A High-Fidelity Virtualized Digital Twin System for ISP Core Networks

Johannes Späth, Georg Carle
Technical University of Munich, Germany
{spaethj, carle}@net.in.tum.de

***Abstract*—ISP core networks are critical for today's Internet, and fault management is a major concern. Digital Twins (DTs) offer a safe environment for what-if analyses and can help generating incident datasets, thereby supporting Machine Learning (ML) applications. Existing approaches often use high abstraction, limiting accuracy and performance, and frequently focus on specific metrics. Furthermore, they often rely on custom interfaces, impeding real-world application. We present a virtualization-based DT framework capable of low abstraction, high fidelity, and near-hardware performance. It spans multiple network planes via standardized NETCONF/YANG interfaces and exports telemetry through YANG models, IPFIX, and BMP.**

***Index Terms*—Digital Twin, YANG, Testbed, Core Network**

I. Introduction

Fault management is a central concern for operators of computer networks, especially for critical infrastructure like ISP core networks. Digital Twins (DTs) offer significant potential to support multiple use cases in this domain. As networks become increasingly complex, understanding the effects of configuration changes is often infeasible. DTs enable what-if analyses to test changes and validate mitigations. Further, Machine Learning (ML)-based network management approaches require large amounts of training data. While live systems generate substantial amounts of data, insights from real incidents are most valuable. Since introducing faults in a production system is not feasible, DTs provide a practical means for generating incident datasets.

Existing DT approaches typically operate at a high abstraction level, like with simulations or emulations, suffering from inaccuracies and low performance. Many approaches focus solely on specific aspects, *e.g.*, latency modeling. In addition, they frequently rely on custom interfaces and do not support ISP-grade monitoring standards.

Our approach uses virtualization in a testbed and thus provides low abstraction, high fidelity, and near-hardware performance. It covers multiple planes via standardized interfaces based on NETCONF/YANG and supports telemetry export using standardized YANG models, IPFIX, and BMP.

II. Background and Related Work

This section covers relevant background and related work.

A. Network Digital Twins

Various techniques have been proposed for replicating computer networks, including simulation [1], [2], emulation [3], [4], virtualization [5], as well as bare-metal deployments.

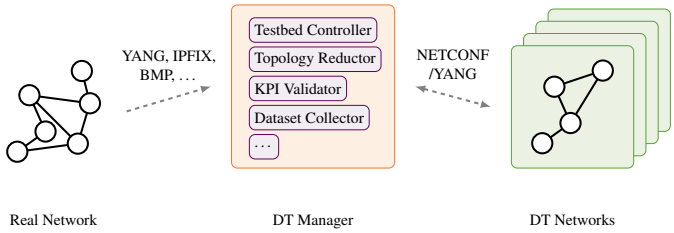

Fig. 1: Overview of the DT system.

Wu et al. [6] present a comprehensive survey on network DTs, in which DTs are defined as replicas of physical network elements. The authors identify key challenges for network DTs and discuss a wide range of application domains. Almasan et al. [7] introduce the concept of a network DT providing accurate and performant network models. Their approach leverages ML techniques such as Graph Neural Networks (GNNs) combined with suitable abstractions to achieve scalability and real-time performance. Our system focuses on ISP core networks and uses standardized YANG models and real-world data from monitoring protocols like IPFIX and BMP.

B. The YANG Modeling Language

YANG [8], [9] is a data modeling language standardized by the IETF for use with NETCONF [10], a network management protocol for configuring network devices. The IETF has standardized a wide range of YANG models covering multiple layers and components of the ISO/OSI stack, including interfaces, routing, and layer 2+3 topology [11], [12], [13], [14], [15], [16], [17].

C. ISP Network Telemetry

We distinguish between configuration data and telemetry in ISP networks. YANG models support configuration and operational state data, whereas dedicated telemetry protocols are limited to the latter. Several standardized telemetry protocols are used in ISP networks, *e.g.*, IPFIX [18] for flow-level data-plane telemetry, and BMP [19] for routing-related control-plane telemetry.

III. Approach

In this section, we describe our high-fidelity virtualized DT system, which is depicted in Figure 1. It consists of a real network, a central management component, and multiple DT networks deployed in a testbed. Network state is exported from the real network using YANG-modeled data, IPFIX telemetry,

and BMP messages, and the architecture can be extended to support additional sources. The DT networks are configured via NETCONF and also export telemetry using YANG.

A. Real Network

This component is a production-grade network consisting of multiple routers. There are no special constraints regarding the routing software used, but it needs to support configuration and telemetry export using the IETF-standardized YANG models.

B. DT Manager Components

At the core of the system is the *DT Manager*, a central component that provides multiple functionalities and supports extensibility, *i.e.*, further features can be added on demand.

a) Testbed Controller: This component orchestrates the virtualized router replicas, configures links, and manages load generation. Depending on the use case, it can also be used to keep the DT synchronized with the real network and to execute actions at specific timestamps.

b) Topology Reductor: This part performs sampling and/or aggregation of the original topology based on predefined goals or the resources available for the DT. Typically, this step also includes flow- and link-rate reduction.

c) KPI Validator: This component evaluates the current state of the real network and the DTs against given Key Performance Indicators (KPIs) using the telemetry.

d) Dataset Collector: This module allows creating datasets for ML using the DT networks. Since DTs are replicas of the production environment, multiple problem instances can be generated safely without affecting live operations. To produce diverse datasets for ML training, the parameter space of problems is defined and samples are balanced across it. Faults can be injected using the Testbed Controller's timed execution feature. After a fault is injected, the corresponding raw data exported from the DT network can be correlated to the fault, allowing multiple datapoints to be extracted from a single long-running measurement.

C. DT Networks

The network DTs are deployed in a testbed using virtualization. Routers are replicated by Linux machines based on FRR [20], which implements a northbound API for YANG export that supports NETCONF via a plugin [21]. Each router runs on an individual Virtual Machine (VM).

IV. Parameter Evaluation

In this section, we evaluate the number of parameters that we expect to be collected for networks of a certain size. For that, we restrict ourselves to three IETF-standardized models (L2 topology [13], L3 topology [12], and interface management [15]) and group their fields by data type (*Identifier*, *Bool*, *Numeric*, and *Other*). We generate random graphs using the Barabási-Albert preferential attachment model [22] with $m = 2$, *i.e.*, for each new node, two edges are added. For each graph, we estimate the number of YANG fields, assuming that every optional field is present and every field with zero or more elements has exactly one instance.

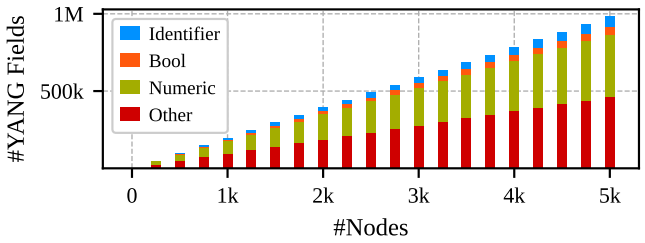

Fig. 2: Number of YANG fields resulting from three IETF models for an increasing number of nodes.

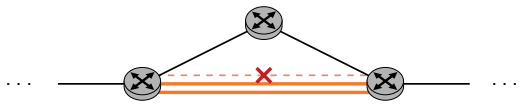

Fig. 3: Overload in a LAG caused by a link failure. The remaining two links are not able to handle the full load.

Figure 2 shows a high number of parameters for three models and small networks (49k fields/250 nodes), which significantly grows for larger networks (1M fields/5k nodes). This underlines the high fidelity of network modeling using our YANG-based approach, however, it introduces complexity for the ML models operating on this data.

V. Example Use Case: LAG Overload Scenario

In this section, we introduce an example scenario involving a Link Aggregation Group (LAG), the relevant YANG models for this scenario, potential symptom fixes, and use cases for our proposed DT system. The scenario considers a LAG comprising three 1 Gbit/s links, carrying a total of 2.5 Gbit/s of traffic (see Figure 3). If one link fails, the remaining two experience an overload situation, since their combined capacity of 2 Gbit/s is exceeded.

In YANG, the LAG is represented using the fields `lag` and `member-link-tp` from the L2 topology model [13]. The `oper-status` field from the interface management model [15] indicates the failed link, while `out-discards` signals the overload on the remaining links.

Potential symptom mitigation strategies include adapting rate limiting on the remaining links if sufficient headroom exists, or re-routing traffic over alternative paths. Using our system, the failure can be automatically detected via the KPI validator, two DT instances can be instantiated to test the mitigations, and the best-performing solution can then be applied to the real network. Additionally, our system can generate datasets of similar scenarios to train ML models for predictive management.

VI. Conclusions and Outlook

In this work, we presented an extensible system for creating high-fidelity virtualized DTs of ISP core networks. Our approach is based on standardized data models and protocols used in production networks. Thus, it provides a basis for fault management in ISP core networks. By automating incident dataset generation, it enables ML-based approaches for fault recovery and thus contributes to efforts toward closed-loop systems and self-driving networks.

Acknowledgments

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), projects HyperNIC (503359370) and SLICES-SUSTAINABILITY (566292327), by the EU Horizon Europe programme, project GreenDIGIT (101131207), by the German Federal Ministry of Research, Technology and Space (BMFTR), project 6G-life (16KIS2414), and by the Bavarian Ministry of Regional Development and Energy, project 6G Future Lab Bavaria. Results presented in this publication were obtained using the SLICES-DE research infrastructure.

References

[1] A. Varga and R. Hornig, "An Overview of the OMNeT++ Simulation Environment," ICST, May 2010. DOI: 10.4108/ICST.SIMUTOOLS2008.3027.

[2] G. F. Riley and T. R. Henderson, "The ns-3 Network Simulator," in *Modeling and Tools for Network Simulation*, K. Wehrle, M. Güneş, and J. Gross, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 15–34. DOI: 10.1007/978-3-642-12331-3_2.

[3] B. Lantz, B. Heller, and N. McKeown, "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets-IX, Monterey, California: Association for Computing Machinery, 2010. DOI: 10.1145/1868447.1868466.

[4] M. Peuster, H. Karl, and S. van Rossem, "MeDICINE: Rapid Prototyping of Production-Ready Network Services in Multi-PoP Environments," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov. 2016, pp. 148–153. DOI: 10.1109/NFV-SDN.2016.7919490.

[5] F. Wiedner, M. Helm, S. Gallenmüller, and G. Carle, "HVNet: Hardware-Assisted Virtual Networking on a Single Physical Host," in *IEEE INFOCOM WKSHPS: Computer and Networking Experimental Research using Testbeds (CNERT 2022) (INFOCOM WKSHPS CNERT 2022)*, Virtual Event, May 2022. DOI: 10.1109/INFOCOMWKSHPS54753.2022.9798351.

[6] Y. Wu, K. Zhang, and Y. Zhang, "Digital Twin Networks: A Survey," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13 789–13 804, 2021. DOI: 10.1109/JIOT.2021.3079510.

[7] P. Almasan et al., "Network Digital Twin: Context, Enabling Technologies, and Opportunities," *IEEE Communications Magazine*, vol. 60, no. 11, pp. 22–27, 2022. DOI: 10.1109/MCOM.001.2200012.

[8] M. Björklund, *YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)*, RFC 6020, Oct. 2010. DOI: 10.17487/RFC6020. [Online]. Available: https://www.rfc-editor.org/info/rfc6020.

[9] M. Björklund, *The YANG 1.1 Data Modeling Language*, RFC 7950, Aug. 2016. DOI: 10.17487/RFC7950. [Online]. Available: https://www.rfc-editor.org/info/rfc7950.

[10] R. Enns, M. Björklund, A. Bierman, and J. Schönwälder, *Network Configuration Protocol (NETCONF)*, RFC 6241, Jun. 2011. DOI: 10.17487/RFC6241. [Online]. Available: https://www.rfc-editor.org/info/rfc6241.

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

[12] A. Clemm, J. Medved, R. Varga, X. Liu, H. Ananthakrishnan, and N. Bahadur, *A YANG Data Model for Layer 3 Topologies*, RFC 8346, Mar. 2018. DOI: 10.17487/RFC8346. [Online]. Available: https://www.rfc-editor.org/info/rfc8346.

[13] J. Dong, X. Wei, Q. Wu, M. Boucadair, and A. Liu, *A YANG Data Model for Layer 2 Network Topologies*, RFC 8944, Nov. 2020. DOI: 10.17487/RFC8944. [Online]. Available: https://www.rfc-editor.org/info/rfc8944.

[14] M. Björklund, *A YANG Data Model for IP Management*, RFC 7277, Jun. 2014. DOI: 10.17487/RFC7277. [Online]. Available: https://www.rfc-editor.org/info/rfc7277.

[15] M. Björklund, *A YANG Data Model for Interface Management*, RFC 8343, Mar. 2018. DOI: 10.17487/RFC8343. [Online]. Available: https://www.rfc-editor.org/info/rfc8343.

[16] S. Litkowski, D. M. Yeung, A. Lindem, Z. Zhang, and L. Lhotka, *YANG Data Model for the IS-IS Protocol*, RFC 9130, Oct. 2022. DOI: 10.17487/RFC9130. [Online]. Available: https://www.rfc-editor.org/info/rfc9130.

[17] L. Lhotka and A. Lindem, *A YANG Data Model for Routing Management*, RFC 8022, Nov. 2016. DOI: 10.17487/RFC8022. [Online]. Available: https://www.rfc-editor.org/info/rfc8022.

[18] P. Aitken, B. Claise, and B. Trammell, *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information*, RFC 7011, Sep. 2013. DOI: 10.17487/RFC7011. [Online]. Available: https://www.rfc-editor.org/info/rfc7011.

[19] J. Scudder, R. Fernando, and S. Stuart, *BGP Monitoring Protocol (BMP)*, RFC 7854, Jun. 2016. DOI: 10.17487/RFC7854. [Online]. Available: https://www.rfc-editor.org/info/rfc7854.

[20] FRRouting, *FRRouting*, Online, visited on: 2026-01-15. [Online]. Available: https://frrouting.org/.

[21] FRRouting, *FRRouting Developer's Guide – Northbound API*, Online, visited on: 2026-01-15, 2017. [Online]. Available: https://docs.frrouting.org/projects/dev-guide/en/latest/northbound/northbound.html.

[22] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999. DOI: 10.1126/science.286.5439.509.