Network Management Research Group (NMRG)
Internet-Draft
Intended status: Informational
Expires: 4 September 2025

M. Liebsch
NEC
M. Stiemerling
N. Schark
h_da
3 March 2025

Challenge: Network Digital Twin - Practical Considerations and Thoughts
draft-liest-nmrg-ndt-challenges-01

Abstract

This document focuses on practical challenges associated with the design, creation and use of Network Digital Twins. According to the identified challenges, a set of suitable functional elements are described to overcome some of these challenges. Experiences from the design, development and evaluation of an SDN-based Network Digital Twin are described and conclude this document.

Table of Contents

1. Terminology

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.

2. Introduction

Digital Twins are well known from Industry applications, where, for example, a robotic system or production machine with sensors and actuators is digitally represented on a computer. The twin is used for monitoring and simulation purpose. In the meantime, the computation and use of a digital twin of a data network gets more traction and is discussed in the research community in particular in the view of automated network management, also in the view of beyond 5th Generation (5G) of a mobile communication system.

The creation of a Network Digital Twin (NDT) implies many challenges, incl. the continuous collection and computation of relevant data points from network components of a possibly large network topology with many network functions and nodes. As a key use case, the user of an NDT may apply changes, such as configurations or load, first to the digital representation of the network and evaluate and assess these changes' impact to the network operation in terms of performance, latency, or stability. Such simulation requires proper modelling of the network's information and behavior. The IRTF's Network Management Research Group (NMRG) is discussing various use cases and is working on a suitable reference architecture for NDTs [I-D.irtf-nmrg-network-digital-twin-arch].

This document focuses on practical challenges associated with the design, creation and use of NDTs. Furthermore, some technology directions and methodologies are discussed as possible solution to overcome some of these challenges. Experiences from the design, development and evaluation of an prototype implementation, that realizes an NDT of a Software-defined Network (SDN) completes this document.

3. Practical Challenges in operating a Network Digital Twin

A user of a Network Digital Twin may be a person, which uses a suitable frontend, or an automated process, such as a network OAM and Orchestration system. The user is probably aware of the detailed or an abstracted view of the physical network topology. The NDT Instance comprises all functions that are needed to monitor the physical network and to generate a digital representation of it, to expose a digital representation of the network to the user, take probe requests from a user to simulate changes in the digital representation of the network, and provide simulation results back to the user. To build a digital representation of the network, the NDT instance monitors network functions in the physical network and collects relevant data points. The resulting digital representation of the network is in the following denoted as Twin Model.

The following assumptions apply:

o The NDT Instance is aware of the network nodes, functions and segments that are in scope of its duty to build and maintain a Twin Model

o The NDT Instance is aware of relevant data points and how to collect them from the physical network. The NDT Instance may probe periodically for these data points or schedule periodic or event-based reporting of these data points, e.g. in a network controller.

o The NDT Instance and the user share the same descriptors and level of detail of the Twin Model's topology.

o The NDT Instance utilizes at least one method and model to apply a user's probe request for changes in the network for simulation. The model may be aligned with tools for discrete event simulation, mathematical models or models from the Artificial Intelligence and Machine Learning Science. A model may be pre-provisioned or automatically generated and improved throughout the NDT Instance operation.

o Based on the provided and assessed simulation results, a user may decide to apply the previously probed or adjusted changes to the physical network. Different options exist for a concrete implementation of such enforcement. The user may have access to a network controller northbound interface or any other API to apply changes to a physical network. One option is based on the principles of Intent Based Networking (IBN), where the NDT Instance provides an API to its user to issue probe requests and further semantic, that allows the NDT Instance to assess the simulation result and enforce changes in the physical network on its own, without providing simulation results back to the user.

Figure 1 depicts generalized principles of generating and using a Network Digital Twin. The NDT Instance maintains a model of the physical network by means of constantly monitored data points. These data points are monitored on the relevant network nodes and network functions (NF) and transmitted directly or indirectly (via a network controller) to the NDT Instance. The current state of a network is exposed to the NDT user. Change requests in the network are issued to the NDT Instance. Such requests may, for example, tune a parameter in a network node, such as load, performance, routing table entry, or load balancing strategy. The simulation results may by provided back to the user to take further action.

The functional architecture of an NDT as described in [I-D.irtf-nmrg-network-digital-twin-arch] comprises northbound interfaces for users or applications to leverage the NDT's features, and southbound interfaces towards the physical network and its components to collect data points and enforce control decisions.

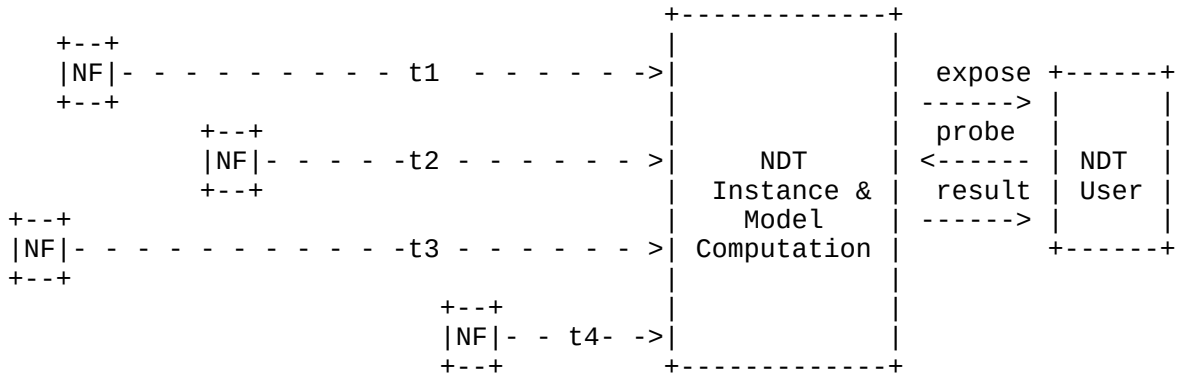

Figure 1: Principle in generating and using a Network Digital Twin

The complete life-cycle of an NDT Instance may imply the following challenges:

o A large number and diversity of nodes and functions needs to be monitored, such as physical or virtual routers, switches, network functions, or even compute, storage and networking resources. Scalability needs to be ensured in dependency of the number and types of data points and the level of details needed for model generation and maintenance.

o A large number of data points needs to be transmitted upstream towards the NDT Instance for data aggregation, storage and computation. The platforms that host components associated with the NDT Instance must provide sufficient compute, storage and networking resources to handle such data volume and its treatment.

o The change rate, e.g., events per second, of any data point plays also a crucial role. As all changes of a data point could potentially transferred into the NDT instance, this may lead to either an overload situation of the instance or it may be even impossible to keep up with all state changes, e.g., if considering TCP flows and their changes with respect to sequence numbers. This will require a proper dimensioning in terms of granularity of the data point's information to be relayed to the NDT instance.

o Transmission latency (t1, t2, t3, ...) from a number of network functions to the NDT Instance may differ dependent on the topological distance of a network function, its monitoring strategy (local data point aggregation) and network performance. In case data points are labeled with timestamps, synchronization of distributed network nodes/functions may be required. Dependent on the modelling technique, differences in transmission latency and timestamps may have impact on the model accuracy.

o Users of an NDT Instance may have different requirements on the relevant network scope, accuracy, attributes associated with the network nodes, functions and segments, as well as with delay tolerance in the NDT providing simulation results back to the user.

o Impact and severity of the above items depend on many factors, such as network size, network performance, expected accuracy, and the deployment strategy of NDT enablers (centralized vs. distributed).

o It is likely that a NDT is used by multiple users in parallel. This is not an issue when these uses concurrently read state from the NDT, but challenges arise when multiple users apply changes to the same NDT. This in fact means that parallel configurations of the NDT happen and it has to be ensured that these configurations are not conflicting each other or, in case of a conflict, are merged in a meaningful way.

4. NDT Instance -- Functional Components and Reference Points

The IRTF’s Network Management Research Group (NMRG) drafted and published a Reference Architecture for NDTs. This section complements the NMRG’s NDT reference architecture without interference and adds a few functional elements in support of the subsequent discussion section. While Section 4.1 depicts an NDT reference architecture with the focus on selected functions and reference points, Section 4.2 discusses the roles of some functional elements and reference points.

4.1. NDT reference architecture -- an extended view

The functional architecture of an NDT as described in [I-D.irtf-nmrg-network-digital-twin-arch] comprises northbound interfaces for users or applications to leverage the NDT’s features, and southbound interfaces towards the physical network and its components to collect data points and enforce control decisions.

Figure 2 distills and depicts a few functional additional elements and reference points that may play a role in tackling the above challenges. The Network Twin Instance (T) comprises an interface (M-T) to a Management System, that initializes the Network Twin Instance, monitors its operation and applies changes as needed, e.g. the scope of the physical network from which a twin should be created, or changes in the physical network topology that need to be taken into account for the twin creation. Monitoring as well as the enforcement of changes in the physical network may be performed through a network controller (NW Controller), which is decoupled from the Network Twin Instance and used by the Network Twin Instance as well as by the NDT User (U) through the NW Controller's northbound interface (T-Ctrl, U-Ctrl). In case of a distributed and collaborative deployment, Network Twin Instances can utilize a federation interface (T-T). The role of the illustrated functional element and reference points in addressing the above challenges is discussed in the subsequent Section 4.2

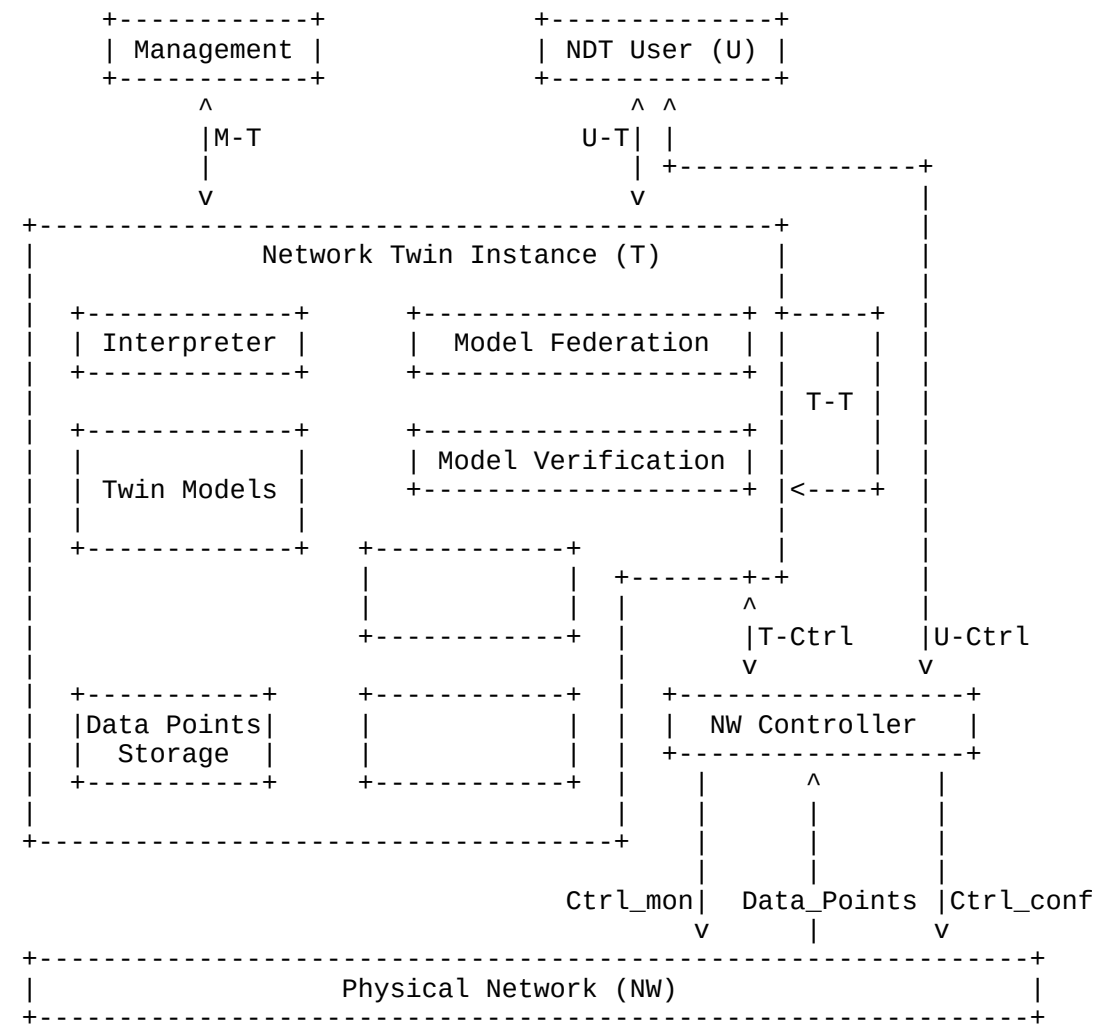

Figure 2: Functional Elements and Reference Points of a Network Digital Twin

4.2. The role of some functional elements and reference points

The following list describes the role of the functional elements and reference points that are highlighted in Figure 2.

o Interpreter -- Receives probe requests with changes that apply to the Network Twin and additional descriptions to guide the Network Twin Instance for computing the result. The U-T reference point may be Intent-based, which requires the Interpreter to classify the request and determine relevant technical attributes and network function before applying the changes to the Twin Model. Any other API may apply to the U-T reference point and can give the NDT User full control on the attributes that apply to a particular model component should change.

o Twin Data and Behavioral Model (TDBM)-- A model of the data and of the complete network and its behavior that should be represented by the NDT, or a sub-network in case of distributed and collaborating Network Twin Instances.

o Network Controller (NWC) -- Exposes northbound interfaces to the Network Twin Instance (T-Ctrl) and the NDT User (U-Ctrl). The northbound interface is used to configure physical network monitoring and data point collection, as well as to apply changes to the physical network. The NWC exposes southbound interfaces to the physical network for scheduling monitoring rules on the network components and function (Ctrl_mon), to collect monitoring result (Data Points), and to enforce changes in the physical network

o Data Points Storage -- Stores and structures monitored data points in alignment with the Twin Model. In case of a distributed and collaborative deployment, the Data Points Storage may hold only data points associated with the respective sub-network.

o Model Verification -- In case a machine learning model is being used, the model may be accurate and valid for some time, but may need to be re-trained in case of re-configurations in the physical network. A new model may be trained in the background while one model is being used at a time. In case a model turns out to be inaccurate, a suitable indication may be exposed to all connected NDT users until a new, more accurate model is in service.

o Model Federation -- A distributed deployment of multiple Network Twin Instances (T) can be advantageous in many aspects. A single instance can model a sub-network comprising only a single network component or network functions, or may take a few neighbor components and their data points into account. Such strategy can increase scalability and accuracy due to the lower volume on network topology information and data points, as well as the resulting smaller Twin Model. Accuracy may benefit from lower transmission latency of data points. Distributed Network Twin Instances can collaborate via the T-T reference points and expose

complete models to neighboring or all Network Twin Instances. Model aggregation can be accomplished by a single or multiple dedicated Network Twin Instances.

For large physical networks and their representation as NDT, a distributed deployment may be advantageous in many aspects. For small sub-networks, In-Network Computing could be a suitable enabler to run machine learning models that represent the NDT of a single or a few neighboring nodes. Training of such sub-network model may be performed on the node with In-Network Computing capabilities or by an external node.

5. Practical Examples - Creation and Use of NDTs

5.1. SDNDT - Implementation of a Software-defined Network Digital Twin

5.1.1. Discussion

A first architectural and practical challenge is I) how to extract information about the current state of the physical network to the Network Digital Twin (NDT) and II) how to feed back changes from the NDT to the physical network.

There are two extremes:

o A fully distributed approach where each single network element writes its state into a distributed database and changes are fed back from this database to the elements. NDT users also have to read and write state to the distributed database.

o A fully centralized approach is also possible where a network controller is in charge of reading and writing from/to the network elements. This network controller would have its own data base and NDT users would solely interact with the network controller.

Further we distinguish between these types of NDTs (though there are more):

o Static Test System

o Live Reacting System

The static test system is the more simple type of the two systems. The overall goal is to be able to create a NDT instance on demand, which represents the physical network as good as possible. It lacks the need for dynamic up- dates in either direction, as only a snapshot is taken on-demand. By design, changes should only be applied to either the physical system or the Digital Twin if

explicitly desired. In a realistic scenario, the NDT has to be updated to keep up with changes done on the physical network, but its sufficient, if the virtual environment gets recreated each time, as it should only represent a static state of the system at a given time. This approach may look overly simplified, but for a first proof-of-concept implementation it may be the best way, as the number of challenges and constraints is low.

The long-term goal is to have a NDT that is (optimally) in total sync with the physical environment or it represents the physical network as close as possible. This would call for an extension of the static test system to a live reacting system, i.e., automatic synchronization of state changes in the physical network to the NDT and triggered changes from the NDT back. However, the live reacting system will need live-data, i.e., real-time updates to the NDT, which may be challenging in terms of amount of information and the required granularity of the data, by the NDT instance.

In order to achieve a first usable implementation we did limit it to a fully-centralized approach for a static live test system. This focusses the implementation efforts to extracting configuration and runtime information from the physical network to the NDT and the feedback from the NDT. However, it of course neglects the real-time aspect, but getting a first workable solution was and is the goal.

5.1.2. Implementation Overview

For the implementation of the live static test system we used the goSDN SDN controller [gosdn] as network controller and on top if it a specialized application, i.e., the venv manager [venv-manager]. The emulation of the network elements can be done on virtual machines, but we did chose to rely on containerlab [containerlab]

Figure 3 shows the principle architecture of the static test system. The roles of, as described in Section 4.2, Interpreter, federation, and verification are currently not implemented, but will follow in the future. It is distinguished in the physical network and the NDT part. From the structure both parts are identically:

o network controller -- is a SDN controller (SDN cntrl in the figure) and has the role of reading and writing configuration and state to and from the network elements. The controller is there in two instances, in the upper part for the physical network and in the lower part for the NDT instance.

o venv-manager -- is a networking app running on the North-Bound-Interface (NBI) of goSDN and is in charge of retrieving the topology of the physical network and to feed it over the interface (2) (right hand side) to the NDT instance’s venv-manager.

o Controller-to-controller interface -- this interface, (1) in the figure, is used to synchronize the NDT instance with the configuration and state data of the physical network and also from the NDT back to the physical part.

o network elements -- are either physical in the physical network domain or emulated ones in the NDT domain (lower part). In our implementation with use either virtual machines or containerlab for emulating the physical network elements.

The M-T and U-T reference points are exposed by the goSDN controller for the NDT instance.

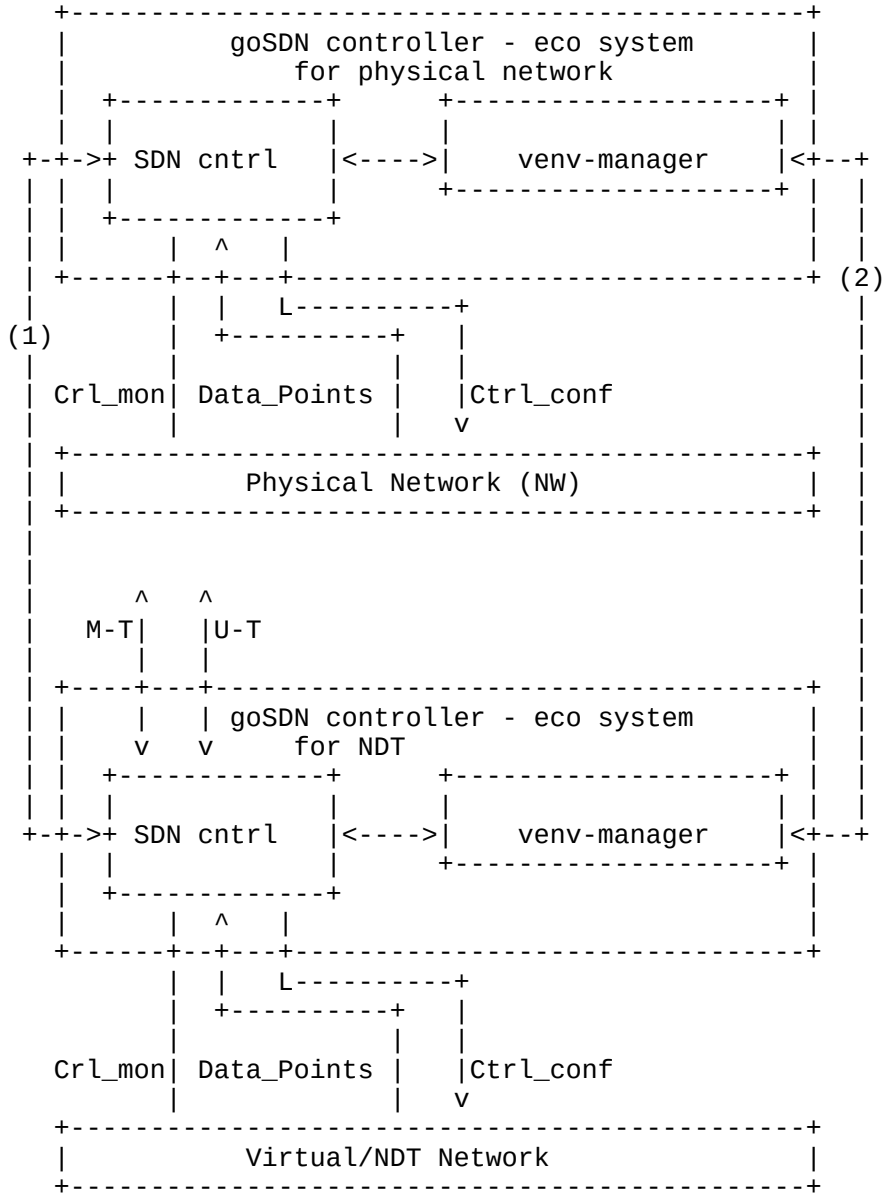
goSDN controller - eco system
for physical network
SDN cntrl
venv-manager
(1)
(2)
Crl_mon
Data_Points
Ctrl_conf
Physical Network (NW)
M-T
U-T
goSDN controller - eco system
for NDT
SDN cntrl
venv-manager
Crl_mon
Data_Points
Ctrl_conf
Virtual/NDT Network

Figure 3: Architecture overview of the static test system

The goSDN controller uses gNMI with Yang models, namely a subset of OpenConfig [openconfig-ym], to represent the configuration and state data of the network elements and thus the whole network. The gNMI interfaces, with the respective data models, is used to implement the Crl_mon and Ctrl_conf interfaces, as well as, the collection of data points. The state of devices is represented in the data models within the controller.

For testing purposes we did use Arista's CEOS docker images within containerlab, as a commercial counter part, for network elements. Further, we have developed our own SDN agent, the gnmi-target [danet-gnmi-target], which runs on Linux (tested for debian and ubuntu) and in the future also on FreeBSD 13.2-RELEASE and newer.

5.1.3. First Findings

The current implementation of a NDT instance is usable for a static test system. The implementation works with one vendor specific operating system and our own gnmi-target as SDN agent on a network element. However, the crucial point will be if and how the required information can be extracted from the physical network. This is usually less a conceptual issue, but more a practical question what is accessible with more less standardized interfaces.

The current aim was not to have full-fledged real-time live reacting system, but to do the first steps, learn, and then move on towards more features, such as live feeds.

Also self-learning behavioral model of the network elements was not developed, as we rely on virtualized versions of the network element's operating system, such as, Aristas CEOS or our own SDN agent on plain Linux. This neglects of course any impact of the hardware of a real network element, e.g., port or forwarding engine behavior.

5.2. QKDN NDT

Quantum Key Distribution Networks (QKDN) are currently being build in various settings, e.g., [darpa-qkd][cn-qkdn-deploy][de-demoquandt-qkdn] but not limited to these, mainly at this time (as of February 2025) for research purposes. However, access to the physical QKD-networks is limited, e.g., as the amount of Quantum Communication Modules (QCM) available is quite limited or as these networks are not accessible by all interested groups.

This memo will introduce some reasonings for Network Digital Twins of QKDNs in the Section Section 5.2.2, but start with a short overview of a simplified architecture of QKDNs in the next Section Section 5.2.1.

5.2.1. Simplified QDKN Architecture

This section here is originally taken from [I-D.danet-qkdn-considerations].

The ITU defines an extensive QKDN architecture in Y.3802 [itu-y-3802]. However, for our discussion we use a simplified architecture here.

The Figure below shows a simplified architecture for a single QKDN domain.

The Quantum Communication Modules (QCM) are in charge of exchanging random numbers between 2 QCM, or n modules for single-source entangled based systems.

The Key Management Systems (KMS) are in charge of allowing a secure end-to-end relay of a secret across the whole domain. They obtain the encryption keys, or some initial input to the encryption key, from their local KMS.

The Network Controller (NW cntrl) can be used to control and managed the operations of the KMS and also the QCM.

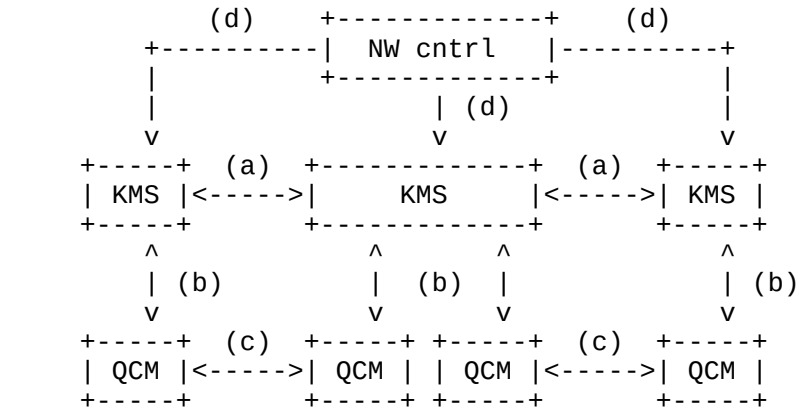

Figure 4: A simplified single Domain QKDN Architecture

The interfaces between the components are:

* (a) KMS-to-KMS interface: this interface is used to facilitate the secure key forwarding between the KMS

* (b) KMS-to-QCM interface: this interface is used by the KMS to obtain the generated random numbers from the QCM

* (c) QCM-to-QCM interface: this interface is used between adjacent Quantum Communication Modules and consists actually out of two interfaces, i.e., the quantum link and the classical channel.

* (d) Network Controller to KMS interface: This interface, if a controller-based approach is used, controls the operation of the KMS.

5.2.2. Use of a NDT in QKDN

Quantum Key Distribution Networks are at this point mainly used in research and explorative settings, but not as production networks. Therefore, some architectural questions and technical details are still to be determined in future developments. On the other hand, Quantum Communication Modules (QCM) are now available and being deployed. Access to these modules can be hard to gain, as they are still limited in numbers available and access on a daily base to run tests or try outs may be still limited.

Network Digital Twins could allow to model the free-space or optical quantum channels for further research, education, development, and training of staff without the need of direct access to the QCM and the infrastructure around.

Why a QKDN NDT is a good idea:

* (a) access to the QCM is the limited

* (b) access to the operational data out of the QCM is the limited

5.3. Network Twin of 5G Mobile System Components

5.3.1. Use of a NDT in a 5G system

5th Generation Mobile Communication Systems are to a large extent deployed as software and virtualized Network Functions (NF) on cloud computing platforms. Individual or groups of NFs can be deployed as virtual instances on top of a Hypervisor or a container runtime. Hardware resources, such as CPU cores and memory, are shared or, if supported, bound to particular NF(s). While production-grade solutions include load balancers and an implementation of suitable scaling as well as failover strategies, unexpected peaks of load may

still occur and result in service interruption. The actual source of such failure can be diverse and, for example, due to an overload of allocated CPU resources or no leftover resources for a dynamic scale-up procedure, lack of memory or even excessive network pressure. Furthermore, in some cases session states are externalized and require read/write operations with either a local volume or an external database for all session state changes, hence overall system performance during busy hours have a dependency on network characteristics and the applied load. However, transaction states are typically treated and stored intrinsically to a NF, which is fast and does not require network operations.

A Network Digital Twin can help the operator of such system to experiment with different workload and investigate the impact to different factors of a deployed 5G System. This can help during planning and operations phase to improve an initial or updated deployment strategy, aiming at a more reliable and stable system.

5.3.2. Positioning of ML-Model for this use case

In particular for a non-real time use of a NDT, where the twin’s states are not permanently synchronized with the real system, DES looks like a good option for NDT creation. In fact, various vendors of hardware components provide models of their switches or routers for simulation purposes. However, large 5G system with a variety of NFs and custom decisions about their deployment, are difficult to model.

Furthermore, building a NDT from a real system that comprises third party components, e.g. one or multiple NFs, may hamper an accurate design of a model following a bottom-up approach due to the lack of knowledge about its software design. Building a model of such components based on an experimental approach can be a suitable tool to mimic their behavior and evaluate impact of various load figures to KPIs of interest.

Proven machine learning models are suitable to abstract from many details but can be trained to model the behavior of interest. Downside of such approach is the costs in creating accurate models, which requires plenty of data, either retrieved from the real production system or from a matching experimental system. Once well designed and trained, the machine learning model can help to classify or predict anomalies with good accuracy and speed.

Suitable machine learning models need to be selected according to the intended use. As example, a recurrent neural network (RNN) with long short-term memory (LSTM) is suitable to recognize particular patterns in a data sequence and predict values [mswim-2020-automec]. Such

characteristics can be leveraged for the prediction on critical situations in a NDT and counteract such situation in a real production network.

5.3.3. Exemplary design and creation

In the context of a 5G System NDT, a model of a single or a group of NFs can be defined as sub-system under test and traffic can be applied to it to enforce a variety of load situations, while a monitoring system continuously captures data points of interest, such as memory- and CPU core utilization, resulting network pressure on dependent interfaces to external NFs or data repositories. While a proper model of NFs can be built based on long-term monitoring of traffic and NF characteristics in a real production system, the advantage of mirroring the relevant NFs from the real production system into an experimental system is that the dedicated sub-system under test can be over-saturated with traffic and load to enforce unexpected NF behavior or even failures, which contributes to a richer data set, resulting in a more accurate model that can better classify or predict a potentially critical situation, which is a key value of a NDT.

For a model of 5G core NFs, the sub-system under test can be exposed to different load figures by means of a traffic generator, that emulates certain operations such as bulk attachments of multiple mobile devices. A radio access network (RAN) emulator is a suitable tool to generate such traffic. Resulting operational steps within a single of a group of NFs can then be monitored and recorded in the view of the target data set with selected features for training a suitable machine learning model. Figure 5 depicts a high-level view of such experimental setup with two instances of a NF on top of a compute system, where a monitor captures data points of interest from CPU resources, memory and network interfaces.

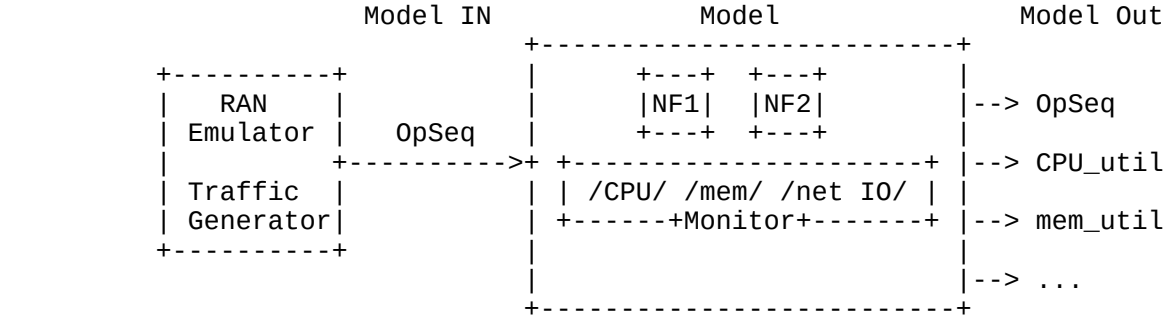

Figure 5: 5G NF model training -- Impact of workload to key resource figures

6. IANA Considerations

This document does not have IANA considerations.

7. Security Considerations

Security considerations are to be done in future revisions of this memo.

However, one can imagine that a NDT instance with a full copy of the configuration and state information of a complete network is a huge trove for any attacker.

8. Acknowledgments

Neil Schark is partially funded by the German BMBF DemoQuanDT project. Martin Stiemerling is partially funded by the German BSI ADWISOR5G project.

9. References

9.1. Normative References

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

[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>.

9.2. Informative References

[I-D.irtf-nmrg-network-digital-twin-arch] Zhou, C., Yang, H., Duan, X., Lopez, D., Pastor, A., Wu, Q., Boucadair, M., and C. Jacquenet, "Network Digital Twin: Concepts and Reference Architecture", Work in Progress, Internet-Draft, draft-irtf-nmrg-network-digital-twin-arch-10, 27 February 2025, <https://datatracker.ietf.org/api/v1/doc/document/draft-irtf-nmrg-network-digital-twin-arch/>.

[gosdn] "goSDN controller GIT repository", <https://netellyfish.org>.

[venv-manager]
"venv mananger implementation", <https://code.fbi.h-da.de/danet/gosdn/-/tree/master/applications/venv-manager>.

[containerlab]
"venv mananger implementation", <https://containerlab.dev/>.

[openconfig-ym]
"OpenConfig Yang Models repository", <https://github.com/openconfig/public/>.

[danet-gnmi-target]
"da/net gnmi-target", <https://code.fbi.h-da.de/danet/gnmi-target>.

[itu-y-3802]
ITU-T, "Quantum Key Distribution Networks - Functional Architecture", December 2020, <https://www.itu.int/rec/T-REC-Y.3802-202012-I/en>.

[I-D.danet-qkdn-considerations]
Stiemerling, M., Seidl, F., Bauch, M., Schark, N., and J. Henrich, "Initial Considerations about QDKN Protocols", Work in Progress, Internet-Draft, draft-danet-qkdn-considerations-00, 21 October 2024, <https://datatracker.ietf.org/doc/html/draft-danet-qkdn-considerations-00>.

[cn-qkdn-deploy]
Chen, Y., "An integrated space-to-ground quantum communication network over 4,600 kilometres", Work in Progress, Internet-Draft, draft-danet-qkdn-considerations-00, 6 January 2021, <https://www.nature.com/articles/s41586-020-03093-8.epdf>.

[darpa-qkd]
Elliott, C. and H. Yeh, "DARPA Quantum Network Testbed", July 2007, <https://apps.dtic.mil/sti/pdfs/ADA471450.pdf>.

[de-demoquandt-qkdn]
"BMBF DemoQuaDT deployment over 900 kms", October 2024, <https://www.linkedin.com/posts/bundesministerium-f%C3%BCr-bildung-und-forschung_sichere-kommunikation-dank-quanten-das-vom-activity-7265701310631419905-eta9/>.

[mswim-2020-automec]
Fattore, U., Liebsch, M., Brik, B., and A.. Ksentini, "AutoMEC: LSTM-based User Mobility Prediction for Service Management in Distributed MEC Resources", November 2020, <https://dl.acm.org/doi/10.1145/3416010.3423246>.

Authors' Addresses

Marco Liebsch
NEC Laboratories Europe GmbH
Kurfuersten-Anlage 36
D-69115 Heidelberg
Germany
Email: marco.liebsch@neclab.eu

Martin Stiemerling
Darmstadt University of Applied Sciences
Schoefferstrasse 3
64295 Darmstadt
Germany
Email: mls.ietf@gmail.com
URI: https://www.stiemerling.org

Neil Schark
Darmstadt University of Applied Sciences
Schoefferstrasse 3
64295 Darmstadt
Germany
Email: Neil.Schark@h-da.de