POLITECNICO DI TORINO
Repository ISTITUZIONALE

On Control and Data Plane Programmability for Data-Driven Networking

Original
On Control and Data Plane Programmability for Data-Driven Networking / Sacco, Alessio; Esposito, Flavio; Marchetto, Guido. - ELETTRONICO. - (2021), pp. 1-6. (2021 IEEE 22nd International Conference on High-Performance Switching and Routing (HPSR 2021) Paris 6-10 June 2021) [10.1109/HPSR52026.2021.9481859].

Availability:
This version is available at: 11583/2904833 since: 2021-09-23T21:06:31Z

Publisher:
IEEE

Published
DOI:10.1109/HPSR52026.2021.9481859

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2021 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

06 May 2026

On Control and Data Plane Programmability for Data-Driven Networking

(Invited Paper)

Alessio Sacco
Politecnico di Torino
alessio_sacco@polito.it

Flavio Esposito
Saint Louis University
flavio.esposito@slu.edu

Guido Marchetto
Politecnico di Torino
guido.marchetto@polito.it

***Abstract*—The soaring complexity of networks has led to more and more complex methods to manage and orchestrate efficiently the multitude of network environments. Several solutions exist, such as OpenFlow, NetConf, P4, DPDK, etc., that allow network programmability at both control and data plane level, driving innovation in many focused high-performance networked applications. However, with the increase of strict requirements in critical applications, also the networking architecture and its operations should be redesigned. In particular, recent advances in machine learning have opened new opportunities to the automation of network management, exploiting existing advances in software-defined infrastructures. We argue that the design of effective data-driven network management solutions needs to collect, merge, and process states from both data and control planes. This paper sheds light upon the benefits of utilizing such an approach to support feature extraction and data collection for network automation.**

***Index Terms*—control plane, network programmability, machine learning**

I. Introduction

The separation of control and data plane, as suggested in Software-Defined Networking (SDN), enables advantages towards more intelligent and more programmable networks, and has attracted interests from both academia and industry. In particular, the combination of such network softwarization and edge computing opened up new opportunities for interactive systems, promising simultaneously low-latency and high-bandwidth reliable telecommunications. Moreover, the edge computing paradigm and the programmability of the data plane with novel network programming languages such as P4 [1] or the Deep Programmable Data Plane Kit (DPDK) are showing promising business use cases, supporting private and latency-sensitive applications.

Nevertheless, supporting 1 ms round-trip time on communication, necessary for critical Tactile Internet applications, remains a major challenge [2]. While adaptive application layer algorithms are rising, a proper network infrastructure handling the traffic of applications with strict requirements cannot be an afterthought in network management. Delivering such promises is even more problematic when the underlying infrastructure is unstable and applications impose tight constraints. Solutions for real-time communications have been proposed for specific applications, *e.g.*, video streaming [3]–[7]. However, the challenges introduced by interactive, reliable, and ultra-low latency communications are yet to be solved, and go beyond the current video streaming advances.

978-1-6654-4005-9/21/$31.00 ©2021 IEEE

For the past decade or so, researchers have assumed that the logical separation of the control plane from the data plane was a given necessity, and despite a few valid, ambitious, clean-slate or incremental, proposals to re-architect the TCP/IP stack exist (see for example RINA [8], XIA [9], MobilityFirst [10], NDN [11] or even more recently NewIP [12]), researchers and network application programmers have been obstructed by a choice: programming the data plane or programming the control plane.

Given this scenario, we present a position paper where we argue that a network application process should not have to make that choice, and that the control-data logical separation should just be a guideline principle, not an architecture impasse. Although such separation mainly refers to how functionalities are split, common network problems, *e.g.*, routing, load balancing, and so on, should operate over a flexible and fluid abstraction of the real architecture of devices. In this paper, we clarify the key entities that need to be present to achieve a logically unified control and data plane. In particular, while it is possible today to compile and run OpenFlow or P4 programs as standalone processes, we envision high-performance switches, routers, or other (software) network elements that could operate sharing states across multiple pipelines, recompiling one program while traffic flows through the other.

While futuristic, such an architecture seems possible to design and implement, given the recent advances on programmable data planes. We discuss how such a unifying programmable plane approach has the potential to boost the performance of several networked applications, and point out how recently proposed technologies such as edge computing, machine learning, and network management protocols could help achieve this goal.

The rest of the paper is structured as follows. Section II presents a few applications that would benefit from the presence of a programmable integrated data and control plane paradigm. We then explain how such a paradigm could help typical network mechanisms, *e.g.*, routing, traffic engineering, in Section III. In Section IV we propose a few alternative approaches based on these new technologies. Finally Section V

concludes our study.

II. Motivating Applications and Networking Problems

The advent of new technologies, *e.g.*, SDN, 5G, and edge computing, has been designed to support the development of applications with very strict requirements, *e.g.*, very low latency and high throughput. Among them, *(i)* Industry 4.0, in which robotic machinery is controlled remotely, *(ii)* the Tactile Internet which can enable haptic interaction with visual feedback, providing the illusion of remote touch, *(iii)* VR/AR and Holographic-Type Communications (HTC) in which the application is supposed to rapidly adapt streamed contents based on changes in user position or viewing angle, *(iv)* telemedicine, where medical devices, or simply medical information, are accessed remotely during a tele-operated session, *(v)* assisted or connected cars, which involves communication and data exchange between vehicles and traffic infrastructure, such as flog platform, are pivotal to realize the vision of smart city and intelligent transportation. Two common traits among these applications are the requirement of a well-defined network latency and reliability, to guarantee an acceptable level of user experience [7].

In this context, a deeply programmable network architecture plays a key role for new technologies and applications, and should help reach the goals of an Ultra-Reliable Communication (URC). The URC mode is built around the concept of reliable service composition (RSC), associated with limited degradation of service quality in worse communication conditions rather than absolute availability/unavailability. Thus, the communication network can offer a certain level of functionality for the service even when it is not possible to fully achieve desiderata. Moreover, the architecture has not only to achieve ultra-reliability, but also to fulfill the stringent requirements of ultra responsiveness and very low latency; this leads to the so-called ultra-reliable and low-latency communication (URLLC) services [13]. The combination of these two aspects makes the problem of deploying such critical applications still open and is particularly important in networks with considerable time-varying delays and packet losses [2].

These interactive communications go beyond traditional video streaming. While conventional video and voice applications naturally allow for graceful degradation of network performance, *e.g.*, adaptive video coding, this may not work for this new class of applications, such as haptic. Since delayed arrival or loss of critical haptic data may run into the system's instability, haptic communication is vulnerable to different types of artifacts, leading to undesirable strong forces and surface roughness, *e.g.*, the erroneous sensation of being in contact with a significantly rough surface.

Similarly, a telemedicine session or, more specifically, a telepathology session, transmits delay and bandwidth-sensitive data to be processed and shared with a remote medical doctor. For this reason, a proper edge computing-enabled system can help partially or fully offload processes at the edge of the network [14], [15]. While it is clear that emerging technologies can drastically improve service interactivity, these solutions still require excessive overhead in essential network operations. Learning-based approaches can help towards automating actions, but it is not sufficient. These recent applications motivate our proposed approach, which attempts to help network management.

III. Integrating Control and Data Planes for Network Automation

As suggested by recent trends, there is now increasing interest in equipping networks with autonomous run-time decision-making capability via the incorporation of artificial intelligence (AI) and machine learning (ML). Given the fact that it is almost impossible for human operators to render network management in real-time, it is likely that future networks will apply AI/ML to *autonomously* identify and locate congestion or malfunctions in the network, and opportunely react. To accurately configure and manage itself, the network needs to pinpoint the malfunction, collect and analyze measurements in a stream way. Once metrics are collected, the network reacts to address the sub-optimal behavior via network programmability.

While for many applications this timescale is appropriate, for many others, collecting network statistics from the control plane and react based on the knowledge mined is too slow. Other approaches based solely on data plane programmability are yet too static, and do not have the same programmability level of their control-plane counterparts. *We envision a re-design of the network architecture, that could combine a data plane for the collection of network statistics with an enhanced control plane*.

We argue that future networked applications would greatly benefit from a network architecture and a general-purpose network management protocol that would use programmability as a mechanism, and planes (management, intent, data, or control planes) as a policy. [1] Which plane a network manager or a network application can program should be flexible, and a decision of the network programmer. Not only data plane and control plane, but also management plane programmability or intent layer programmability should be considered [16].

High-performance network hardware solutions, *e.g.*, switches and routers, that support such a vision of integrated data-control planes do not yet exist. Today, we do have switches supporting either control plane protocols programmability, such as OpenFlow-enabled network components, or data plane, *e.g.*, Tofino [17].

By programming interfaces and programmable substrates, existing Software-Defined Networking (SDN) enables specifying and provisioning network-wide forwarding behaviors, also adapting how the network hardware itself forwards traffic.

[1] From the classical network management literature, it is known that a network architecture is composed by (i) an identification of the mechanisms, and (ii) the separation of functionalities. A network mechanism is an invariance — an invariant aspect of a networking problem, while a policy is a variant aspect of such mechanism. For example, acknowledgment is a mechanism, when to acknowledge is a policy.

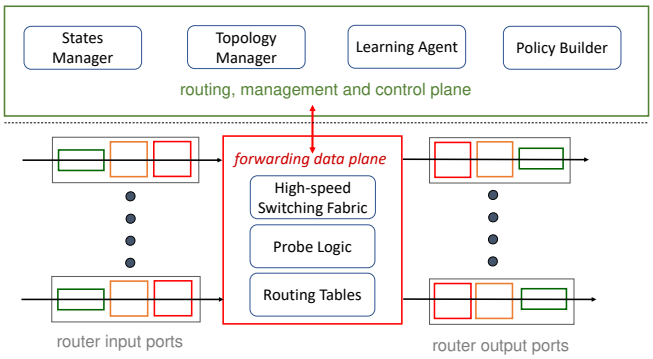

Fig. 1: Control plane (green) and data plane (red) processes interact with each other and optionally with a learning agent for an integrated data-driven network management.

One approach has been to use SDN solutions, composed of a centralized point for management [18]. At its core, SDN aims to make programming networks as easy as programming computers. However, these approaches based on centralized controllers are inherently too slow to respond to fine-grained traffic changes, as in short traffic bursts. Moreover, even when the software control plane is locally on the switches, the ability to select new routes is often limited and not fast enough.

To overcome these limitations, recent work has developed mechanisms that, operating entirely in the data plane, enable real-time adaptation [19]–[22]. By using fine-grained performance information on hardware timescales, these solutions can deliver considerable performance benefits over static mechanisms and centralized approaches. However, these techniques are limited to trivial performance-aware policies that are unable to learn during the execution and, consequently, adapt to multiple scenarios.

A data plane approach must be combined with control plane operations to realize a data-driven solution able to address the raised complexity. Fig. 1 highlights the main functionalities that should be present in the architecture. As shown in the figure, the data plane offloads complex tasks, such as routing and management, to the control plane, and the two planes communicate without disrupting normal switch operations.

Clearly, the logic and the knowledge regarding the environment reside in the control plane. Since its operations are generally orders of magnitude slower than the duration of network events, the learning agent can perform the training procedure required to learn the dynamics of the environment. During this phase, it needs to save the state (current and past) and the topology. On the other hand, the data plane processes messages and passes metrics, *e.g.*, counters or control logic inside the packet header, to the control space. In this plane also reside the tables used to route packets and the logic to write/read the telemetry information of packets.

We envision that this capability of the networks enables the advanced intelligent operations described previously. To do so, in our proposed architecture, packets can carry both probes and algorithm parameters that are then extracted by the traversed nodes. This piece of information can be introduced into packets as an additional header, *i.e.*, between the traditional packet header and user payload, for example by means of solutions such as BPP [23] and In-Situ OAM [24]. Fig. 2 summarizes the main components of our additional header: information about the network state and control plane parameters. The latter, for example, contains commands and metadata that provide guidance to intermediate network devices for how to process the packet. Using these techniques, our architecture also allows the user to drive the execution of programs on network nodes.

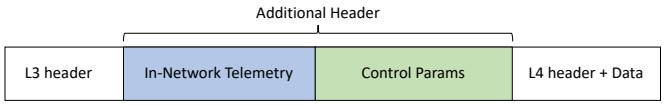

Fig. 2: Additional header structure to include in the packet. It carries information for both data and control plane.

The last feature required in our proposed architecture is the possibility to adapt and reconfigure such programs running at network nodes. This can be done by deploying solutions such as P4 [1], a technology for reconfigurability of parsing and processing of packets. The P4 language enables to program packet processing pipelines, allowing to define custom parsing rules and new protocol logic, thus being compliant with a solution like ours, based on the presence of additional flexible headers. In particular, P4's control model follows the SDN architecture and involves a separate control plane to deploy commands directly on networking devices. Since our header defines a control model that involves commands and data carried in the packets themselves, processed as packets traverse the network, P4 can facilitate the implementation of our proposed framework, making its usage smoother. Moreover, P4 offers a switch abstraction that is independent of the actual hardware. Indeed, P4 programs are compiled to a target-independent representation (front-end), and then compiled again to different specific platforms, *e.g.*, fixed-function switch ASICs, NPUs, reconfigurable switches, software switches, NetFPGA [25].

By combining programmable switches with extensible headers, control information, carried in the header of the packets, is locally processed. This would remove the central controller of traditional SDN architectures, *e.g.*, OpenFlow, where the switches need an external entity to decide flowing rules. In such a way, networks can finally implement a data-driven strategy that adapts on-the-fly the encountered (or predicted to be) problem and pave the way to accomplish strict requirements of critical applications.

IV. Enabling High-Performance Data-Driven Networking

In the following we describe four possible algorithms to prove the viability of this architecture, and to demonstrate that the network programming framework can indeed be used to support applications with real-time networking demands without needing to deploy custom software in networking devices or even controllers.

A. Use Case 1: Data-Driven routing

Traditional solutions operate on the routing or flow path decisions, using source routing and policy-based routing for the path control. However, to make these decisions adaptive to the current network utilization, the algorithm requires the ability to collect information network traffic that can be achieved via NetFlow [26], SNMP, or custom protocols. A common scenario, due to the ease of use of SDN in prototyping, considers OVS switches [27] featuring OpenFlow, combined with an SDN-controller to obtain the above requisites. These features are key for dynamic and performance-aware routing solutions.

Recent advances towards active networks and their dynamic behaviors have been possible thanks to advances in SDN, which have enabled more flexible and predictable network control, made it easier to extend the network with new functionality, and made it possible to verify the correctness of network behaviors. It is not surprising, then, that we are witnessing an increased number of networking frameworks combining an SDN-enabled profiling phase and an ML reaction phase. For example, in [28], the architecture adapts the routing strategy of the underlying edge network based on the estimated future available bandwidth. An SDN controller gathers global traffic condition information, keeps track of past link loads, and takes a new route if the current path is predicted to be congested.

The desire to avoid bottlenecks as for a centralized controller has pushed new studies to consider a multi-agent setting. [29], [30] present valid solutions using multi-agent approaches. While splitting the logic among multiple controllers partially reduces the overhead and solves the single-point-of-failure issue, it still requires additional control traffic that can create conflict with the application traffic.

To further optimize these decisions, in-network telemetry can be processed directly in the data plane. Hula [21], a state-of-the-art solution for utilization-aware routing in data centers, uses load-aware criterion to always choose the least-utilized path among all shortest paths. Similar to Hula, also CONGA [19] is a recent data-plane technique that overcomes ECMP's limitations by using link utilization information to balance load across paths. These solutions, sharing a distributed approach that avoids a central controller, are extremely responsive because they operate in the data plane, permitting it to make load-balancing decisions every few microseconds.

However, since the processing occurs in the data plane, the implemented policies are limited to mere strategies contemplating, for example, links utilization. By combining control and data planes, more complex decisions can be taken, to either anticipate likely predicted congestion or optimize a long-term revenue. Extensible headers are, thus, the enabling technology, carrying latency metadata, link information, and algorithm parameters as part of the packet itself. This allows the solution to react to unforeseen variations in the path, adjusting the forwarding as needed. Besides, to mitigate the heavy computational resources utilization and reduce the amount of data to transport, Federated Learning (FL) helps aggregate the learning processes of different nodes and thus, achieve robustness to different network conditions [31]. In such a way, the node only aggregates model parameters, and data remain local to reduce the bytes exchanged and the operations performed.

Moreover, if, as in [32], the header includes latency metadata, such as the latency objective, network devices can "slow down" or "speed up" packets as needed to meet the given end-to-end latency objective. While this solution can clearly optimize the routing decisions, so that the packet flows on the path with expected lower latency, it can also save network resources, *e.g.*, available bandwidth. In cases where a requested latency is physically impossible to meet, the node immediately drops these packets to reduce congestion, instead of forwarding them all the way to the destination only to have them dropped there because they are late. In these circumstances, the network can also provide exceeded-latency notifications.

B. Use Case 2: Auto-Scaling Networks

Focusing on network reliability and network elasticity, *i.e.*, the subproblem of autonomous networks that deals with the ability to auto-scale resources up and down, it is important that scaling happens in harmony with changes in the environment, *e.g.*, traffic demand. The advantages brought by such techniques are multiple. By deactivating resources that may increase unnecessary (energy) costs, they reduce resource management costs (scale down). At the same time, the network can provide redundant facilities to reroute traffic when workload peaks to unexpected levels (scale up). For example, [33] presents a proactive ML-based approach that auto-scales VNFs in telco and cloud networks in response to dynamic traffic changes.

In this context, reinforcement learning (RL) has been widely studied and applied as enabling technology, given its ability to automatically make decisions. By online capturing the performance model of a target application and its policy without any *a priori* knowledge, RL can well fit auto-scaling problems. The problem of dynamic resource allocation was largely studied in the literature, where [34], [35] are examples of a profitable usage of RL. To face the greater or smaller demand, these studies attempt to optimize the allocation of tasks, services, and Virtual Machines (VMs).

Thanks to the advantages in SDN, auto-scaling solutions can also be applied on network switches, as in [36]. In this solution, SDN controllers monitor the status of nodes, that are scaled up and down to accommodate the traffic demand and reacts to possible failures. Decisions aim to optimize network performance and users' experience, exploiting SDN to promptly change the configuration. While results are promising and benefits are noticeable, existing solutions are limited to the presence of a (logical or physical) central controller entity. As in routing problems, processing information via P4 would dramatically shorten control operations and data-plane

adaptation. The deployment of extensible headers, thus, is fundamental both for monitoring and for changing reasoning.

C. Use Case 3: Physical Layer Decisions

The design and development of novel Reinforcement Learning (RL) algorithms are key for effective and efficient real-time optimization of the physical layer of next-generation wireless networks. Existing work has explored RL, and its well-known variant Deep Reinforcement Learning (DRL), only from a theoretical perspective, substantially leaving the investigation of several critical system-level issues [37].

There is a significant knowledge gap in architectures, efficient protocols, and kernel implementations to dynamically update network Management-Information Base (MIB) states. Examples of MIB are the routing table, or Routing Information Base (RIB), and the forwarding table (FIB). To make informed and efficient decisions on wireless edge network management, such MIBs could be populated with network *and* radio signal parameters arriving from functions or middleboxes implemented across the whole networking stack. Common parameters to consider are the Signal-to-Noise Ratio (SNR), or other physical layer key-performance indicators, in-network states, such as a statistic about the queuing delay, or end-to-end parameters, such as the congestion window size.

The challenge resides in integrating network mechanisms that have different scope, *i.e.*, range of operation with machine learning parameters. For example, *(i)* an asynchronous Deep Reinforcement Learning (DRL) algorithm that, during the training phase, learns how to select the best policy according to the specific task at hand, and *(ii)* a synchronous, hardware-based subsystem, that makes decisions during the execution phase periodically gathering data from the network, enforcing the execution of the machine learning policy. While it is clear that the execution should run within the data plane, the training process, given its time-consuming nature, should occur in the control plane. Where to run an online training phase is an open question: if an external machine, possibly equipped with GPU, is used, then the model needs to be sent to the (IoT/networked) device by means of dedicated protocol, for example [23]. If instead, the networked device is also responsible for the training process, enough resources should be allocated in order to meet strict latency and computational (wireless) constraints.

One potential mechanism to tune with such envisioned integration of control and data planes via a DRL action would be the selection of the best channel based on the signals received. For example, avoiding weak Wi-Fi signals. To do so, it is necessary to design and implement, using real software-defined radio, SDN-driven solutions to serve the needs of such wireless learning engines for signal recognition. Other possible DRL actions could optimize fairness in spectrum sharing to tackle spectrum scarcity in the sub-6 GHz bands [38]. Some key unaddressed challenges of spectrum sensing include (i) extremely low latency constraints over large bandwidths to detect tiny spectrum holes and (ii) strict real-time digital signal processing (DSP) constraints; (iii) its underlying algorithms need to be highly accurate, and flexible enough to work with different wireless bands. While legacy spectrum sensing techniques rely on ad-hoc, protocol-dependent feature extraction techniques, they do not generalize to different protocols and spectrum environments. A DRL algorithm with integrated data and control planes could overcome these limitations.

Moreover, it is worth noticing how in this scenario, our proposed protocol (Fig. 2) should meet specific requirements of the wireless technology, as in the 802.11 standard. Therefore, the additional header would be placed on top of the MAC header, as already suggested by recent solutions [39].

D. Use Case 4: High-Performance Transport

Congestion control protocols are often classified into two categories according to the metrics used: in-network and end-to-end congestion controls. However, since there exist advantages on both sides, we argue that to better support the decision process of network agents, both in-network and end-to-end statistics can be combined. Inspired by recent work [40], this approach would bring benefits to congestion avoidance decisions, increasing the visibility of the network. To enable highly performant transport, our header can contain metrics inserted by network nodes, and when one of these nodes receives the packet, it includes its information or modifies existing fields. Having all these values in one place, *i.e.*, packet header, the processing can contemplate both classes of states, and network telemetry is dramatically simplified.

V. Conclusion and Next Steps

One of the main challenges that future networking applications will face concerns the ability to support high-precision services that adhere to stringent service level objectives. The highly flexible format of packets would help combine the ability to express complex packet processing semantics with compact encoding. This approach, along with a new architecture and logic of switches, unleashes several exciting research opportunities, such as the design of new packet processing pipelines with capabilities to process incoming information. To meet application requirements, routing protocols can autonomously adapt to the current network condition and the transported data, changing the logic on the fly. However, more research is needed to study the applicability in real networks and to capture the missing aspects while considering specific applications. Our next steps include designing new data-driven algorithms to support critical scenarios, as well as the definition of action primitives in packets format and switch actions. We also plan to build solutions and test for a more extensive evaluation.

Acknowledgment

This material is based upon work supported in part by NSF Awards CNS-1836906 and CNS-1908574.

References

[1] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[2] G. P. Fettweis, "The tactile internet: Applications and challenges," *IEEE Vehicular Technology Magazine*, vol. 9, no. 1, pp. 64–70, 2014.

[3] Z. Akhtar, Y. S. Nam, R. Govindan, S. Rao, J. Chen, E. Katz-Bassett, B. Ribeiro, J. Zhan, and H. Zhang, "Oboe: auto-tuning video abr algorithms to network conditions," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '18)*, 2018, pp. 44–58.

[4] F. Y. Yan, H. Ayers, C. Zhu, S. Fouladi, J. Hong, K. Zhang, P. Levis, and K. Winstein, "Learning in situ: a randomized experiment in video streaming," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI '20)*, 2020, pp. 495–511.

[5] K. Du, A. Pervaiz, X. Yuan, A. Chowdhery, Q. Zhang, H. Hoffmann, and J. Jiang, "Server-driven video streaming for deep learning inference," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '20)*, 2020, pp. 557–570.

[6] J. Kim, Y. Jung, H. Yeo, J. Ye, and D. Han, "Neural-enhanced live streaming: Improving live video ingest via online learning," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '20)*, 2020, pp. 107–125.

[7] C. Han, Y. Wu, Z. Chen *et al.*, "Network 2030 a blueprint of technology, applications and market drivers towards the year 2030 and beyond," 2018.

[8] J. Day, I. Matta, and K. Mattar, "Networking is ipc: a guiding principle to a better internet," in *Proceedings of the 4th International on Conference on emerging Networking EXperiments and Technologies (CoNEXT '08)*. ACM New York, NY, USA, 2008, pp. 1–6.

[9] A. Anand, F. Dogar, D. Han, B. Li, H. Lim, M. Machado, W. Wu, A. Akella, D. G. Andersen, J. W. Byers *et al.*, "Xia: An architecture for an evolvable and trustworthy internet," in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks (HotNets '11)*, 2011, pp. 1–6.

[10] A. Venkataramani, J. F. Kurose, D. Raychaudhuri, K. Nagaraja, M. Mao, and S. Banerjee, "MobilityFirst: A Mobility-Centric and Trustworthy Internet Architecture," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 74–80, 2014.

[11] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos *et al.*, "Named Data Networking (NDN) Project," *Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC*, vol. 157, p. 158, 2010.

[12] R. Li, K. Makhijani, and L. Dong, "New ip: A data packet framework to evolve the internet," in *2020 IEEE 21st International Conference on High Performance Switching and Routing (HPSR)*. IEEE, 2020, pp. 1–8.

[13] P. Popovski, "Ultra-reliable communication in 5g wireless systems," in *1st International Conference on 5G for Ubiquitous Connectivity*. IEEE, 2014, pp. 146–151.

[14] A. Sacco, F. Esposito, P. Okorie, and G. Marchetto, "Livemicro: An edge computing system for collaborative telepathology," in *Proceedings of the 2nd USENIX Workshop on Hot Topics in Edge Computing (HotEdge '19)*. USENIX Association, 2019.

[15] A. Sacco, F. Esposito, G. Marchetto, G. Kolar, and K. Schwetye, "On edge computing for remote pathology consultations and computations," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 9, pp. 2523–2534, 2020.

[16] Intent-based networking - concepts and definitions. [Online]. Available: https://tools.ietf.org/html/draft-irtf-nmrg-ibn-concepts-definitions-01

[17] Tofino barefoot, world's fastest p4-programmable ethernet switch asics. [Online]. Available: http://barefootnetworks.com/products/brief-tofino/

[18] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, 2013.

[19] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, R. Pan, N. Yadav *et al.*, "CONGA: Distributed congestion-aware load balancing for datacenters," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication (SIGCOMM '14)*. ACM New York, NY, USA, 2014, pp. 503–514.

[20] K.-F. Hsu, R. Beckett, A. Chen, J. Rexford, and D. Walker, "Contra: A programmable system for performance-aware routing," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. USENIX Association, 2020, pp. 701–721.

[21] N. Katta, M. Hira, C. Kim, A. Sivaraman, and J. Rexford, "Hula: Scalable load balancing using programmable data planes," in *Proceedings of the Symposium on SDN Research (SOSR '16)*. Association for Computing Machinery, 2016, pp. 1–12.

[22] L. Yu, J. Sonchack, and V. Liu, "Mantis: Reactive programmable switches," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '20)*, 2020, pp. 296–309.

[23] R. Li, A. Clemm, U. Chunduri, L. Dong, and K. Makhijani, "A new framework and protocol for future networking applications," in *Proceedings of the 2018 Workshop on Networking for Emerging Applications and Technologies (NEAT '18)*, 2018, pp. 21–26.

[24] F. Brockners, S. Bhandari, C. Pignataro, H. Gredler, J. Leddy, S. Youell, T. Mizrahi, D. Mozes, P. Lapukhov, R. Chang *et al.* (2020) Data fields for in-situ oam, draft-ietf-ippm-iom-data-09. [Online]. Available: https://tools.ietf.org/html/draft-ietf-ippm-ioam-data-09

[25] S. Ibanez, G. Brebner, N. McKeown, and N. Zilberman, "The p4 –> netfpga workflow for line-rate packet processing," in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2019, pp. 1–9.

[26] Cisco ios netflow. [Online]. Available: http://www.cisco.com/web/go/netflow

[27] Open vswitch: An open virtual switch. [Online]. Available: http://www.openvswitch.org/

[28] A. Sacco, F. Esposito, and G. Marchetto, "Rope: An architecture for adaptive data-driven routing prediction at the edge," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 986–999, 2020.

[29] X. Zhao, C. Wu, and F. Le, "Improving inter-domain routing through multi-agent reinforcement learning," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2020, pp. 1129–1134.

[30] X. You, X. Li, Y. Xu, H. Feng, J. Zhao, and H. Yan, "Toward packet routing with fully distributed multiagent deep reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.

[31] A. Sacco, F. Esposito, and G. Marchetto, "A federated learning approach to routing in challenged sdn-enabled edge networks," in *Proceedings of the 6th IEEE Conference on Network Softwarization (NetSoft '20)*. IEEE, 2020, pp. 150–154.

[32] A. Clemm and T. Eckert, "High-precision latency forwarding over packet-programmable networks," in *2020 IEEE/IFIP Network Operations and Management Symposium (NOMS '20)*. IEEE, 2020, pp. 1–8.

[33] S. Rahman, T. Ahmed, M. Huynh, M. Tornatore, and B. Mukherjee, "Auto-scaling vnfs using machine learning to improve qos and reduce cost," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6.

[34] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks (HotNets '16)*. ACM New York, NY, USA, 2016, pp. 50–56.

[35] X. Dutreilh, S. Kirgizov, O. Melekhova, J. Malenfant, N. Rivierre, and I. Truck, "Using reinforcement learning for autonomic resource allocation in clouds: towards a fully automated workflow," in *ICAS 2011, The Seventh International Conference on Autonomic and Autonomous Systems*, 2011, pp. 67–74.

[36] A. Sacco, F. Matteo, Flocco andEsposito, and G. Marchetto, "Supporting sustainable virtual network mutations with mystique," *IEEE Transactions on Network and Service Management*, 2021.

[37] F. Restuccia and T. Melodia, "DeepWiERL: Bringing Deep Reinforcement Learning to the Internet of Self-Adaptive Things," *IEEE Conference on Computer Communications (INFOCOM '20)*, pp. 844–853, 2020.

[38] H. Qi, X. Zhang, and Y. Gao, "Low-Complexity Subspace-Aided Compressive Spectrum Sensing Over Wideband Whitespace," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 12, pp. 11 762–11 777, 2019.

[39] E. Coronado, A. Thomas, S. Bayhan, and R. Riggio, "aios: An intelligence layer for sd-wlans," in *2020 IEEE/IFIP Network Operations and Management Symposium (NOMS)*. IEEE, 2020, pp. 1–9.

[40] A. Sacco, F. Matteo, F. Esposito, and G. Marchetto, "A federated learning approach to routing in challenged sdn-enabled edge networks," in *IEEE Conference on Computer Communications (INFOCOM '21)*. IEEE, 2021.