

Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 7, 2020

A. Clemm
Futurewei
L. Ciavaglia
Nokia
L. Granville
Federal University of Rio Grande do Sul (UFRGS)
J. Tantsura
Apstra, Inc.
November 4, 2019

Intent-Based Networking - Concepts and Overview
draft-clemm-nmrg-dist-intent-03

Abstract

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

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 May 7, 2020.

Copyright Notice

Copyright (c) 2019 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. Key Words	4
3. Definitions and Acronyms	4
4. Introduction of Concepts	5
4.1. Intent and Intent-Based Management	5
4.2. Related Concepts	6
4.2.1. Service Models	6
4.2.2. Policy and Policy-Based Management	8
4.2.3. Distinguishing between Intent, Policy, and Service Models	10
5. Principles	11
6. Lifecycle	14
7. Intent-Based Networking - Functionality	18
7.1. Intent Fulfillment	18
7.2. Intent Assurance	18
8. Items for Discussion	19
9. IANA Considerations	19
10. Security Considerations	19
11. References	19
11.1. Normative References	19
11.2. Informative References	19
Authors' Addresses	21

1. Introduction

Traditionally in the IETF, interest with regard to management and operations 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 for this is SNMP-based management and the 200+ MIBs that have been defined by the IETF over the years. More recent examples include YANG data model definitions for aspects such as interface configuration, ACL configuration, or Syslog configuration.

There is a sense and reality that in modern network environments managing networks by configuring myriads of "nerd knobs" on a device-

by-device basis is no longer sustainable. Big challenges arise with keeping device configurations not only consistent across a network, but consistent with the needs of services and service features they are supposed to enable. Adoptability to changes at scale is a fundamental property of a well designed IBN system, that requires ability to consume and process analytics that are context/intent aware at near real time speeds. At the same time, operations need to be streamlined and automated wherever possible to not only lower operational expenses, but allow for rapid reconfiguration of networks at sub-second time scales and to ensure networks are delivering their functionality as expected.

Accordingly, 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]. In addition, a lot of 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 management of the network as a whole exceptionally easy, 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 vision has since caught on with the industry in a big way, leading to a significant number 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 systems (offering full lifecycle of intent), SDN controllers (offering a single point of control and administration for a network) as well as network management and Operations Support Systems (OSS).

However, 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. High-level operational objectives would propagate in top-down fashion from upper to lower layers. The associated abstraction hierarchy was key

to decompose management complexity into separate areas of concerns. 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 management" 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.

What has been missing, however, is putting these concepts into a more current context and updating it to account for current technology trends. This document attempts to clarify the concepts behind intent. It differentiates it from related concepts. It also provides an overview of first-order principles of Intent-Based Networking as well as associated functionality. In addition, a number of research challenges are highlighted. 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 Intent-Based Networking.

2. Key Words

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Definitions and Acronyms

ACL: Access Control List

Intent: An abstracted, declarative and vendor agnostic set of rules used to provide full lifecycle (Design/Build/Deploy/Validate) to a network and services it provides.

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

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

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

IBS: Intent Based 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.

4. Introduction of Concepts

The following section provides an overview of the concept of intent respectively intent-based management. It also provides an overview of the related concepts of service models, and of policies respectively policy-based management, and explains how they relate to intent and intent-based management.

4.1. Intent and Intent-Based Management

In the context of Autonomic Networks, Intent 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. However, to avoid using "intent" simply as a synonym for "policy, a clearer distinction needs to be introduced that distinguishes intent clearly from other types of policies.

For one, while Intent-Based Management clearly aims to lead towards networks that are dramatically simpler to manage and operate requiring only minimal outside intervention, the concept of "intent" is not limited to autonomic networks, but applies to any network. Networks, even when considered "autonomic", are not clairvoyant and have no way of automatically knowing particular operational goals nor what 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 by what informally constitutes "intent".

More specifically, intent is a declaration of operational goals that a network should meet and outcomes that the network is supposed to deliver, without specifying how to achieve them. Those goals and outcomes are defined in a manner that is purely declarative - they specify what to accomplish, not how to achieve it. "Intent" thus applies several important concepts simultaneously:

- o It provides data abstraction: Users and operators do not need to be concerned with low-level device configuration and nerd knobs.

- o It provides functional abstraction from particular management and control logic: Users and operators do not need to be concerned even with how to achieve a given intent. What is specified is a desired outcome, with the intent-based system automatically figuring out a course of action (e.g. a set of rules, an algorithm) for how to achieve the outcome.

In an autonomic network, intent should be rendered by the network itself, i.e. translated into device-specific rules and courses of action. Ideally, it should not even 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 abstraction. Because intent holds for the network as a whole, not individual devices, it needs to be automatically disseminated across all devices in the network, which can themselves decide whether they need to act on it. This facilitates management even further, since it obviates the need for a higher-layer system to break down and decompose higher-level intent, and because there is no need to even discover and maintain an inventory of the network to be able to manage it.

Tentative definition for intent-based networks Networks configuring and adapting autonomously to the user or operator intentions (i.e., a desired state or behavior) without the need to specify every technical detail of the process and operations to achieve it (i.e., the "machines" will figure out on their own how to realize the user goal).

Other definitions of intent exist such as [TR523] and will be investigated in future revisions of this document. Likewise, some definitions of intent allow for the presence of a centralized function that renders the intent into lower-level policies or instructions and orchestrates them across the network. While to the end user the concept of "intent" appears the same regardless of its method of rendering, this interpretation opens a slippery slope of how to clearly distinguish "intent" from other higher-layer abstractions. Again, these notions will be further investigated in future revisions of this document and in collaboration with NMRG.

4.2. Related Concepts

4.2.1. 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/vendor agnostic way that can be used independent 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, 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, who might be billed for the service.

Instantiating a service typically involves multiple aspects:

- o A user (or northbound system) needs to define and/or request a service to be instantiated.
- o Resources need to be allocated, such as IP addresses, AS numbers, VLAN or VxLAN pools, interfaces, bandwidth, or memory.
- o 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 with the service).
- o [I-D.ietf-teas-te-service-mapping-yang] is an example of such mapping - a data model to map customer service models (e.g., the L3VPM Service Model) to Traffic Engineering (TE) models (e.g., the TE Tunnel or the Abstraction and Control of Traffic Engineered Networks Virtual Network model)
- o Bindings need to be maintained between upper and lower-level objects.
- o Once instantiated, the service needs to be validated and assured to ensure that the network indeed delivers the service as requested.

They involve a system, such as a controller, that provides provisioning logic. Orchestration itself is generally conducted using a "push" model, in which the controller/manager initiates the operations as required, pushing down the specific configurations to the device. (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. 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 catalogues, tariffs, service contracts, and Service Level Agreements (SLAs) including contractual agreements regarding remediation actions.

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 the intent system. Instead, management of service models requires development of sophisticated algorithms and control logic by network providers or system integrators.

4.2.2. Policy and Policy-Based Management

Policy-based management (PBM) 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 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

(that get assessed and that 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 specify precisely what needs to be done when and in which circumstance. 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. In effect, policies define management as a set of simple control loops.

Policies typically involve a certain degree of abstraction in order to cope with 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, 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 to apply 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 definition across different networking domains, WAN, DC or public cloud.

Policy-based management 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 monitor their proper operation. That 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, PBM 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 specification of rules that allow to define various triggers for certain courses of actions. Unlike intent, the definition of those rules (and courses of actions) 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 PBM.

4.2.3. 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. Service models have less in common with policy and intent than policy and intent do with each other.

Summarized differences:

- o 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 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.
- o 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 themselves, without requiring intervention by outside system. Policy lets users define what to do under what circumstances, but it does not specify a desired outcome.
- o Intent is a higher-level declarative policy 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 the need to enumerate specific events, conditions,

and actions. Which algorithm or rules to apply can be automatically "learned/derived from intent" by the intent system. In the context of autonomic networking, ideally, intent is rendered by the network itself; also the dissemination of intent across the network and any required coordination between nodes is resolved by the network itself without the need for outside systems.

One analogy to capture the difference between policy and intent 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. They are able to make automatic inferences based on those rules, but are not able to "learn" 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. However, they do require a learning or training phase and explanations of actions that the system actually takes provide a different set of challenges.

5. Principles

The following 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 state and actual state of the system and determining drift between them. SSoT and the drift information provide the basis for corrective actions. If the intent-based is equipped with prediction capabilities or means, it can further develop strategies to anticipate, plan and pro-actively act on the 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 query, poll or filter the measured and correlated information to create so-called "views". These views can serve the operators and/or the users of the intent-based system. To create intents as single sources of truth, the intent-based system must follow well-specified and well-documented processes and models. In other contexts

[Lenrow15], SSoT is also referred to as the invariance of the intent.

2. One touch but not one shot. In an ideal intent-based system, the user expresses its intents 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 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 be mistaken 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 interactive process. Depending on the level of abstraction, the intent expressions will 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 the 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 Oversight. 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 its 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 oversight allows to operate a system that will have the necessary levels of autonomy to conduct its tasks and operations without requiring 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 oversight to satisfy the user requirements for reporting and escalation of relevant information. to be added: description for feedback, reporting, guarantee scope (check points, guard rails, dynamically provisioned, context rich, regular operation vs. exception/ abnormal, information zoom in-out, and link to SVoT. Accountable for decisions and efficiency, late binding (leave it to the

system where to place functionality, how to accomplish certain goals).

4. Learning. An intent-based system is a learning system. By contrast to imperative type of system, such as Event-Condition-Action policy rules, where the user define beforehand the expected behavior of the system to various event and conditions, in an intent-based system, the user only declare what the system should 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 implies also the availability in the intent-based system of capabilities or means for learning, reasoning and knowledge representation and management. The learning abilities of an intent-based systems 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. Cope with unawareness of users (smart recommendations).
5. Explainability. Need expressive network capabilities, requirements and constraints to be able to compose/decompose intents, map user's expectation to system capabilities. capability exposure. not just automation of steps that need to be taken, but of bridging the semantic gap between "intent" and actionable levels of instructions Context: multi providers, need discovery and semantic descriptions Explainability: why is a network doing what it is doing
6. Abstraction - users do not need to be concerned with how intent is achieved

Additional principles will be described in future revision of this document addressing aspects such as: Target groups not individual devices, agnostic to implementation details, user-friendly, user vocabulary vs. language of the device/network, explainability, validation and troubleshooting, how to resolve and point out conflicts (between intents), reconcile the reality of what is possible with the fiction of what the user would want, "moderate", awareness of operating within system boundaries, outcome-driven

((what not how, for the user); (what and how/where, for the operator).not imperative/instruction based.)).

The above principles will be further used to understand implications on the design of intent-based systems and their supporting architecture, and derive functional and operational requirements.

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 are divided into two functional (horizontal) planes and into three (vertical) spaces.

The functional planes provide structure for the main functional concerns that are associated with intent: how to fulfill intent, and how to assure it.

- o 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. This includes:
 - * Functions that recognize intent from interaction with the user and functions that allow users to refine their intent and articulate it in such ways so that it becomes actionable by an Intent-Based System. Those functions can involve 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 tradeoffs, and to facilitate refinements.
 - * Functions that translate user intent into courses of actions and requests to take against the network, which will be meaningful to network configuration and provisioning systems. Possibly, this includes learning functions and algorithms that optimize the courses of actions to take in order to result in the best outcomes, specifically in cases where multiple ways of achieving those outcomes are conceivable.
 - * Functions that perform and orchestrate the configuration and provisioning steps that were determined by the previous intent translation step.

- o Assurance is concerned with the functions that are necessary ensure that the network indeed complies with the desired intent once it has been fulfilled. This includes:
 - * Functions that monitor and observe the network and its exhibited behavior.
 - * Functions that assess and validate whether the observation indicate compliance with intent. This can include functions that perform analysis and aggregation of raw observation data.
 - * Functions that trigger corrective action as needed.
 - * Functions that abstract the observations and analysis results in a way that makes it possible for users to relate them to intent. 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.
 - * Functions that report intent compliance status and that provide adequate summarization and visualization to the user.

The spaces indicate the different perspectives and interactions with different roles that are involved to address the functions:

- o 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 whether their intent is having the desired effect.
- o The translation or Intent-Based System (IBS) space involves the functions that bridge the gap between intent users and network operations. This includes the functions used to translate an intent into a course of action, the algorithms used to plan and optimize those courses of actions also in consideration of feedback, the functions to analyze and abstract observations to validate compliance with intent and take corrective actions as necessary.
- o 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.

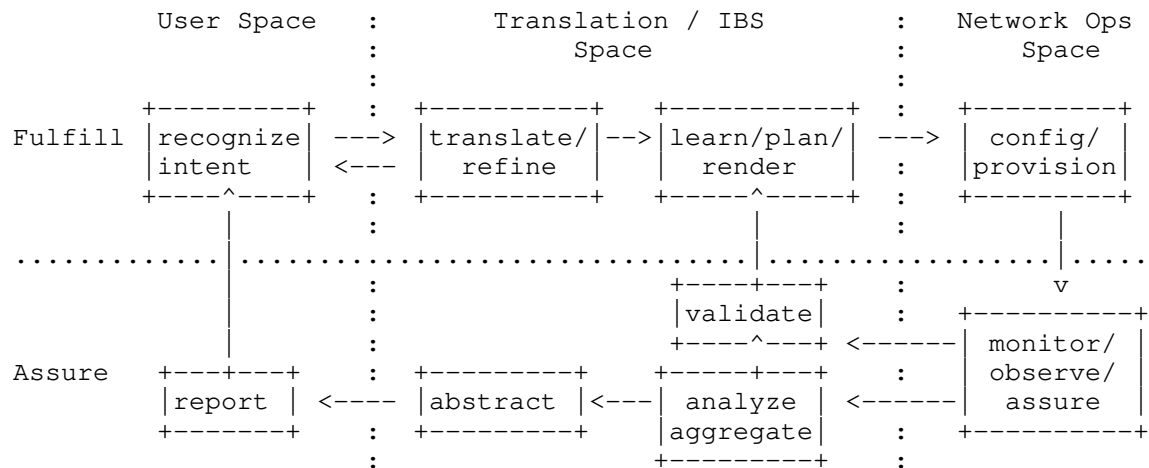


Figure 1: Intent Lifecycle

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

- o The "inner" intent control loop between IBS and Network Operations space is completely automated and does not involve any human in the loop. It involves automatic analysis and validation of intent based on observations from the network operations space, and feeding those observations into the function that plans the rendering of networking intent in order to make adjustments as needed in the configuration of the network.
- o The "outer" intent control loop involves also the user space and includes the user taking action and adjusting their intent based on feedback from the IBS.

Slight alternatives in intent lifecycles and the functions involved are conceivable. Figure 2 depicts one such alternative with an emphasis in intent fulfilment. (Todo: Intent attributes, intent states. Distinguish flow from users to network, and from network to user.)

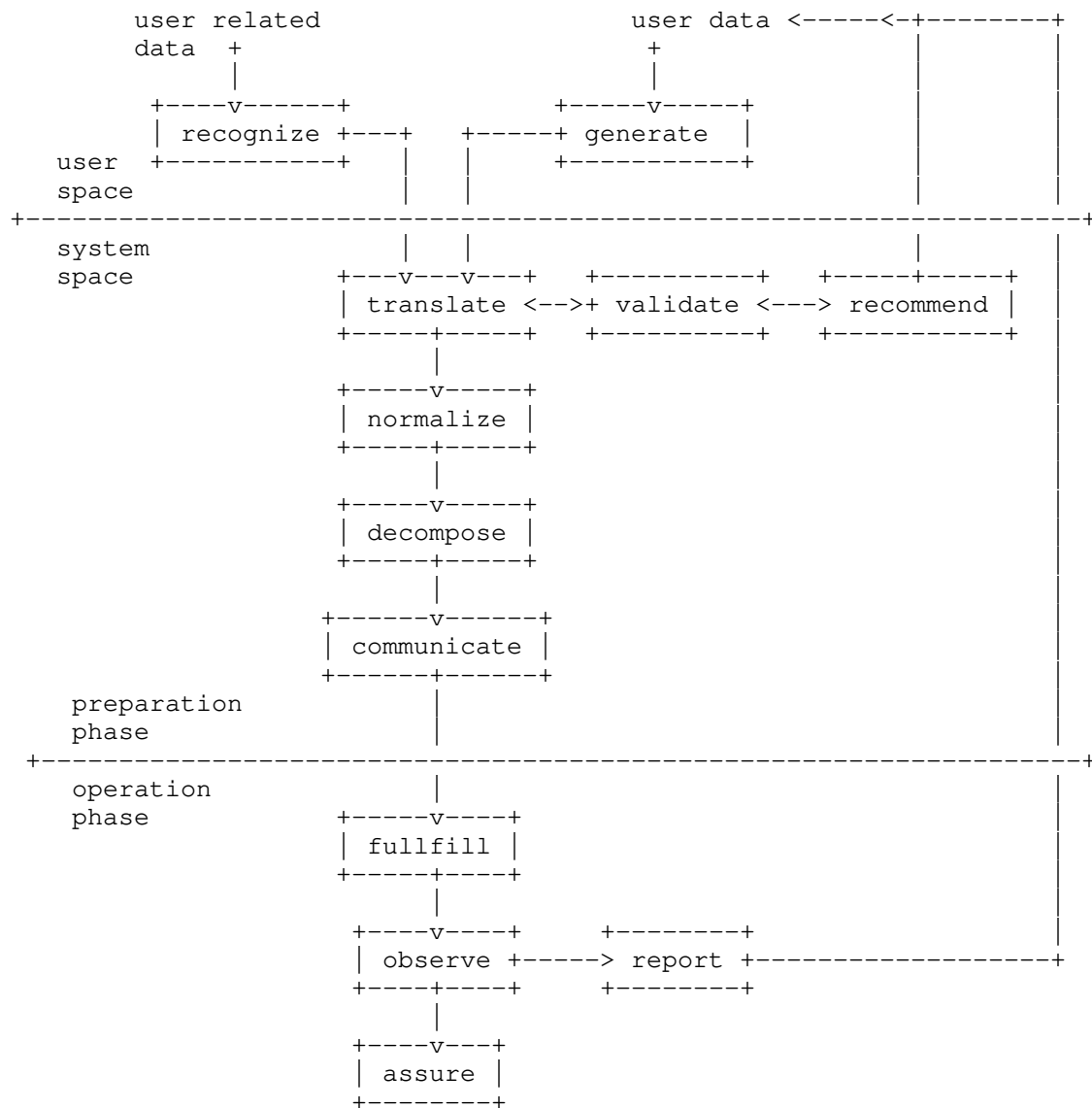


Figure 2: Intent Lifecycle (alt.)

7. Intent-Based Networking - Functionality

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

- o Intent Fulfillment provides functions and interfaces that allow users to communicate intent to the network, and that orchestrates the intent, i.e. that breaks down intent abstractions into lower-level network and device abstractions and performs or coordinates the configuration operations across the network.
- o Intent Assurance provides functions and interfaces that allow users to validate and monitor that the network is indeed adhering to and complying with intent. Control plane or lower-level management operations can cause behavior that inadvertently conflicts with intent which was orchestrated earlier. Accordingly, "intent drift" may occur. Network operators need to be able to detect when such drift occurs, or is about to occur, and be provided with the necessary functions to resolve such conflicts. This can occur by either bringing the network back into compliance, or by articulating modifications to the original intent to moderate between conflicting interests.

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

7.1. Intent Fulfillment

RBD

7.2. Intent Assurance

Ability to reason about system' state by employing closed-loop validation in the presence of an inevitable change is a fundamental property of an Intent Assurance part of an IBN system. Since service expectations are created during intent consumption and modeling phase, closed-loop intent validation should start immediately, with the service instantiation. Telemetry consumed could then be enriched with an additional context and must always be processed in context of the Intent it has been instantiated. Direct relationship between the Intent and telemetry gathered enables correlation between changes in states and the Intent and provides contextual base for reasoning about the changes.

8. Items for Discussion

Arguably, given the popularity of the term intent, its use could be broadened to encompass also known concepts ("intent-washing"). For example, it is conceivable to introduce intent-based terms for various concepts that, although already known, are related to the context of intent. Each of those terms could then designate an intent subcategory, for example:

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

Whether to do so is an item for discussion by the Research Group.

9. IANA Considerations

Not applicable

10. Security Considerations

Not applicable

11. References

11.1. 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>>.
- [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>>.

11.2. 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.
- [eTOM] TMForum, "GB 921 Business Process Framework, Release 17.0.1.", February 2018.
- [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", draft-ietf-teas-te-service-mapping-yang-02 (work in progress), September 2019.
- [Lenrow15] Lenrow, D., "Intent As The Common Interface to Network Resources, Intent Based Network Summit 2015 ONF Boulder: IntentNBI", February 2015.
- [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>>.
- [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>>.
- [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.

Authors' Addresses

Alexander Clemm
Futurewei
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: ludwig@clemm.org

Laurent Ciavaglia
Nokia
Route de Villejust
Nozay 91460
FR

Email: laurent.ciavaglia@nokia.com

Lisandro Zambenedetti Granville
Federal University of Rio Grande do Sul (UFRGS)
Av. Bento Goncalves
Porto Alegre 9500
BR

Email: granville@inf.ufrgs.br

Jeff Tantsura
Apstra, Inc.

Email: jefftant.ietf@gmail.com

Network Management Research Group
Internet-Draft
Intended status: Informational
Expires: January 9, 2020

M-S. Kim
ETRI
Y-H. Han
KoreaTech
Y-G. Hong
ETRI
July 8, 2019

Intelligent Reinforcement-learning-based Network Management
draft-kim-nmrg-rl-05

Abstract

This document presents intelligent network management based on Artificial Intelligent (AI) such as reinforcement-learning approaches. In a heterogeneous network, intelligent management with Artificial Intelligent should usually provide real-time connectivity, the type of network management with the quality of real-time data, and transmission services generated by an application service. With that reason intelligent management system is needed to support real-time connection and protection through efficient management of interfering network traffic for high-quality network data transmission in the both cloud and IoE network systems. Reinforcement-learning is one of the machine learning algorithms that can intelligently and autonomously provide to management systems over a communication network. Reinforcement-learning has developed and expanded with deep learning technique based on model-driven or data-driven technical approaches so that these trendy techniques have been widely to intelligently attempt an adaptive networking models with effective strategies in environmental disturbances over variety of networking areas. For Network AI with the intelligent and effective strategies, intent-based network (IBN) can be also considered to continuously and automatically evaluate network status under required policy for dynamic network optimization. The key element for the intent-based network is that it provides a verification of whether the represented network intent is implementable or currently implemented in the network. Additionally, this approach need to provide to take action in real time if the desired network state and actual state are inconsistent.

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 January 9, 2020.

Copyright Notice

Copyright (c) 2019 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. Conventions and Terminology	4
3. Theoretical Approaches	4
3.1. Reinforcement-learning	4
3.2. Deep-reinforcement-learning	4
3.3. Advantage Actor Critic (A2C)	5
3.4. Asynchronously Advantage Actor Critic (A3C)	5
3.5. Intent-based Network (IBN)	6
4. Reinforcement-learning-based process scenario	6
4.1. Single-agent with Single-model	7
4.2. Multi-agents Sharing Single-model	7
4.3. Adversarial Self-Play with Single-model	7
4.4. Cooperative Multi-agents with Multiple-models	7
4.5. Competitive Multi-agents with Multiple-models	8
5. Use Cases	8
5.1. Intelligent Edge-computing for Traffic Control using Deep-reinforcement-learning	8
5.2. Edge computing system in a field of Construction-site using Reinforcement-learning	8
5.3. Deep-reinforcement-learning-based remote Control system over a software-defined network	9

6. IANA Considerations	11
7. Security Considerations	11
8. References	11
8.1. Normative References	11
8.2. Informative References	11
Authors' Addresses	13

1. Introduction

Reinforcement-learning for intelligently autonomous network management, in general, is one of the challengeable methods in a dynamic complex and cluttered network environments. With the intelligent approach needs the development of computational systems in a single or large distributed networking nodes, where these environments involve limited and incomplete knowledge.

The reinforcement-learning can become a challenge-able and effective technique to transfer and share information via the global environment, as it does not require a priori-knowledge of the agent behavior or environment to accomplish its tasks [Megherbi]. Such a knowledge is usually acquired and learned repeatedly and autonomously by trial and error. The reinforcement-learning is also one of the machine learning techniques that will be adapted to the various networking environments for automatic networks [S.Jiang].

Deep-reinforcement-learning recently proposes has been extended from reinforcement-learning that can emerge as more powerful model-driven or data-driven model in a large state space, to overcome the classical behavior reinforcement-learning process. However, the classical reinforcement-learning slightly has a limitation to be adopted in networking areas, since the networking environments consist of significantly large and complex components in fields of routing configuration, optimization and system management, so that deep-reinforcement-learning can provide much more state information for learning process.[MS]

There are many different networking management problems to intelligently solve, such as connectivity, traffic management, fast Internet without latency and etc. Reinforcement-learning-based approaches can surely provide some of specific solutions with multiple cases against human operating capacities although it is a challengeable area due to a multitude of reasons such as large state space, complexity in the giving reward, difficulty in control actions, and difficulty in sharing and merging of the trained knowledge in a distributed memory node to be transferred over a communication network.[MS]

In addition, Intent-based network bridge to solve some of network problems and gaps between network business model and technical scheme. Intents should be applied to application service levels, security policies, compliance, operational processes, and other business needs. The network should constantly monitor and adjust to meet the intent in following the monitoring system. There are some of requirements to satisfy Intent-based network as following: (1) Transfer, (2) policy activation (automatically), (3) guarantee (Continuous monitoring and verification) [Cisco]. Through continuously monitoring with network data, we are able to collect network information and to analyze the collected information by artificial intelligent approach. If the analysis result shows that the new network configuration parameter needs to be changed or reconfigured by deriving the optimized value.

2. Conventions and Terminology

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

3. Theoretical Approaches

3.1. Reinforcement-learning

Reinforcement-learning is an area of machine learning concerned with how software agents should take actions in an environment so as to maximize some notion of cumulative reward.[Wikipedia] The reinforcement-learning is normally used with a reward from centralized node (the global brain), and capable of autonomous acquirement and incorporation of knowledge. It is continuously self-improving and becoming more efficient as the learning process from an agent experience to optimize management performance for autonomous learning process.[Sutton][Madera]

3.2. Deep-reinforcement-learning

Some of advanced techniques using reinforcement-learning encounter and combine with deep-learning in neural networks that has made it possible to extract high-level features from raw data in compute vision [A Krizhevsky]. There are many challenges under the deep-learning models such as convolution neural network, recurrent neural network and etc., on the reinforcement-learning approach. The benefit of the deep learning applications is that lots of networking models, but the problematic issue is complex and cluttered networking structures used with large amounts of labelled training data.

Recently, the advances in training deep neural networks to develop a novel artificial agent, termed a deep Q-network (deep-reinforcement-learning network), can be used to learn successful policies directly from high-dimensional sensory inputs using end-to-end reinforcement learning [V.Mnih].

The deep-reinforcement-learning (deep Q-network) can provide more extended and powerful scenarios to build networking models with optimized action controls, huge system states and real-time-based reward function. Moreover, the technique has a significant advantage to set highly sequential data in a large model state space. [MS] In particular, the data distribution in reinforcement-learning is able to change as learning behaviors, that is a problem for deep learning approaches assumed by a fixed underlying distribution [V. Mnih].

3.3. Advantage Actor Critic (A2C)

Advantage Actor Critic is one of the intelligent reinforcement-learning models based on policy gradient model. The intelligent approach can optimize deep neural network controller in terms of reinforcement-learning algorithms, and show that parallel actor-learners have a stabilizing effect on training and they can be allowing all of the methods to successfully train neural network controllers [Volodymyr Mnih]. Even if the prior deep-reinforcement-learning algorithm with experience replay memory tremendously has performance in challenging of the control service domains, it still needs to use more memory and computational power due to off-policy learning methods. To make up for this algorithms, a new algorithm has appeared.

The Advantage Actor Critic (consisting of actor and critic) method would implement generalized policy iteration alternating between a policy evaluation and a policy improvement step. Actor is a policy-based method that can improve the current policy for available the best next action. Critic in the value-based approach can evaluate the current policy and reduce the variance by a bootstrapping method. It is more stable and effective algorithm than the pure policy-based gradient methods.[MS]

3.4. Asynchronously Advantage Actor Critic (A3C)

Asynchronously Advantage Actor Critic is the updated algorithm based on Advantage Actor Critic. The main algorithm concept is to run multiple environments in parallel to run the agent asynchronously instead of experience replay. The parallel environment reduces the correlation of agent's data and induces each agent to experience various states so that the learning process can become a stationary process. This algorithm is a beneficial and practical point of view

since it allows learning performance even with a general multi-core CPU. In addition, it can be applied to continuous space as well as discrete action space, and also has the advantages of learning both feedforward and recurrent agent.[MS]

A3C algorithm is possibly a number of complementary improvement to the neural network architecture and it has been shown to accurately produce and estimate of Q-values by including separate streams for the state value and advantage in the network to improve both value-based and policy-based methods by making it easier for the network to represent feature coordinates [Volodymyr Mnih].

3.5. Intent-based Network (IBN)

Intent-based Network is a new technical approach that can adapt the network flexibly through configuration parameters derived from data analysis for network machine learning. Software-defined Networking (SDN) is a similar concept with Intent-based Network, however, Software-defined Networking has not yet tipped in the sector that relies on network automation. With the approach, network machine learning is integrated with network analysis, routing, wireless communications, and resource management. However, unlike the field of computer vision, which can easily acquire sufficient data, it is difficult to obtain data over a real network. Therefore, there are limitations to apply machine learning technique to network field with the data. Reinforcement Learning (RL) can diminish much attention and the importance of securing high-quality data, so that both concepts of reinforcement learning and intent-based network might solve the limitation and integrate a gap between network machine learning and network technique.

Intent-based network is also describing how to apply the setting values for network management/operation in a procedural way. For that reason, the approach is also the core of Intent processing that automatically interprets it and declares it declaratively. Even if the basic concepts of intent-based network reflects and to be announced regarding intent, there is no standardized form of Intent processing technology. While intent-based network has the advantage of providing a higher level of abstraction in network management/operation and providing ease of use, a more specific and clear definition of the technology is likely to be needed.

4. Reinforcement-learning-based process scenario

With a single agent or multiple agents trained for intelligent network management, a variety of training scenarios are possible, depending on how agents are interacted and how many models are linked

to the agents. The followings are possible RL training scenarios for network management.

4.1. Single-agent with Single-model

This is the traditional scenario of training a single agent who tries to achieve one goal related to network management. It receives all of information and rewards from a network (or a simulated network), and decides its appropriate action for the current network status.

4.2. Multi-agents Sharing Single-model

In this scenario, multiple agents share a single model and a single goal linked to the model. But, each of them is connected to an independent part of network or an independent whole network, so that they receive different information and rewards from such an independent one. The multiple agents experience differently on their connected networks. However, it does not mean their training behavior for network management will diverge. Each of their experience is used to train the single model. This scenario is a kind of parallelized version of the traditional 'Single-Agent with Single-Model' scenario, which can speed-up the RL training process and stabilize the single model's behavior.

4.3. Adversarial Self-Play with Single-model

This scenario contains two interacting agents with inverse reward functions linked to a single model. This scenario makes an agent have the perfectly matched opposing agent: itself, and trains the agent to become increasingly more skilled for network management. Inverse rewards are used to punish the opposing agent when an agent receives as positive reward, and vice versa. The two agents are linked to a single model for network management, and the model are trained and stabilized while both agents interact in a conflicting manner.

4.4. Cooperative Multi-agents with Multiple-models

In this scenario, two or more interacting agents share a common reward function linked to multiple different models for network management. In this scenario, a common goal is set up and all agents are trained to achieve the goal together that is hard to be achieved alone. Usually, each agent has access only to partial information of network status and determines an appropriate action by using its own model. Each of actions will be independently taken in order to accomplish a management task and collaboratively achieve the common goal.

4.5. Competitive Multi-agents with Multiple-models

This scenario contains two or more interacting agents with diverse reward function linked to multiple different models. In this scenario, agents will compete with one another to obtain some limited set of network resources and try to achieve their own goal. In a network, there will be tasks that have different management objectives. This leads multi-objective optimization problems, which are generally difficult to solve analytically. This scenario is suitable for solving such a multi-objective optimization problem related to network management by allowing each agent solve a single-objective problem, but complete with each other.

5. Use Cases

5.1. Intelligent Edge-computing for Traffic Control using Deep-reinforcement-learning

Edge computing is a concept that allows data from a variety of devices to be directly analyzed at the site or near the data, rather than being sent to a centralized data center such as the cloud. As such, edge computing will support data flow acceleration by processing data with low latency in real-time. In addition, by supporting efficient data processing on large amounts of data that can be processed around the source, and internet bandwidth usage will be also reduced.

Deep-reinforcement-learning would be useful technique to improve system performance in an intelligent edge-controlled service system for fast response time, reliability and security. Deep-reinforcement-learning is model-free approach so that many algorithms such as DQN, A2C and A3C can be adopted to resolve network problems in time-sensitive systems.

5.2. Edge computing system in a field of Construction-site using Reinforcement-learning

In a construction site, there are many dangerous elements such as noisy, gas leak and vibration needed by alerts, so that real-time monitoring system to detect the alerts using machine learning techniques can provide more effective solution and approach to recognize dangerous construction elements.

Representatively, to monitor these elements CCTV (closed-circuit television) should be locally and continuously broadcasting in a situation of construction site. At that time, it is in-effective and wasteful even if the CCTV is constantly broadcasting unchangeable scenes in high definition. However, the streaming should be

converted to high quality streaming data to rapidly show and detect the dangerous situation, when any alert should be detected due to the dangerous elements. To approach technically deep-reinforcement-learning can provide a solution to automatically detect these kinds of dangerous situations with prediction in an advance. It can also provide the transform data including with the high-rate streaming video and quickly prevent the other risks. Deep-reinforcement-learning is an important role to efficiently manage and monitor with the given dataset in real-time.

5.3. Deep-reinforcement-learning-based remote Control system over a software-defined network

With the nonlinear control system such as cyber physical system provides an unstable system environment with initial control state due to its nonlinear nature. In order to stably control the unstable initial state, the prior-complex mathematical control methods (Linear Quadratic Regulator, Proportional Integral Differential) are used for successful control and management, but these approaches are needed with difficult mathematical process and high-rate effort. Therefore, using deep-reinforcement-learning can surely provide more effective technical approach without difficult initial set of control states to be compared with the other methods.

The ultimate purpose of the reinforcement-learning is to interact with the environment and maximize the target reward value. Observing the state in the step and the action by the policy are performed, and the reward judge a value through the compensation given in the environment. Deep-reinforcement-learning using Convolutional Neural Network (CNN) can provide more performing learning process to make stable control and management.

As part of the system, it shows how the physical environment and the cyber environment interact with the reinforcement-learning module over a network. The actions to control the physical environment, delivered to the Enhanced Learning model based on DQN, transfer to data to the physical environment using networking communication tools as below.

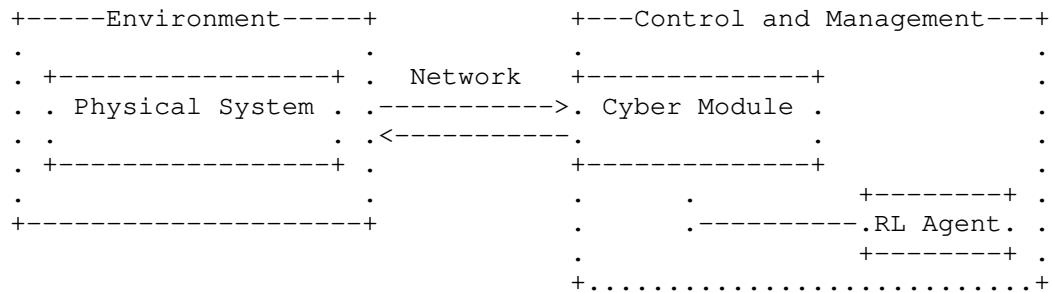


Figure 1: DRL-based Cyber Physical Management Control System

With the use-case, the reinforcement learning agent interacts with the physical remote device while exchanging network packets. The Software-defined network controller can manage the network traffic transmission, so that the system is naturally composed of a cyber environment and physical environment, and two environments closely and synchronously. [Ju-Bong]

For the intelligent traffic management in the system, software-defined networking for automation (basic concept for IBN) should be used to control and manage of connection between the cyber physical system and edge computing module. The intelligent approach consists of software that intelligently controls the network and technique that allows software to set up and control the network. The concept of can be centralized to control of network operation by software programming, centralizes switch/router control function based on existing hardware. It is possible to manage the network according to the requirements without the detailed network configuration.

In addition, software-defined networking switch is able to enable the network traffic control to be controlled and managed by software-based controllers. This approach is really similar with intent-based networking since both approaches can share the similar principle using software to run the network, however, intent-based networking offers an abstraction layer under the implemented policy and instruction across all the physical hardware within the infrastructure for automated networking. To achieve superior intent-based networking over a real network, the physical control system will be implemented to automatically manage and provide IoE edge smart traffic control service for high quality real time connection.

6. IANA Considerations

There are no IANA considerations related to this document.

7. Security Considerations

[TBD]

8. References

8.1. 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>>.

8.2. Informative References

[I-D.jiang-nmlrg-network-machine-learning]
Jiang, S., "Network Machine Learning", ID draft-jiang-nmlrg-network-machine-learning-02, October 2016.

[Megherbi]
"Megherbi, D. B., Kim, Minsuk, Madera, Manual., A Study of Collaborative Distributed Multi-Goal and Multi-agent based Systems for Large Critical Key Infrastructures and Resources (CKIR) Dynamic Monitoring and Surveillance, IEEE International Conference on Technologies for Homeland Security", 2013.

[Teiralbar]
"Megherbi, D. B., Teiralbar, A. Boulenouar, J., A Time-varying Environment Machine Learning Technique for Autonomous Agent Shortest Path Planning, Proceedings of SPIE International Conference on Signal and Image Processing, Orlando, Florida", 2001.

[Nasim]
"Nasim ArianpooEmail, Victor C.M. Leung, How network monitoring and reinforcement learning can improve tcp fairness in wireless multi-hop networks, EURASIP Journal on Wireless Communications and Networking", 2016.

[Minsuk]
"Dalila B. Megherbi and Minsuk Kim, A Hybrid P2P and Master-Slave Cooperative Distributed Multi-Agent Reinforcement Learning System with Asynchronously Triggered Exploratory Trials and Clutter-index-based Selected Sub goals, IEEE CIG Conference", 2016.

- [April] "April Yu, Raphael Palefsky-Smith, Rishi Bedi, Deep Reinforcement Learning for Simulated Autonomous Vehicle Control, Stanford University", 2016.
- [Markus] "Markus Kuderer, Shilpa Gulati, Wolfram Burgard, Learning Driving Styles for Autonomous Vehicles from Demonstration, Robotics and Automation (ICRA)", 2015.
- [Ann] "Ann Nowe, Peter Vrancx, Yann De Hauwere, Game Theory and Multi-agent Reinforcement Learning, In book: Reinforcement Learning: State of the Art, Edition: Adaptation, Learning, and Optimization Volume 12", 2012.
- [Kok-Lim] "Kok-Lim Alvin Yau, Hock Guan Goh, David Chieng, Kae Hsiang Kwong, Application of Reinforcement Learning to wireless sensor networks: models and algorithms, Published in Journal Computing archive Volume 97 Issue 11, Pages 1045-1075", November 2015.
- [Sutton] "Sutton, R. S., Barto, A. G., Reinforcement Learning: an Introduction, MIT Press", 1998.
- [Madera] "Madera, M., Megherbi, D. B., An Interconnected Dynamical System Composed of Dynamics-based Reinforcement Learning Agents in a Distributed Environment: A Case Study, Proceedings IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, Italy", 2012.
- [Al-Dayaa] "Al-Dayaa, H. S., Megherbi, D. B., Towards A Multiple-Lookahead-Levels Reinforcement-Learning Technique and Its Implementation in Integrated Circuits, Journal of Artificial Intelligence, Journal of Supercomputing. Vol. 62, issue 1, pp. 588-61", 2012.
- [Chowdappa] "Chowdappa, Aswini., Skjellum, Anthony., Doss, Nathan, Thread-Safe Message Passing with P4 and MPI, Technical Report TR-CS-941025, Computer Science Department and NSF Engineering Research Center, Mississippi State University", 1994.
- [Mnih] "V.Mnih and et al., Human-level Control Through Deep Reinforcement Learning, Nature 518.7540", 2015.

- [Stampa] "G Stamp, M Arias, etc., A Deep-reinforcement Learning Approach for Software-defined Networking Routing Optimization, cs.NI", 2017.
- [Krizhevsky] "A Krizhevsky, I Sutskever, and G Hinton, Imagenet classification with deep convolutional neural networks, In Advances in Neural Information Processing Systems, 1106-1114", 2012.
- [Volodymyr] "Volodymyr Mnih and et al., Asynchronous Methods for Deep Reinforcement Learning, ICML, arXiv:1602.01783", 2016.
- [MS] "Intelligent Network Management using Reinforcement-learning, draft-kim-nmrg-rl-03", 2018.
- [Ju-Bong] "Deep Q-Network Based Rotary Inverted Pendulum System and Its Monitoring on the EdgeX Platform, International Conference on Artificial Intelligence in Information and Communication (ICAIIC)", 2019.

Authors' Addresses

Min-Suk Kim
Etri
161 Gajeong-Dong Yuseung-Gu
Daejeon 305-700
Korea

Phone: +82 42 860 5930
Email: mskim16@etri.re.kr

Youn-Hee Han
KoreaTech
Byeongcheon-myeon Gajeon-ri, Dongnam-gu
Choenan-si, Chungcheongnam-do
330-708
Korea

Phone: +82 41 560 1486
Email: yhhan@koreatech.ac.kr

Yong-Geun Hong
ETRI
161 Gajeong-Dong Yuseung-Gu
Daejeon 305-700
Korea

Phone: +82 42 860 6557
Email: yghong@etri.re.kr

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 20, 2019

B. Liu
Huawei Technologies
B. Carpenter
Univ. of Auckland
October 17, 2018

Roadmap to a Networkless World
draft-liu-nmrg-networkless-roadmap-01

Abstract

This document aims to illustrate possible approaches to make network management and operations more autonomic in several aspects. The ultimate goal is that the network could run all by itself, so that users and administrators may feel that there is no network to take care of at all (a.k.a. "Networkless"). The approaches are described in a form of different levels (inspired by the Self-Driven Car levels). The higher the level is, the more autonomic management capabilities the network would have.

Although some specific technologies are categorized into different levels, it is not the document's intent to rank them; rather, this document is more about discussing the possible next stage and the ultimate vision. Hopefully, this document could collect people's consensus in the industry and provide guidance for future technology developments.

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 April 20, 2019.

Copyright Notice

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

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

Table of Contents

1. Introduction	2
2. Level-by-Level approach to Networkless	3
2.1. Self-Organization Levels	3
2.2. Self-Configuration Levels	4
2.3. Self-Optimization and Levels	5
2.4. Self-Diagnostic Levels	5
2.5. Self-Healing Levels	6
3. Key Capabilities to Achieve Networkless	6
3.1. Network Perception	6
3.2. Decision and Reasoning	7
3.3. Operation Interface	7
4. Possible next-step technologies	8
4.1. Self-Organization	8
4.2. Self-Configuration	9
4.3. Self-Optimization	11
4.4. Self-Diagnostic/Healing	11
5. Security Considerations	11
6. IANA Considerations	11
7. Acknowledgements	11
8. Informative References	12
Authors' Addresses	12

1. Introduction

As the network is evolving rapidly, the system is becoming more and more complex; thus managing a network is more and more challenging. It has been a common feeling in the industry that the operational expense of running networks is becoming a vital pain point. To address the management complexity challenges, there are new technologies emerging. For example, Autonomic Networking [RFC7575], which is under standardization in IETF Anima working group [Anima],

is following an approach to allow the network elements do more management related operations by themselves. SDN techniques have significantly improved the efficiency of network service delivery in some scenarios. Network function virtualization, network slicing, and the related orchestration techniques are expected to do the same. In future, the intent-based network concept, which focuses more on the operational simplicity perspective, should allow users or administrators to control the network system in a radically simple way (that is, driven by abstract intent, rather than by detailed configurations).

This document is not proposing a new technology, rather, it collects available technologies and illustrates possible future technologies and the final effect on network users or administrators. The ultimate goal is that the network could run all by itself, so that users or administrators may feel like there no network to take care of at all (a.k.a. "Networkless").

In Section 2, network management is divided into several aspects for discussion, from an administrator's perspective. In each aspect there are automation and autonomicity levels to illustrate past (Level 0), current state of art (Level 1) and possible future technologies (Level 2-4). Section 3 focuses on some common and vital capabilities the network system needs to have, in order to support the goals described in Section 2.

2. Level-by-Level approach to Networkless

2.1. Self-Organization Levels

Self-organization represents the ability of network elements to autonomically connect with each other, form domains, or even decide the topology, hierarchy or architecture.

o Level 1: LAN auto-connection

- E.g. current Ethernets can connected with each other without any configurations once the cables are connected.

o Level 2: IP auto-routing & NE auto-connection to NMS

- IGP and BGP protocols allow the routers to connect with each other autonomically, assuming prefixes are assigned to links.
- NEs automatically get connected with the NMS, current solutions includes DCN [Q: What is DCN?], Anima ACP [I-D.ietf-anima-autonomic-control-plane] etc. [Q: Mention Netconf "call home"?]

- o Level 3: Network Areas Self-Division and Key NEs election
 - E.g. IGP Area self-division; controller election
- o Level 4: Network Architecture and NE roles Self-identification
 - E.g. autonomically identify topology characteristics and divide network layers; autonomically identify roles such as access gateway, aggregation gateway, core gateway etc.

[Note] More detailed technical discussion regarding to Level-3 and Level-4 please refer to Section 4.1 .

- o Level 5: Self-Construction of Network Topologies
 - E.g. for wireless network or overlay virtual networks

2.2. Self-Configuration Levels

- o Level 1: CLI
 - remote log-in, do configs one by one
- o Level 2: NE Configs Auto-delivery
 - Administrators design detailed configurations of each NE, using NMS/Controller automatically deliver the configurations
- o Level 3/4: NE Configs Auto-Compiling
 - Administrators design network architecture and solutions, the network autonomically compiles detailed NE configurations (centrally or in a distributed manner).
 - All detailed configurations are created by software.
 - More and more machine-native configurations rather than human interfaces.

[Note] More detailed technical discussion please refer to Section 4.2 .

- o Level 5: Network Self-Orchestration
 - Administrators/Apps only input highly abstracted service requests (e.g., build a wireless backhaul network), then the network would deduce all configurations.

2.3. Self-Optimization and Levels

This sub-section focuses on traffic forwarding performance of the network, mainly include path selection and QoS related issues.

- o Level 1: Static Traffic Engineering
- o Level 2: Auto Traffic Load Balance
 - Controller dynamically adjust paths to achieve balanced traffic load and congestion, according to specific algorithms;
 - NE can achieve port-based load balancing locally
- o Level 3/4: Comprehensive SLA/QoS Self-Optimization
 - The network autonomically optimizes delay, bandwidth etc. according to Administrator's or App's requirements;
 - The network autonomically achieves measurement according to the optimization goal.
- o Level 5: Autonomous Optimization
 - The network generates optimization policies by itself, and keeps the performance at the best level;
 - Meanwhile, achieves balance between performance and cost.

2.4. Self-Diagnostic Levels

This sub-section focuses on network fault diagnostic.

- o Level 1: NMS-assisted manual diagnostic
 - Administrators use tools like ping/tracroute for mannual diagnostics
- o Level 2: Automatic Data Analysis
 - Software collects data around the whole network, and use data mining or machine learning and decision tree to aggregate alarms and analyze the cause.
- o Level 3/4: Precise Fault Location
 - Precise alarms to report the exact fault events.

- Precise location to reveal the real root cause.
- o Level 5: Fault Prediction
 - Network uses current data (e.g. bit error rates, temperature alarms) to predict failures.

2.5. Self-Healing Levels

- o Level 1: NMS-assisted manual healing
 - Administrators use NMS to manually recover the configurations or do the adjustment.
- o Level 2: Protocol-based Healing
 - Fixed healing functions built into NEs, such as BFD, and FRR etc.
- o Level 3: Programmable Healing
 - Administrators can set specific healing policies based on a set of general and abstracted rules of dealing with fault.
 - Automatic call-out of technician.
- o Level 4/5: Fault Avoidance
 - According to the prediction, avoid the fault by backup, adjust traffic, early call-out of technician, etc.

3. Key Capabilities to Achieve Networkless

3.1. Network Perception

- o Level 1: NE-based Statistics and Probe
 - E.g. NE port statistics; end to end probe. Based on known fixed topology.
- o Level 2: Network Visualization
 - Telemetry, logs/event analysis etc.
 - Display current topology.
- o Level 3: Real-time Holographic Network Data

- Network Digital Twin;
 - NE deeply sense local traffic and fault etc.
 - o Level 4: Network Modeling and Pattern Recognition
 - Comprehensive modeling for complex network problems;
 - Pattern recognition to identify current network status
 - o Level 5: Network Event/Traffic Trend Prediction
 - Based on ML trained on past observation of similar networks.
- 3.2. Decision and Reasoning
- o Level 1: Fixed Control Loops
 - The control loop functions are embedded in specific protocols/modules, such as IGP, DHCP, Anima BRSKI [I-D.ietf-anima-bootstrapping-keyinfra] , and Anima ACP [I-D.ietf-anima-autonomic-control-plane] etc.
 - o Level 2: Programmable Control Loops
 - Algorithms (in Controller or Autonomic Service Agent) for specific functions and scenarios
 - might embed some Machine Learning capabilities, or outsource ML to a central resource.
 - o Level 3: Machine Learning [Q: Maybe this should be Level 2/3?]
 - General control loops, driven by specific Intents (e.g. Intent provides the Reward definition of the reinforcement learning)
 - o Level 4: Machine Inference [Q: Maybe this should be Level 4?]
 - Configuration, optimization, diagnostic, healing policies inference
 - o Level 5: (To be filled)
- 3.3. Operation Interface
- o Level 1: CLI

- Manual management oriented interface; batch processing within a machine (e.g. Shell)
- o Level 2: NE-level Primitive API
 - Controller oriented NE-level API containing detailed configurations. (E.g. Openflow, Netconf/YANG)
- o Level 3: NE-level Declarative API
 - Orchestrator oriented NE-level declarative API
 - Orchestrator doesn't need to care about detailed NE specific configurations
- o Level 4: Network-level Declarative API
 - User/Administrator oriented declarative API, to make the network be called as a service.
- o Level 5: Machine-native Autonomous API
 - The machines would autonomously construct the content of the APIs to fulfill the need of collaboration between modules.
 - This level would likely be based on ML trained on similar networks with similar applications.

4. Possible next-step technologies

This section discusses some possible next-steps for the technologies described in Section 2. Basically, the next-steps are Level-3 or Level-4 for each aspect.

4.1. Self-Organization

Current technologies (such as the Level-2 Self-organization) can decently deal with the problem of how a device can get connected to the NOC and then get managed. After that, it still relies on human planning to properly configure the basic network connectivity, such as IP addresses, IGP etc. (This part of basic configurations is always called "underlay configurations" comparing to the overlay services.)

Thus, to simplify human work, it is expected that the system can do some "planning" work. Some critical aspects of network planning are as following, which are pre-conditions for both underlay configurations and overlay configurations.

- Routing domain division: the system can divide the devices into groups, according to some mechanisms.
- Network hierarchy recognition: the system can learn there is hierarchy in the network; and it can even recognize which are higher hierarchy and which are lower.
- Roles recognition: some device roles are directly related to the topology, such as access gateway, aggregation gateway, core gateway.

If the system can figure out the above things, then it would be much easier to create the specific configurations. The IP addresses could be assigned in a good order (e.g. from higher hierarchy to lower, keep the addresses in a certain prefix for a specific domain); the IGP could be inheritably configured according to the routing domain divisions.

4.2. Self-Configuration

This section is mostly regarding service configurations (e.g. VPN configurations).

The following figure shows a typical architecture of how current state-of-the-art technologies do configurations for services.

(preamble)

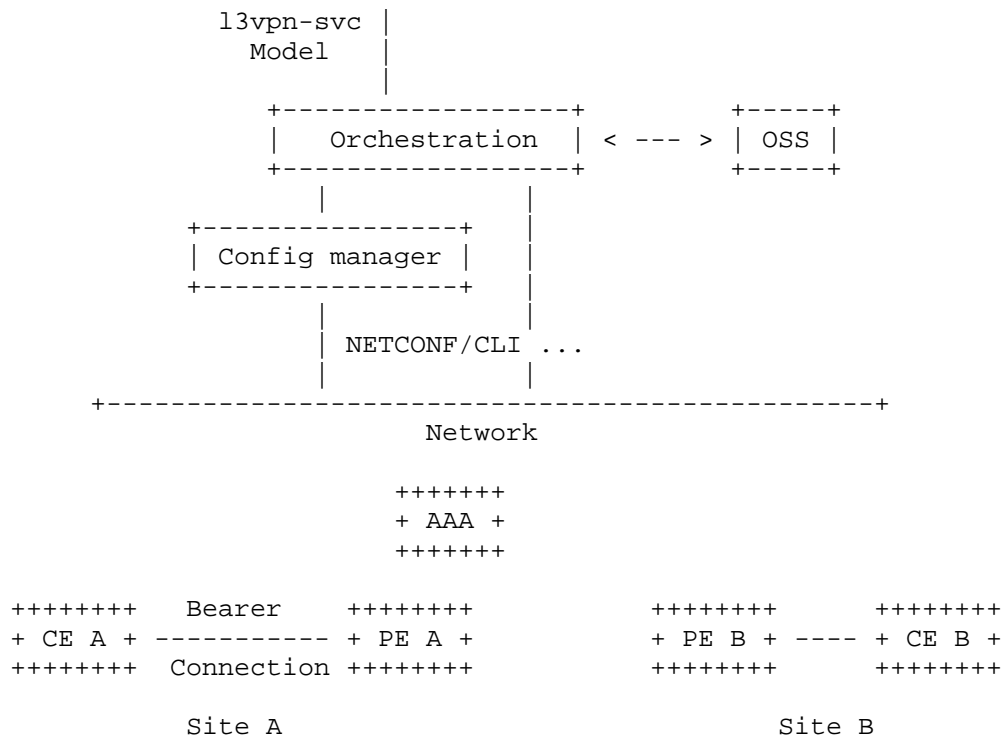


Figure 1: L3VPN Service Configuration Architecture (from RFC8299)

For this approach, there are several issues:

1. Too much details in currently defined service models, which implies
 - Cost a lot of human labor
 - The more details, the harder to achieve a unified and correct model
2. Orchestrator/Controller is hard to scale
 - Binding to specific service and underlay models; need to develop new instance when service/underlay varies
 - Need to compile each single model in each device
3. Southbound data models are hard to be unified

- Each vendor's capabilities are different
- Each operator's needs are different
- A long-term puzzle from the SNMP era, not fixed by Netconf/YANG

To address Issue-1, we'll need some easy expression of the network service, this surely fits into the Intent-based network field. (TBD.)

To address Issue-2 we might need a intermediate common layer to separate the binding between specific service-level models and device-level configurations. (TBD.)

4.3. Self-Optimization

TBD.

4.4. Self-Diagnostic/Healing

TBD.

5. Security Considerations

Security mechanisms such as firewall placement, firewall or route filtering rules, authorization to join the network, key distribution, VPN encryption policy etc. are potentially subject to all of the above. However, raising security management to Levels 3 or 4 requires great confidence that the autonomic mechanisms are themselves foolproof. It is to be expected that security management remains at Level 0, 1 or 2 longer than most other aspects. An exception is threat or DoS detection, where ML techniques should be applicable in the short term.

6. IANA Considerations

No IANA assignment is needed.

7. Acknowledgements

The initial idea of this work and the "networkless" concept were from Xiaofei Xu.

8. Informative References

- [Anima] ["https://datatracker.ietf.org/wg/anima/about/"](https://datatracker.ietf.org/wg/anima/about/).
- [I-D.ietf-anima-autonomic-control-plane]
Eckert, T., Behringer, M., and S. Bjarnason, "An Autonomic Control Plane (ACP)", draft-ietf-anima-autonomic-control-plane-18 (work in progress), August 2018.
- [I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Behringer, M., Bjarnason, S., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-16 (work in progress), June 2018.
- [I-D.ietf-anima-reference-model]
Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L., and J. Nobre, "A Reference Model for Autonomic Networking", draft-ietf-anima-reference-model-08 (work in progress), October 2018.
- [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>>.

Authors' Addresses

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

Email: leo.liubing@huawei.com

Brian Carpenter
Department of Computer Science
University of Auckland
PB 92019
Auckland 1142
New Zealand

Email: brian.e.carpenter@gmail.com

NMRG
Internet-Draft
Updates: draft-pedro-nmrg-anticipated-
adaptation-01 (if approved)
Intended status: Informational
Expires: December 30, 2018

P. Martinez-Julia, Ed.
NICT
June 28, 2018

Exploiting External Event Detectors to Anticipate Resource Requirements
for the Elastic Adaptation of SDN/NFV Systems
draft-pedro-nmrg-anticipated-adaptation-02

Abstract

The adoption of SDN/NFV technologies by current computer and network system infrastructures is constantly increasing, becoming essential for the the particular case of edge/branch network systems. The systems supported by these infrastructures require to be adapted to environment changes within a short period of time. Thus, the complexity of new systems and the speed at which management and control operations must be performed go beyond human limits. Thus, management systems must be automated. However, in several situations current automation techniques are not enough to respond to requirement changes. Here we propose to anticipate changes in the operation environments of SDN/NFV systems in response to external events and reflect it in the anticipation of the amount of resources required by those systems for their ulterior adaptaion. The final objective is to avoid service degradation or disruption while keeping close-to-optimum resource allocation to reduce monetary and operative cost as much as possible. Here we discuss how to achieve such capabilities by the integration of the Autonomic Resource Control Architecture (ARCA) to the management and operation (MANO) of NFV systems. We showcase it by building a multi-domain SDN/NFV infrastructure based on OpenStack and deploying ARCA to adapt a virtual system based on the edge/branch network concept to the operational conditions of an emergency support service, which is rarely used but that cannot leave any user unattended.

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 December 30, 2018.

Copyright Notice

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

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

Table of Contents

1. Introduction	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. The Autonomic Resource Control Architecture (ARCA)	6
4. External Event Detectors	8
5. Anticipating Requirements	8
6. Information Model	9
6.1. Tree Structure	10
6.1.1. event-payloads	10
6.1.1.1. basic	10
6.1.1.2. seismometer	11
6.1.1.3. bigdata	11
6.1.2. external-events	11
6.1.3. notifications/event	12
6.2. YANG Module	12
7. ARCA Integration With ETSI-NFV-MANO	13
7.1. Functional Integration	14
7.2. Target Experiment and Scenario	16
7.3. OpenStack Platform	18
7.4. Initial Results	19
8. Relation to Other IETF/IRTF Initiatives	22

9. IANA Considerations	22
10. Security Considerations	22
11. Acknowledgements	22
12. References	22
12.1. Normative References	23
12.2. Informative References	23
Author's Address	24

1. Introduction

The incorporation of Software Defined Networking (SDN) and Network Function Virtualization (NFV) to current infrastructures to build virtual computer and network systems is constantly increasing. The need to automate the management and control of such systems has motivated us to design the Autonomic Resource Control Architecture (ARCA), as presented in ICIN 2018 [ICIN-2018]. Automation requirements are enough justified by the increasing size and complexity of systems, which in turn are essential in the current digital world. Moreover, the particular requirements and market benefits of network virtualization have been crystallized in the uprising of SDN/NFV infrastructures. Nowadays they broad reception of the combined SDN/NFV technology supposes a huge leap towards the empowerment and homogenization of virtualization technologies. Therefore, we have modeled ARCA to fit within the reference architecture for management and orchestration of NFV elements, the Virtual Network Functions (VNFs).

Behind the scenes, NFV is based on a highly distributed and network empowered version of the well-known Cloud infrastructures and platforms, also complemented by their centralized counterparts. This takes to virtual networks the high degree of flexibility already found for computer systems. It is highly desirable at the time NFV is being exploited by many organizations to build their private infrastructures, as well as by network service providers to build the services they later commercialize. However, to actually exploit the potential monetary and operative cost reduction that is associated to such infrastructures, the amount of resources used by production services must be kept close to the optimum, so the physical resources are exploited as much as possible.

The fast detection of changes in the requirements of the virtual systems deployed on the aforementioned SDN/NFV infrastructures, and the consequent adaptation of allocated resources to the new situations, becomes essential to actually exploit their cost and operative benefits, while also avoiding service unresponsiveness due to underlying resource overloading. It is widely accepted that the size and complexity of systems and services makes it difficult for humans to accomplish such task within their objective time

boundaries. Therefore, they must be automated. Luckily, the architecture and underlying platforms supporting the SDN/NFV technologies enable the required automation. In fact, some solutions already exist to perform several batched or scripted tasks without human intervention. However, those solutions still have high dependences on low-level human involvement. This remarks the challenge found in control and management automation, which is continuously revised and enlarged.

ARCA provides as a small step towards the resolution of the aforementioned problem. It advances the State of the Art in automation of resource control and management by providing a supervised but autonomous mechanism that reduces the time required to perform corrective and/or adaptive changes in virtual computer and network systems from hours/minutes to seconds/milliseconds. Moreover, it is able to take advantage of the event notifications provided by external detectors to anticipate the amount of resources that the controlled SDN/NFV system will require in response to such event. We propose to bring such benefit to the reference architecture promoted by ETSI for the management and orchestration of NFV services (see ETSI-NFV-MANO [ETSI-NFV-MANO]) by integrating ARCA as the Virtual Infrastructure Manager (VIM). We showcase this proposal by discussing the evaluation results obtained by ARCA when running on a real and physical experimentation infrastructure based on OpenStack [OPENSTACK]. We thus justify the need to adapt the interfaces supported by the NFV-MANO to include real-world event detectors, which are external to the virtualization platform and virtual resources.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in 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 should 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. Although network functions can be implemented in a central computing facility, they can 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. 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.

4. External Event Detectors

As mentioned above, current mechanisms used to achieve 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.

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.

5. Anticipating 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.

Here we propose to integrate ARCA with NFV-MANO to take advantage of the notifications provided by the aforementioned external event detectors, by correlating them to the target amount of resources required by the managed system and enforcing the necessary

adaptations 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.

6. Information Model

In this section we introduce the basic model needed to support the implementation of the anticipation algorithm. 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.

6.1. Tree Structure

```
module: ietf-nmrg-nict-resource-anticipation
  +--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.

6.1.1. 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.

6.1.1.1. 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 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.song-ntf]).

6.1.1.2. 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.

6.1.1.3. 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.

6.1.2. 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.

6.1.3. 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.

6.2. YANG Module

```

.

module ietf-nmrg-nict-resource-anticipation {
  namespace "urn:ietf:params:xml:ns:yang:ietf-nmrg-nict-resource-anticipation";
  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;
    }
  }
}
```

.

7. 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.

7.1. 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.song-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.

7.2. 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.

7.3. 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.

7.4. 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 assignation (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 limitations 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.

8. Relation to Other IETF/IRTF Initiatives

TBD

9. IANA Considerations

This memo includes no request to IANA.

10. Security Considerations

The major security concerns of the integration of external event detectors and ARCA to manage 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 the current solution is inherited and thus determined by the security measures established by the systems conforming such planes.

11. Acknowledgements

TBD

12. References

12.1. 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>>.

12.2. 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., Lucena, J., Ameigeiras, P., Lopez, D., and L. Contreras, "COMS Architecture", draft-geng-coms-architecture-02 (work in progress), March 2018.
- [I-D.qiang-coms-netslicing-information-model]
Qiang, L., Galis, A., Geng, L., kiran.makhijani@huawei.com, k., Martinez-Julia, P., Flinck, H., and X. Foy, "Technology Independent Information Model for Network Slicing", draft-qiang-coms-netslicing-information-model-02 (work in progress), January 2018.

[I-D.song-ntf]

Song, H., Zhou, T., and Z. Li, "Toward a Network Telemetry Framework", draft-song-ntf-01 (work in progress), March 2018.

[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.

Author's Address

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

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 3, 2019

H. Song, Ed.
T. Zhou
ZB. Li
Huawei
G. Fioccola
Telecom Italia
ZQ. Li
China Mobile
P. Martinez-Julia
NICT
L. Ciavaglia
Nokia
A. Wang
China Telecom
July 2, 2018

Toward a Network Telemetry Framework
draft-song-ntf-02

Abstract

This document suggests the necessity of an architectural framework for network telemetry in order to meet the current and future network operation requirements. The defining characteristics of network telemetry shows a clear distinction from the conventional network OAM concept; hence the network telemetry demands new techniques and protocols. This document clarifies the terminologies and classifies the categories and components of a network telemetry framework. The requirements, challenges, existing solutions, and future directions are discussed for each category. The network telemetry framework and the taxonomy help to set a common ground for the collection of related works and put future technique and standard developments into perspective.

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 January 3, 2019.

Copyright Notice

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

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

Table of Contents

1. Motivation	3
1.1. Use Cases	3
1.2. Challenges	5
1.3. Glossary	5
1.4. Network Telemetry	6
2. The Necessity of a Network Telemetry Framework	8
3. Network Telemetry Framework	9
3.1. Existing Works Mapped in the Framework	11
3.2. Management Plane Telemetry	12
3.2.1. Requirements and Challenges	12
3.2.2. Push Extensions for NETCONF	13
3.2.3. gRPC Network Management Interface	13
3.3. Control Plane Telemetry	14
3.3.1. Requirements and Challenges	14
3.3.2. BGP Monitoring Protocol	14
3.4. Data Plane Telemetry	15
3.4.1. Requirements and Challenges	15
3.4.2. Technique Classification	16
3.4.3. The IPFPM technology	16
3.4.4. Dynamic Network Probe	18
3.4.5. IP Flow Information Export (IPFIX) protocol	18

3.4.6. In-Situ OAM	18
3.5. External Data and Event Telemetry	19
3.5.1. Requirements and Challenges	19
4. Security Considerations	20
5. IANA Considerations	20
6. Contributors	20
7. Acknowledgments	20
8. References	20
8.1. Normative References	20
8.2. Informative References	20
Authors' Addresses	23

1. Motivation

The advance of AI/ML technologies gives networks an unprecedented opportunity to realize network autonomy with closed control loops. An intent-driven autonomous network is the logical next step for network evolution following SDN, aiming to reduce (or even eliminate) human labor, make the most efficient use of network resources, and provide better services more aligned with customer requirements. Although we still have a long way to reach the ultimate goal, the journey has started nevertheless.

The storage and computing technologies are already mature enough to be able to retain and process a huge amount of data and make real-time inference. Tools based on machine learning technologies and big data analytics are powerful in detecting and reacting on network faults, anomalies, and policy violations. In turn, the network policy updates for planning, intrusion prevention, optimization, and self-healing can be applied. Some tools can even predict future events based on historical data.

However, the networks fail to keep pace with such data need. The current network architecture, protocol suite, and system design are not ready yet to provide enough quality data. In the remaining of this section, first we identify a few key network operation use cases that network operators need the most. These use cases are also the essential functions of the future autonomous networks. Next, we show why the current network OAM techniques and protocols are not sufficient to meet the requirements of these use cases. The discussion underlines the need of a new brood of techniques and protocols which we put under an umbrella term - network telemetry.

1.1. Use Cases

All these use cases involves the data extracted from the network data plane and sometimes from the network control plane and management plane.

Intent and Policy Compliance: Network policies are the rules that constraint the services for network access, provide differentiate within a service, or enforce specific treatment on the traffic. For example, a service function chain is a policy that requires the selected flows to pass through a set of network functions in order. An intents is a high-level abstract policy which requires a complex translation and mapping process before being applied on networks. While a policy is enforced, the compliance needs to be verified and monitored continuously.

SLA Compliance: A Service-Level Agreement (SLA) defines the level of service a user expects from a network operator, which include the metrics for the service measurement and remedy/penalty procedures when the service level misses the agreement. Users need to check if they get the service as promised and network operators need to evaluate how they can deliver the services that can meet the SLA.

Root Cause Analysis: Network failure often involves a sequence of chained events and the source of the failure is not straightforward to identify, especially when the failure is sporadic. While machine learning or other data analytics technologies can be used for root cause analysis, it up to the network to provide all the relevant data for analysis.

Load Balancing, Traffic Engineering, and Network Planning: Network operators are motivated to optimize their network utilization for better ROI or lower CAPEX, as well as differentiation across services and/or users of a given service. The first step is to know the real-time network conditions before applying policies to steer the user traffic or adjust the load balancing algorithm. In some cases network micro-bursts need to be detected in a very short time-frame so that fine grained traffic control can be applied to avoid possible network congestion. The long term network capacity planning and topology augmentation also rely on the accumulated data of the network operation.

Event Tracking and Prediction: Network visibility is critical for a healthy network operation. Numerous network events are of interest to network operators. For example, Network operators always want to learn where and why packets are dropped for an application flow. They also want to be warned by some early signs that some component is going to fail so the proper fix or replacement can be made in time.

1.2. Challenges

The conventional OAM techniques, as described in [RFC7276], are not sufficient to support the above use cases for the following reasons:

- o Most use cases need to continuously monitor the network and dynamically refine the data collection in real-time and interactively. The poll-based low-frequency data collection is ill-suited for these applications. Streaming data directly pushed from the data source is preferred.
- o Various data is needed from any place ranging from the packet processing engine to the QoS traffic manager. Traditional data plane devices cannot provide the necessary probes. An open and programmable data plane is therefore needed.
- o Many application scenarios need to correlate data from multiple sources (e.g., from distributed nodes or from different network plane). A piecemeal solution is often lacking the capability to consolidate the data from multiple sources. The composition of a complete solution, as partly proposed by ARCA [I-D.pedro-nmrg-anticipated-adaptation], will be empowered and guided by a comprehensive framework.
- o The passive measurement techniques can either consume too much network resources and render too much redundant data, or lead to inaccurate results. The active measurement techniques are indirect, and they can interfere with the user traffic. We need techniques that can collect direct and on-demand data from user traffic.

1.3. Glossary

Before further discussion, we list some key terminology and acronyms used in this documents. We make an intended distinction between network telemetry and network OAM.

AI: Artificial Intelligence. Use machine-learning based technologies to automate network operation.

BMP: BGP Monitoring Protocol

DNP: Dynamic Network Probe

DPI: Deep Packet Inspection

gNMI: gPRC Network Management Interface

gRPC: gRPC Remote Procedure Call

IDN: Intent-Driven Network

IPFIX: IP Flow Information Export Protocol

IPFPM: IP Flow Performance Measurement

IOAM: In-situ OAM

NETCONF: Network Configuration Protocol

Network Telemetry: A general term for a new brood of network visibility techniques and protocols, with the characteristics defined in this document. Network telemetry enables smooth evolution toward intent-driven autonomous networks.

NMS: Network Management System

OAM: Operations, Administration, and Maintenance. A group of network management functions that provide network fault indication, fault localization, performance information, and data and diagnosis functions. Most conventional network monitoring techniques and protocols belong to network OAM.

SNMP: Simple Network Management Protocol

YANG: A data modeling language for NETCONF

YANG FSM: A YANG model to define device side finite state machine

YANG PUSH: A method to subscribe pushed data from remote YANG datastore

1.4. Network Telemetry

For a long time, network operators have relied upon protocols such as SNMP [RFC1157] to monitor the network. SNMP can only provide limited information about the network. Since SNMP is poll-based, it incurs low data rate and high processing overhead. Such drawbacks make SNMP unsuitable for today's automatic network applications.

Network telemetry has emerged as a mainstream technical term to refer to the newer techniques of data collection and consumption, distinguishing itself from the convention techniques for network OAM. It is expected that network telemetry can provide the necessary network visibility for autonomous networks, address the shortcomings

of conventional OAM techniques, and allow for the emergence of new techniques bearing certain characteristics.

One key difference between the network telemetry and the network OAM is that the network telemetry assumes an intelligent machine in the center of a closed control loop, while the network OAM assumes the human network operators in the middle of an open control loop. The network telemetry can directly trigger the automated network operation; The conventional OAM tools only help human operators to monitor and diagnose the networks and guide manual network operations. The different assumptions lead to very different techniques.

Although the network telemetry techniques are just emerging and subject to continuous evolution, several defining characteristics of network telemetry have been well accepted:

- o Push and Streaming: Instead of polling data from network devices, the telemetry collector subscribes to the streaming data pushed from the data source in network devices.
- o Volume and Velocity: The telemetry data is intended to be consumed by machine rather than by human. Therefore, the data volume is huge and the processing is often in realtime.
- o Normalization and Unification: Telemetry aims to address the overall network automation needs. The piecemeal solutions offered by the conventional OAM approach are no longer suitable. Efforts need to be made to normalize the data representation and unify the protocols.
- o Model-based: The data is model-based which allows applications to configure and consume data with ease.
- o Data Fusion: The data for a single application can come from multiple data sources (e.g., cross domain, cross device, and cross layer) and needs to be correlated to take effect.
- o Dynamic and Interactive: Since the network telemetry means to be used in a closed control loop for network automation, it needs to run continuously and adapt to the dynamic and interactive queries from the network operation controller.

In addition, the ideal network telemetry solution should also support the following features:

- o In-Network Customization: The data can be customized in network at run-time to cater to the specific need of applications. This

needs the support of a programmable data plane which allows probes to be deployed at flexible locations.

- o Direct Data Plane Export: The data originated from data plane can be directly exported to the data consumer for efficiency, especially when the data bandwidth is large and the real-time processing is required.
- o In-band Data Collection: In addition to the passive and active data collection approaches, the new hybrid approach allows to directly collect data for any target flow on its entire forwarding path.
- o Non-intrusive: The telemetry system should not fall into the trap of the "observer effect". That is, it should not change the network behavior or affect the forwarding performance.

2. The Necessity of a Network Telemetry Framework

Big data analytics and machine-learning based AI technologies are applied for network operation automation, relying on abundant data from networks. The single-sourced and static data acquisition cannot meet the data requirements. It is desirable to have a framework that integrates multiple telemetry approaches from different layers, and allows flexible combinations for different applications. The framework will benefit application development for the following reasons.

- o The future autonomous networks will require a holistic view on network visibility. All the use cases and applications need to be supported uniformly and coherently under a single intelligent agent. Therefore, the protocols and mechanisms should be consolidated into a minimum yet comprehensive set. A telemetry framework can help to normalize the technique developments.
- o Network visibility presents multiple viewpoints. For example, the device viewpoint takes the network infrastructure as the monitoring object from which the network topology and device status can be acquired; the traffic viewpoint takes the flows or packets as the monitoring object from which the traffic quality and path can be acquired. An application may need to switch its viewpoint during operation. It may also need to correlate a service and its network experience to acquire the comprehensive information.
- o Applications require network telemetry to be elastic in order to efficiently use the network resource and reduce the performance impact. Routine network monitoring covers the entire network with

low data sampling rate. When issues arise or trends emerge, the telemetry data source can be modified and the data rate can be boosted.

- o Efficient data fusion is critical for applications to reduce the overall quantity of data and improve the accuracy of analysis.

So far, some telemetry related work has been done within IETF. However, this work is fragmented and scattered in different working groups. The lack of coherence makes it difficult to assemble a comprehensive network telemetry system and causes repetitive and redundant work.

A formal network telemetry framework is needed for constructing a working system. The framework should cover the concepts and components from the standardization perspective. This document clarifies the layers on which the telemetry is exerted and decomposes the telemetry system into a set of distinct components that the existing and future work can easily map to.

3. Network Telemetry Framework

Telemetry can be applied on the data plane, the control plane, and the management plane in a network, as well as other sources out of the network, as shown in Figure 1.

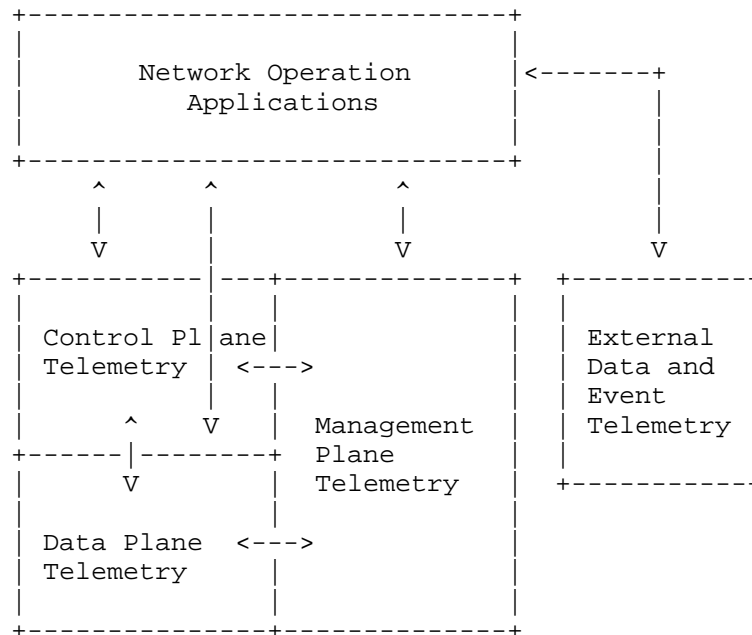


Figure 1: Layer Category of the Network Telemetry Framework

Note that the interaction with the network operation applications can be indirect. For example, in the management plane telemetry, the management plane may need to acquire data from the data plane. On the other hand, an application may involve more than one plane simultaneously. For example, an SLA compliance application may require both the data plane telemetry and the control plane telemetry.

At each plane, the telemetry can be further partitioned into five distinct components:

Data Source: Determine where the original data is acquired. The data source usually just provides raw data which needs further processing. A data source can be considered a probe. A probe can be statically installed or dynamically installed.

Data Subscription: Determine the protocol and channel for applications to acquire desired data. Data subscription is also responsible to define the desired data that might not be directly available from data sources. The subscription data can be described by a model. The model can be statically installed or dynamically installed.

Data Generation: The original data needs to be processed, encoded, and formatted in network devices to meet application subscription requirements. This may involve in-network computing and processing on either the fast path or the slow path in network devices.

Data Export: Determine how the ready data are delivered to applications.

Data Analysis and Storage: In this final step, data is consumed by applications or stored for future reference. Data analysis can be interactive. It may initiate further data subscription.

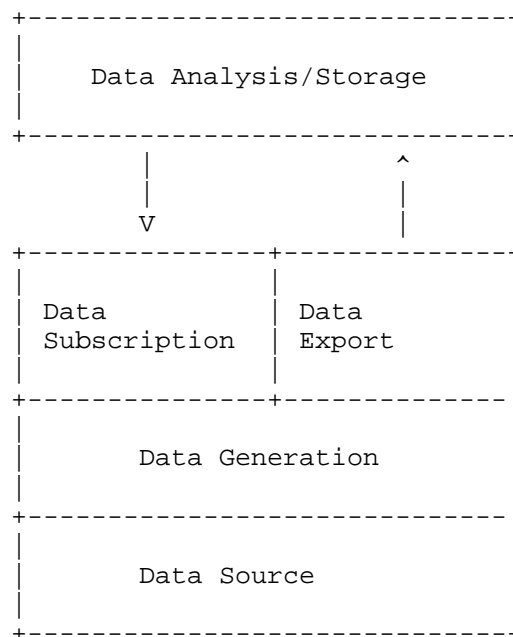


Figure 2: Components in the Network Telemetry Framework

Since most existing standard-related work belongs to the first four components, in the remainder of the document, we focus on these components only.

3.1. Existing Works Mapped in the Framework

The following table provides a non-exhaustive list of existing works (mainly published in IETF and with the emphasis on the latest new technologies) and shows their positions in the framework.

	Management Plane	Control Plane	Data Plane
Data Source	YANG Data Store	Control Proto. Network State	Flow/Packet Statistics States DPI
Data Subscribe	gRPC YANG PUSH	NETCONF/YANG BGP	NETCONF/YANG YANG FSM
Data Generation	Soft DNP	Soft DNP	In-situ OAM IPFPM Hard DNP
Data Export	gRPC YANG PUSH UDP	BMP	IPFIX UDP

Figure 3: Existing Work

3.2. Management Plane Telemetry

3.2.1. Requirements and Challenges

The management plane of the network element interacts with the Network Management System (NMS), and provides information such as performance data, network logging data, network warning and defects data, and network statistics and state data. Some legacy protocols are widely used for the management plane, such as SNMP and Syslog, but these protocols do not meet the requirements of the automatic network operation applications.

New management plane telemetry protocols should consider the following requirements:

Convenient Data Subscription: An application should have the freedom to choose the data export means such as the data types and the export frequency.

Structured Data: For automatic network operation, machines will replace human for network data comprehension. The schema languages such as YANG can efficiently describe structured data and normalize data encoding and transformation.

High Speed Data Transport: In order to retain the information, a server needs to send a large amount of data at high frequency. Compact encoding formats are needed to compress the data and improve the data transport efficiency. The push mode, by replacing the poll mode, can also reduce the interactions between clients and servers, which help to improve the server's efficiency.

3.2.2. Push Extensions for NETCONF

NETCONF [RFC6241] is one popular network management protocol, which is also recommended by IETF. Although it can be used for data collection, NETCONF is good at configurations. YANG Push [I-D.ietf-netconf-yang-push] extends NETCONF and enables subscriber applications to request a continuous, customized stream of updates from a YANG datastore. Providing such visibility into changes made upon YANG configuration and operational objects enables new capabilities based on the remote mirroring of configuration and operational state. Moreover, distributed data collection mechanism [I-D.zhou-netconf-multi-stream-originators] via UDP based publication channel [I-D.ietf-netconf-udp-pub-channel] provides enhanced efficiency for the NETCONF based telemetry.

3.2.3. gRPC Network Management Interface

gRPC Network Management Interface (gNMI) [I-D.openconfig-rtgwg-gnmi-spec] is a network management protocol based on the gRPC [I-D.kumar-rtgwg-grpc-protocol] RPC (Remote Procedure Call) framework. With a single gRPC service definition, both configuration and telemetry can be covered. gRPC is an HTTP/2 [RFC7540] based open source micro service communication framework. It provides a number of capabilities that makes it well-suited for network telemetry, including:

- o Full-duplex streaming transport model combined with a binary encoding mechanism provided further improved telemetry efficiency.
- o gRPC provides higher-level features consistency across platforms that common HTTP/2 libraries typically do not. This characteristic is especially valuable for the fact that telemetry data collectors normally reside on a large variety of platforms.
- o The built-in load-balancing and failover mechanism.

3.3. Control Plane Telemetry

3.3.1. Requirements and Challenges

The control plane telemetry refers to the health condition monitoring of different network protocols, which covers Layer 2 to Layer 7. Keeping track of the running status of these protocols is beneficial for detecting, localizing, and even predicting various network issues, as well as network optimization, in real-time and in fine granularities.

One of the most challenging problems for the control plane telemetry is how to correlate the E2E Key Performance Indicators (KPI) to a specific layer's KPIs. For example, an IPTV user may describe his User Experience (UE) by the video fluency and definition. Then in case of an unusually poor UE KPI or a service disconnection, it is non-trivial work to delimit and localize the issue to the responsible protocol layer (e.g., the Transport Layer or the Network Layer), the responsible protocol (e.g., ISIS or BGP at the Network Layer), and finally the responsible device(s) with specific reasons.

Traditional OAM-based approaches for control plane KPI measurement include PING (L3), Tracert (L3), Y.1731 (L2) and so on. One common issue behind these methods is that they only measure the KPIs instead of reflecting the actual running status of these protocols, making them less effective or efficient for control plane troubleshooting and network optimization. An example of the control plane telemetry is the BGP monitoring protocol (BMP), it is currently used to monitoring the BGP routes and enables rich applications, such as BGP peer analysis, AS analysis, prefix analysis, security analysis, and so on. However, the monitoring of other layers, protocols and the cross-layer, cross-protocol KPI correlations are still in their infancies (e.g., the IGP monitoring is missing), which require substantial further research.

3.3.2. BGP Monitoring Protocol

BGP Monitoring Protocol (BMP) [RFC7854] is used to monitor BGP sessions and intended to provide a convenient interface for obtaining route views.

The BGP routing information is collected from the monitored device(s) to the BMP monitoring station by setting up the BMP TCP session. The BGP peers are monitored by the BMP Peer Up and Peer Down Notifications. The BGP routes (including Adjacency_RIB_In [RFC7854], Adjacency_RIB_out [I-D.ietf-grow-bmp-adj-rib-out], and Local_Rib [I-D.ietf-grow-bmp-local-rib] are encapsulated in the BMP Route Monitoring Message and the BMP Route Mirroring Message, in the form

of both initial table dump and real-time route update. In addition, BGP statistics are reported through the BMP Stats Report Message, which could be either timer triggered or event driven. More BMP extensions can be explored to enrich the applications of BGP monitoring.

3.4. Data Plane Telemetry

3.4.1. Requirements and Challenges

An effective data plane telemetry system relies on the data that the network device can expose. The data's quality, quantity, and timeliness must meet some stringent requirements. This raises some challenges to the network data plane devices where the first hand data originate.

- o A data plane device's main function is user traffic processing and forwarding. While supporting network visibility is important, the telemetry is just an auxiliary function and it should not impede normal traffic processing and forwarding (i.e., the performance is not lowered and the behavior is not altered due to the telemetry functions).
- o The network operation applications requires end-to-end visibility from various sources, which results in a huge volume of data. However, the sheer data quantity should not stress the network bandwidth, regardless of the data delivery approach (i.e., through in-band or out-of-band channels).
- o The data plane devices must provide the data in a timely manner with the minimum possible delay. Long processing, transport, storage, and analysis delay can impact the effectiveness of the control loop and even render the data useless.
- o The data should be structured and labeled, and easy for applications to parse and consume. At the same time, the data types needed by applications can vary significantly. The data plane devices need to provide enough flexibility and programmability to support the precise data provision for applications.
- o The data plane telemetry should support incremental deployment and work even though some devices are unaware of the system. This challenge is highly relevant to the standards and legacy networks.

The industry has agreed that the data plane programmability is essential to support network telemetry. Newer data plane chips are

all equipped with advanced telemetry features and provide flexibility to support customized telemetry functions.

3.4.2. Technique Classification

There can be multiple possible dimensions to classify the data plane telemetry techniques.

Active and Passive: The active and passive methods (as well as the hybrid types) are well documented in [RFC7799]. The passive methods include TCPDUMP, IPFIX [RFC7011], sflow, and traffic mirror. These methods usually have low data coverage. The bandwidth cost is very high in order to improve the data coverage. On the other hand, the active methods include Ping, Traceroute, OWAMP [RFC4656], and TWAMP [RFC5357]. These methods are intrusive and only provide indirect network measurement results. The hybrid methods, including in-situ OAM [I-D.brockners-inband-oam-requirements], IPFPM [RFC8321], and Multipoint Alternate Marking [I-D.fioccola-ippm-multipoint-alt-mark], provide a well-balanced and more flexible approach. However, these methods are also more complex to implement.

In-Band and Out-of-Band: The telemetry data, before being exported to some collector, can be carried in user packets. Such methods are considered in-band (e.g., in-situ OAM [I-D.brockners-inband-oam-requirements]). If the telemetry data is directly exported to some collector without modifying the user packets, Such methods are considered out-of-band (e.g., postcard-based INT). It is possible to have hybrid methods. For example, only the telemetry instruction or partial data is carried by user packets (e.g., IPFPM [RFC8321]).

E2E and In-Network: Some E2E methods start from and end at the network end hosts (e.g., Ping). The other methods work in networks and are transparent to end hosts. However, if needed, the in-network methods can be easily extended into end hosts.

Flow, Path, and Node: Depending on the telemetry objective, the methods can be flow-based (e.g., in-situ OAM [I-D.brockners-inband-oam-requirements]), path-based (e.g., Traceroute), and node-based (e.g., IPFIX [RFC7011]).

3.4.3. The IPFPM technology

The Alternate Marking method is efficient to perform packet loss, delay, and jitter measurements both in an IP and Overlay Networks, as

presented in IPFPM [RFC8321] and [I-D.fioccola-ippm-multipoint-alt-mark].

This technique can be applied to point-to-point and multipoint-to-multipoint flows. Alternate Marking creates batches of packets by alternating the value of 1 bit (or a label) of the packet header. These batches of packets are unambiguously recognized over the network and the comparison of packet counters for each batch allows the packet loss calculation. The same idea can be applied to delay measurement by selecting ad hoc packets with a marking bit dedicated for delay measurements.

Alternate Marking method needs two counters each marking period for each flow under monitor. For instance, by considering n measurement points and m monitored flows, the order of magnitude of the packet counters for each time interval is $n*m*2$ (1 per color).

Since networks offer rich sets of network performance measurement data (e.g packet counters), traditional approaches run into limitations. One reason is the fact that the bottleneck is the generation and export of the data and the amount of data that can be reasonably collected from the network. In addition, management tasks related to determining and configuring which data to generate lead to significant deployment challenges.

Multipoint Alternate Marking approach, described in [I-D.fioccola-ippm-multipoint-alt-mark], aims to resolve this issue and makes the performance monitoring more flexible in case a detailed analysis is not needed.

An application orchestrates network performance measurements tasks across the network to allow an optimized monitoring and it can calibrate how deep can be obtained monitoring data from the network by configuring measurement points roughly or meticulously.

Using Alternate Marking, it is possible to monitor a Multipoint Network without examining in depth by using the Network Clustering (subnetworks that are portions of the entire network that preserve the same property of the entire network, called clusters). So in case there is packet loss or the delay is too high the filtering criteria could be specified more in order to perform a detailed analysis by using a different combination of clusters up to a per-flow measurement as described in IPFPM [RFC8321].

In summary, an application can configure initially an end to end monitoring between ingress points and egress points of the network. If the network does not experiment issues, this approximate monitoring is good enough and is very cheap in terms of network

resources. But, in case of problems, the application becomes aware of the issues from this approximate monitoring and, in order to localize the portion of the network that has issues, configures the measurement points more exhaustively. So a new detailed monitoring is performed. After the detection and resolution of the problem the initial approximate monitoring can be used again.

3.4.4. Dynamic Network Probe

Hardware based Dynamic Network Probe (DNP) [I-D.song-opsawg-dnp4iq] provides a programmable means to customize the data that an application collects from the data plane. A direct benefit of DNP is the reduction of the exported data. A full DNP solution covers several components including data source, data subscription, and data generation. The data subscription needs to define the custom data which can be composed and derived from the raw data sources. The data generation takes advantage of the moderate in-network computing to produce the desired data.

While DNP can introduce unforeseeable flexibility to the data plane telemetry, it also faces some challenges. It requires a flexible data plane that can be dynamically reprogrammed at run-time. The programming API is yet to be defined.

3.4.5. IP Flow Information Export (IPFIX) protocol

Traffic on a network can be seen as a set of flows passing through network elements. IP Flow Information Export (IPFIX) [RFC7011] provides a means of transmitting traffic flow information for administrative or other purposes. A typical IPFIX enabled system includes a pool of Metering Processes collects data packets at one or more Observation Points, optionally filters them and aggregates information about these packets. An Exporter then gathers each of the Observation Points together into an Observation Domain and sends this information via the IPFIX protocol to a Collector.

3.4.6. In-Situ OAM

Traditional passive and active monitoring and measurement techniques are either inaccurate or resource-consuming. It is preferable to directly acquire data associated with a flow's packets when the packets pass through a network. In-situ OAM (ioAM) [I-D.brockners-inband-oam-requirements], a data generation technique, embeds a new instruction header to user packets and the instruction directs the network nodes to add the requested data to the packets. Thus, at the path end the packet's experience on the entire forwarding path can be collected. Such firsthand data is invaluable to many network OAM applications.

However, iOAM also faces some challenges. The issues on performance impact, security, scalability and overhead limits, encapsulation difficulties in some protocols, and cross-domain deployment need to be addressed.

3.5. External Data and Event Telemetry

Events that occur outside the boundaries of the network system are another important source of telemetry information. Correlating both internal telemetry data and external events with the requirements of network systems, as presented in Exploiting External Event Detectors to Anticipate Resource Requirements for the Elastic Adaptation of SDN/NFV Systems [I-D.pedro-nmrg-anticipated-adaptation], provides a strategic and functional advantage to management operations.

3.5.1. Requirements and Challenges

As with other sources of telemetry information, the data and events must meet strict requirements, especially in terms of timeliness, which is essential to properly incorporate external event information to management cycles. Thus, the specific challenges are described as follows:

- o The role of external event detector can be played by multiple elements, including hardware (e.g. physical sensors, such as seismometers) and software (e.g. Big Data sources that analyze streams of information, such as Twitter messages). Thus, the transmitted data must support different shapes but, at the same time, follow a common but extensible ontology.
- o Since the main function of the external event detectors is actually to perform the notifications, their timeliness is assumed. However, once messages have been dispatched, they must be quickly collected and inserted into the control plane with variable priority, which will be high for important sources and/or important events and low for secondary ones.
- o The ontology used by external detectors must be easily adopted by current and future devices and applications. Therefore, it must be easily mapped to current information models, such as in terms of YANG.

Organizing together both internal and external telemetry information will be key for the general exploitation of the management possibilities of current and future network systems, as reflected in the incorporation of cognitive capabilities to new hardware and software (virtual) elements.

4. Security Considerations

TBD

5. IANA Considerations

This document includes no request to IANA.

6. Contributors

The other main contributors of this document are listed as follows.

- o James N. Guichard, Huawei
- o Yunan Gu, Huawei

7. Acknowledgments

We would like to thank Victor Liu and others who have provided helpful comments and suggestions to improve this document.

8. References

8.1. 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>>.

8.2. Informative References

- [I-D.brockners-inband-oam-requirements]
Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mozes, D., Mizrahi, T., <>, P., and r. remy@barefootnetworks.com, "Requirements for In-situ OAM", draft-brockners-inband-oam-requirements-03 (work in progress), March 2017.

- [I-D.fioccola-ippm-multipoint-alt-mark]
Fioccola, G., Cociglio, M., Sapio, A., and R. Sisto, "Multipoint Alternate Marking method for passive and hybrid performance monitoring", draft-fioccola-ippm-multipoint-alt-mark-04 (work in progress), June 2018.

- [I-D.ietf-grow-bmp-adj-rib-out]
Evens, T., Bayraktar, S., Lucente, P., Mi, K., and S. Zhuang, "Support for Adj-RIB-Out in BGP Monitoring Protocol (BMP)", draft-ietf-grow-bmp-adj-rib-out-01 (work in progress), March 2018.
- [I-D.ietf-grow-bmp-local-rib]
Evens, T., Bayraktar, S., Bhardwaj, M., and P. Lucente, "Support for Local RIB in BGP Monitoring Protocol (BMP)", draft-ietf-grow-bmp-local-rib-01 (work in progress), February 2018.
- [I-D.ietf-netconf-udp-pub-channel]
Zheng, G., Zhou, T., and A. Clemm, "UDP based Publication Channel for Streaming Telemetry", draft-ietf-netconf-udp-pub-channel-03 (work in progress), July 2018.
- [I-D.ietf-netconf-yang-push]
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "YANG Datastore Subscription", draft-ietf-netconf-yang-push-17 (work in progress), July 2018.
- [I-D.kumar-rtgwg-grpc-protocol]
Kumar, A., Kolhe, J., Ghemawat, S., and L. Ryan, "gRPC Protocol", draft-kumar-rtgwg-grpc-protocol-00 (work in progress), July 2016.
- [I-D.openconfig-rtgwg-gnmi-spec]
Shakir, R., Shaikh, A., Borman, P., Hines, M., Lebsack, C., and C. Morrow, "gRPC Network Management Interface (gNMI)", draft-openconfig-rtgwg-gnmi-spec-01 (work in progress), March 2018.
- [I-D.pedro-nmrg-anticipated-adaptation]
Martinez-Julia, P., "Exploiting External Event Detectors to Anticipate Resource Requirements for the Elastic Adaptation of SDN/NFV Systems", draft-pedro-nmrg-anticipated-adaptation-02 (work in progress), June 2018.
- [I-D.song-opsawg-dnp4iq]
Song, H. and J. Gong, "Requirements for Interactive Query with Dynamic Network Probes", draft-song-opsawg-dnp4iq-01 (work in progress), June 2017.

- [I-D.zhou-netconf-multi-stream-originators]
Zhou, T., Zheng, G., Voit, E., Clemm, A., and A. Bierman,
"Subscription to Multiple Stream Originators", draft-zhou-
netconf-multi-stream-originators-02 (work in progress),
May 2018.
- [RFC1157] Case, J., Fedor, M., Schoffstall, M., and J. Davin,
"Simple Network Management Protocol (SNMP)", RFC 1157,
DOI 10.17487/RFC1157, May 1990,
<<https://www.rfc-editor.org/info/rfc1157>>.
- [RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M.
Zekauskas, "A One-way Active Measurement Protocol
(OWAMP)", RFC 4656, DOI 10.17487/RFC4656, September 2006,
<<https://www.rfc-editor.org/info/rfc4656>>.
- [RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J.
Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)",
RFC 5357, DOI 10.17487/RFC5357, October 2008,
<<https://www.rfc-editor.org/info/rfc5357>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed.,
and A. Bierman, Ed., "Network Configuration Protocol
(NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,
<<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken,
"Specification of the IP Flow Information Export (IPFIX)
Protocol for the Exchange of Flow Information", STD 77,
RFC 7011, DOI 10.17487/RFC7011, September 2013,
<<https://www.rfc-editor.org/info/rfc7011>>.
- [RFC7276] Mizrahi, T., Sprecher, N., Bellagamba, E., and Y.
Weingarten, "An Overview of Operations, Administration,
and Maintenance (OAM) Tools", RFC 7276,
DOI 10.17487/RFC7276, June 2014,
<<https://www.rfc-editor.org/info/rfc7276>>.
- [RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext
Transfer Protocol Version 2 (HTTP/2)", RFC 7540,
DOI 10.17487/RFC7540, May 2015,
<<https://www.rfc-editor.org/info/rfc7540>>.
- [RFC7799] Morton, A., "Active and Passive Metrics and Methods (with
Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799,
May 2016, <<https://www.rfc-editor.org/info/rfc7799>>.

- [RFC7854] Scudder, J., Ed., Fernando, R., and S. Stuart, "BGP Monitoring Protocol (BMP)", RFC 7854, DOI 10.17487/RFC7854, June 2016, <<https://www.rfc-editor.org/info/rfc7854>>.
- [RFC8321] Fioccola, G., Ed., Capello, A., Cociglio, M., Castaldelli, L., Chen, M., Zheng, L., Mirsky, G., and T. Mizrahi, "Alternate-Marking Method for Passive and Hybrid Performance Monitoring", RFC 8321, DOI 10.17487/RFC8321, January 2018, <<https://www.rfc-editor.org/info/rfc8321>>.

Authors' Addresses

Haoyu Song (editor)
Huawei
2330 Central Expressway
Santa Clara
USA

Email: haoyu.song@huawei.com

Tianran Zhou
Huawei
156 Beiqing Road
Beijing, 100095
P.R. China

Email: zhoutianran@huawei.com

Zhenbin Li
Huawei
156 Beiqing Road
Beijing, 100095
P.R. China

Email: lizhenbin@huawei.com

Giuseppe Fioccola
Telecom Italia
Via Reiss Romoli, 274
Torino 10148
Italy

Email: giuseppe.fioccola@telecomitalia.it

Zhenqiang Li
China Mobile
No. 32 Xuanwumenxi Ave., Xicheng District
Beijing, 100032
P.R. China

Email: lizhenqiang@chinamobile.com

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

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

Laurent Ciavaglia
Nokia
Villardeaux 91460
France

Email: laurent.ciavaglia@nokia.com

Aijun Wang
China Telecom
Beiqijia Town, Changping District
Beijing, 102209
P.R. China

Email: wangaj.bri@chinatelecom.cn