loop" performance management.

## 6. Classification of NMI

In this section, we divide the network measurement intent into static NMI and dynamic NMI according to different requirement characteristics.

### 6.1. Static NMI

Static NMI refers to the measurement purposes remain unchanged and is independent of the network state/external environment. Static NMI can be translated into determined network performance indicator values, such as concrete delay values, network bandwidth occupancy, throughput and so on.

Because the static NMI can be translated into the measurement of the determined network performance parameters, the whole process is relatively simple and error-prone, and only needs to verify whether the measurement results meet the requirements.

### 6.2. Dynamic NMI

Dynamic NMI refers to the measurement purpose remains unchanged but the measurement process changes dynamically according to the network state/external environment. Dynamic NMI can also be translated into the measurement of determined network performance parameters, however, the values of network performance parameters will change with the changes of network states and external environment.

For example, the measurement of busy network performances mentioned in the previous. Although the corresponding network parameters for judging whether the network is busy are determined, the corresponding network parameters have different values according to different network states and external environments.

Due to the dynamic nature of dynamic NMI, its processing process is more complex than static NMI. It is not only necessary to verify the accuracy of demand analysis, but also to verify whether the final measurement results meet the requirements.

## 7. Summary

This memo introduces the network measurement intent, and give two concrete examples to illustrate the process of network measurement intent. On the basis of existing intent drafts, this memo can be used as a use case for IBN. NMI is a big and typical use case of IBN, and the classification of different examples of NMI may vary.

## 8. Security Considerations

TBD.

## 9. IANA Considerations

This document has no requests to IANA.

## 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, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8889] Fioccola, G., Ed., Cociglio, M., Sapio, A., and R. Sisto, "Multipoint Alternate-Marking Method for Passive and Hybrid Performance Monitoring", RFC 8889, DOI 10.17487/RFC8889, August 2020, <<https://www.rfc-editor.org/info/rfc8889>>.

### 10.2. Informative References

- [I-D.irtf-nmrg-ibn-concepts-definitions]  
Clemm, A., Ciavaglia, L., Granville, L. Z., and J. Tantsura, "Intent-Based Networking - Concepts and Definitions", draft-irtf-nmrg-ibn-concepts-definitions-06 (work in progress), December 2021.
- [I-D.irtf-nmrg-ibn-intent-classification]  
Li, C., Havel, O., Olariu, A., Martinez-Julia, P., Nobre, J. C., and D. R. Lopez, "Intent Classification", draft-irtf-nmrg-ibn-intent-classification-06 (work in progress), February 2022.

## Authors' Addresses

Danyang Chen  
China Mobile  
Beijing 100053  
China

Email: [chendanyang@chinamobile.com](mailto:chendanyang@chinamobile.com)



Hongwei Yang  
China Mobile  
Beijing 100053  
China

Email: yanghongwei@chinamobile.com

Kehan Yao  
China Mobile  
Beijing 100053  
China

Email: yaokehan@chinamobile.com

Giuseppe Fioccola  
Huawei Technologies  
Riesstrasse, 25  
Munich 80992  
Germany

Email: giuseppe.fioccola@huawei.com

Internet Research Task Force  
Internet-Draft  
Intended status: Informational  
Expires: 6 September 2022

C. Zhou  
H. Yang  
X. Duan  
China Mobile  
D. Lopez  
A. Pastor  
Telefonica I+D  
Q. Wu  
Huawei  
M. Boucadair  
C. Jacquenet  
Orange  
5 March 2022

Digital Twin Network: Concepts and Reference Architecture  
draft-zhou-nmrg-digitaltwin-network-concepts-07

Abstract

Digital Twin technology has been seen as a rapid adoption technology in Industry 4.0. The application of Digital Twin technology in the networking field is meant to develop various rich network applications and realize efficient and cost effective data driven network management and accelerate network innovation.

This document presents an overview of the concepts of Digital Twin Network, provides the basic definitions and a reference architecture, lists a set of application scenarios, and discusses the benefits and key challenges of such technology.

Status of This Memo

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

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

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

This Internet-Draft will expire on 6 September 2022.

## Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	3
2.1. Acronyms & Abbreviations . . . . .	3
2.2. Definitions . . . . .	4
3. Introduction and Concepts of Digital Twin Network . . . . .	4
3.1. Background of Digital Twin . . . . .	4
3.2. Digital Twin for Networks . . . . .	5
3.3. Definition of Digital Twin Network . . . . .	6
4. Benefits of Digital Twin Network . . . . .	9
4.1. Optimized Network Total Cost of Operation . . . . .	10
4.2. Optimized Decision Making . . . . .	10
4.3. Safer Assessment of Innovative Network Capabilities . . . . .	10
4.4. Privacy and Regulatory Compliance . . . . .	11
4.5. Customized Network Operation Training . . . . .	11
5. Challenges to Build Digital Twin Network . . . . .	11
6. A Reference Architecture of Digital Twin Network . . . . .	13
7. Interaction with IBN . . . . .	16
8. Sample Application Scenarios . . . . .	17
8.1. Human Training . . . . .	17
8.2. Machine Learning Training . . . . .	17
8.3. DevOps-Oriented Certification . . . . .	18
8.4. Network Fuzzing . . . . .	18
9. Research Perspectives: A Summary . . . . .	18
10. Security Considerations . . . . .	18
11. Acknowledgements . . . . .	19
12. IANA Considerations . . . . .	19
13. Open issues . . . . .	19
14. Informative References . . . . .	20
Appendix A. Change Logs . . . . .	22
Authors' Addresses . . . . .	23

## 1. Introduction

The fast growth of network scale and the increased demand placed on these networks require them to accommodate and adapt dynamically to customer needs, implying a significant challenge to network operators. Indeed, network operation and maintenance are becoming more complex due to higher complexity of the managed networks and the sophisticated services they are delivering. As such, providing innovations on network technologies, management and operation will be more and more challenging due to the high risk of interfering with existing services and the higher trial costs if no reliable emulation platforms are available.

A Digital Twin is the real-time representation of a physical entity in the digital world. It has the characteristics of virtual-reality interrelation and real-time interaction, iterative operation and process optimization, full life-cycle and comprehensive data-driven network infrastructure. Currently, digital twin has been widely acknowledged in academic publications. See more in Section 3.

A digital twin for networks platform can be built by applying Digital Twin technologies to networks and creating a virtual image of physical network facilities (called herein, emulation). Basically, the digital twin for networks is an expansion platform of network simulation. The main difference compared to traditional network management systems is the interactive virtual-real mapping and data driven approach to build closed-loop network automation. Therefore, a digital twin network platform is more than an emulation platform or network simulator.

Through the real-time data interaction between the physical network and its twin network(s), the digital twin network platform might help the network designers to achieve more simplification, automatic, resilient, and full life-cycle operation and maintenance. More specifically, the digital twin network can, thus, be used to develop various rich network applications and assess specific behaviors (including network transformation) before actual implementation in the physical network, tweak the network for better optimized behavior, run 'what-if' scenarios that cannot be tested and evaluated easily in the physical network. In addition, service impact analysis tasks can also be facilitated.

## 2. Terminology

### 2.1. Acronyms & Abbreviations

IBN: Intent-Based Networking

IA: Artificial Intelligence

CI/CD: Continuous Integration / Continuous Delivery

ML: Machine Learning

OAM: Operations, Administration, and Maintenance

PLM: Product Lifecycle Management

## 2.2. Definitions

This document makes use of the following terms:

**Digital Twin:** a virtual instance of a physical system (twin) that is continually updated with the latter's performance, maintenance, and health status data throughout the physical system's life cycle.

**Digital twin network:** a digital twin that is used in the context of networking. This is also called, digital twin for networks. See more in Section 3.3.

## 3. Introduction and Concepts of Digital Twin Network

### 3.1. Background of Digital Twin

The concept of the "twin" dates to the National Aeronautics and Space Administration (NASA) Apollo program in the 1970s, where a replica of space vehicles on Earth was built to mirror the condition of the equipment during the mission [Rosen2015].

In 2003, Digital Twin was attributed to John Vickers by Michael Grieves in his product lifecycle management (PLM) course as "virtual digital representation equivalent to physical products" [Grieves2014]. Digital twin can be defined as a virtual instance of a physical system (twin) that is continually updated with the latter's performance, maintenance, and health status data throughout the physical system's life cycle [Madni2019]. By providing a living copy of physical system, digital twins bring numerous advantages, such as accelerated business processes, enhanced productivity, and faster innovation with reduced costs. So far, digital twin has been successfully applied in the fields of intelligent manufacturing, smart city, or complex system operation and maintenance to help with not only object design and testing, but also management aspects [Tao2019].

Compared with 'digital model' and 'digital shadow', the key difference of 'digital twin' is the direction of data between the physical and virtual systems [Fuller2020]. Typically, when using a digital twin, the (twin) system is generated and then synchronized using data flows in both directions between physical and digital components, so that control data can be sent, and changes between the physical and digital objectives and systems are automatically represented. This behavior is unlike a 'digital model' or 'digital shadow', which are usually synchronized manually, lacking of control data, and might not have a full cycle of data integrated.

At present (2022), there is no unified definition of digital twin framework. The industry, scientific research institutions, and standards developing organizations are trying to define a general or domain-specific framework of digital twin. [Natis-Gartner2017] proposed that building a digital twin of a physical entity requires four key elements: model, data, monitoring, and uniqueness. [Tao2019] proposed a five-dimensional framework of digital twin {PE, VE, SS, DD, CN}, in which PE represents physical entity, VE represents virtual entity, SS represents service, DD represents twin data, and CN represents the connection between various components. [ISO-2021] issued a draft standard for digital twin manufacturing system, and proposed a reference framework including data collection domain, device control domain, digital twin domain, and user domain.

### 3.2. Digital Twin for Networks

Communication networks can provide a solid foundation for implementing various 'digital twin' applications. At the same time, in the face of increasing business types, scale and complexity, a network itself also needs to use digital twin technology to seek better solutions beyond physical network. Since 2017, the application of digital twin technology in the field of communication networks has gradually been researched. Some examples are listed below.

In academy, [Dong2019] established the digital twin of 5G mobile edge computing (MEC) network, used the twin offline to train the resource allocation optimization and normalized energy-saving algorithm based on reinforcement learning, and then updated the scheme to MEC network. [Dai2020] established a digital twin edge network for mobile edge computing system, in which a twin edge server is used to evaluate the state of entity server, and the twin mobile edge computing system provides data for training offloading strategy. [Nguyen2021] discusses how to deploy a digital twin for complex 5G networks. [Hong2021] presents a digital twin platform towards automatic and intelligent management for data center networks, and then proposes a simplified the workflows of network service

management. In addition, international workshops dedicated to digital twin in network field have already appeared, such as IEEE DTPI 2021 - Digital Twin Network Online Session [DTPI2021], or are being proposed such as IEEE NOMS 2022 - TNT workshop [TNT2022].

Although the application of digital twin technology in networking has started, the research of digital twin for networks technology is still in its infancy. Current applications focus on specific scenarios (such as network optimization), where network digital twin is just used as a network simulation tool to solve the problem of network operation and maintenance. Combined with the characteristics of digital twin technology and its application in other industries, this document believes that digital twin network can be regarded as an organic whole of the overall network system and become a general architecture involving the whole life cycle of physical network in the future, serving the application of network innovative technologies such as network planning, construction, maintenance and optimization, improving the automation and intelligence level of the network.

### 3.3. Definition of Digital Twin Network

So far, there is no standard definition of "digital twin network" within the networking industry. This document defines "digital twin network" as a virtual representation of the physical network. Such virtual representation of the network is meant to be used to analyze, diagnose, emulate, and then control the physical network based on data, models, and interfaces. To that aim, a real-time and interactive mapping is required between the physical network and its virtual twin network.

Referring the characteristics of digital twin in other industries and the characteristics of the networking itself, the digital twin network should involve four key elements: data, mapping, models and interfaces as shown in Figure 1.

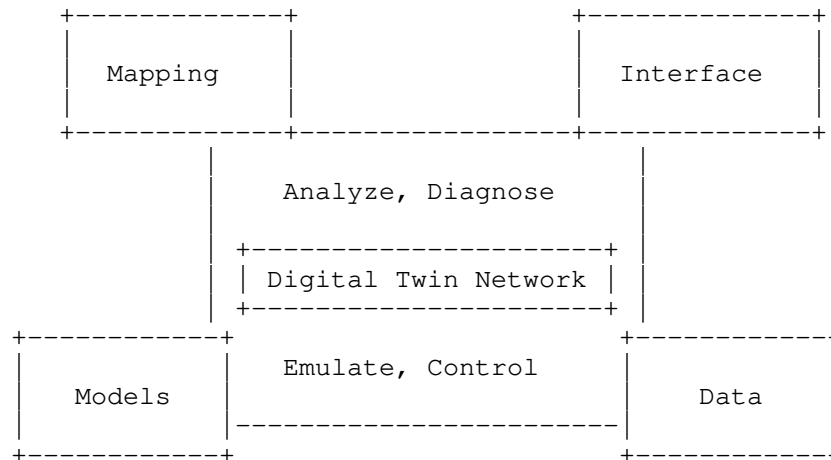


Figure 1: Key Elements of Digital Twin Network

**Data:** A digital twin network should maintain historical data and/or real time data (configuration data, operational state data, topology data, trace data, metric data, process data, etc.) about its real-world twin (i.e. physical network) that are required by the models to represent and understand the states and behaviors of the real-world twin.

The data is characterized as the single source of "truth" and populated in the data repository, which provides timely and accurate data service support for building various models.

**Models:** Techniques that involve collecting data from one or more sources in the real-world twin and developing a comprehensive representation of the data (e.g., system, entity, process) using specific models. These models are used as emulation and diagnosis basis to provide dynamics and elements on how the live physical network operates and generates reasoning data utilized for decision-making.

Various models such as service models, data models, dataset models, or knowledge graph can be used to represent the physical network assets and, then, instantiated to serve various network applications.

**Interfaces:** Standardized interfaces can ensure the interoperability of digital twin network. There are two major types of interfaces:

- \* The interface between the digital twin network platform and the physical network infrastructure.



- \* The interface between digital twin network platform and applications.

The former provides real-time data collection and control on the physical network. The latter helps in delivering application requests to the digital twin network platform and exposing the various platform capabilities to applications.

Mapping: Used to identify the digital twin and the underlying entities and establish a real-time interactive relation between the physical network and the twin network or between two twin networks. The mapping can be:

- \* One to one (pairing, vertical): Synchronize between a physical network and its virtual twin network with continuous flows.
- \* One to many (coupling, horizontal): Synchronize among virtual twin networks with occasional data exchange.

Such mappings provide a good visibility of actual status, making the digital twin suitable to analyze and understand what is going on in the physical network. It also allows using the digital twin to optimize the performance and maintenance of the physical network.

The digital twin network constructed based on the four core technology elements can analyze, diagnose, emulate, and control the physical network in its whole life cycle with the help of optimization algorithms, management methods, and expert knowledge. One of the objectives of such control is to master the digital twin network environment and its elements to derive the required system behavior, e.g., provide:

- \* repeatability: that is the capacity to replicate network conditions on-demand.
- \* reproducibility: i.e., the ability to replay successions of events, possibly under controlled variations.

Note: Real-time interaction is not always mandatory for all twins. When testing some configuration changes or trying some innovative techniques, the digital twins can behave as a simulation platform without the need of real time telemetry data. And even in this scenario, it is better to have interactive mapping capability so that the validated changes can be tested in real network whenever required by the testers. In most other cases (e.g., network optimization, network fault recovery), real-time interaction between virtual and real network is mandatory. This way, digital twin network can help achieve the goal of autonomous network or self-driven network.

#### 4. Benefits of Digital Twin Network

Digital twin network can help enabling closed-loop network management across the entire lifecycle, from deployment and emulation, to visualized assessment, physical deployment, and continuous verification. By doing so, network operators and end-users to some extent, as allowed by specific application interfaces, can maintain a global, systemic, and consistent view of the network. Also, network operators and/or enterprise user can safely exercise the enforcement of network planning policies, deployment procedures, etc., without jeopardizing the daily operation of the physical network.

The main difference between digital twin network and simulation platform is the use of interactive virtual-real mapping to build closed-loop network automation. Simulation platforms are the predecessor of the digital twin network, one example of such a simulation platform is network simulator [NS-3], which can be seen as a variant of digital twin network but with low fidelity and lacking for interactive interfaces to the real network. Compared with those classical approaches, key benefits of digital twin network can be summarized as follows:

- 1) Using real-time data to establish high fidelity twins, the effectiveness of network simulation is higher; then the simulation cost will be relatively low.
- 2) The impact and risk on running networks is low when automatically applying configuration/policy changes after the full analysis and required verifications (e.g., service impact analysis) within the twin network.
- 3) The faults of the physical network can be automatically captured by analyzing real-time data, then the correction strategy can be distributed to the physical network elements after conducting adequate analysis within the twins to complete the closed-loop automatic fault repair.

The following subsections further elaborate such benefits in details.

#### 4.1. Optimized Network Total Cost of Operation

Large scale networks are complex to operate. Since there is no effective platform for simulation, network optimization designs have to be tested on the physical network at the cost of jeopardizing its daily operation and possibly degrading the quality of the services supported by the network. Such assessment greatly increases network operator's Operational Expenditure (OPEX) budgets too.

With a digital twin network platform, network operators can safely emulate candidate optimization solutions before deploying them in the physical network. In addition, operator's OPEX on the real physical network deployment will be greatly decreased accordingly at the cost of the complexity of the assessment and the resources involved.

#### 4.2. Optimized Decision Making

Traditional network operation and management mainly focus on deploying and managing running services, but hardly support predictive maintenance techniques.

Digital twin network can combine data acquisition, big data processing, and AI modeling to assess the status of the network, but also to predict future trends, and better organize predictive maintenance. The ability to reproduce network behaviors under various conditions facilitates the corresponding assessment of the various evolution options as often as required.

#### 4.3. Safer Assessment of Innovative Network Capabilities

Testing a new feature in an operational network is not only complex, but also extremely risky. Service impact analysis is required to be adequately achieved prior to effective activation of a new feature.

Digital twin network can greatly help assessing innovative network capabilities without jeopardizing the daily operation of the physical network. In addition, it helps researchers to explore network innovation (e.g., new network protocols, network AI/ML applications) efficiently, and network operators to deploy new technologies quickly with lower risks. Take AI/ ML application as example, it is a conflict between the continuous high reliability requirement (i.e., 99.999%) and the slow learning speed or phase-in learning steps of AI/ML algorithms. With digital twin network, AI/ML can complete the learning and training with the sufficient data before deploying the model in the real network. This would encourage more network AI innovations in future networks.

#### 4.4. Privacy and Regulatory Compliance

The requirements on data confidentiality and privacy on network providers increase the complexity of network management, as decisions made by computation logics such as an SDN controller may rely upon the packet payloads. As a result, the improvement of data-driven management requires complementary techniques that can provide a strict control based upon security mechanisms to guarantee data privacy protection and regulatory compliance. This may range from flow identification (using the archetypal five-tuple of addresses, ports and protocol) to techniques requiring some degree of payload inspection, all of them considered suitable to be associated to an individual person, and hence requiring strong protection and/or data anonymization mechanisms.

With strong modeling capability provided by the digital twin network, very limited real data (if at all) will be needed to achieve similar or even higher level of data-driven intelligent analysis. This way, a lower demand of sensitive data will permit to satisfy privacy requirements and simplify the use of privacy-preserving techniques for data-driven operation.

#### 4.5. Customized Network Operation Training

Network architectures can be complex, and their operation requires expert personnel. Digital twin network offers an opportunity to train staff for customized networks and specific user needs. Two salient examples are the application of new network architectures and protocols or the use of "cyber-ranges" to train security experts in threat detection and mitigation.

### 5. Challenges to Build Digital Twin Network

According to [Hu2021], the main challenges in building and maintaining digital twins can be summarized as the following five aspects:

- \* Data acquisition and processing
- \* High-fidelity modeling
- \* Real-time, two-way connection between the virtual and the real twins
- \* Unified development platform and tools
- \* Environmental coupling technologies

Compared with other industrial fields, digital twin in networking field has its unique characteristics. On one hand, network elements and system have higher level of digitalization, which implies that data acquisition and virtual-real connection are relatively easy to achieve. On the other hand, there are many kinds of network elements and topologies in the network field; and the complex giant system of network carries a variety of business services. So, the construction of a digital twin network system needs to consider the following major challenges:

**Large scale challenge:** A digital twin of large-scale networks will significantly increase the complexity of data acquisition and storage, the design and implementation of relevant models. The requirements of software and hardware of the digital twin network system will be even more constraining. Therefore, efficient and low cost tools in various fields should be required. Take data as an example, massive network data can help achieve more accurate models. However, to lower the cost of virtual-real communication and data storage, efficient tools on data collection and data compression methods must be used.

**Interoperability:** Due to the inconsistency of technical implementations and the heterogeneity of vendor technologies, it is difficult to establish a unified digital twin network system with a common technology in a network domain. Therefore, it is needed firstly to propose a unified architecture of digital twin network, in which all components and functionalities are clear to all stakeholders; then define standardized and unified interfaces to connect all network twins via ensuring necessary compatibility.

**Data modeling difficulties:** Based on large-scale network data, data modeling should not only focus on ensuring the accuracy of model functions, but also has to consider the flexibility and scalability to compose and extend as required to support large scale and multi-purpose applications. Balancing these requirements further increases the complexity of building efficient and hierarchical functional data models. As an optional solution, straightforwardly clone the real network using virtualized resources is feasible to build the twin network when the network scale is relatively small. However, it will be of unaffordable resource cost for larger scales network. In this case, network modeling using mathematical abstraction or leveraging the AI algorithms will be more suitable solutions.

**Real-time requirements:** Network services normally have real-time requirements, the processing of model simulation and verification through a digital twin network will increase the service latency. Meanwhile, the real-time requirements will further increase

performance requirements on the system software and hardware. Moreover, it is also challenge to keep network digital twins in sync given the nature of distributed systems and propagation delays. To address these requirements, the function and process of the data model need to be based on automated processing mechanism under various network application scenarios. On the one hand, it is needed to design a simplified process to reduce the time cost for tasks in network twin as much as possible; on the other hand, it is recommended to define the real-time requirements of different applications, and then match the corresponding computing resources and suitable solutions as needed to complete the task processing in the twin.

Security risks: A digital twin network has to synchronize all or subset of the data related to involved physical networks in real time, which inevitably augments the attack surface, with a higher risk of information leakage, in particular. On one hand, it is mandatory to design more secure data mechanism leveraging legacy data protection methods, as well as innovative technologies such as block chain. On the other hand, the system design can limit the data (especially raw data) requirement on building digital twin network, leveraging innovative modeling technologies such as federal learning.

In brief, to address the above listed challenges, it is important to firstly propose a unified architecture of digital twin network, which defines the main functional components and interfaces (Section 6). Then, relying upon such an architecture, it is required to continue researching on the key enabling technologies including data acquisition, data storage, data modeling, interface standardization, and security assurance.

## 6. A Reference Architecture of Digital Twin Network

Based on the definition of the key digital twin network technology elements introduced in Section 3.3, a digital twin network architecture is depicted in Figure 2. This digital twin network architecture is broken down into three layers: Application Layer, Digital Twin Layer, and Physical Network Layer.

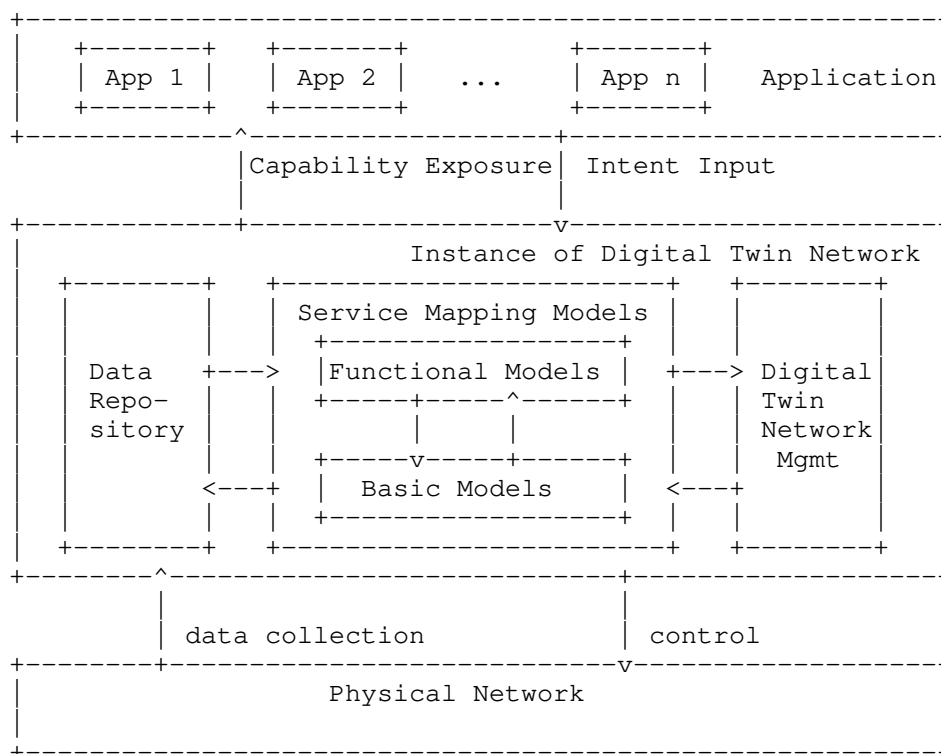


Figure 2: Reference Architecture of Digital Twin Network

**Physical Network:** All or subset of network elements in the physical network exchange network data and control messages with a network digital twin instance, through twin-physical control interfaces. The physical network can be a mobile access network, a transport network, a mobile core, a backbone, etc. The physical network can also be a data center network, a campus enterprise network, an industrial Internet of Things, etc.

The physical network can span across a single network administrative domain or multiple network administrative domains.

This document focuses on the IETF related physical network such as IP bearer network and datacenter network.

**Digital Twin Layer:** This layer includes three key subsystems: Data Repository subsystem, Service Mapping Models subsystem, and Digital Twin Network Management subsystem.

One or multiple digital twin network instances can be built and maintained:

- \* Data Repository subsystem is responsible for collecting and storing various network data for building various models by collecting and updating the real-time operational data of various network elements through the twin southbound interface, and providing data services (e.g., fast retrieval, concurrent conflict handling, batch service) and unified interfaces to Service Mapping Models subsystem.
- \* Service Mapping Models complete data modeling, provide data model instances for various network applications, and maximizes the agility and programmability of network services. The data models include two major types: basic and functional models.
  - Basic models refer to the network element model(s) and network topology model(s) of the network digital twin based on the basic configuration, environment information, operational state, link topology and other information of the network element(s), to complete the real-time accurate characterization of the physical network.
  - Functional models refer to various data models used for network analysis, emulation, diagnosis, prediction, assurance, etc. The functional models can be constructed and expanded by multiple dimensions: by network type, there can be models serving for a single or multiple network domains; by function type, it can be divided into state monitoring, traffic analysis, security exercise, fault diagnosis, quality assurance and other models; by network lifecycle management, it can be divided into planning, construction, maintenance, optimization and operation. Functional models can also be divided into general models and special-purpose models. Specifically, multiple dimensions can be combined to create a data model for more specific application scenarios.

New applications might need new functional models that do not exist yet. If a new model is needed, 'Service Mapping Models' subsystem will be triggered to help creating new models based on data retrieved from 'Data Repository'.



- \* Digital Twin Network Management fulfils the management function of digital twin network, records the life-cycle transactions of the twin entity, monitors the performance and resource consumption of the twin entity or even of individual models, visualizes and controls various elements of the network digital twin, including topology management, model management and security management.

Notes: 'Data collection' and 'change control' are regarded as southbound interfaces between virtual and physical network. From implementation perspective, they can optionally form a sub-layer or sub-system to provide common functionalities of data collection and change control, enabled by a specific infrastructure supporting bi-directional flows and facilitating data aggregation, action translation, pre-processing and ontologies.

Application Layer: Various applications (e.g., Operations, Administration, and Maintenance (OAM)) can effectively run over a digital twin network platform to implement either conventional or innovative network operations, with low cost and less service impact on real networks. Network applications make requests that need to be addressed by the digital twin network. Such requests are exchanged through a northbound interface, so they are applied by service emulation at the appropriate twin instance(s).

## 7. Interaction with IBN

Implementing Intent-Based Networking (IBN) is an innovative technology for life-cycle network management. Future networks will be possibly Intent-based, which means that users can input their abstract 'intent' to the network, instead of detailed policies or configurations on the network devices.

[I-D.irtf-nmrg-ibn-concepts-definitions] clarifies the concept of "Intent" and provides an overview of IBN functionalities. The key characteristic of an IBN system is that user intent can be assured automatically via continuously adjusting the policies and validating the real-time situation.

IBN can be envisaged in a digital twin network context to show how digital twin network improves the efficiency of deploying network innovation. To lower the impact on real networks, several rounds of adjustment and validation can be emulated on the digital twin network platform instead of directly on physical network. Therefore, digital twin network can be an important enabler platform to implement IBN systems and speed up their deployment.

## 8. Sample Application Scenarios

Digital twin network can be applied to solve different problems in network management and operation.

### 8.1. Human Training

The usual approach to network OAM with procedures applied by humans is open to errors in all these procedures, with impact in network availability and resilience. Response procedures and actions for most relevant operational requests and incidents are commonly defined to reduce errors to a minimum. The progressive automation of these procedures, such as predictive control or closed-loop management, reduce the faults and response time, but still there is the need of a human-in-the-loop for multiples actions. These processes are not intuitive and require training to learn how to respond.

The use of digital twin network for this purpose in different network management activities will improve the operators performance. One common example is cybersecurity incident handling, where "cyber-range" exercises are executed periodically to train security practitioners. Digital twin network will offer realistic environments, fitted to the real production networks.

### 8.2. Machine Learning Training

Machine Learning requires data and their context to be available in order to apply it. A common approach in the network management environment has been to simulate or import data in a specific environment (the ML developer lab), where they are used to train the selected model, while later, when the model is deployed in production, re-train or adjust to the production environment context. This demands a specific adaption period.

Digital twin network simplifies the complete ML lifecycle development by providing a realistic environment, including network topologies, to generate the data required in a well-aligned context. Dataset generated belongs to the digital twin network and not to the production network, allowing information access by third parties, without impacting data privacy.

### 8.3. DevOps-Oriented Certification

The potential application of CI/CD models network management operations increases the risk associated to deployment of non-validated updates, what conflicts with the goal of the certification requirements applied by network service providers. A solution for addressing these certification requirements is to verify the specific impacts of updates on service assurance and SLAs using a digital twin network environment replicating the network particularities, as a previous step to production release.

Digital twin network control functional block supports such dynamic mechanisms required by DevOps procedures.

### 8.4. Network Fuzzing

Network management dependency on programmability increases systems complexity. The behavior of new protocol stacks, API parameters, and interactions among complex software components are examples that imply higher risk to errors or vulnerabilities in software and configuration.

Digital twin network allows to apply fuzzing testing techniques on a twin network environment, with interactions and conditions similar to the production network, permitting to identify and solve vulnerabilities, bugs and zero-days attacks before production delivery.

## 9. Research Perspectives: A Summary

Research on digital twin network has just started. This document presents an overview of the digital twin network concepts and reference architecture. Looking forward, further elaboration on digital twin network scenarios, requirements, architecture, and key enabling technologies should be investigated by the industry, so as to accelerate the implementation and deployment of digital twin network.

## 10. Security Considerations

This document describes concepts and definitions of digital twin network. As such, the following security considerations remain high level, i.e., in the form of principles, guidelines or requirements.

Security considerations of the digital twin network include:

- \* Secure the digital twin system itself.

- \* Data privacy protection.

Securing the digital twin network system aims at making the digital twin system operationally secure by implementing security mechanisms and applying security best practices. In the context of digital twin network, such mechanisms and practices may consist in data verification and model validation, mapping operations between physical network and digital counterpart network by authenticated and authorized users only.

Synchronizing the data between the physical and the digital twin networks may increase the risk of sensitive data and information leakage. Strict control and security mechanisms must be provided and enabled to prevent data leaks.

## 11. Acknowledgements

Many thanks to the NMRG participants for their comments and reviews. Thanks to Daniel King, Quifang Ma, Laurent Ciavaglia, Jerome Francois, Jordi Paillisse, Luis Miguel Contreras Murillo, Alexander Clemm, Qiao Xiang, Ramin Sadre, Pedro Martinez-Julia, Wei Wang, Zongpeng Du, and Peng Liu.

Diego Lopez and Antonio Pastor were partly supported by the European Commission under Horizon 2020 grant agreement no. 833685 (SPIDER), and grant agreement no. 871808 (INSPIRE-5Gplus).

## 12. IANA Considerations

This document has no requests to IANA.

## 13. Open issues

- \* The draft focuses on concept and architecture of digital twin network, not including enabling technologies. Actually, each 'enabling technology' is worth of a separate draft to study in details in future. A decision is needed that whether to add a section to describe the enabling technologies in brief.
- \* Related to above issue, if section of enabling technologies is added, recent technologies (e.g. Network connectivity, Real-time data communication, Collaboration management, conflict detection and resolution, etc.) recently discussed in the IRTF/IETF should be described.
- \* In section of 'Sample Application Scenarios', to dig deeper into one or two use cases.

- \* On the research side, the idea behind digital twin networks is reminiscent of earlier work from the 1990s that should be referenced/acknowledged. Examples include the Shadow MIB concept, Inductive Modeling Technique, etc.

#### 14. Informative References

- [Dai2020] Dai, Y. Dai., Zhang, K. Zhang., Maharjan, S. Maharjan., and Yan Zhang. Zhang, "Deep Reinforcement Learning for Stochastic Computation Offloading in Digital Twin Networks. IEEE Transactions on Industrial Informatics, vol. 17, no. 17", August 2020.
- [Dong2019] Dong, R. Dong., She, C. She., HardjawanaLiu, W. Hardjawana., Li, Y. Li., and B. Vucetic. Vucetic, "Deep Learning for Hybrid 5G Services in Mobile Edge Computing Systems: Learn from a Digital Twin. IEEE Transactions on Wireless Communications, vol. 18, no. 10", July 2019.
- [DTPI2021] "IEEE International Conference on Digital Twins and Parallel Intelligence - Digital Twin Network Session, <https://www.dtpi.org/video/10>", July 2021.
- [Fuller2020] Fuller, A. Fuller., Fan, Z., Day, C., and C. Barlow, "Digital Twin: Enabling Technologies, Challenges and Open Research," in IEEE Access, vol. 8, pp. 108952-108971", 2020.
- [Grieves2014] Grieves, M. Grieves., "Digital twin: Manufacturing excellence through virtual factory replication", 2003, <<https://www.3ds.com/fileadmin/PRODUCTS-SERVICES/DELMIA/PDF/Whitepaper/DELMIA-APRISO-Digital-Twin-Whitepaper.pdf>>.
- [Hong2021] Hong, H., Wu, Q., Dong, F., Song, W., Sun, R., Han, T., Zhou, C., and H. Yang, "NetGraph: An Intelligent Operated Digital Twin Platform for Data Center Networks. In ACM SIGCOMM 2021 Workshop on Network-Application Integration (NAI' 21), Virtual Event, USA. ACM, New York, NY, USA", 2021.
- [Hu2021] Hu, W., Zhang, T., Deng, X., Liu, Z., and J. Tan, "Digital twin: a state-of-the-art review of its enabling technologies, applications and challenges. Journal of Intelligent Manufacturing and Special Equipment, Vol. 2 No. 1, pp. 1-34", 2021.

- [I-D.irtf-nmrg-ibn-concepts-definitions]  
Clemm, A., Ciavaglia, L., Granville, L. Z., and J. Tantsura, "Intent-Based Networking - Concepts and Definitions", Work in Progress, Internet-Draft, draft-irtf-nmrg-ibn-concepts-definitions-06, 15 December 2021, <<https://www.ietf.org/archive/id/draft-irtf-nmrg-ibn-concepts-definitions-06.txt>>.
- [ISO-2021] ISO, "Digital Twin manufacturing framework - Part 2: Reference architecture: ISO/CD 23247-2. <https://www.iso.org/standard/78743.html>", 2021.
- [Madni2019]  
Madni, A. Madni., Madni, C. Madni., and S. Lucero. Lucero, "Leveraging digital twin technology in model-based systems engineering. Systems, vol. 7, no. 1, p. 7", January 2019.
- [Natis-Gartner2017]  
Natis, Y. Natis., Velosa, A. Velosa., and W. R. Schulte. Schulte, "Innovation insight for digital twins - driving better IoT-fueled decisions. <https://www.gartner.com/en/documents/3645341>", 2017.
- [Nguyen2021]  
Nguyen, H. X. Nguyen., Trestian, R. Trestian., To, D. To., and M. Tatipamula. Tatipamula, "Digital Twin for 5G and Beyond. IEEE Communications Magazine, vol. 59, no. 2", February 2021.
- [NS-3] "Network Simulator, NS-3. <https://www.nsnam.org/>".
- [Roson2015]  
Rosen, R. Rosen., Wichert, G. Von Wichert., Lo, G. Lo., and K.D. Bettenhausen. Bettenhausen, "About the importance of autonomy and DTs for the future of manufacturing. IFAC-Papersonline, Vol. 48, pp. 567-572.", 2015.
- [Tao2019] Tao, F. Tao., Zhang, H. Zhang., Liu, A. Liu., and A. Y. C. Nee. Nee, "Digital Twin in Industry: State-of-the-Art. IEEE Transactions on Industrial Informatics, vol. 15, no. 4.", April 2019.
- [TNT2022] "IEEE International workshop on Technologies for Network Twins, <https://sites.google.com/view/tnt-2022/>", 2022.

## Appendix A. Change Logs

v06 - v07: Addressed reviewer's comments from adoption call, including below major changes.

- \* Resequenced the sections via adding more subsections on concepts of digital twin network, removing the 'Requirements Language' section, and moving ahead the 'Challenges' section.
- \* Cited more papers, or industrial information on digital twin concepts and digital twin for networks.
- \* Added more information on describing the challenges and key characteristics digital twin network.
- \* Removed previous open issue on investigating related digital twin network work and identify the differences and commonalities, and added several new open issues for future studys.
- \* Other Editorial changes.

v05 - v06: Addressed comments form meeting and maillist, to request adoptoin call.

- \* Remove acronym DTN to avoid conflict with 'Delay Tolerant Network';
- \* Elaborate the descriptoin of Digital Twin Network architecture that supports multiple instances;
- \* Other Editorial changes.

04 - v05

- \* Clarify the difference between digital twin network platform and traditional network management system;
- \* Add more references of researches on applying digital twin to network field;
- \* Clarify the benefit of 'Privacy and Regulatory Compliance';
- \* Refine the description of reference architecture;
- \* Other Editorial changes.

v03 - v04

- \* Update data definition and models definitions to clarify their difference.
- \* Remove the orchestration element and consolidated into control functionality building block in the digital twin network.
- \* Clarify the mapping relation (one to one, and one to many) in the mapping definition.
- \* Add explanation text for continuous verification.

v02 - v03

- \* Split interaction with IBN part as a separate section.
- \* Fill security section;
- \* Clarify the motivation in the introduction section;
- \* Use new boilerplate for requirements language section;
- \* Key elements definition update.
- \* Other editorial changes.
- \* Add open issues section.
- \* Add section on application scenarios.

#### Authors' Addresses

Cheng Zhou  
China Mobile  
Beijing  
100053  
China  
Email: zhouchengyjy@chinamobile.com

Hongwei Yang  
China Mobile  
Beijing  
100053  
China  
Email: yanghongwei@chinamobile.com



Xiaodong Duan  
China Mobile  
Beijing  
100053  
China  
Email: duanxiaodong@chinamobile.com

Diego Lopez  
Telefonica I+D  
Seville  
Spain  
Email: diego.r.lopez@telefonica.com

Antonio Pastor  
Telefonica I+D  
Madrid  
Spain  
Email: antonio.pastorperales@telefonica.com

Qin Wu  
Huawei  
101 Software Avenue, Yuhua District  
Nanjing  
Jiangsu, 210012  
China  
Email: bill.wu@huawei.com

Mohamed Boucadair  
Orange  
Rennes 35000  
France  
Email: mohamed.boucadair@orange.com

Christian Jacquenet  
Orange  
Rennes 35000  
France  
Email: christian.jacquenet@orange.com

Internet Research Task Force  
Internet-Draft  
Intended status: Informational  
Expires: 28 April 2022

Y. Zhu  
D. Chen  
C. Zhou  
China Mobile  
P. Martinez-Julia, Ed.  
NICT  
25 October 2021

An Efficient Data collection method for Digital Twin Network  
draft-zhu-nmrg-digitaltwin-data-collection-01

Abstract

Digital Twin Network is a network system with Physical Network and Twin Network, which can be mapped interactively in real time. The construction of Digital Twin Network requires real-time data of Physical Network to update the state of Twin Network. However the existing method collects the full amount of data from the Physical Network for modeling, and does not consider the problems such as time-lag, insufficient storage resources, low computational efficiency and waste of bandwidth resources. This document introduces an efficient data collection, aggregation and correlation method in which the Twin Network sends instructions to the Physical Network to collect data on demand, and then the Physical Network completes instructions such as knowledge representation, Telemetry Streaming Element of Physical Network completes data aggregation and correlation. Finally Telemetry Streaming Element sends the processed data to the Twin Network.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on 28 April 2022.

#### Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

#### Table of Contents

1. Introduction . . . . .	3
2. Definitions and Acroyms . . . . .	3
3. Overview . . . . .	3
4. Conclusion . . . . .	7
5. Security Considerations . . . . .	7
6. IANA Considerations . . . . .	8
7. References . . . . .	8
7.1. Normative References . . . . .	8
7.2. Informative References . . . . .	8
Index . . . . .	8
Authors' Addresses . . . . .	8

## 1. Introduction

With the deployment of Internet of Things, cloud computing and data center, etc., the scale of the current network is expanded gradually. However, the increase of network scale leads to the increasing complexity of the current network, and that induces plenty of problems. In order to improve the autonomy ability of network and reduce the negative effect on Physical Network, we consider that an endogenous intelligent and autonomous network architecture which achieves self-optimization and decision is indispensable. Digital twin, as an innovative technology, has the potential to realize this architecture because it can optimize and validate policies through real-time and interactive mapping with physical entities.[I-D.zhou-nmrg-digitaltwin-network-concepts]

Data is the cornerstone of Digital Twin Network construction. In the face of large network scale, data collection, storage and management are faced with great challenges. If the full-data collection method is adopted, huge storage space and bandwidth resource is needed, especially for complex scenarios that require real-time data and traffic from multi-source heterogeneous devices. Therefore, it is extremely important to propose a lightweight and efficient data collection, aggregation and correlation method.

## 2. Definitions and Acroyms

PN: Physical Network

IMC: Instruction Management Center

DSC: Data Storage Center

TN: Twin Network

TSE: Telemetry Streaming Element

RDF: Resource Description Framework

CPE: Complex Event Processing

## 3. Overview

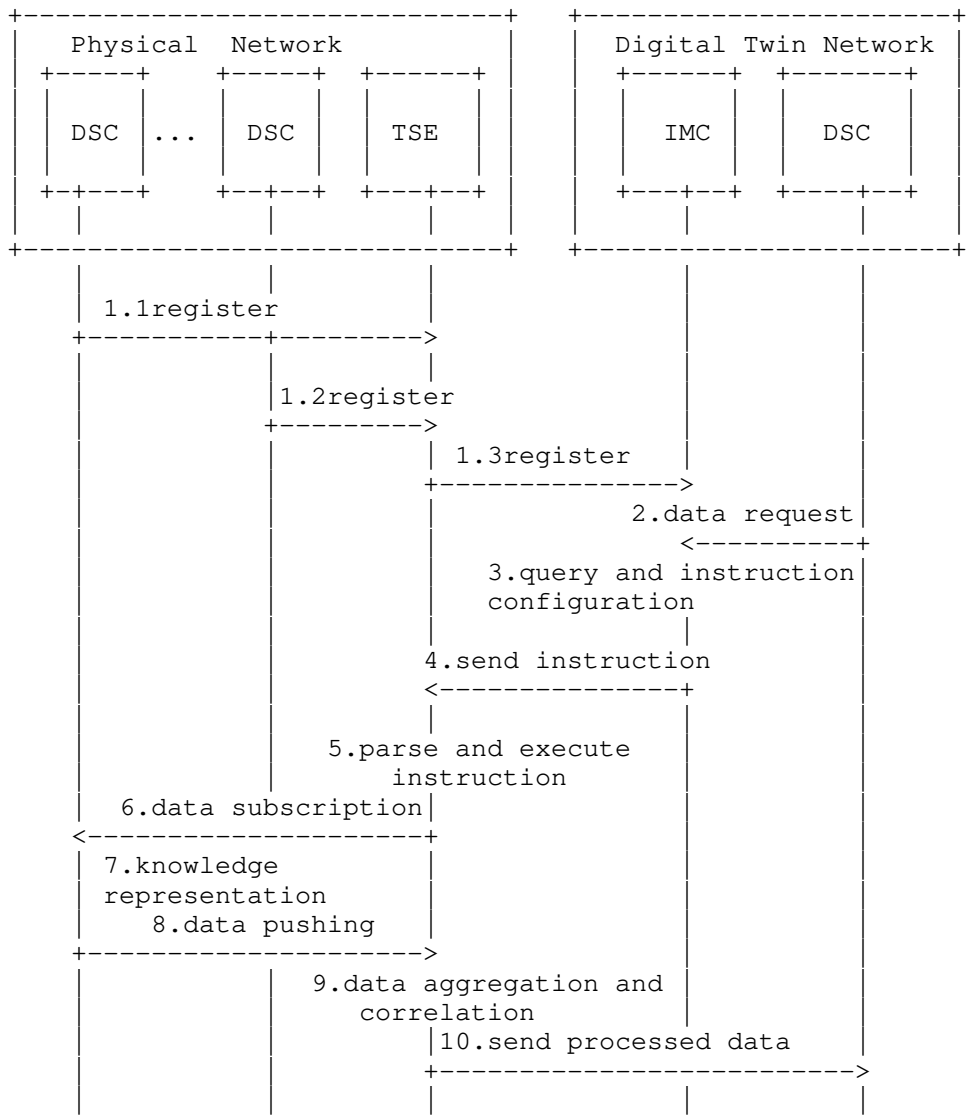
Digital Twin Network is a network system with Physical Network and Twin Network, which can be mapped interactively in real time. The construction of Digital Twin Network requires real-time data of Physical Network to update the state of Twin Network. However the existing method collects the full amount of data from the Physical Network for modeling, and does not consider the problems such as

time-lag, insufficient storage resources, low computational efficiency and waste of bandwidth resources caused by data transmission. In order to solve these problems, this memo introduces an efficient data collection method for Digital Twin Network. This data collection method is to send instructions model in the Twin Network to the Physical Network to collect data on demand, and then the Physical Network completes instructions such as data cleaning or knowledge representation, and then sends the representation data to the Digital Twin Network.

Digital Twin Network consists of Physical Network and Twin Network. The Physical Network includes multiple Data Storage Centers and Telemetry Streaming Element[I-D.ietf-opsawg-ntf], and the Twin Network includes the Instruction Management Center and Data Storage Center. Telemetry Streaming Element has multiple functions, including data collection, data aggregation, data correlation, knowledge representation and query, etc. In addition, the Complex Event Processing(CPE) engine is integrated into TSE to perform query function. The Instruction Management has two functions. On the one hand, the Instruction Management Center of the Twin Network is mainly used to manage the registration of the Data Storage Center in the Physical Network, and its registration information can include various key information such as the IP address of the Data Storage Center in the Physical Network, data type, and various index names in the data , data source name and data size, etc; on the other hand, it is mainly used to adaptively configure data collection instructions according to the collection requirements of the Data Storage Center in the Twin Network, and search for IP addresses to send instructions. The instruction-carrying information includes rule-based mathematical expressions, executable models in .exe format, dynamic collection frequency, parameter lists, program text files in .m format, text files with parameter configuration, and other types of files. Instructions are flexible and programmable, and can be created, modified, combined, and deleted at any time according to requirements. When the Data Storage Center of the Twin Network initiates data collection requests to the Instruction Management Center, the Instruction Management Center searches for IP addresses of Data Storage Center from registration information according to critical information such as data type and data name, and functional instructions for data processing or knowledge representation can be implemented depending on the demand configuration. The Data Storage Center of the Twin Network is mainly used to store the effective information after data processing and knowledge representation returned by the Telemetry Streaming Element in the Physical Network.

Data Storage Center in the Physical Network has two functions. On the one hand, it can store data, such as performance indicators, operational status, log, traffic scheduling, business requirements,

etc. On the other hand, it has the function of automatically parsing the instructions sent by the Telemetry Streaming Element. Then the operating environment of the instruction is configured according to the instruction needs, and data processing or knowledge representation is performed based on the instruction. Data processing mainly includes data cleaning, filling missing data, normalization, conflict verification, etc. Knowledge representation refers to the representation of the original data as a data structure that can be used for efficient computation. Such representation results are closer to machine language, which is conducive to the rapid and accurate construction of the model. The role of knowledge representation is to represent the original data as a data structure that can be used to efficiently calculate. Such representation results closer to the machine language, which is conducive to the rapid and accurate construction of the model.



The specific process is as follows:

- \* The Data Storage Centers in the Physical Network registers with the Telemetry Streaming Element in the Physical Network. The Telemetry Streaming Element registers with the Instruction management center. The registration information includes the IP address of the Data Storage Center, the data type, the data source, or the data size, etc.

- \* The Data Storage Center in the Twin Network sends the data collection request to the Instruction Management Center.
- \* According to the data collection request, the Instruction Management Center intelligently query the registration information for addressing, and configures the data processing instruction.
- \* The Instruction Management Center in the Twin Network sends the corresponding instruction according to the query result to the Telemetry Streaming Element in the Physical Network.
- \* After receiving the instructions, the Telemetry Streaming Element in the Physical Network will parse them and execute them according to the instructions, and query the location of data stored. The query function can be performed by the Complex Event Processing (CEP) engine, which receives all telemetry data and processes it with all queries provided.
- \* The Telemetry Streaming Element sends data subscription to DSC of the Physical Network.
- \* DSC of Physical Network performs knowledge representation of local data, for example, in RDF form, also sends raw data to TSE for knowledge representation.
- \* DSC of Physical Network push data or knowledge to TSE.
- \* TSE aggregates and correlates the collected data or knowledge. Then according to the actual needs, decide whether to perform knowledge representation.
- \* TSE sends the processed data or knowledge to DSC of Twin Network.

#### 4. Conclusion

This memo introduces an efficient data collection method for Digital Twin Network. This data collection method is to send instructions model in the Twin Network to the Physical Network to collect data on demand, and then the Physical Network completes instructions such as data cleaning or knowledge representation, and then sends the representation data to the Digital Twin Network. And the data collection process between the Physical Network and the Twin Network is introduced in detail.

#### 5. Security Considerations

TBD.



## 6. IANA Considerations

This document has no requests to IANA.

## 7. References

### 7.1. 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, <<https://www.rfc-editor.org/info/rfc2119>>.

### 7.2. Informative References

[I-D.ietf-opsawg-ntf]  
Song, H., Qin, F., Martinez-Julia, P., Ciavaglia, L., and A. Wang, "Network Telemetry Framework", Work in Progress, Internet-Draft, draft-ietf-opsawg-ntf-09, 13 October 2021, <<https://www.ietf.org/archive/id/draft-ietf-opsawg-ntf-09.txt>>.

[I-D.zhou-nmrg-digitaltwin-network-concepts]  
Zhou, C., Yang, H., Duan, X., Lopez, D., Pastor, A., Wu, Q., Boucadair, M., and C. Jacquenet, "Digital Twin Network: Concepts and Reference Architecture", Work in Progress, Internet-Draft, draft-zhou-nmrg-digitaltwin-network-concepts-04, 7 July 2021, <<https://www.ietf.org/archive/id/draft-zhou-nmrg-digitaltwin-network-concepts-04.txt>>.

## Index

?

?

???

Section 3

## Authors' Addresses

Yanhong Zhu  
China Mobile  
Beijing  
100053  
China

Email: zhuyan hong@chinamobile.com

Danyang Chen  
China Mobile  
Beijing  
100053  
China

Email: chendanyang@chinamobile.com

Cheng Zhou  
China Mobile  
Beijing  
100053  
China

Email: zhouchengyjy@chinamobile.com

Pedro Martinez-Julia (editor)  
NICT  
4-2-1, Nukui-Kitamachi, Koganei, Tokyo,  
184-8795  
Japan

Email: pedro@nict.go.jp