Innovative Approaches for Network Analysis and Optimization: Leveraging Deep Learning and Programmable Hardware

1st Ariel Góes de Castro
Universidade Estadual de Campinas (UNICAMP)
Campinas - SP, Brazil
https://orcid.org/0000-0002-5391-5082

2nd Christian Esteve Rothenberg
Universidade Estadual de Campinas (UNICAMP)
Campinas - SP, Brazil
https://orcid.org/0000-0003-3109-4305

***Abstract*—Network demand for real-time applications like self-driving cars and cloud gaming strains existing networks. Latency and congestion hurt user experience. Realistic testing is vital to improving networks, but real-world data is scarce. In this context, we propose to analyze existing network data and identify traffic patterns and anomalies. We believe this knowledge can be used to feed generative adversarial network (GAN) models, which can create realistic synthetic data, supplementing existing real traces while protecting end-user privacy. This augmented data can then be used, for instance, to empower improved routing algorithms designed to benefit from programmable hardware (e.g., SmartNICs) and collected data plane metrics, paving the way for improved network performance and enhanced user experience with more autonomous decisions. This paper presents our initial analysis of synthetic network data generation technologies and summarizes the main ideas guiding my Ph. D. research.**

***Index Terms*—programmable hardware, neural networks, and network trace generation.**

I. Introduction

The surge in network demand fueled by real-time, data-intensive applications like healthcare, autonomous vehicles, and cloud gaming has intensified the need for enhanced network performance. However, production networks often suffer from issues like latency and congestion, impacting user experience and service quality. Also, obtaining real network data for these tasks is often challenging for several reasons. First, network data may contain sensitive or private information of the users or organizations, which raises ethical and legal issues for sharing or publishing them. Second, network data may be scarce or outdated [1], especially for emerging or evolving network scenarios (e.g., 5G, IoT, SDN). Third, network data may be biased or incomplete [2], [3], which limits the generalization and robustness of the network analysis models.

This work was partially supported by the Innovation Center, Ericsson S.A., and by the Sao Paulo Research Foundation (FAPESP), grant 2021/00199-8, CPE SMARTNESS.

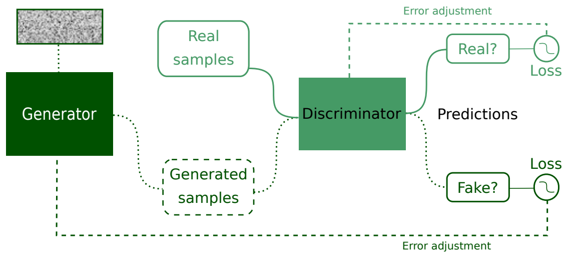

Fig. 1. GAN architecture overview.

To effectively develop new algorithms and solutions that cater to the diverse requirements of such applications under realistic conditions, it is imperative to test them in environments that closely resemble real-world scenarios. It necessitates using real devices and incorporating communication logs. To overcome the challenges mentioned above, network data generation techniques have been proposed to create synthetic or realistic network data that can be used for network analysis tasks. These techniques mimic the characteristics and behaviors of real network data, such as packet headers, payloads, flows, protocols, and traffic patterns (e.g., video streaming, VoIP). Network data generation techniques can also introduce variations and anomalies to the network data to simulate different network conditions and scenarios.

Network data generation techniques can be classified into two main categories: model-based and trace-based. *Model-based* [4], [5] techniques use mathematical models or statistical methods to generate network data from scratch. Model-based techniques can capture real network data's general properties and distributions, such as packet inter-arrival times, packet sizes, flow duration, and flow rate. Model-based techniques can also incorporate different network parameters and configurations, such as the number of hosts, connections, traffic types, and traffic volumes. However, model-based techniques may not be able to reproduce the specific features and dynamics of real network data, such as protocol-specific fields, application-specific contents, and temporal or spatial correlations. On the other hand, *trace-based* techniques [6] use existing network traces or PCAPs as inputs to generate new network data. It can preserve the realistic and detailed aspects of real network data, such as packet headers, payloads, flows, protocols, and traffic patterns. With this new paradigm, we can modify or manipulate the existing network traces or PCAPs to create new network data with different characteristics or behaviors while anonymizing sensitive or private information (e.g., end-user IPs) in the network traces.

A promising trace-based approach lies in harnessing the recent strides made in Generative Adversarial Networks (GANs). Just as GANs [7] have revolutionized the generation of high-quality images, they hold the potential to reshape the landscape of PCAP (packet capture) generation. Figure 1 summarizes a GAN architecture overview. It encompasses two distinct Neural Networks — an entity designated as the generator and another as the discriminator — that engage in an interplay following the principles of game theory, as delineated in [8]. The fundamental operational paradigm involves the generator network producing synthetic data samples to deceive the discriminator. In parallel, the discriminator network undertakes the role of a judge, assessing the similarity between real and generated/synthetic data. The main objective is to create a scenario wherein the discriminator's capacity to discern actual data from its synthetic counterparts is markedly diminished.

When employed as a synthetic data generator, GANs work as a simulator. Within this context, the synthetic data produced emulates the inherent distribution of the original dataset [9], thereby ensuring the preservation of privacy considerations. Moreover, these networks prove instrumental in tasks such as dataset augmentation and balancing, culminating in a dataset characterized by enhanced representational capacity. Consequently, the resultant model emerges as a conduit to share intricate dynamics of real environments while obfuscating inherent complexities and maintaining data quality integrity.

The rest of the paper is organized as follows. Section II summarizes the state of the art on synthetic traffic generation. Section III presents the methodology we hope to use to answer the main questions identified. Finally, Section IV concludes the article, recapitulating the topics covered and illustrating the future steps of our research.

II. State of the art

Recently, realistic traffic generation relied primarily on neural network architectures such as Generative Adversarial Networks (GANs) [6], [9], [10] or diffusion models [11]. For instance, the GAN architecture works as follows: a generator produces synthetic data that mimics the real data distribution, while the discriminator tries to distinguish between real and synthetic data. Both are trained adversarially until they reach an equilibrium where the discriminator cannot differentiate (i.e., discriminate) between real and synthetic data. Similarly, diffusion models have a controlled and gradual training process but are more computationally expensive.

PcapGAN [6] proposes a method for generating realistic PCAP files that preserve the style and structure of real PCAP files. The technique uses a style-based GAN architecture that can control the style of the generated packets at different levels of abstraction. SIP-GAN [10] introduces a method for generating realistic SIP (Session Initiation Protocol) traffic that can be used for testing VoIP (Voice over IP) systems. The technique uses a conditional GAN (cGAN) architecture that can generate SIP messages with different types, such as INVITE, ACK, BYE, CANCEL, and OPTIONS. NetDiffusion [11] leverages a fixed-length packet representation [12] to transform packets into images that can be easily manipulated. Despite that, its protocol rule-compliance approach is post-generative, which means the generated packet is not protocol-compliant and must be modified to reflect the desired output.

While the mentioned work has demonstrated the efficacy of neural networks in network analysis tasks, it is important to note they have primarily been executed on generic-purpose CPUs. However, the landscape of network hardware is rapidly evolving, presenting new opportunities to leverage specialized hardware for enhanced performance and efficiency. One such promising avenue is the integration of neural networks with programmable network hardware, such as SmartNICs, to tackle a diverse range of network challenges.

By offloading neural network processing tasks, significant performance improvements can be achieved [13], leading to faster decision-making and reduced latency. It is particularly advantageous for real-time applications such as intrusion and anomaly detection, where timely responses are critical. Furthermore, programmable data plane hardware provides access to low-level network data with minimal overhead, enabling neural networks to operate directly on raw packet streams with local decisions. For instance, FcNN [14] is a distributive data-centric computing framework for reconfigurable SmartNIC-based systems. It allows the complete detaching of NN kernel execution control logic system scheduling and network communication to the SmartNICs. It boosts performance by avoiding control dependency with CPUs for various neural network kernels and applications, including DNNs and GNNs.

III. Objectives, Research Questions and Methodology

The proposed research aims to address gaps in the state-of-the-art by exploring the feasibility of generating realistic network traces with different generative approaches.

A. *Research Questions*

The proposed research plan deals with the following research questions.

1) How can we handle specific challenges like sequence generation [6] and temporal dependencies [11], in the context of network trace generation? Can those restrictions be embedded into the model, or must they be treated in a post-generation manner?
2) Is it possible to create a transparent traffic generation tool for the end user?
3) Should we offload network applications (e.g., routing) with the aid of neural network models into programmable network hardware (e.g., FPGAs, SmartNICs)? If so, what are the best offloading strategies (i.e., hybrid or total offloading) for each application?
4) What potential challenges will it present (e.g., energy consumption, memory limitation) for different hardware architectures (e.g., SmartNICs, FPGAs)?

B. *Work Plan*

This section provides an overview of the work plan stages that will direct the research activities for this Ph.D. The results will be published accordingly in all of these phases. To attain the objectives mentioned earlier and research questions, we will use the following methodology.

Addressing Challenges in Network Trace Generation: The first step towards an intelligent system that generates realistic packet traces is to (i) gather a diverse set of real-world network traces (PCAP files) representing various communication scenarios and protocols and (ii) preprocess the collected dataset to extract relevant features and normalize data to ensure consistency across different traces. To do that, we intend to leverage nPrint [12] packet representation. It provides a standardized bit-level representation of every network packet, ensuring all potential header fields (even if not present in the original packet). For instance, while a TCP packet will not have UDP header bits, the nPrint still includes placeholders for these bits, ensuring a uniform input structure for ML models. Regarding the datasets, Kaggle is a promising online data science platform that could serve as a starting point for gathering freely available PCAPs. On this website, we found a promising dataset with around 7GB of network traces split in heterogeneous applications such as Skype and Amazon. Another option would be to explore code platforms such as GitHub and GitLab or other platforms such as Paperswithcode, which groups papers and their respective codes – i.e. if they are open source. However, it is yet to be known whether the available data may be biased, with a lot of repeated information that does not help the neural model to learn different traffic scenarios. For example, if we consider a dataset about bank fraud and consider two classes (true, false) where most of the labels are "false", then the trained model may become biased, learning/specializing more in certain characteristics that do not facilitate the identification of true positives for fraud. The same idea applies for the computer network context. An ideal dataset distribution should be able to represent the entire – i.e., at least most of – data distribution over time.

An end-user traffic generator tool: Besides having data from each application, it is necessary to create a neural network that understands the demands of each application to generate network traffic that captures the nuances of each request. Initially, the most promising architecture tested to create images more faithful to the originals is Variational Autoencoders (VAEs) [15]. However, there are some limitations. First, we must train the model offline and re-execute the PCAPs in the infrastructure.

Second, the user has little control over the type of application to be generated or what types of traffic the network can generate. To achieve this, we could integrate the idea of creating or modifying an existing large language model (LLM) [16] capable of understanding high-level user requests in the network context and generating network traffic with desired characteristics. For example, ideally, we should be able to generate traffic as follows (or similarly) "Netflix traffic, 1GBps, latency 50ms, jitter 5ms". In the previous example, the user would not (ideally) need to worry about how their data is being generated, where the solution's internal processes would be transparent to them. However, the user could be guaranteed that the data distribution he requested would be generated, in the same way as the traffic restrictions (e.g. jitter, latency).

Offloading Network Applications to Programmable Hardware: At this point, we will conduct a comprehensive review of available programmable network hardware options, including SmartNICs and FPGAs that can be leveraged, considering their specifications, capabilities, and programmability features to determine suitability for offloading network applications. Then, we will identify the most prominent network applications suitable for offloading (i.e., partially or even totally) and adapt the selected applications for execution on programmable hardware, optimizing for performance and resource utilization.

Analysis of Hardware Limitations and Trade-offs: Considering the acquisition of the necessary hardware, profiling tests will be carried out on the available platforms and Measure key performance metrics such as processing speed, memory bandwidth, and energy consumption under varying workloads for partially/totally offloaded applications, considering factors like data transfer rate and protocol overhead, determining the feasibility and practical implications of offloading network applications to programmable hardware in real-world deployment scenarios.

IV. Conclusion

This article presents the main research directions for my first-year Ph.D. In this context, the idea of traffic generation can still be widely explored for different protocols and applications, facilitating the generation of traffic for the end user transparently. Furthermore, there is the possibility of offloading different applications onto programmable hardware. We believe we could benefit from this equipment and accelerate the traffic generation process.

References

[1] H. N. Qureshi, U. Masood, M. Manalastas, S. M. A. Zaidi, H. Farooq, J. Forgeat, M. Bouton, S. Bothe, P. Karlsson, A. Rizwan, *et al.*, "Towards addressing training data scarcity challenge in emerging radio access networks: A survey and framework," *IEEE Communications Surveys & Tutorials*, 2023.

[2] T. Bühler, R. Schmid, S. Lutz, and L. Vanbever, "Generating representative, live network traffic out of millions of code repositories," in *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*, pp. 1–7, 2022.

[3] E. Dai and S. Wang, "Say no to the discrimination: Learning fair graph neural networks with limited sensitive attribute information," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pp. 680–688, 2021.

[4] G. Carneiro, "Ns-3: Network simulator 3," in *UTM lab meeting April*, vol. 20, pp. 4–5, 2010.

[5] Y. Yin, Z. Lin, M. Jin, G. Fanti, and V. Sekar, "Practical gan-based synthetic ip header trace generation using netshare," in *Proceedings of the ACM SIGCOMM 2022 Conference*, pp. 458–472, 2022.

[6] B. Dowoo, Y. Jung, and C. Choi, "Pcapgan: Packet capture file generator by style-based generative adversarial networks," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pp. 1149–1154, IEEE, 2019.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[9] Y. Yin, Z. Lin, M. Jin, G. Fanti, and V. Sekar, "Practical gan-based synthetic ip header trace generation using netshare," in *Proceedings of the ACM SIGCOMM 2022 Conference*, SIGCOMM '22, (New York, NY, USA), p. 458–472, Association for Computing Machinery, 2022.

[10] A. Meddahi, H. Drira, and A. Meddahi, "Sip-gan: Generative adversarial networks for sip traffic generation," in *2021 International Symposium on Networks, Computers and Communications (ISNCC)*, pp. 1–6, IEEE, 2021.

[11] X. Jiang, S. Liu, A. Gember-Jacobson, A. N. Bhagoji, P. Schmitt, F. Bronzino, and N. Feamster, "Netdiffusion: Network data augmentation through protocol-constrained traffic generation," *arXiv preprint arXiv:2310.08543*, 2023.

[12] J. Holland, P. Schmitt, N. Feamster, and P. Mittal, "New directions in automated traffic analysis," CCS '21, (New York, NY, USA), p. 3366–3383, Association for Computing Machinery, 2021.

[13] C. Campolo, A. Iera, and A. Molinaro, "Network for distributed intelligence: a survey and future perspectives," *IEEE Access*, 2023.

[14] A. Guo, T. Geng, Y. Zhang, P. Haghi, C. Wu, C. Tan, Y. Lin, A. Li, and M. Herbordt, "Fcsn: A fpga-centric smartnic framework for neural networks," in *2022 IEEE 30th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 1–2, IEEE, 2022.

[15] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[16] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, *et al.*, "A survey on evaluation of large language models," *ACM Transactions on Intelligent Systems and Technology*, vol. 15, no. 3, pp. 1–45, 2024.