Programmable Data Planes for Increased Digital Resilience in OT Networks

Filip Holik, Marco M. Cook, Xicheng Li, Awais Aziz Shah, and Dimitrios Pezaros

Abstract

Critical national infrastructure is heavily reliant on operational technology (OT) to monitor and control physical industrial processes. Despite the importance of OT systems, they have inherently been designed to maintain safety assurance rather than mitigate security threats. Recent advances in programmable networks offer highly customizable and flexible solutions for cybersecurity. However, the integration of data plane and network stack programmability into OT environments has not been fully explored. The conventional programmable data plane (PDP) approach based on the P4 language and the reconfigurable match-action tables (RMT) switch model has proven highly effective for data center environments. However, performance is subject to specific target implementations, while the architectural model lacks inherent support for flexible and stateful functionality required by cybersecurity use cases. Alternative (host) network programmability paradigms, such as extended Berkley packet filter (eBPF), are gaining traction as more suitable to OT use cases. In this article, we explore the suitability of PDP and offload technologies for improved digital resilience in OT environments. We provide a comprehensive cross-comparison between PDP solutions and advocate that eBPF-based network programmability can achieve flexible, dynamic, and low-cost in-network functions for digital security and resilience. We present an eBPF-based proof-of-concept asset discovery service that is compared with alternative solutions. Furthermore, we discuss the OT network security functions enabled through eBPF to address potential future research directions.

Introduction

Modern society relies on the uninterrupted operation of complex inter-dependent systems that underpin Critical National Infrastructure (CNI), such as water treatment, energy, and transportation. CNI has witnessed systemic digitization while the underlying Operational Technologies (OT) and Industrial Control Systems (ICS) that drive and monitor physical processes are now ubiquitous. Despite being such critical systems, the integration of security controls into OT networks and components at design time has often been overlooked, with reliability and safety being the primary objectives. OT digitization has resulted in significant convergence of industrial environments with Internet-facing networks. In an increasingly insecure world, adversaries have looked to exploit this interconnectivity through cyber campaigns that aim to manipulate the operations of controllers connected to physical apparatus, demonstrated through bespoke ICS malware and, more recently, living-off-the-land attacks [1]. Hence, deploying resilient networks and components into the digital infrastructure of OT is critical in managing and mitigating current and future threats.

As OT components typically lack interfaces for end users to deploy cybersecurity mechanisms, for example, encryption and endpoint detection, controls are often retrofitted into the OT network by deploying monolithic middleware such as firewalls. While these are ubiquitous in conventional IT networks, their deployment in OT networks introduces significant cost overheads due to additional power consumption, space factor, and required reliability, longevity, and ruggedness for operation in extreme environments [2].

The paradigm of programmable networks offers significant advancements in dynamic per-packet processing for a diverse set of use cases, including security and resource orchestration, while retaining high transmission speeds [3]. The emergence of Programmable Data Plane (PDP) technologies has provided additional opportunities for customized network management and security by offloading functions from software controllers to network switches and host networking stacks. The main benefit of this shift is that packet processing can be performed at line rate using bespoke network programming languages such as P4 to define the packet processing logic [4].

However, current PDP solutions are typically implemented through dedicated target architectures (e.g., Intel Tofino), requiring proprietary hardware to be deployed while suffering functional limitations, including a lack of support for floating point and loop operations [4], increasing the complexity of deploying flexible in-network security controls. Such solutions are unlikely to be adopted in OT environments in the immediate future due to the long life cycles of industrial equipment on top of the monetary and logistical challenges of validating and deploying new hardware. For instance, OpenFlow-based Software-Defined Networking (SDN) has been around for 15 years and has been successfully introduced into data centers and generic IT environments, but it is only now starting to be applied to OT through research stud-

ISSN: 0163-6804
Digital Object Identifier: 10.1109/MCOM.001.2400446

The authors are with University of Glasgow, UK.

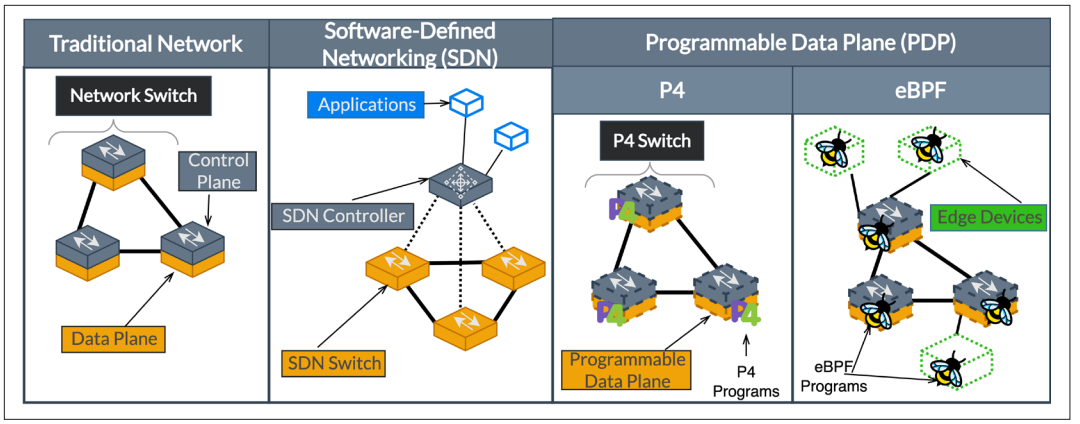

FIGURE 1. Abstraction of Networking Paradigms illustrating the control and data plane architecture, as well as typical topologies.

ies [5]. To address these challenges, the extended Berkley Packet Filter (eBPF) can provide a flexible framework for dynamic and interoperable security for OT networks and wider CNI. While eBPF is not explicitly a networking technology, it enables the execution of sandboxed programs with bound execution time within operating systems kernels and supports interaction with the kernel network layers, promoting the suitability of eBPF for time-sensitive operations often found in OT environments.

In this article, we propose that programmable networking technologies can provide flexible cybersecurity function chaining for OT compared with existing middleware implementations such as firewalls to improve digital resilience. We provide a comprehensive comparison of the functional aspects of eBPF and P4 and consider the benefits of integrating programmability into OT networks to enable dynamic and accelerated cybersecurity functions. Moreover, we emphasize that the properties of eBPF, including flexible deployment, code safety verification, and extensive functionality, make it more suitable to address the specific requirements of OT environments compared to SDN and P4 technologies. To further support this, we demonstrate a proof-of-concept eBPF-based PDP solution within an emulated OT network topology, specifically electric substation environments. Our results from using an asset discovery service highlight the superior performance of eBPF in maintaining low network latency to support critical communications between devices. Finally, we present novel perspectives related to the implementation of cybersecurity function chaining using eBPF, demonstrated through multiple OT use cases to enable digital resilience of future systems.

The remainder of this article is structured as follows. The next section provides background on data plane programmability and compares eBPF with existing PDP technologies. After that, a proof-of-concept solution using eBPF for flexible OT cybersecurity provisioning is presented, followed by a discussion on open challenges and security use cases within OT networks. Finally, we conclude this article.

Data Plane Programmability

Networking Architecture

Architectures of network devices, such as those found in switches and routers, are comprised of a *control plane* and a *data plane*. The control plane provides the logical decision regarding how to forward network packets to subsequent destinations. The data plane executes the processing functionality and is implemented in a sequence of *match-action* tables which represent low-level primitives that are easy to implement on hardware, thus providing high (line-rate) performance. In traditional networks, this functionality is hard-coded into the switching Application-Specific Integrated Circuit (ASIC) and hence has limited customization for application-specific domains. Conversely, network programmability provides modifications to the conventional fixed switching architecture to achieve custom traffic processing.

Network Programmability Paradigms

Network programmability is a concept from the late 1990s, but has gained significant traction in recent years with the availability and increasing affordability of programmable silicon. In this article, we classify network programmability paradigms into SDN and PDP, where the latter is represented by P4 and eBPF technologies. These paradigms are shown in Fig. 1 and described below.

Software-Defined Networking (SDN): SDN introduced the physical separation of control and data planes, relocating the control plane from the network switch to a centralized SDN controller. SDN became popular with the introduction of the OpenFlow protocol, which defined the data plane, control plane, and communication between them, the so-called southbound interface. Despite its significant popularity within data center networks, OT-SDN has only recently seen increased traction with the introduction of OT-SDN switches and controllers. Although OT-SDN has demonstrated improvements in fast failover and self-healing, its cybersecurity abilities for edge devices are limited [5]. SDN does not support full data plane programmability as the processing pipeline must follow pre-defined structures called *flow tables*. These tables define processing logic in the form of the match-action fields specified by the OpenFlow protocol. An action of a matching packet can then point to the next flow table, allowing sequencing. Compared to traditional switching architectures, SDN flow tables allow relatively complex customized processing; for example, tables can be chained, searched in parallel, executed in groups, or placed in specific locations (ingress and egress tables). However, the strictly defined match fields pose a more significant restriction as network protocol header

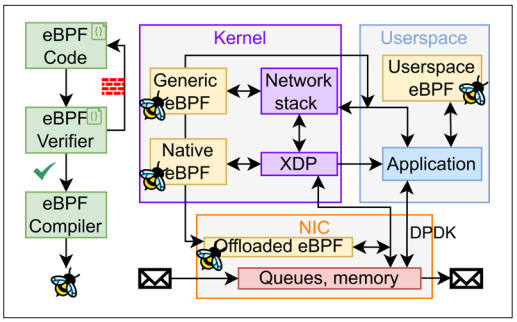

FIGURE 2. eBPF architecture — represents the eBPF function.

fields are defined. This limits the customizability of SDN functions to match against OT-specific protocol headers such as Modbus, and hence, SDN will not be considered further in this article.

Programming Protocol-Independent Packet Processors (P4): More recently, the paradigm of PDPs has emerged that enables a significant increase in network programmability by logically separating the data plane from the control plane but being kept on the same hardware. The conventional approach for constructing PDPs is P4, a high-level language for data plane processing definition. P4 has brought full programmability of the match-action tables through custom parsers and deparsers. Early versions of the P4 language (P4_14) defined programming constructs and a Protocol Independent Switch Architecture (PISA). On the contrary, P4_16 defines the programming constructs only, while features such as register access and checksum recalculation are encapsulated in the form of extern objects and functions. State-of-the-art P4 architectures such as Portable Switch Architecture (PSA) and Tofino Native Architecture (TNA) use parallelism and are tailored to Application-Specific Integrated Circuits (ASICs) to achieve high performance at a few terabits per second. While P4 is suitable for implementing highly specific functionality, including one-the-fly processing and telemetry of industrial protocols found in OT networks, it is currently only supported by a limited number of target devices such as Intel Tofino switches and SmartNICs. The lack of native Linux OS kernel support for high-speed software implementations limits deployment flexibility when compared to eBPF.

Extended Berkeley Packet Filter (eBPF): In comparison to SDN and P4, eBPF is not specifically a networking technology but instead comprises a virtual machine-like construct with a specific instruction set that is executed inside the Linux kernel. eBPF enables unprecedented flexibility, is not bound to the match-action concept mandated by P4, and can be deployed on any device with a Linux kernel version 3.18+ (released in 2014), with various attachment points for eBPF programs, as shown in Fig. 2. This enables deployment on edge devices such as Industrial Internet of Things (IIoT) and next generation Programmable Logic Controllers (PLCs), but in this article, we primarily consider deployments on industrial networking devices, unless stated otherwise.

eBPF and P4 Comparison

While eBPF is not strictly confined to networking functions, it has recently gained significant popularity for PDP use cases and fast packet processing [6]. In the following sections, we explore the considerations for selecting between PDP technologies such as eBPF and P4 to enable flexible cybersecurity function synthesis, particularly for OT environments, based on three dimensions summarized in Table 1. Where applicable, we use the example of Asset Discovery (AD) to discuss the specific properties of each technology within the context of digital resilience.

Deployment Properties: eBPF can be deployed on low-cost resource-constrained devices such as Single-Board Computers (SBC) and does not require expensive and bespoke silicon to reach multi-gigabit rates in commodity P4 architectures. Industrial components have been intentionally designed with a very limited programmable interface, often through embedded Real-Time Operating Systems (RTOS) tailored to continuous and deterministic logic execution. However, open PLCs are becoming increasingly attractive to end users due to their flexibility. For instance, leading industrial vendors such as Rockwell Automation and Beckhoff Automation have started to introduce openly programmable PLCs that enable parallel execution of Windows environments through virtualization. This drive toward openness supports the use of flexible frameworks such as eBPF for cybersecurity function chaining, where retrofitting functions can be integrated to extend existing system kernels on both edge and network devices. This would make the AD service more efficient as it could run on multiple devices in different locations for improved network observability.

In comparison, current off-the-shelf P4-programmable devices mainly focus on networking applications in high-performance data centers and not on edge devices. This would limit the AD service reach only to the backbone traffic.

eBPF can be deployed flexibly within the system architecture as shown in Fig. 2. It supports userspace offload directly on the Network Interface Card (NIC) and in kernel — either as generic, or native with eXpress Data Path (XDP) subsystem which allows eBPF programs to process received packets at the earliest point in the network stack before expensive memory allocations have been performed.

The eBPF userspace implementation (such as uBPF) can also leverage the Data Plane Development Kit (DPDK) library, as illustrated in Fig. 2, which passes received traffic directly to userspace, hence significantly improving performance. These advances improve the versatility of eBPF for OT networks with heterogeneous devices in different life cycle stages and diverse requirements for functionality and performance.

Code Safety and Flexibility Properties: Unlike other PDP paradigms, eBPF has inherent code safety mechanisms performed by the eBPF verifier that checks parameters, including program termination, execution simulation, and helper function call verification [6]. Verification makes eBPF particularly suitable for OT networks where safety assurance is critical, which cannot be achieved with alternative PDP technologies. For instance, P4 does not support any verification natively. Although there are 3rd party efforts to add this functionality, there is no single solution for all P4 architectures (e.g., PSA) [7]. Furthermore, eBPF can use kernel helper function calls, which further extend functionality by providing operations such as IP header checksum

recalculation or random number generation [6]. Kernel functions ensure safe and efficient functionality that is ideal for more complex OT functions. Conversely, it is the vendor's responsibility in P4 to provide target-specific extensions as *extern* objects and functions. Hence, P4 does not provide built-in alternatives, reducing flexibility and increasing the risk of programmer error [8].

From a programming perspective, eBPF is more flexible than P4 and can be used to program more complex constructions; for instance, implementing custom regular expressions. Additionally, eBPF supports loops (limited number of executions) and chaining of up to 32 eBPF programs via tail calls, further increasing code possibilities [6].

Architectural Properties: One significant advantage of eBPF is that code within programs can be modified during runtime without the need to reboot the system. This promotes the suitability of eBPF for network security use cases such as anomaly inference, where data models may need to be updated more frequently to mitigate challenges such as concept drift. In particular, this flexibility is important to ensure high availability required within OT systems, where multiple eBPF programs can be deployed at the same time to promote redundancy. P4-programmable hardware such as Intel Tofino switches provide low maximum execution times for every packet, which, if exceeded, the packet will be dropped. While this feature ensures line rate performance, eBPF is not limited to pre-defined execution times and, hence, is more flexible.

For efficient data storage and sharing between eBPF programs and userspace applications, eBPF uses dynamically allocated maps stored in DRAM [6]. P4 uses a similar concept called tables. However, the size of these is limited by the fixed onboard memory of programmable switches. Moreover, for packet parsing (extracting header fields of a datagram), P4 uses the Packet Header Vector (PHV) with a limited (target-specific) memory size for storing extracted header fields and metadata. When P4 forwards packets to a user application, it creates a copy sent to the CPU port, which then subsequently traverses the host network stack. Conversely, eBPF maps can be accessed without the need to copy the packet. Therefore, eBPF is more suitable for storing large amounts of network traffic and for sharing the data with the application more efficiently, which is important for AD. In P4, such a service could not be done in the same way as the memory capacity would not allow static allocation for every MAC address.

Feature	P4	eBPF
Deployment		
Target deployment	Data centers	Any device
Target area	Networking-only	Flexible (OS & networking)
Target architecture	Per-device specific	Generic
Hardware acceleration	Yes	Smart NIC only
Coding		
Safety verification	No (limited projects)	Yes
Programming language	P4	C-like
Loops (max cycles)	No	8,388,608
Maximum instructions	Platform specific	1,000,000*
Regular Expressions	No	Yes (custom)
Architecture		
Runtime change	Reload required	Seamless
Execution time	Bounded	Unlimited
Data storage structures	T&R (limited)	Maps (flexible)
Parser memory	Limited by PHV	Unlimited

TABLE 1. Technical comparison of P4 and eBPF. * Up to 32 eBPF programs can be chained thus allowing 32,000,000 inst. PHV = Packet Header Vector, T&R = Tables and Registers.

Programmable Data Planes for OT Networks

The comparison of P4 and eBPF presented previously highlights the suitability of eBPF for OT PDP deployments for two main reasons. Firstly, the target-independence and flexibility of eBPF enables deployment on diverse types of device. Secondly, the code safety verification and a more complex instruction set make eBPF more suitable for implementation of resilient network functions within safety-critical OT environments. Finally, the speeds that need to be supported in OT networks are bound and hence bespoke hardware support is not required.

To demonstrate eBPF functionality in OT networks, we have developed a proof-of-concept using a smart grid network topology with emulated communication, illustrated in Fig. 3. Specifically, we used SGSim (Smart Grid Simulator) [9] to generate a topology containing one digital primary substation, two digital secondary substations, a control center, and realistic IEC104, GOOSE (Generic Substation Events) and SV (Sampled Values) communication, all running in the Mininet network emulator. We have built on our prior work and integrated a pre-existing eBPF PDP framework called *BPFabric* [10] to provide an SDN-like centralized orchestration of programmable devices that run eBPF micro-functions. BPFabric is independent of platform, protocol, and language and provides automated eBPF code compilation and delivery into network devices. The complete setup with an installation guide is provided in our GitHub [11].

OT Network Services

While BPFabric provides several generic network functions such as forwarding and traffic mirroring, we have developed and integrated two additional services into our OT proof-of-concept: an Access Control List (ACL) and an Asset Discovery (AD) that maintains an eBPF map storing the number of received bytes and GOOSE packets for every MAC address. If a packet is classified as the GOOSE protocol, the counters will be adequately increased, and a notification message will be sent to the BPFabric controller with the updated content of the eBPF map. We deployed these services to different networking devices, as illustrated in Fig. 3. This forms the processing pipeline for every connected device and provides a flexible solution to cybersecurity function chaining.

Network Latency Evaluation

We evaluated the performance of our solution using GOOSE traffic, which represents highly time-critical services in digital substations with an

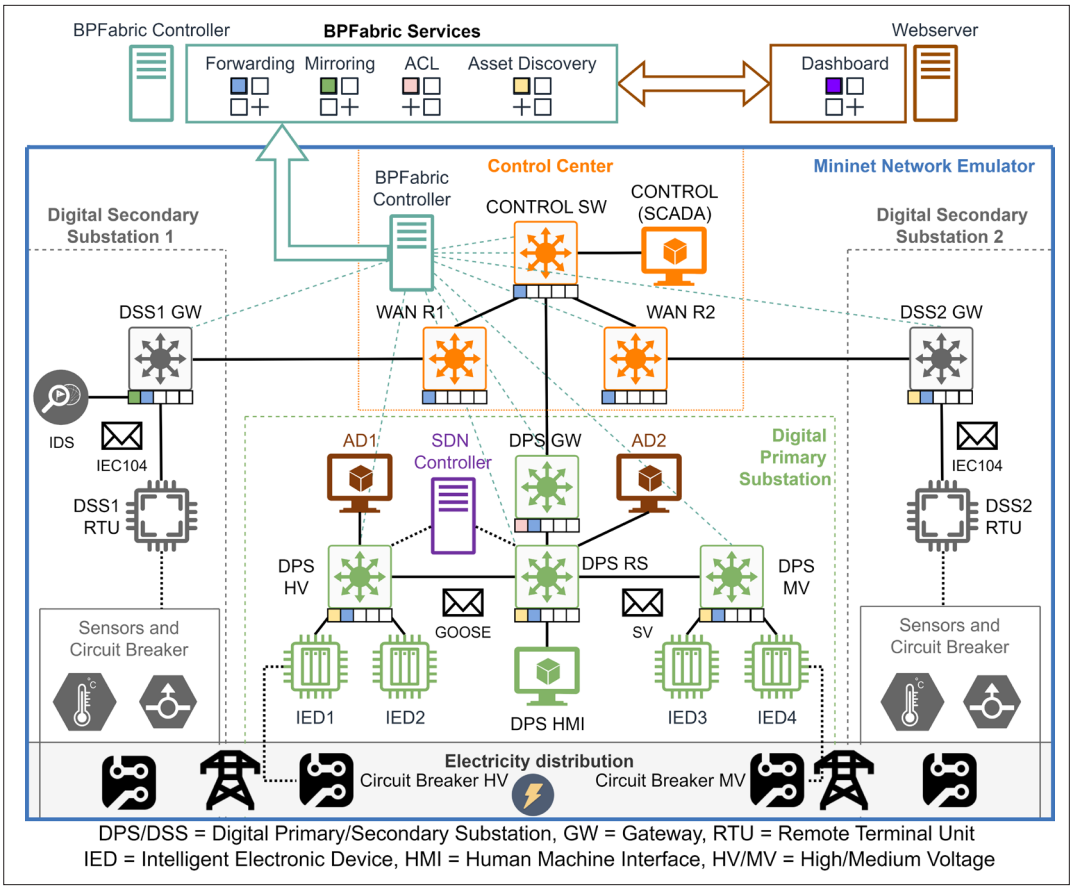

FIGURE 3. Emulated topology of the proof-of-concept solution.

enforced latency limit of 3 ms to ensure the timely shutdown of safety-critical components [12]. We measured end-to-end latency between IED1 (GOOSE publisher) and DPS HMI (GOOSE subscriber) within the network topology illustrated in Fig. 3. The DPS HV and DPS RS eBPF switches have AD and forwarding services enabled. We compared the performance of eBPF-based PDP functions with the three alternative networking paradigms outlined earlier in Fig. 1: traditional networking, P4, and SDN. We first deployed a network that does not support custom AD and hence was implemented on dedicated hosts (AD1 and AD2). Each AD device logged the traffic before forwarding it to the next hop.

Moreover, we compared against an SDN implementation deployed with a Ryu controller. Finally, we have developed a BMv2 software implementation to simulate the functionality of real P4 switches. Due to P4 architecture limitations in static memory allocation, the AD service could not be implemented on the switch itself as it could not allocate enough memory for the entire MAC address range. For this reason, we conduct two experiments on the P4 implementation. Firstly, we used the AD devices for logging off-switch while BMv2 P4 switches were responsible for appropriate packet forwarding between the IEDs and AD devices. In the second experiment, we implemented a limited AD service with static MAC address allocation for known grid devices inline on-switch. This implementation would, therefore, not detect unknown devices but is only for performance comparison.

Results and Discussion

Our results are presented in Fig. 4, where the maximum latency limit for GOOSE messages is shown at 3 ms. Figure 4a demonstrates a period of normal network operation for 30 seconds (periodical 1-second reporting). This is followed by a 1-second GOOSE event involving a burst transmission, which is detailed in Fig. 4b, before returning to normal operation for another 30 seconds.

The latency of the SDN approach was an order of magnitude higher than the eBPF PDP solution and above the required threshold. This was caused by every GOOSE message being forwarded via the controller since the AD service could not be conducted on the switch due to OpenFlow not supporting custom counters. Hence, the forwarding was performed on both networking devices — DPS HV and DPS RS. These switches had to use packet-in events to forward the message to the controller for logging purposes, after which the controller returned the message in the packet-out event to be forwarded to the next device via the correct port.

The traditional networking approach followed a similar packet path as with the SDN solution, where both DPS HV and DPS RS first forwarded the message to the AD devices. In this case, we observed that latency was approximately 3 times higher than SDN during normal operation. In the GOOSE burst transmission event that is detailed in Fig. 4b, latency increased linearly, reaching upwards of 900 ms. This was caused by the high overhead of *socket* object creation, resulting in packets being stored in the queue. The latency

stabilized when the network went back to normal operation at 31 seconds. While no packets were dropped due to sufficient queue capacity, in reality, this would highly likely result in network unavailability due to traffic bottlenecks.

The off-switch BMv2 P4 implementation further increased packet latency, as evident from the burst transmission in Fig. 4b, where the latency reached 5.7 seconds and required additional time (3 seconds) to return to normal processing rates. The on-switch variant performed significantly better with consistent latency but still significantly higher than the GOOSE limit and comparable to SDN performance. Hence, the substandard performance of P4 software implementations through BMv2, and lack of ruggidized P4 hardware coupled with significant device costs makes P4 unsuitable for demanding OT environments.

The results demonstrate that eBPF performed significantly better than other solutions as it enabled seamless integration of the AD service without introducing additional devices or altering the packet path. Moreover, despite using the slowest eBPF implementation (uBPF) due to limitations with Mininet, the eBPF approach outperformed the other network solutions and was the only one able to comply with the GOOSE latency requirement for every single message.

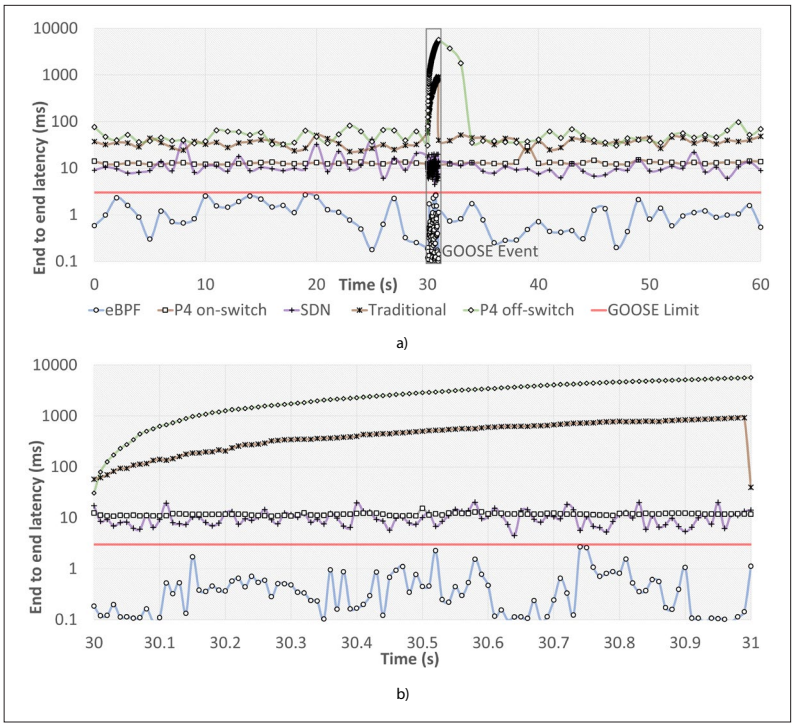

FIGURE 4. Comparison of packet latency using different network service solutions: a) GOOSE Normal operation and burst event; b) Burst GOOSE event transmission.

OT Networks of the Future

We have discussed how flexible security functions can be deployed to specific areas of the network. Furthermore, dependencies and links can be created between functions to establish chains, such as in the example illustrated in Fig. 5, where we map packet functions to high-level areas of digital resilience. For instance, given a particular attack type being identified, such as Denial of Service (DoS), organizations may invoke mitigation controls as part of an automated security function chain to control the movement of packets and remediate the impacts. In this section, we discuss the potential OT cybersecurity functions that are enabled through inexpensive PDP capabilities and link these functions to specific eBPF properties examined in Table 1. We approach these functions as open challenges to facilitate future research addressing digital resilience in existing and next-generation OT networks.

Moreover, we envision an architecture where the functions of each networking device are deployed using an eBPF-based agent and managed through a centralized controller. Such an architecture will not require the replacement of industrial equipment while supporting the addition of new functions. In the next stage, this architecture could be extended to OT edge devices, particularly with the advancements in open industrial controllers that use Linux-based software architectures. This would allow comprehensive infrastructure observability to a level of running processes on every device, thus enabling even more advanced resilience functions to be developed.

Monitoring and Data Collection

Industrial Ethernet has become more ubiquitous within existing and greenfield OT sites compared to legacy Fieldbus protocols, improving efficiency and resulting in vast amounts of data traversing through interconnected networks.

Conventionally, monitoring and data collection within OT has been achieved through the installation of passive and inflexible network Test Access Points (TAPs) that typically send copied packets to a centralized service for processing. These methods are costly to implement in large-scale industrial environments and can quickly become saturated with excessive network traffic volumes, which is becoming additionally challenging with the rise of the Industrial Internet of Things (IIoT). As our earlier proof-of-concept results show, eBPF can address the aforementioned challenges by observing traffic anywhere in the network and performing per-packet analysis, including on-edge devices, to cause minimal disruption. A custom eBPF monitoring service can perform data pre-processing (potentially at line rate) using efficient eBPF maps and report only aggregated information to the user application, thus saving bandwidth and computing resources.

In-Network Encryption and Authentication

One significant vulnerability that is common to many OT networks across different CNI sectors is the lack of encryption. Retrofitting encryption algorithms into industrial protocols used by OT edge devices can be very challenging due to proprietary technologies often used by OT vendors and would require changes to protocol standards such as Modbus. Furthermore, encryption can be computationally demanding, making it challenging to deploy on resource-constrained devices that are ubiquitous to OT and industrial environments, such as PLCs. Using middleware such as proxy servers can offer encryption solutions that are easier to implement in existing OT systems [13]. However, these come at a cost where packets are offloaded first to be encrypted, similar to the SDN implementation in our proof-of-concept. Implementing

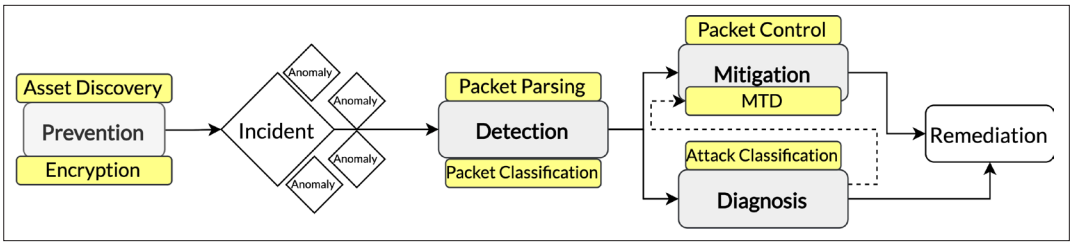

FIGURE 5. OT security function chaining mapped to high-level security concepts — example functions are in yellow.

encryption on network devices with eBPF would eliminate the requirement for additional devices, such as proxies, and would limit the performance penalty. Moreover, the flexible instruction set and integrated checksum recalculation functions of eBPF can be used to implement various encryption standards with dynamic key lengths.

Anomaly Inference

The use of in-network anomaly detection is not a new concept and has gained significant attention in recent years [14]. In parallel, research has also explored the use of deploying Machine Learning (ML) algorithms into network hardware [14] specifically for detection tasks, typically implemented through P4. However, the resource constraints of hardware targets and language limitations mean that the complexity of ML algorithms and the size of models are restricted, making them less suitable for large industrial environments where high dimensionality within datasets is common. Furthermore, adapting ML models and algorithms toward evolving digital threats is important to maintain accurate predictions and low false negatives/positives. Such inference is challenging to do during runtime on programmable switching hardware without impacting the packet processing performance. Given these challenges with using P4 switches in OT networks, eBPF can provide the building blocks for effective and flexible anomaly detection frameworks, both in-network and at the edge, that can dynamically change monitored network parameters and use advanced ML algorithms such as deep learning.

Real-Time Attack Mitigation and Tolerance

Although crucial, detecting anomalies is one part of a defensive resilience chain. A subsequent function would be to mitigate the effects of the occurring anomaly, which can be achieved through network controls. For instance, recent work has demonstrated anomalous traffic mitigation for OT networks using packet count measurements in a simulated P4 environment [2]. While this initial work shows promise, we propose that more complex mitigation algorithms can be deployed on eBPF PDPs using statistical measurements that are not limited to analyzing network data. Reinforcement Learning (RL) has previously proven effective for achieving network availability under link flooding amplification attacks [15]. However, similar approaches could be explored within OT contexts. The complexity of such algorithms would require flexible instruction sets and data storage structures, which are provided in eBPF but not in P4.

Conclusion

In this article, we have explored the potential benefits of incorporating PDP into OT infrastructures to improve digital resilience. We have reviewed the functional differences between eBPF and P4 technologies and argued that the former could provide greater flexibility and safety when deploying cybersecurity functions such as in-network encryption and asset discovery, all at a much lower cost than P4 alternatives. We demonstrated a proof-of-concept solution that enables dynamic cybersecurity in an emulated smart grid environment, evaluating an asset discovery use case. The results emphasize that eBPF-enabled PDPs provide significant performance benefits compared to traditional network management solutions and SDN controller architectures without requiring the installation of bespoke devices and middleboxes into the OT network. Considering the performance benefits of ePBF, we discussed how eBPF-based PDPs can enable digital resilience in OT networks through flexible stateful function chaining.

Acknowledgment

This work has been supported in part by the Royal Academy of Engineering under grant number RCSRF2223-1645, Dstl under order number DSTL0000014002, EDF under order number 4840659490, and UK Research and Innovation (UKRI) under reference number 10091225. Filip Holik and Marco M. Cook are co-first authors.

References

[1] J. Slowik, "Evolution of ICS Attacks and the Prospects for Future Disruptive Events," *Threat Intelligence Centre Dragos Inc*, 2019; accessed 2024-11-12: https://www.dragos.com/resources/whitepaper/evolution-of-ics-attacks-and-the-prospects-for-future-disruptive-events/.
[2] Z. Hu *et al.*, "Industrial Network Protocol Security Enhancement Using Programmable Switches." *Proc. 2023 IEEE Int'l. Conf. Commun., Control, and Computing Technologies for Smart Grids*, IEEE, 2023, pp. 1–7.
[3] X. Chen *et al.*, "Empowering Network Security With Programmable Switches: A Comprehensive Survey," *IEEE Commun. Surveys & Tutorials*, vol. 25, no. 3, 2023, pp. 1653–1704.
[4] P416 Language Specification, 2021, The P4 Language Consortium; accessed 2024-10-03: https://p4.org/p4-spec/docs/P4-16-v1. 2.2.pdf.
[5] W. J. Hutton *et al.*, "Deploying Software-Defined Networking in Operational Technology Environments," *J. Information Warfare*, vol. 20, no. 2, 2021, pp. 93–106.
[6] M. A. M. Vieira *et al.*, "Fast Packet Processing with eBPF and XDP: Concepts, Code, Challenges, and Applications," *ACM Computing Surveys*, vol. 53, no. 1, 2020, pp. 1–36.
[7] Q. Wang *et al.*, "Foundational Verification of Stateful P4 Packet Processing," *Proc. 14th Int'l. Conf. Interactive Theorem Proving, Schloss-Dagstuhl-Leibniz Zentrum für Informatik*, 2023.
[8] Y. Zhou *et al.*, "P4DB: On-the-Fly Debugging for Programmable Data Planes," *IEEE/ACM Trans. Networking*, vol. 27, no. 4, 2019, pp. 1714–27.
[9] F. Holik, S. Y. Yayilgan, and Gu. B. Olsborg, "Emulation of Digital Substations Communication for Cyber Security Awareness," *Electronics*, vol. 13, no. 12, 2024.
[10] S. Jouet and D. P. Pezaros, "BPFabric: Data Plane Programmability for Software Defined Networks," *Proc. 2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, 2017, pp. 38–48.
[11] Networked Systems Research Laboratory, GitHub: Uof-

Gnetlab/ SGFabric, 2024; accessed 2024-10-04: https://github.com/UofG-netlab/SGFabric.
[12] M. F. Elrawy *et al.*, "A Geometrical Approach to Enhance Security Against Cyber Attacks in Digital Substations," *IEEE Access*, vol. 12, 2024, pp. 18,724–38.
[13] X. Niu, M. M. Cook, and D. Pezaros, "Examining the Suitability of Stream Ciphers for Modbus-TCP Encryption on Resource Constrained Devices," *Proc. 17th European Workshop on Systems Security*, 2024, pp, 51–57.
[14] C. Zheng *et al.*, "In-Network Machine Learning Using Programmable Network Devices: A Survey," *IEEE Commun. Surveys & Tutorials*, 2023.
[15] K. A Simpson, S. Rogers, and D. P. Pezaros, "Per-Host DDoS Mitigation by Direct-Control Reinforcement Learning," *IEEE Trans. Network and Service Management*, vol. 17, no. 1, 2019, pp. 103–17.

Biographies

Filip Holik received a Ph.D. degree from the University of Pardubice in 2018. Currently, he is a Research Associate with the School of Computing Science, University of Glasgow. His research interests include software-defined networking and programmable data planes based on eBPF and their applicability for the cybersecurity of critical infrastructure, especially power grid networks.

Marco M. Cook [M] received a Ph.D. degree in computing science from the University of Glasgow in 2023, funded through an EPSRC iCASE Award in partnership with the Defence Science and Technology Laboratory (Dstl). Currently, he is a Research Associate with the School of Computing Science, University of Glasgow. His research focuses on cybersecurity for industrial control systems, specifically anomaly detection and digital forensics.

Xicheng Li is a Ph.D. student in the School of Computing Science at the University of Glasgow. He received his bachelor's degree in software engineering from the University College Dublin in 2021. His research focuses on software-defined networks and programmable data planes, with a particular emphasis on utilizing programmable switches to offload generic network services.

Awais Aziz Shah is a lecturer in the School of Computing Science at the University of Glasgow. His research is focused on using software-defined networks and virtualization technologies such as containers to virtualize modern network infrastructures for optimal virtual network function chain deployments. His recent projects focus on resilience in IT/OT environments in critical national Infrastructures, including Industrial Control Systems and Smart Grids.

Dimitrios Pezaros [M'04, SM'14] received B.Sc. and Ph.D. degrees in computer science from Lancaster University in 2000 and 2005, respectively. He is a Full Professor of computer networks with the School of Computing Science at the University of Glasgow, where he currently holds the DSTL & CINIF/RAEng Research Chair in Digital Resilience for Critical National Infrastructure. He has received significant funding for his research from various funding agencies and industry, and has published widely in the areas of computer communications, network and service management, and network resilience. He is a Chartered Engineer, a fellow of the BCS and IET, and a Senior Member of ACM.