Towards Autonomous Computer Networks in Support of Critical Systems

Towards Autonomous Computer Networks in Support of Critical Systems

Alessio Sacco * Guido Marchetto *
* Department of Control and Computer Engineering, Politecnico di Torino, Italy

***Abstract*—A recent trend dictating evolution of management and orchestration of computer networks is constituted by the softwarization and virtualization of them, which have drastically simplified the deployment and real-time reconfiguration of network functions, allowing them to continuously adapt and to deal with dynamic demands in an automated way. Alongside, recent management and orchestration approaches for softwarized networks employ Artificial Intelligence (AI) and Machine Learning (ML) to further reduce reaction time and improve the accuracy of decisions, where the network operations can be automated to the point of realizing autonomous driving networks. However, while automating operations can improve the overall system (it is acknowledged that 70% of network faults are caused by manual errors), AI/ML methods are not the panaceas, and we are still far from having a fully operating and efficient automated architecture. In this dissertation, we present a novel class of software network solutions that share the goal of enabling intelligent and autonomous computer networks, exploring how to exploit the power of AI/ML to handle the growing complexity of critical systems. We start with a new network management scheme for adaptive routing and autonomous scaling of virtual network resources. Then, acting on the hosts, we propose to adjust the TCP congestion control with a ML-based solution, whose goal is to select the proper congestion window learning from end-to-end features and (when available) network signals. We believe that the proposed solutions, and their combination, can lay the foundation for automated systems that better suit modern edge environments and cellular networks by providing unprecedented flexibility and adaptation to even unseen and unknown network conditions.**

***Index Terms*—machine learning, computer networks, congestion control**

I. Introduction

In recent years, communication networks have undergone a radical evolution towards more programmability and flexibility, mainly due to the so-called "network softwarization". The combination of such network softwarization and new architectures opened up new opportunities. For example, edge computing, a novel paradigm in which resources, instead of residing in the cloud, are moved closer to the sender (i.e., at the edge of the network) can facilitate the rise of new interactive systems, promising simultaneously (ultra) low-latency, high-bandwidth, and reliable telecommunications. Together, 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) [2] are showing promising business use cases, supporting several applications [3], [4]. Among all examples of these networked systems and applications, in this dissertation the focus is on the so-called "critical systems", deployed for life improvement and sometimes even life-saving services. Consider, e.g., a remote surgery operation, where the system should lead to an improvement in the accuracy and dexterity of a surgeon while minimizing trauma to the patient [5]. Similarly, a telepathology session, in which histological imagery is transmitted over delay and bandwidth-sensitive path to be processed and shared with a remote medical doctor for real-time diagnosis or pre-computation of digital pathology [6]. On the one hand, these novel paradigms are opening new applications and business opportunities. On the other hand, however, they are opening new network and application management problems.

It thus appears how there is a need for new effective and efficient management of softwarized networks and services to cope with the unfolding plethora of opportunities provided by softwarization. Such flexibility does not necessarily have to be addressed by selecting a configuration once, but systems can be adapted continuously and be able to deal with dynamic demands in an automated way [7]. In this perspective, we argue that Artificial Intelligence (AI) and Machine Learning (ML) can play a central role in managing and orchestrating softwarized networks for the following reasons.

First, the tools that network operators use to gather data from the network have not changed appreciably in decades, even as both demands on the network and traffic volumes have increased. The network data collection, the analysis of such data, and the decision of whether and how to adapt the network's configuration in response to changing network conditions (e.g., a shift in traffic demand, an attack), still remain three decoupled steps. Operators perform network management and network optimization tasks on several timescales, often relying on operators' experience and cumbersome adjustments.

Second, new security and performance requirements create a growing need for new approaches to real-time network management that exploit the growing capabilities in programmable networks and systems to support the analysis of real-time streaming data. Despite recent advances in algorithmic and system aspects of streaming applications, the set of queries that network management requires is significantly more extensive than current methods can handle.

Third and lastly, due to the increasing popularity of Internet-connected devices and the various applications that run over the network, the expectations for network reliability and performance are greater than ever. To achieve these goals, network operators must continuously collect and analyze the various data streams from the network, possibly with lightweight yet

accurate methods.

Therefore, we believe that networks can be equipped with AI/ML to achieve autonomous decision-making capabilities at run-time, as also suggested by recent trends [8]–[10]. Considering that it is nearly 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. For example, 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. These mechanisms can help solve the challenges introduced by interactive, reliable, and low-latency communication of critical systems [5]. A proper network infrastructure handling the traffic of applications with strict requirements cannot be an afterthought in network management.

In this dissertation, we present a novel class of software network solutions that share the goal of enabling intelligent and autonomous networks in order to facilitate the deployment of such interactive systems. The research question driving the work can be summarized as: *how these new algorithms (such as machine learning) and the new technologies (such as edge computing and SDN) can be used to effectively solve (traditional and not) network management problems*. In particular, our research contribution covers solutions that exploit the power of AI/ML to handle the growing complexity of communication networks in the context of resource-constrained, dynamic, and mission-critical environments of modern networks. In this dissertation we focus on the design and deployment of learning algorithms for some specific network operations, ranging from the the management of virtualized and softwarized network resources and their allocation, to the end-host modification with the similar aim of mitigating network congestion.

II. Network Management Operations

As users and traffic demands grow, the need to optimize our communication networks magnifies, denoting the evidence that networks dictate our technological world. Recent advantages in artificial intelligence (AI) and machine learning (ML) are paving the path to autonomous networks: networks that measure, analyze and control themselves autonomously. Such a network automation brings enormous benefits since it is almost impossible for human operators to render real-time network management [11]–[13].

A. Auto-Scaling Networks

A particular problem in network management is network reliability and elasticity, *i.e.*, the subproblem of autonomous networks that deals with the ability to auto-scale resources up and down, in harmony with changes in the environment, $e.g.$, traffic demand. The advantages brought by the auto-scaling techniques are multiple. They reduce the cost of resource management, by deactivating resources that may increase

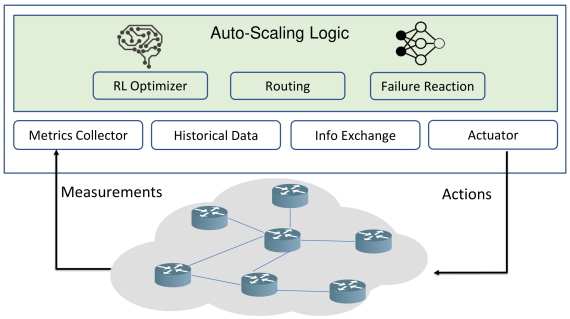


Fig. 1: System overview. The Software-Defined Network controller receives as input traffic statistics and outputs new flow routes and power on/off commands.

unnecessary (energy) costs. At the same time, the network can provide redundant facilities to reroute traffic when workload peaks to unexpected levels.

Traditional threshold-based and recent ML-based auto-scaling policies are often unable to address the high complexity of networks and consequently to satisfy the carrier-grade requirements of reliability and stability. Furthermore, state-of-the-art solutions hardly combine these features altogether, such as [14] whose primary goal is the energy efficiency, or [15], which automatically scales Virtual Network Function instances via an ML classifier. Although reinforcement learning is emerging as a valuable technique to solve many networking problems, as in [16], there is no solution incorporating network information to automatically and efficiently orchestrate network resources in a decentralized manner.

To this end, we propose Mystique, a network management schema that, using Multi-Agent Reinforcement Learning (MARL), auto-scales to accommodate the traffic demand and reacts to possible failures [17]. On the one hand, Mystique unburdens network nodes that are over-congested with traffic, to preserve the high bandwidth and high availability of the applications. On the other hand, it leverages healing strategies to repair failing nodes and links. Each MARL agent, a process running within an SDN controller, can learn an auto-scaling policy from experience, without any *a priori* knowledge or human intervention. By continuously monitoring the state of the network, the agent can make sharp decisions on how to optimize network performance and users' experience, exploiting SDN to promptly change the configuration. Moreover, the distributed nature of MARL makes it possible to exploit a (possibly) large number of SDN switches spread across the topology as probes. The system automatically re-balances both existing and new flows across nodes, while the agents communicate among them to obtain information about the other sub-network. The decisions taken by Mystique aim to maximize overall Quality of Experience (QoE) across multiple users and achieve a desired level of QoE fairness, while reducing the energy costs for active links and nodes.

Results validate our decentralized control plane, showing how Mystique can promptly adapt and modify its behavior to handle variations in workloads. Compared to other benchmark

solutions, our algorithm can jointly improve the user satisfaction and more wisely utilize the network resources.

The main functions of Mystique are to auto-scale according to the traffic demand and react to failures when they occur. We developed and implemented these features in a system depicted in Fig. 1. In this context, the controller monitors the state of each switch in its sub-network to detect if one of the following events occurs: the switch is overloaded (congestion), the switch in under-utilized and can be deleted (cost-saving), the switch fails and the connectivity can be no longer guaranteed (failure). However, the control plane is distributed to several controllers. Each of them controls a subset of switches and communicates with the other controllers, via the *Info exchange* process, to obtain a consistent network view. For any change in the controlled network region, e.g., new link, the controller notifies its peers. They also exchange the information required for computing the QoE for connected users. The reinforcement learning (RL) module selects the best action, *i.e.*, active network resources, but interacts with other processes to collect the information required for the decision and to notify about the outcome. In fact, we avail multiple processes to better separate concerns, but they cooperate to achieve our stated goal by implementing the following functionalities.

Routing. Each agent dynamically creates and destroys virtual switches and virtual links in response to network fault or substantial network traffic changes. This means that, in these events, the agent is also responsible for re-steering the traffic and deciding what flows to move in response to these actions. At the beginning, the route for each flow is selected by the controller based on the shortest path algorithm. In the case of multiple available paths between source and destination, a load balancing strategy is applied, *i.e.*, flows are equally distributed among the multiple paths. In the following, we separate the events to face during the execution with the aim of a more clear presentation.

In the case of a *link or node failure*, the same resource is re-created. For a link failure, a new edge is created connecting the same source node and destination node. The neighbor of the switch modifies the forwarding rules reflecting the new port ids. For a switch failure, a new one is generated with the same links that the faulty switch had. This implies that all the flows previously installed on the old switch are moved to the new one, and the neighbor nodes that were connected need to be re-instructed with the new ports.

When a *new link or switch is created*, the topology is analyzed, and the flows that can take advantage of the new path(s) are identified. Among them, a subset of flows, *i.e.*, in order to select half of the identified traffic, is transferred to the new alternative path. However, we remark that the number of flows to move is a consequence of a load balancing strategy that attempts to equally redistribute flows.

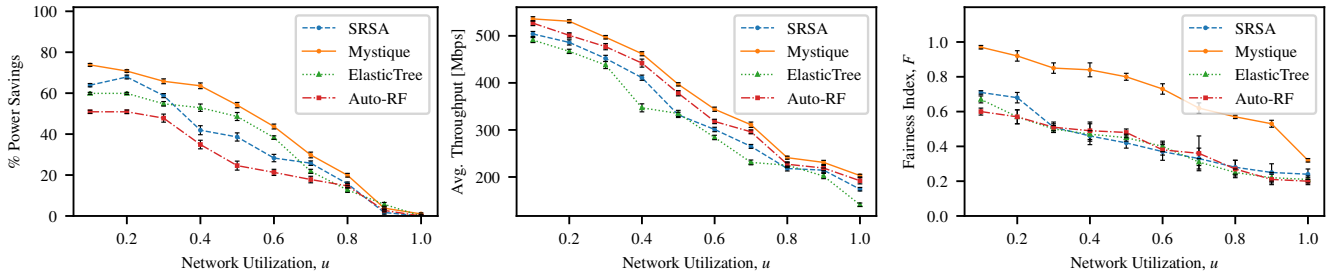
Finally, when *a link or switch is deactivated* due to energy saving considerations, all the flows traversing the deactivated item are considered, and a new path for each of them is set via the shortest path strategy.

Failure Reaction. We desire to react accordingly to the degree of the issue and take a proportional action. For this reason, while the utilization of the switch (and connected links) is handled by the RL model, a separate module manages the failure detection and reaction. Inspired by previous work [18], we consider 5 possible faults that can take place in our scenario: (1) communication with the controller ended, (2) timeout of the response expired, (3) port fault, (4) flows of a particular host have blocked unexpectedly by the switch, (5) unexpected behavior of the switch. As the controller continuously monitors the state of the switches, it can replace the switch in case of (1)-(2) (three consecutive timeout expirations)-(5). Instead, in case of (3)-(4), the link originating from the fault port is re-created.

The mechanism of fault reaction is in addition to the fast failover provided by new versions of OpenFlow. In this procedure, it is possible to install rules whose forwarding behavior depends on the local state of the switch. Hence, it allows fast failure recovery, as long as the SDN controller is able to anticipate every possible failure and precompute appropriate backup paths. However, OpenFlow fast failover is just to react to link failures, and no other events are taken into account, for example, switch failure. Even though this is equivalent to a failure of all the adjacent links, we argue that the controller can benefit from our model and adapt the routing and the application logic dynamically, as the network evolves. The fast failover is orthogonal to a reactive solution, as our model is. For this reason, both strategies are utilized for an improved fault reaction.

RL Optimizer. The optimizer's role is to find the network subset that satisfies current traffic conditions while avoiding the waste of resources. In our solution, this decision is taken by theh reinforcement learning agent using the one-step Q-learning algorithm. As input, it receives the topology, the power model of switches, and the current traffic conditions. These measurements are collected by the switches and reported periodically to the network controller, where resides the *metric collector* component. The collected data then feed the model on the agent, that outputs the best decision for the network itself, exploiting *historical data* to learn the goodness of a particular action upon the occurrence of similar conditions. The decisions regard activation of links in the sub-network. When the decision is made, the *actuator* receives the output consisting of the set of active components and performs the appropriate commands. Moreover, the actuator also pushes the new routes into the network.

In Fig. 2 we evaluate our solution against three state-of-the-art solutions adequately adapted to our use case: an ML classifier-based method to perform auto-scaling, [15], SRSA [19], and ElasticTree [14]. SRSA is a reinforcement learning approach to auto-scaling VMs in a telco cloud platform [19], while ElasticTree attempts to solve a power optimization problem using a greedy bin-packing heuristic. In particular, we report in the (a) energy efficiency, (b) application throughput, and (c) QoE fairness for the considered methods. By considering the plots, we can notice how our system outperforms the related algorithms in all the examined metrics.



(a) (b) (c)

Fig. 2: Performance comparison with other benchmark algorithms in terms of (a) energy saved with respect to original setting, (b) mean application throughput achieved, (c) fairness index for the flows.

In particular, fairness is distinctly one of the most improved quantities in Mystique, as one of the desiderata. Furthermore, none of the other algorithms can efficiently optimize more metrics simultaneously, but they can successfully improve only some of them. Conversely, Mystique stably outperforms other solutions, demonstrating its ability to optimize management costs and QoS parameters altogether in multiple network scenarios.

III. Host-Based Operations

A performing congestion control protocol is fundamental for proper network operation as it ensures telecommunication stability, fairness in computer network resource utilization, high throughput, and a low switch queuing delay. This is especially important in new throughput and/or delay-sensitive applications [20]. Although many solutions have been proposed in the last decade, Transport Control Protocol (TCP) still constitutes the overwhelming majority of current Internet and Long Term Evolution (LTE) communications, and the vast majority of congestion control mechanisms are implemented on TCP. In the following we present Owl, a new approach to control the congestion on hosts [21].

A. High-Performance Transport Pdrotocol

Various studies have shown how TCP performs poorly in scenarios that require adaptability or that departs from the original network conditions on which it was designed in the '70s [22]–[24]. In particular, problems may occur in cellular and wireless networks, where TCP misinterprets the stochastic packet losses as congestion, hence leading to performance degradation [24]. This issue has motivated many authors to propose innovative congestion control approaches that follow a domain-specific design philosophy, in which the design is limited to a specific network scenario and it leverages its specific characteristics to boost the performance. Examples are in data centers [25], [26] and edge networks [24]. Several TCP variations (e.g., PCC [27] and Copa [28], to mention a few) have been recently proposed to overcome these shortcomings. Nevertheless, the fixed rule strategies used by these solutions are often inadequate to adapt to the rapidly changing environment.

To solve the problem of an adequate congestion window update strategy, we have developed *Owl*, a novel transport protocol based on *reinforcement learning (RL)*. Although

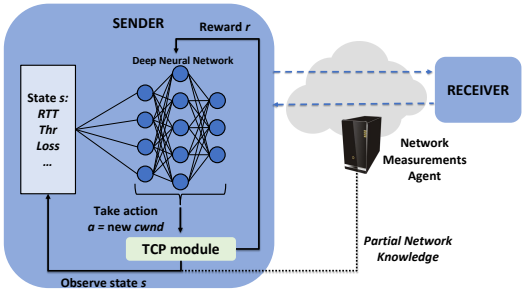


Fig. 3: Owl Overview: reinforcement learning sender's agent interaction with the network.

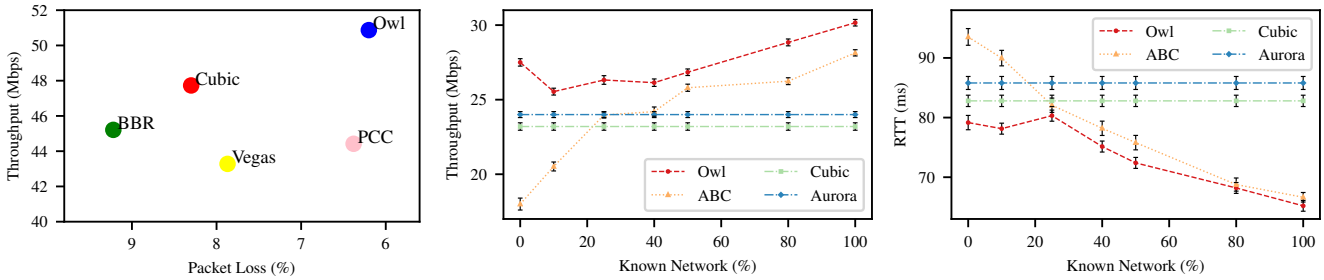
many transport protocols have been designed, with [29], [30] or without learning algorithms [25], [31], the most recent solutions using RL do not exploit network intelligence fully.

Our transport protocol *Owl* is able to increase the throughput and fairness while reducing the number of packets lost and delay by learning from several end-to-end and in-network metrics. In particular, our contributions are summarized as follows. We designed and implemented as a kernel module *Owl*[1], a new congestion control protocol that leverages partial network knowledge to train a reinforcement learning model based on Deep Q-Learning [32], improving the network performance with respect to recent work [33]. The outcome of Owl model is the next congestion window value, a crucial and volatile parameter for any reliable telecommunication.

Recently, RL has permeated many congestion control mechanisms, such as Orca [34] and Aurora [33], where in Aurora, the previous Performance-oriented Congestion Control (PCC) protocol was extended with a Deep-RL approach. Our RL approach differs from others for the agent state: we effectively combine features from both the transport and the network layers, without significant burdens to the Linux kernel module.

In Figure 3, we detail the main actions performed by the sender. The collected metrics are fed to the neural network of the model and the protocol starts. The host runs a RL model to decide the next congestion window (*cwnd*). The congestion window is one of the per-connection state variables that is used by TCP to limit the amount of data a sender can transmit

[1]As owls that (are wise and) can see with poor light conditions, our protocol operates with partially visible networks.



(a) GENI Testbed (b) Network Knowledge (c) Network Knowledge

Fig. 4: **(a) GENI testbed evaluation.** Throughput-loss rate trade-off for kernel-level solutions over real networks. Owl optimizes the two quantities simultaneously. **(b)-(c) Network Knowledge Impact on Performance.** Throughput and RTT of Owl protocol for increasing percentage of known network. Somehow surprisingly, the highest performance gaps with respect to other algorithms are obtained when the percentage of network knowledge is either low or very high.

before receiving an ACK. Since TCP was designed based on specific network conditions and handles all packet losses as network congestion, in wireless lossy links it unnecessarily lowers its rate by reducing the *cwnd* at each packet loss, negatively affecting the end-to-end performance. We, instead, want to encourage the RL agent to explore diverse ways to influence the connection by assigning different magnitudes to the performed change. Indeed, not only the learning agent should predict when increasing or decreasing the *cwnd*, but also to what extent. For example, our algorithm must learn when the network state suggests that a large part of the bandwidth is unused to aggressively increment the window size, while it must only slightly increase it when the network approaches any congestion. Our network module starts with an initial *cwnd* of 10. Due to the opted approach, the protocol learns how to make control decisions from experience and, thus, eliminates the need for necessary pre-coded rules to adapt to the variety of network environments.

To establish the practicality of our approach and understand how Owl performs over wide-area Internet paths with real cross-traffic and real packet schedulers, we deploy our solution on the GENI testbed. In these experiments, we evaluate how congestion control schemes behave across two federated GENI aggregates and three senders transmit simultaneously. We measure the performance of each schema when competing with another flow to accentuate the possible congestion occurrences. To evaluate our protocol in these realistic settings, we average the throughput and delay over 60-second flows, while the senders share a bottleneck link with 3ms RTT and a bandwidth of 100 Mbps. We summarize in Figure 4a the performance of our protocol when compared to other protocols available on Linux, as Cubic [35], Vegas [36], BBR [37], and an online learning protocol as PCC [27]. Our prototype evaluation deployed in real settings demonstrated that our implementation can jointly achieve high throughput and a low loss rate when compared to other solutions, balancing the two components effectively.

Next, we discuss our experiments regarding the impact of the required network state knowledge that Owl needs to train the RL system effectively. Figure 4b display the throughput and Figure 4c the RTT, when different transport protocols run over a network composed of 20 nodes emulated on our local Mininet virtual network testbed. Specifically, we compare against Cubic as a reference end-to-end congestion control, Aurora [33], as a reference RL-based congestion control, and ABC [31], as a reference of in-network congestion control. The performance of Cubic and Aurora are not affected by the lack of in-network knowledge since they are both end-to-end congestion control algorithms. On the other hand, ABC performs worse than Owl when the number of ABC-compliant routers is relatively low. This result is important for two aspects: our protocol works even as a pure end-to-end strategy, it can handle a partial network information and not all intermediate routers necessarily have to be Owl-compatible Our measurements reveal also that our solution outperforms both end-to-end approaches (such as Cubic) and novel in-network protocols (such as ABC), even when less than 50% of switches are utilized to collect statistics. On the other hand, if a partial network knowledge (more than 50%) is available, Owl drastically speeds up the transmission in terms of throughput and reduces latency. The worst result occurs approximately when half of the devices are controlled, as the agent cannot assign the proper importance to the coming information, resulting in occasionally misleading values. Nonetheless, even though in this scenario the information does not help improve the overall performance, Owl has higher throughput and reduced delay than other protocols.

IV. Conclusion and Future Work

This dissertation, whose extended version is publicly available at [38], exposed a new direction for managing critical systems. We explored novel methodologies aimed to solve some typical networking management problems with the inclusion of reactive mechanisms based on Machine Learning methods. In particular, we demonstrated how ML-based algorithms can be effectively used in optimizing network operations, ranging from the management of virtualized and softwarized networks to the congestion mitigation at the end-hosts. As we deepened only a subset of the existing problems, we hope that future research will cover the remaining challenges that, if solved, could lead to the definition of a fully autonomous

and functional network that can enable a better machine-human hybrid architecture. Among the future challenges, we can cite, for example, the need to cope with a few or partial data, which can hinder an accurate learning process. Taking network management decisions when the telemetry is limited may move current solutions towards production.

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] "Data plane development kit." https://www.dpdk.org/.

[3] A. Sacco, F. Esposito, and G. Marchetto, "On control and data plane programmability for data-driven networking," in *2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR)*. IEEE, 2021, pp. 1–6.

[4] A. V. Ventrella, F. Esposito, A. Sacco, M. Flocco, G. Marchetto, and S. Gururajan, "APRON: an Architecture for Adaptive Task Planning of Internet of Things in Challenged Edge Networks," in *Proceedings of the 8th IEEE International Conference on Cloud Networking (CloudNet '19)*, 2019, pp. 1–6.

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

[6] 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.

[7] N. Feamster, A. Gupta, J. Rexford, and W. Willinger, "Nsf workshop on measurements for self-driving networks," in *Workshop on Measurements for Self-Driving Networks was held at Princeton University on April*, vol. 4, 2019, p. 5.

[8] W. Kellerer, P. Kalmbach, A. Blenk, A. Basta, M. Reisslein, and S. Schmid, "Adaptable and data-driven softwarized networks: Review, opportunities, and challenges," *Proceedings of the IEEE*, vol. 107, no. 4, pp. 711–731, 2019.

[9] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 1, p. 16, 2018.

[10] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications surveys & tutorials*, vol. 21, no. 3, pp. 2224–2287, 2019.

[11] A. Sacco, F. Esposito, and G. Marchetto, "A Federated Learning Approach to Routing in Challenged SDN-Enabled Edge Networks," in *6th IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2020, pp. 150–154.

[12] A. Sacco, M. Flocco, F. Esposito, and G. Marchetto, "An Architecture for Adaptive Task Planning in Support of IoT-based Machine Learning Applications for Disaster Scenarios," *Computer Communications*, vol. 160, pp. 769–778, 2020.

[13] 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.

[14] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "Elastictree: Saving energy in data center networks," in *Nsdi*, vol. 10, 2010, pp. 249–264.

[15] 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.

[16] 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)*, 2016, pp. 50–56.

[17] A. Sacco, F. Matteo, Flocco andEsposito, and G. Marchetto, "Supporting sustainable virtual network mutations with mystique," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 2714–2727, 2021.

[18] B. Chandrasekaran and T. Benson, "Tolerating sdn application failures with legosdn," in *Proceedings of the 13th ACM workshop on hot topics in networks (HotNets '14)*, 2014, pp. 1–7.

[19] P. Tang, F. Li *et al.*, "Efficient auto-scaling approach in the telco cloud using self-learning algorithm," in *IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–6.

[20] 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.

[21] A. Sacco, F. Matteo, F. Esposito, and G. Marchetto, "Owl: Congestion control with partially invisible networks via reinforcement learning," in *IEEE INFOCOM-IEEE Conference on Computer Communications*. IEEE, 2021.

[22] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck, "An in-depth study of lte: effect of network protocol and application behavior on performance," in *ACM SIGCOMM Computer Communication Review*, vol. 43. ACM, 2013, pp. 363–374.

[23] H. Jiang, Y. Wang, K. Lee, and I. Rhee, "Tackling bufferbloat in 3g/4g networks," in *Proceedings of the 2012 Internet Measurement Conference*. ACM, 2012, pp. 329–342.

[24] Y. Zaki, T. Pötsch, J. Chen, L. Subramanian, and C. Görg, "Adaptive congestion control for unpredictable cellular networks," in *ACM SIGCOMM Computer Communication Review*, vol. 45. ACM, 2015, pp. 509–522.

[25] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data Center TCP (DCTCP)," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication (SIGCOMM '10)*. ACM New York, NY, USA, 2010, pp. 63–74.

[26] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, "Better never than late: Meeting deadlines in datacenter networks," in *ACM SIGCOMM Computer Communication Review*, vol. 41. ACM, 2011, pp. 50–61.

[27] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira, "Pcc: Re-architecting congestion control for consistent high performance," in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, Oakland, CA, 2015, pp. 395–408.

[28] V. Arun and H. Balakrishnan, "Copa: Practical delay-based congestion control for the internet," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, 2018, pp. 329–342.

[29] Y. Kong, H. Zang, and X. Ma, "Improving tcp congestion control with machine intelligence," in *Proceedings of the 2018 Workshop on Network Meets AI & ML*, 2018, pp. 60–66.

[30] W. Li, F. Zhou, K. R. Chowdhury, and W. Meleis, "Qtcp: Adaptive congestion control with reinforcement learning," *IEEE Transactions on Network Science and Engineering*, vol. 6, no. 3, pp. 445–458, 2018.

[31] P. Goyal, A. Agarwal, R. Netravali, M. Alizadeh, and H. Balakrishnan, "ABC: A simple explicit congestion controller for wireless networks," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, 2020, pp. 353–372.

[32] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[33] N. Jay, N. Rotman *et al.*, "A deep reinforcement learning perspective on internet congestion control," in *International Conference on Machine Learning (ICML)*, 2019, pp. 3050–3059.

[34] S. Abbasloo, C.-Y. Yen, and H. J. Chao, "Classic meets modern: A pragmatic learning-based congestion control for the internet," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '20)*, 2020, p. 632–647.

[35] S. Ha, I. Rhee, and L. Xu, "Cubic: a new tcp-friendly high-speed tcp variant," *ACM SIGOPS operating systems review*, vol. 42, no. 5, pp. 64–74, 2008.

[36] L. S. Brakmo and L. L. Peterson, "Tcp vegas: End to end congestion avoidance on a global internet," *IEEE Journal on selected Areas in communications*, vol. 13, no. 8, pp. 1465–1480, 1995.

[37] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control," *Queue*, vol. 14, no. 5, pp. 20–53, 2016.

[38] A. Sacco, "Towards autonomous computer networks in support of critical systemss," Ph.D. dissertation, Politecnico di Torino, 2022. [Online]. Available: https://hdl.handle.net/11583/2968454