A Dynamic Data-Driven Framework for Enhancing Energy System Security and Resilience through Programmable Networks

Reuben Samson Raj[1] and Dong Jin[1]

[1] Dept. of Electrical Engr. and Computer Science, University of Arkansas, AR, USA
{rs077, dongjin}@uark.edu

Abstract. Cyber-security for Industrial Control Systems (ICS) such as Supervisory Control and Data Acquisition (SCADA) systems has been an important avenue of research over the last couple of decades. The need for secure ICS systems stems from several factors - legacy protocols, limited segmentation between operational technology (OT) and information technology (IT) networks and increasing connectivity of ICS devices to corporate networks and the Internet, among others. Given the unique challenges to ICS security and the unique characteristics of ICS systems and network traffic, this Chapter presents one of the first dynamic data-driven security framework for ICS built on P4-programmable switches. Our framework consists of a switch-controller feedback loop mechanism, along the lines of DDDAS-based approaches, that enables real-time cyber-attack detection and mitigation for a variety of attack scenarios. The P4 switch, with its custom packet processing capabilities, generates statistics and metrics based on network traffic patterns. The controller employs these insights to further detect and mitigate attacks by updating forwarding rules on the switch in real time. Our prototype solutions covering a range of defense approaches for Denial-of-Service attacks on a Modbus network, achieving low-latency detection, flexible defense policies, and adaptive rate limiting, demonstrating the promising potential of P4-based DDDAS towards real-time cyber-defense for ICS networks.

Keywords: DDDAS, Dynamic data driven applications systems, Infosymbiotic systems, Industrial Control Systems, Programmable Switches; Security; Denial-of-Service Attacks; Cyber-Defense

1 Introduction

Industrial Control Systems (ICS), such as Supervisory Control and Data Acquisition (SCADA), are used by a significant majority (80%-90%) of utility operators [1, 2]. These systems typically employ legacy protocols, such as Modbus [21], Distributed Network Protocol 3 (DNP3) [39], or International Electrotechnical Commission (IEC) 60870-5. These protocols were originally designed for operation in isolation without the need for interaction with external Wide Area Networks (WANs) or the Internet. However, the recent modernization of ICS systems into highly interconnected systems has rendered them vulnerable to evolving cyber-attacks [3, 4].

Although upgrading the legacy ICS devices to support advanced protocols involving security mechanisms, such as authentication and encryption, seem like logical choices, there are challenges to this approach. For instance, utility operators might be hesitant to perform such upgrades across host devices due to real-time constraints. Also, these devices are generally resource-constrained in terms of compute and storage power. Given these limitations on host-based security, the responsibility of security shifts to the network. We are thus motivated to pursue network-centric approaches for ICS network security, with a particular focus on programmable networks. The approaches proposed here apply DDDAS-based methods [13], and do not preclude future implementations which can provide both, host-based and network-based security, which can result in added robustness.

1.1 *Programmable Networks*

Since the late 1990s, communication networks have seen the evolution from fixed-function systems to more programmable and ”softwarized” systems. This culminated in the first major breakthrough as Software Defined Networking (SDN) around the early 2010s [5]. SDN introduced and refined the paradigm of viewing the network as two layers of abstraction - (i) a Control Plane (that manages how traffic should be handled) and - (ii) a Data Plane (that forwards traffic based on decisions made by the Control Plane), as shown in Figure 1.

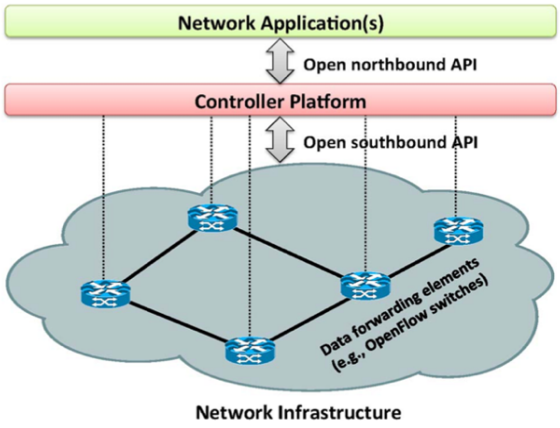


Fig. 1: Software-Defined Networking Architecture [6]

SDN's decoupling, along with centralized control and enhanced programmability, opened up several novel and pragmatic approaches to network management and security problems [6]. While SDN enabled a programmable control plane, programmable data planes emerged as the next phase of evolution. SmartNICs such as NVIDIA Blue-Field DPU [7], FPGA-based platforms like NetFPGA [8] and NetFPGA Sume [9], and devices programmed using P4 [10] now allow for highly performant customized and granular packet processing. This chapter deals with leveraging the P4 programmable packet-forwarding data plane towards ICS network security.

A P4 network switch comprises a programmable data plane based on unique application-specific integrated circuits (ASICs) and a programmable control plane based on a general-purpose CPU environment. This hybrid architecture enables a Dynamic Data-Driven Application Systems (DDDAS) approach to develop a security framework that enhances the attack resilience of industrial control networks in power systems. DDDAS is a paradigm that involves dynamically incorporating real-time data into computations to guide the measurement and control processes of application systems [11], and has been applied in power grid analysis, cyber security, and many other applications [12–16]. In our P4-based security framework, the data plane generates real-time and dynamic data-driven stats and metrics, which are then analyzed using advanced data analytics and learning algorithms on the control plane to detect attacks. The control plane then steers the measurement process in a feedback control loop by installing network updates on the match-action pipelines of the data plane to mitigate attacks.

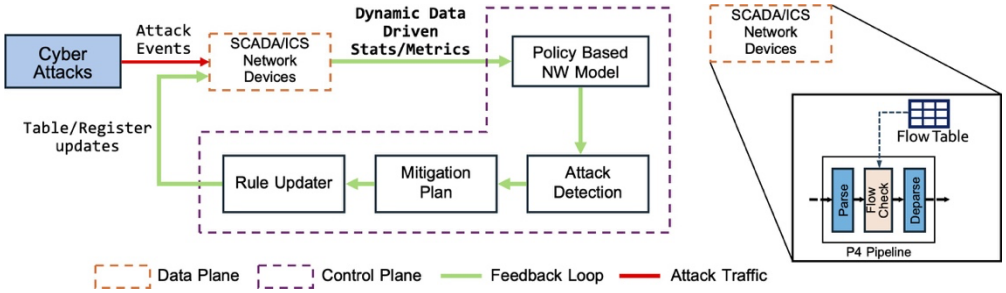


Fig. 2: ICS Security Architecture Using P4-based DDDAS

1.1 P4 for ICS

Studies on network traffic for Operational Technologies (OT) such as ICS [17, 18] have shown that OT traffic differs from typical Information Technology (IT) traffic. ICS traffic is more predictable, has a high level of periodicity, and often uses simple, proprietary protocols. These characteristics prove advantageous as they enable us to define security policies that can be centrally managed by a controller and implemented on the P4 switch. Given the stability of network topology, known communication paths, and traffic patterns, the controller can maintain a constant, global security model of the network.

Given the benefits of the DDDAS paradigm, P4 switch capabilities, and ICS traffic characteristics, we thus explore a P4-based DDDAS system for ICS Security. Our proposed framework is a data-driven feedback loop system comprising of (i) high-speed programmable switches that can parse custom protocols for ICS networks, generate dynamic data, and (ii) a controller that can run applications based on these data, and

accordingly update the forwarding or processing behavior of the switches (network data plane). Figure 2 shows our proposed architecture. The switch data plane (orange dotted box) detects any cyber-attack events in the form of anomalous traffic. P4's custom processing capabilities generate key statistics and metrics about the traffic, which are then sent to a controller application. The controller performs further analysis on attack detection and mitigation plans to arrive at a new forwarding behavior, which is then installed in the form of rules on the switch.

This chapter builds upon the work presented in [40], which was published in the Proceedings of the DDDAS 2024 Conference. While the conference paper introduced the P4-based DDDAS framework for Modbus systems and demonstrated a basic Denial-of-Service (DoS) mitigation mechanism, this chapter extends that work by incorporating additional DoS defense strategies. Specifically, we present a per-Function Code rate-limiting mechanism using P4's meter and queue rate configurations, and an adaptive threshold approach against high-rate Modbus requests, highlighting the flexibility of P4 for handling a wider range of attack scenarios in ICS systems. This chapter thus aligns with DDDAS principles by offering a more comprehensive set of mitigation strategies that dynamically adapt to real-time traffic conditions.

This chapter is organized as follows: Section II, presents the general framework of our proposed design. In Section III, provides a background on the Modbus protocol and the proof-of-concept demonstration of a P4-based DDDAS Denial-of-Service defense mechanism. Section IV highlights the per-Function Code rate-limiting approach. Section V presents the adaptive threshold approach for attacks involving high-rate Modbus requests. Section VI outlines existing works in ICS network security. Finally, Section VII presents concluding remarks on generalizability to other protocols and offers future directions for related research.

2 General Framework

Figure 3 depicts the overall framework of our proposed P4-based DDDAS system for ICS security, with an expanded view of the architecture shown in Fig. 2. The framework consists of three main areas: (1) the data plane, (2) the control plane, and (3) the interface between the two.

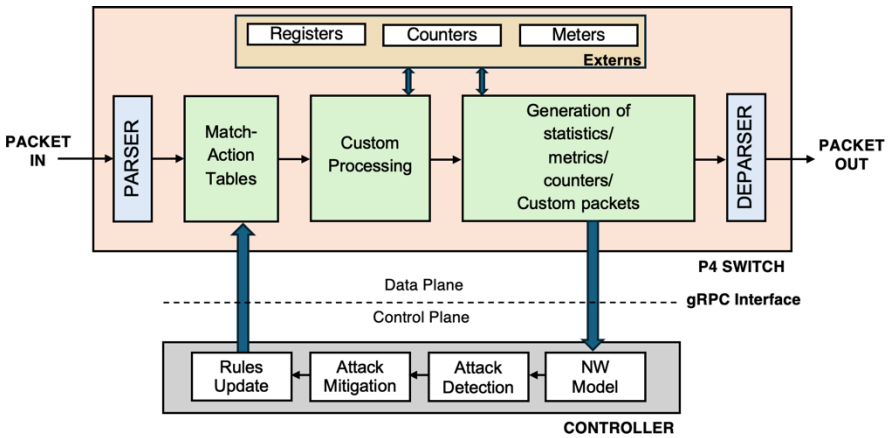


Fig. 3: General Framework for P4-based DDDAS

2.1 Data Plane

The data plane of our P4-based DDDAS framework consists of several key components: the parser, match-action tables, externs, custom processing, and dynamic data generation.

Parser: P4 provides a programmable packet parser that is flexible to parse most protocols. Given that most ICS protocols are unencrypted and follow a simple structure, P4 is an effective choice for parsing. Parsing enables the extraction of key fields of interest, which would then be used in later stages of the pipeline.

Match-Action Tables: These are a set of rules and corresponding actions. Usually, the extracted fields from the parsing stage are used to match against the table entries. For instance, an Internet Protocol (IP)-forwarding table that matches a packet based on the destination IP field specifies what actions to be taken when there is a match(”hit”)/mismatch(”miss”) against a table entry. The actions could be a simple Drop or Forward or perform some custom processing.

Custom Processing: In addition to match-action based validation, P4 can perform a wide range of custom processing involving registers, custom validations, packet modifications, etc. Index-based registers serve as quick storage and retrieval units.

Dynamic Data Generation: P4 supports the generation of custom statistics, metrics, counters, and custom packets that can be interfaced with an external controller for additional processing. In the P4-based DDDAS framework, these serve as dynamic data inputs to the controller for further detection and mitigation.

Externs: To facilitate the generation of such custom statistics and metrics, P4 switch models contain ’Externs’ such as Registers, Counters, and Meters. Registers are indexed arrays for storing and retrieving data and are especially useful for stateful processing. Counter externs enable counting packets or bytes of incoming traffic, at the desired level of granularity. Meter externs allow traffic flows to be classified based on the rate of arrival of packets. The classification is a standardized Two Rate Three Color (trTCM) coloring scheme as described in RFC 2698 [19], which we describe further in Section 4.

2.2 Control Plane

NW Model: The controller runs a policy-based network model of the entire ICS network under consideration. The policy encompasses a defined set of communication constraints within the network infrastructure, such as authorized host-to-host communication, allowed TCP port access, and allowable Function Code interactions. The network model can be initialized from device configurations, known traffic patterns, periodicity information, ports, or even custom logic.

Attack Detection and Mitigation: The dynamic data generated by the switch, coupled with the controller’s computational capabilities, enables the detection of various ICS-specific attacks. Subsequently, the controller formulates rapid mitigation plans by deploying new forwarding rules to the switch.

While the terms ”Controller” and ”Control Plane” typically have distinct meanings, in this Chapter, we use the two terms interchangeably for simplicity, as the methods developed here support both levels. A Controller is a specific device or software entity responsible for managing one or more switches and controlling their forwarding behavior within the data plane. On the other hand, the Control Plane is a conceptual or logical domain where control messages are exchanged between controllers and the data plane of the switches. A Control Plane can consist of multiple controllers, each managing one or

more switches.

2.3 Switch-Controller Interface

The controller communicates with the switch data plane over a gRPC (gRPC Remote Procedure Call [20]) interface. gRPC is used as the transport layer for remote procedure calls, and this allows the controller to dynamically manage the switch's behavior
- update table entries, modify configuration, etc. The switch can transmit various dynamic data over this interface, including packet/byte counters, traffic flow statistics, and customizable application-level information. This information can provide insight into real-time network status, thus enabling attack detection and mitigation at the controller.

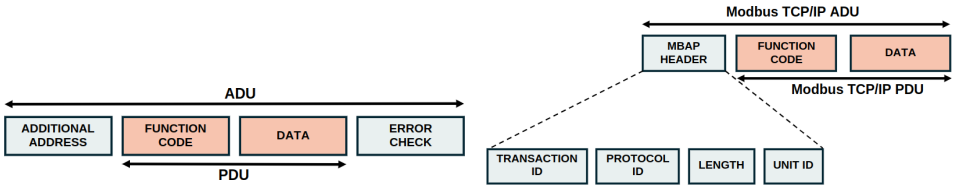
We thus have a dynamic data-driven applications system where the P4 switch provides a first line of cyber defense, and the controller can perform the next level of advanced detection and mitigation. In the subsequent sections, we present how leveraging such a P4-based DDDAS paradigm can help us design defense solutions against various forms of Denial-of-Service and Flooding attacks in Modbus TCP systems.

3 P4-DDDAS for Modbus DoS Defense

In this section, we provide a brief background about the Modbus and Modbus TCP protocols before presenting the threat model and our proposed DDDAS-based defense.

3.1 Background on Modbus

Introduced in 1979, Modbus [21] is a serial communication protocol designed to connect industrial control system (ICS) devices such as Programmable Logic Controllers (PLCs) and sensors. It follows a request-response communication model, with devices assigned Master and Slave roles. The Master sends commands using specific function codes to read from, or write to, the registers of Slave devices. Additionally, Modbus supports various management and diagnostic functions through these codes.



(a) General MODBUS frame (b) Modbus TCP/IP frame

Fig. 4: Modbus packet structures

Modbus TCP is an upgrade over the serial Modbus and uses TCP/IP over Ethernet networks. As a result, Modbus TCP can support faster data transfer rates and longer coverage distances. Figure 4 shows the packet structure for both Modbus

Serial and Modbus TCP. Serial Modbus frames begin with the device address, followed by function code, data payload, and a checksum for error checking. Modbus TCP frames, however, prepend an MBAP (Modbus Application Protocol) header to the function code and data, which includes transaction, protocol identifiers, and length, omitting the device address and checksum for TCP/IP's error handling. Despite the enhancement, Modbus TCP is still vulnerable to attacks due to the lack of inherent security mechanisms.

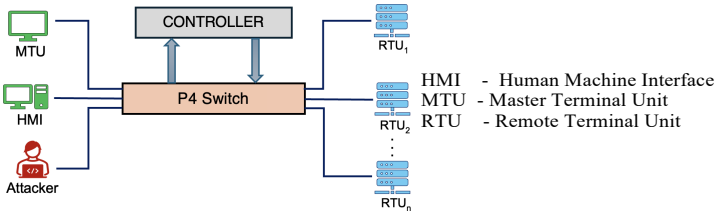


Fig. 5: P4-DDDAS based Modbus TCP Network

Figure 5 shows a Modbus Master-Slave network, where all Master (MTU, HMI) and Slave (RTU) hosts are connected via a P4 switch. The P4 switch is emulated using the BMv2 [22] software switch platform and Mininet [23]. It features local controller software capable of receiving dynamic data-driven inputs, detecting attack traffic, devising mitigation plans, and updating forwarding rules accordingly.

As mentioned earlier in Sec. 1, ICS traffic is mostly periodic. For instance, the work in [17] discusses the concept of "polling" in Modbus networks, where the Master periodically sends request messages to read the register contents on a Slave. In terms of Modbus semantics, this translates to - a certain Master sending a certain Function Code at a predefined periodicity to a certain Slave node. Based on this characteristic, in our case study, we assume that each function code on a slave is defined by a specified periodicity. The utility operator configures each slave device with a set of supported Function Codes and corresponding periodicity values. These values represent the minimum inter-packet delay (in seconds or milliseconds) per function code per client-slave transaction.

3.2 Threat Model

The threat model in our demonstration involves a Modbus-level DoS, where a rogue host spoofs the Master node's IP address and continuously sends Modbus request packets to a Slave device at rates higher than the specified periodicity values. In our context, DoS attacks are primarily intended to overwhelm target slave devices, which typically have limited buffer sizes for processing incoming packets. The objective of our solution is to filter out any anomalous traffic arriving at the switch at a rate higher than the specified periodicity.

We make the following assumptions on the attacker's capability:

- Has full or partial knowledge of the network topology (IP address, TCP ports, etc.)
- Has ability to spoof legitimate Modbus clients (Master devices)
- Has sufficient compute power to send Modbus requests at high rates, to overwhelm Modbus server (Slave) devices.

3.3 Switch and Controller Algorithm

Fig. 6 shows the sequence of operations at the switch and controller. The architecture embodies a closed-looped feedback system, which is a hallmark of DDDAS, by continuously monitoring network traffic at the switch, adapting thresholds and dynamically updating routing and filtering rules based on the collected runtime date.

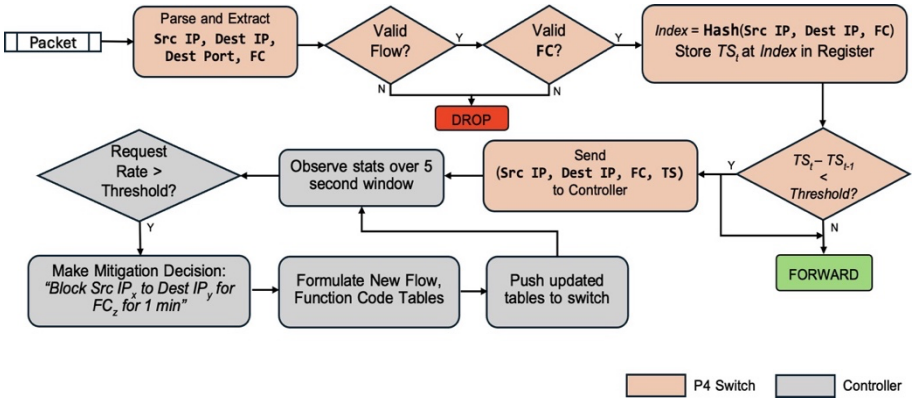


Fig. 6: Switch-Controller Sequence for Modbus DoS Defense

For every Modbus request packet received, the switch performs the following steps:

- Parses and extracts key fields: Source IP address (Src IP), Destination IP address (Dest IP), Destination Port (Dest Port), and Function Code (FC).
- Checks the validity of these fields against pre-populated tables and drops the packet if there is any mismatch.
- Calculates the inter-arrival delay for packets matching the unique combination of Source-IP, Destination-IP, and Function Code.
- Formulates a 4-tuple (Src IP, Dest IP, FC, Timestamp) and sends this information
  to the controller if the arrival rate exceeds a preset threshold T.

At the controller:

- The controller continuously listens to incoming stats from the P4 switch over a pre-configured TCP port.
- The controller monitors stats over a 5-second moving window and compares against a threshold rate. Both the controller threshold and the switch-side threshold are configurable parameters
- If the switch reports > 20 instances of high-rate traffic within a 5-second window for a particular 3-tuple (Src IP, Dest IP, FC), the controller identifies it as attack traffic.
- The controller formulates updated Flow and Function Code tables and pushes the table entries onto the switch.
- The switch continues to forward higher-than-threshold rate traffic until it

receives the updated table rules. Upon receiving the updated rules, the switch blocks all subsequent packets that match the 3-tuple.

The threshold choice of 20 instances is an illustrative value, primarily to demonstrate the operation of our framework in this study. In actual implementations with continuous monitoring, one may employ an adaptive threshold, such as percentage-based increase in traffic, which would be more practical.

Table 1: Round-Trip Latency between Switch-Controller

# Master Devices	# Slave Devices	# Function Codes	# Combinations	Avg. Latency (ms)	Std. Dev. (ms)
2	3	2	12	1.39	0.44
2	5	10	100	1.41	0.88
8	10	13	~1000	2.6	1.64
21	21	25	~11000	7.82	8.8

3.4 Evaluation of Switch-Controller Latency

We evaluated the performance of our algorithm by measuring the round-trip latency between the switch and the controller. The round-trip latency encompasses the time from the switch sending 4-tuple stats to the controller, the controller performing attack detection, to the switch receiving updated blocking rules. Table 1 presents findings across various scenarios, varying the number of master nodes, slave nodes, and sup- ported function codes per slave. The latency remains minimal (millisecond range) even with increasing complexity, ensuring scalability. In the context of Modbus systems, our focus is on preventing devices from being overwhelmed by abnormal traffic flows, swiftly identified by our P4-based in-network solution. Given the challenges in establishing accuracy metrics without a definitive ground truth, we prioritize latency evaluation as it aligns with the real-time constraints of ICS operations.

4 Per-Function-Code Rate-Limiting

Exploiting sensitive function codes is another threat that an attacker can exploit within Modbus systems. These are function codes that trigger resource-intensive operations on the slave, such as diagnostics or configuration changes. Overwhelming a slave device with such function codes can lead to reduced service availability, increased processing delays, or even device failures.

4.1 High-Level Design

To mitigate the consequences of *transaction flooding*, we adopt a rate-limiting mechanism that selectively controls traffic volume based on function code priority, instead of outright dropping packets. This rate-limiting is enforced through a data-driven feed-back loop between the switch and the control plane. Programmable data planes allow fine-grained enforcement of rate limits at different flow levels. In this approach, we offload most of the detection logic onto the switch data plane. In contrast, the control plane is primarily responsible for dynamically adjusting the switch's egress queue rate upon detecting an attack. For the setup shown in Figure 5, consider a scenario where a Master sends Modbus traffic to Slave RTU-1, involving multiple function codes — some

of lower priority (acceptable for rate-limiting) and others highly sensitive (thus critical to regulate). Figure 7 shows the design overview for the per-Function Code rate limiting defense involving the P4 switch and controller. After parsing the packet headers and checking for valid flow and function code, the meter is applied to measure the traffic rate of all flows directed to a given slave device (destination IP address). Under high traffic conditions, possibly an attacker attempting to overwhelm a slave with a sensitive function code, the P4 switch detects the excessive traffic specific to a sensitive function code and notifies the control plane of a threshold violation.

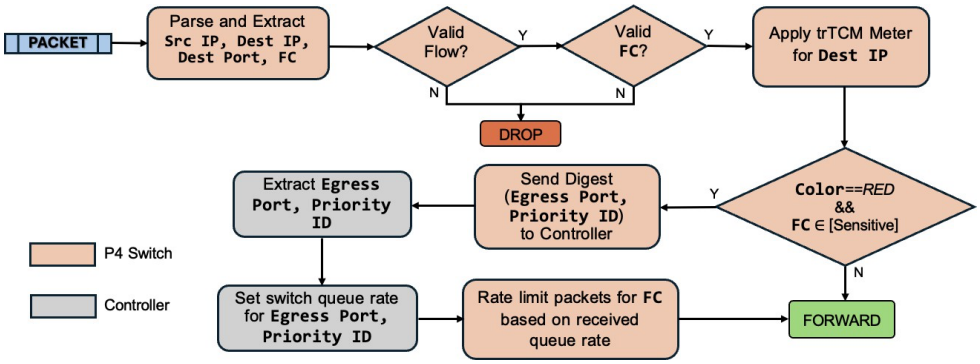


Fig. 7: Switch-Controller Sequence for per-Function Code Rate Limiting

We devise the defense logic such that if there is sustained high-rate traffic above a preset threshold (tagged Red by the meter) directed to a particular device, the switch sends a digest notification to the control plane indicating – (i) the egress port on the switch that needs to be rate-limited and (ii) the priority ID that uniquely maps to traffic belonging to low-priority/sensitive function codes. The control plane then immediately issues a control message to enforce rate-limiting at the affected egress port, but only for packets matching the identified priority ID.

4.2 P4 and Controller Implementation

As briefly described in Section 2, P4 switches provide meter externs that can classify flows based on the rate of arrival to the switch. The classification is the Two Rate Three Color (trTCM) mechanism as described in RFC 2698 [19]. The tcTCM measures an IP packet stream against two rates - Peak Information Rate (PIR) and the Committed Information Rate (CIR) - along with their corresponding peak (PBS) and committed burst sizes (CBS) . Each packet is labeled green, yellow, or red. Any packet that surpasses the PIR is marked red. If it doesn't exceed the PIR, then the packet is marked either yellow or green, depending on whether it exceeds the CIR or not.

Meters in P4 provide flexibility in configuration to color mark flows at our desired level of granularity. Some examples of granularity could be:

- all flows to a given destination IP address
- all flows between a pair of IP addresses
- all flows directed to a given TCP port, etc.

```
{
" table ": " FC_priority ",
" key ":
      {
    " hdr.ipv4.dstAddr ": " exact",
    " hdr.modbus.functionCode ": "
       exact"
  },
" actions ":
     [ "set_priority",
      "No Action"],
" size ": 1024,
" default_action ": "NoAction()"
}
```

Listing 1: FC-Priority Table

```
table_add MyIngress.FC_Priority
 My_Ingress.set_priority 10.0.2.1 1
  => 0

table_add MyIngress.FC_Priority
 My_Ingress.set_priority 10.0.2.2 4
  => 0

table_add MyIngress.FC_Priority
 My_Ingress.set_priority 10.0.2.2 8
  => 1
```

Listing 2: Runtime CLI Commands

Fig. 8: P4 table structure and corresponding table entries

The meter extern is programmed to classify the net Modbus traffic towards a particular slave device connected to one of the switch egress ports. Note that this traffic would include all Function codes. Assume that only Function Codes 1, 4, and 8 are allowed. Similar to the pipeline logic in Figure 6, we first filter out traffic with Function Codes not in the allowed list. Among the allowed codes, we treat Function Code 8 as sensitive or low priority. Specifically, Function Code 8 is used for diagnostics, and when combined with sub-Function Code 04 (Force Listen Mode), it instructs the slave device to enter Listen Mode, halting its response to requests. Thus, all traffic containing this FC should be rate-limited under load conditions, to prevent the slave from being overwhelmed. We achieve this by first mapping each function code + destination IP combination to a priority ID. This mapping is done in the P4 switch pipeline program via a match-action table structure (Listing 1) and corresponding runtime CLI table addition commands (Listing 3).

The action `set_priority` is defined as:

```
action set_priority (bit<3> prio) {
       standard_metadata.priority = prio;
}
```

Listing 3: P4 Action Block for setting priority to a packet

4.3 BMv2 Switch Runtime CLI Configuration

A trTCM meter typically requires configuring four parameters - Peak Information Rate (PIR), Peak Burst Size (PBS), Committed Information Rate (CIR), and Committed Burst Size (CBS) to classify traffic. To have a fixed threshold, we configure the same values for PIR and CIR, so that all traffic is either Green or Red.

To make the meter monitor and classify the net Modbus traffic destined to a slave IP address, we configure:

```
meter_set_rates MyIngress.my meter 1 0.001:50 0.001:50
```

where:

`MyIngress.my meter` is the name of the meter instance
`1` denotes the index of the meter entry, allowing multiple entries with different configurations
`0.001:50` specifies first rate-burst pair; `0.001` sets the CIR in packets/µs, and `50`
sets the CBS in packets
`0.001:50` specifies second rate-burst pair; `0.001` sets the PIR in packets/µs, with a PBS of `50` packets

The queue rate modification command issued by the controller on receiving a notification from the data plane is as follows:

```
set_queue_rate 50 2 1
```

where:

`50` - denotes queue rate in packets/s
`2` - indicates the egress port on the switch for which the queue rate is set
`1` - denotes priority ID that was set by the P4 program as shown in Listing 3

The above command effectively limits the maximum rate on egress port 2 for packets marked with a priority ID of 1. With a packet size of 66 bytes for a Modbus request, this translates to around 26 Kbps ($50 \frac{\text{packets}}{\text{s}} \times 66 \frac{\text{bytes}}{\text{packet}} \times 8 \frac{\text{bits}}{\text{byte}} = 26400$ bits/s).

4.4 Evaluation and Discussion

We demonstrate the rate-limiting mechanism of our solution by sending traffic containing three different function codes - 1,4, and 8 - from a master to a slave. We designate Function Code 8 (FC=8) as sensitive - the flow that needs to be rate-limited. Figure 9 shows the throughput plot for each of the function code flows sent from the master and the net throughput received at the slave (dashed line). As we introduce flows of 200 Kbps for Function codes 1 (green) and 4 (orange), the net throughput received at the Slave increases accordingly. For FC=8 (grey), when we inject occasional bursts around the 9s and 11s mark, the net throughput at the slave still increases accordingly, yet the switch does not enforce the rate limiting. However, at around the 11s mark, when there is sustained high-rate traffic from FC=8 flow, we observe that the rate-limiting mitigation is enforced after a short delay. Consequently, the throughput received at the slave for FC=8 flow is maintained at a relatively constant limit of around 26 Kbps, aligning with the configured queue rate of 50 packets/s for this flow. The net throughput is also thus maintained at a constant rate, even when the high-rate flow for FC=8 continues to persist. The mitigation delay (approx. 1-1.5 seconds) is the total time incurred between the P4 switch detecting

sustained high-rate traffic for FC=8, notifying the control plane, and the control plane issuing a CLI command to limit the queue rate for this sensitive flow. Thus, overall, our solution tolerates occasional spikes or bursts but quickly rate-limits any sustained high-rate sensitive flow.

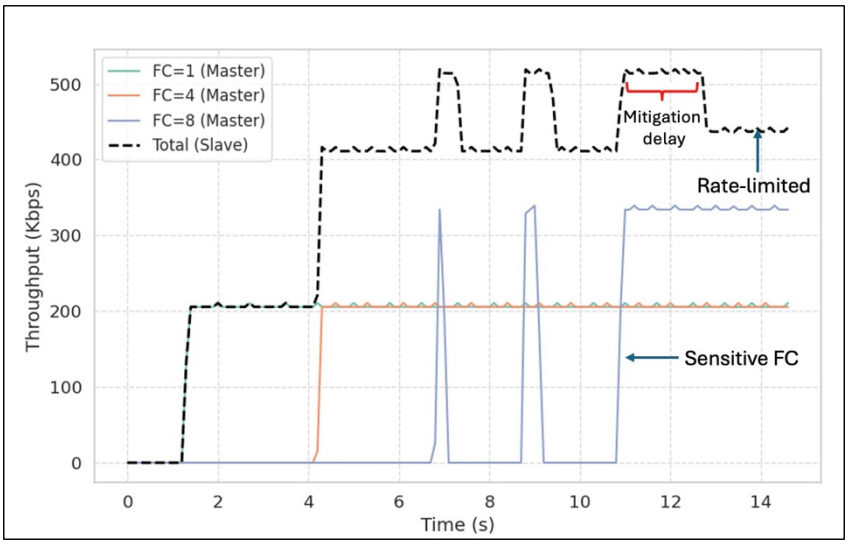


Fig. 9: Per-Function Code Rate-Limiting

The mitigation delay is influenced by the number of burst packets the switch can handle before it begins marking subsequent packets as Red, excluding any communication delays between the switch and the controller. Configuring a larger burst size (PBS and CBS values) in the meter settings results in a longer mitigation delay. Therefore, choosing an optimal burst size is a trade-off between allowing short bursts and timely rate-limiting.

The meter configuration parameters, specifically the PIR and CIR for the meter, as well as the queue rate for an egress switch port, also significantly impact the results. The PIR and CIR influence the thresholds for packet classification, while the queue rate directly affects the egress throughput. It is worth mentioning that these parameter values for configuring the meter and queue rates were chosen somewhat arbitrarily, primarily to demonstrate the potential of our proposed P4-based DDDAS for ICS. In real-world or large-scale deployments, one should iteratively tune these parameters based on network topology, expected traffic rates, required Quality of Service, and the anticipated duration of bursty phases to maximize defense effectiveness while minimizing unnecessary drops.

Another plausible design is to program the switch to continuously monitor and periodically report the traffic rate (for all flows) to the controller. Although the P4 data-can count flows at any desired granularity, the counting approach shifts most of the attack detection to the control-plane, where advanced algorithms, statistical analysis or even machine-learning models can be run on general-purpose hardware. Such sophisticated detection methods aid in catching complex attacks but incur increased communicated overhead between switch and controller, even during benign

traffic. Thus, a robust, data-driven defense must carefully balance this overhead against the gains in detection accuracy.

5 Adaptive Traffic Policing

In the scenario described in Section 3, we employed a fixed flat-rate threshold for dropping high-rate Modbus Request packet flows. Enforcing a fixed threshold alone runs the risk of dropping legitimate traffic that could have bursty periods. An alternative to this approach is to employ an adaptive mechanism that integrates both a flat-rate threshold and an exponential weighted moving average (EWMA) threshold.

Given the bursty nature of network traffic, EWMA helps maintain a drop threshold that is proportional to the rate of packet arrival, allowing greater tolerance to short-term bursts while still protecting against sustained high-rate attacks. The Exponentially Weighted Moving Average (EWMA) threshold is updated using the following formula:

$$T_{new} = (1 - \alpha)T_{old} + \alpha M \quad (1)$$

where:

T_{old} is the current threshold, representing the maximum number of Modbus request packets allowed within a predefined interval,
T_{new} is the updated threshold for the next interval,
M is the number of Modbus request packets received within that interval,
α is a smoothing parameter in the range $(0, 1)$ that controls the influence of recent observations.

In our evaluation, we present a rate-limiting mechanism that is range-based, as it demonstrates P4's flexibility to enforce both a blanket drop policy and an adaptive drop policy, proportional to the arrival rate of Modbus requests. Similar to the CIR and PIR thresholds in meters, we defined Upper (U) and Lower (L) threshold values of 40 packets/s and 60 packets/s.

The rate-limiting logic is as follows:

- Packet rate ≤ 40 packets/s : forward
- Packet rate ≥ 60 packets/s : drop
- 40 < Packet rate < 60 packets/s : apply EWMA threshold (α = 0.0625)

Figure 10 shows the instantaneous packet rates sent by the master (red) and received by the slave (green). To illustrate range-based adaptive policing by the P4 switch, Modbus Request packets were replayed by the master device at increasing bit rates over 5-second intervals. When the master sends Modbus request packets below the lower threshold *L*, the switch forwards all such packets to the slave with no rate-limiting, as can be seen from the green lines directly following the red lines. When the master sending rate exceeds the upper threshold *U*, no green lines follow the red, indicating the switch performs a blanket drop for all packets exceeding *U*. For the packets arriving at a rate between *L* and *U*, the switch proportionally rate-limits using the EWMA threshold, as can be seen by the intermittent sections of green lines.

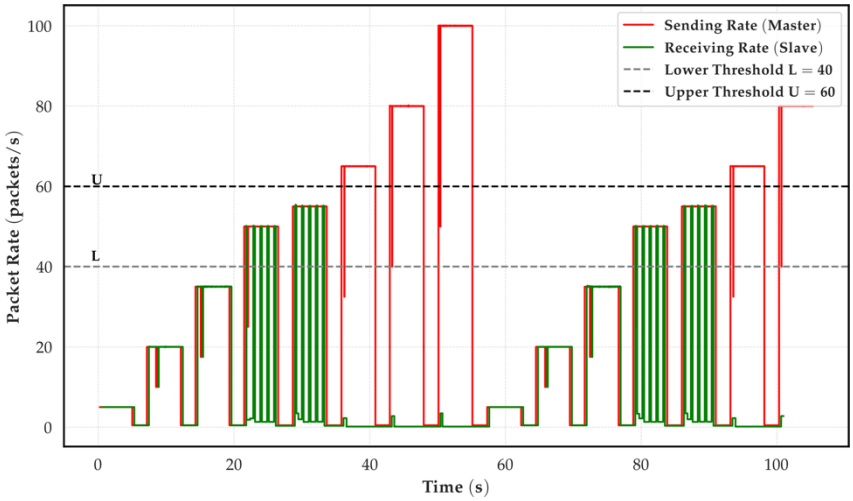


Fig. 10: Range-based adaptive traffic policing

6 Related Work

ICS networks have a long operational history, and several approaches exist in the literature dealing with their security. Very broadly, we classify such related work into Conventional approaches and Programmable Networks-based (SDN/P4) approaches.

6.1 Conventional ICS Security Approaches

Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) are popular approaches in detecting and safeguarding against malicious traffic [24–26]. In [27], the authors exploit Bro [28], a customizable IDS solution, to implement a parser and anomaly detection for SCADA protocols. A flow-based whitelisting approach is proposed in [18] and [29], while [30] presents a variation using flow-based whitelisting by inverting the black-listing logic.

Deep packet inspection (DPI) based approaches are another prominent method, enabling the detection and mitigation of highly specialized attacks concealed within communication flows [31, 32]. Additionally, a cyber-physical model-based approach is introduced in [33] to assess network intrusions by correlating network behavior with physical system states.

6.2 SDN/P4-based ICS Security Solutions

Conventional approaches listed above most commonly involve a single point of deployment, typically at traffic aggregation points, to intercept and mirror the traffic. However, programmable network approaches such as Software-Defined Networking (SDN) and P4 often adopt a distributed architecture, offering a broader defense perimeter without the need for traffic mirroring.

In [34], the authors propose an SDN-based solution to prevent eavesdroppers from fully capturing communication flows between SCADA components. The work in [35] introduces a strategic framework that models the interaction between a hypervisor

monitoring virtual SDN controllers and potential sources of Distributed Denial-of-Service (DDoS) attacks.

Furthermore, the work in [36] presents a two-level P4-based IDS for Modbus TCP systems. The first level handles flow and application-level checks within the P4 switch, while the second level involves a controller that performs deep packet inspection (DPI) and dynamically updates flow entries in the switch for enhanced threat detection and mitigation. In [41], the authors present a cross-domain policy enforcement pipeline on hardware P4 (Tofino) for Modbus systems. The policies involve flow, packet structure, application and timing-based checks. In [37], the authors use P4 to enhance security of DNP3-based SCADA systems by implementing packet-level security measures—such as inspection, filtering, hashing, and encryption—to effectively mitigate DNP3-specific cyber-attacks such as malformed packets, denial-of-service, and man-in-the-middle attacks. In [38], the authors develop an IDS for DNP3 data via a Decision Tree model implemented entirely on the P4 data plane using optimized data structures, for detecting DDoS and False Data Injection attacks.

7 Conclusion

In this Chapter, we proposed a P4-based DDDAS framework for securing ICS networks and demonstrated through a DoS defense system in a Modbus environment. Our proposed framework leverages the capabilities of both programmable P4 switches and controller applications. By integrating programmable switches and controller applications in a real-time feedback loop, our approach delivers low-latency, flexible and adaptive mitigation. While our prototype focused on Modbus, much of the P4 pipeline design, including field validation, traffic monitoring, and rate-limiting modules are protocol-agnostic and can be reused with minimal changes. Substantial new development would be required only in the parsing and header-specific stages. This modularity enables extension to other ICS protocols, including DNP3 and IEC 60870-5. Also, future work will implement a multi-protocol hardware-based deployment and explore statistical detection models leveraging advanced P4 telemetry.

Acknowledgements. The authors are grateful for the support of the National Science Foundation (NSF) under Grant CNS-2247721, CNS-2034870, and the U.S. Department of Energy, Office of Cybersecurity, Energy Security, and Emergency Response (CESER) under Award Number DE-CR00000031. The authors also thank Dr. Yanfeng Qu for his valuable comments and feedback.

References

[1] Newton-Evans Research Company (2019) 94% of North American Electric Utilities Surveyed Use DNP3 for SCADA. https://www.newton-evans.com/94-of-north-american-electric-utilities-surveyed-use-dnp3-for-scada/ Accessed: Aug 19, 2025

[2] American Public Power Association (2022) Finding Value in Utility Data. https://www.publicpower.org/periodical/article/finding-value-utility-data Accessed: Aug 19, 2025

[3] Drias Z, Serhrouchni A, Vogel O (2015) Taxonomy of attacks on Industrial Control Protocols. In: International Conference on Protocol Engineering (ICPE) and International

[4] Huitsing P, Chandia R, Papa M, Shenoi S (2008) Attack taxonomies for the Modbus protocols. International Journal of Critical Infrastructure Protection 1 (2008)

[5] Feamster N, Rexford J, Zegura E (2014). The road to SDN: an intellectual history of programmable networks. ACM SIGCOMM Computer Communication Review, 44(2), 87-98.

[6] Kreutz D, Ramos FMV, Verissimo PE, Rothenberg CE, Azodolmolky S, Uhlig S (2014). Software-defined networking: A comprehensive survey. Proceedings of the IEEE, 103(1), 14-76.

[7] NVIDIA Corporation: NVIDIA BlueField Data Processing Unit (DPU) (2024) https://www.nvidia.com/en-us/networking/products/data-processing-unit/ Accessed: Aug 23, 2025

[8] NetFPGA Project: NetFPGA - Open-Source Platform for Networking Research (2024). https://netfpga.org/ Accessed: Aug 23, 2025

[9] Zilberman, N., Audzevich, Y., Covington, G.A., Moore, A.W (2014) Netfpga sume: Toward 100 gbps as research commodity. IEEE micro 34(5), 32–41

[10] Bosshart P, et al (2014) P4: Programming protocol-independent packet processors. ACM SIGCOMM Computer Communication Review 44(3), 87–95

[11] Darema F (2004) Dynamic data driven applications systems: A new paradigm for application simulations and measurements. In: International Conference on Computational Science, pp. 662–669. Springer

[12] Blasch EP, Darema F, Ravela S, Aved AJ (2022) Handbook of Dynamic Data Driven Applications Systems: Volume 1. Springer, https://doi.org/10.1007/978-3-030-74568-4

[13] Darema F, Blasch EP, Ravela S., Aved AJ (2023) Handbook of Dynamic Data Driven Applications Systems: Volume 2. Springer, https://doi.org/10.1007/978-3-031-27986-7

[14] Fujimoto R, Barjis J, Blasch E, Cai W, Jin D, Lee S, Son YJ (2018) Dynamic data driven application systems: research challenges and opportunities. In: 2018 Winter Simulation Conference (WSC) (pp. 664-678). IEEE.

[15] Blasch, E, Al-Nashif Y, Hariri, S (2014) Static versus dynamic data information fusion analysis using DDDAS for cyber security trust. Procedia Computer Science, 29, 1299-1313

[16] Qu Y, Liu X, Yan J, Jin D (2020) Dynamic data-driven self-healing application for phasor measurement unit networks. In: International Conference on Dynamic Data Driven Applications Systems, pp. 85–92. Springer

[17] Barbosa RRR, Sadre R, Pras A. (2012) A first look into SCADA network traffic. In 2012 IEEE Network Operations and Management Symposium (pp. 518-521). IEEE.

[18] Barbosa, RRR, Sadre R, Pras A (2013) Flow whitelisting in SCADA networks. International Journal of Critical Infrastructure Protection, 6(3-4), 150-158.

[19] Heinanen DJ, Guerin DR (1999) A Two Rate Three Color Marker

https://doi.org/10.17487/RFC2698

[20] gRPC - A High Performance, Open-Source Universal RPC Framework (2024) https://grpc.io Accessed: Aug 23, 2025

[21] modbus.org: MODBUS Application Protocol Specification (2006) https://modbus.org/docs/Modbus_Application_Protocol V1 1b.pdf Accessed: Aug 23, 2025

[22] p4.org: Behavioral Model (bmv2). https://github.com/p4lang/behavioral-model Accessed: Aug 23, 2025.

[23] Lantz B, Heller B, McKeown N (2010) A Network in A Laptop: Rapid Prototyping For Software-Defined Networks. In: 9th ACM SIGCOMM Workshop on Hot Topics in Networks, pp. 1–6

[24] Yusheng W, Kefeng F, Yingxu L, Zenghui L, Ruikang Z, Xiangzhen Y, Lin L (2017) Intrusion Detection of Industrial Control System Based on Modbus TCP Protocol. In 2017 IEEE 13th International Symposium on Autonomous Decentralized System (ISADS) pp. 156-162. IEEE

[25] Goldenberg N, Wool A (2013) Accurate Modeling of Modbus/TCP For Intrusion Detection In SCADA Systems. International Journal of Critical Infrastructure Protection, , 6(2), 63-75

[26] Fovino IN, Carcano A, Mure TDL, Trombetta A, Masera M (2010) Modbus/DNP3 State-Based Intrusion Detection System In: 24th IEEE International Conference on Advanced Information Networking and Applications. IEEE

[27] Udd R, Asplund M, Nadjm-Tehrani S, Kazemtabrizi M, Ekstedt M (2016) Exploiting Bro for Intrusion Detection in a SCADA System. In: 2nd ACM International Workshop on Cyber-Physical System Security, pp. 44–51

[28] Paxson V (1999) Bro: A System for Detecting Network Intruders in Real-Time. Computer networks 31(23-24), 2435–2463

[29] Kang D, Kim B, Na J, Jhang K (2014) Whitelists Based Multiple Filtering Techniques In SCADA Sensor Networks. Journal of Applied Mathematics, 2014(1), 597697

[30] Lemay A, Rochon J, Fernandez JM (2016) A Practical flow white list Approach for SCADA Systems. In: 4th International Symposium for ICS & SCADA Cyber Security Research 2016

[31] Nyasore ON, Zavarsky P, Swar B, Naiyeju R, Dabra S (2020) Deep packet inspection in industrial automation control system to mitigate attacks exploit- ing modbus/tcp vulnerabilities. In: 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS) pp. 241-245. IEEE

[32] Wakchaure M, Sarwade S, Siddavatam I (2016) Reconnaissance of industrial control system by deep packet inspection. In: IEEE International Conference on Engineering and Technology. IEEE

[33] Sheng C, Yao Y, Fu Q, Yang W (2021) A cyber-physical model for SCADA system and its intrusion detection. Computer Networks, 185**,** 107677

[34] da Silva, E. G., Knob, L. A. D., Wickboldt, J. A., Gaspary, L. P., Granville, L. Z., & Schaeffer-Filho, A. (2015, May). Capitalizing on SDN-based SCADA systems: An anti-eavesdropping case-study. In 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM) (pp. 165-173). IEEE.

[35] Niazi, R.A., Faheem, Y. (2019) A bayesian game-theoretic intrusion detection system for hypervisor-based software defined networks in smart grids. IEEE Access **7**, 88656–88672

[36] Ndonda, G.K., Sadre, R (2018) A two-level intrusion detection system for industrial control system networks using p4. In: 5th International Symposium for ICS & SCADA Cyber Security Research 2018 5, pp. 31–40

[37] Hu, Z., Lin, H., Waind, L., Qu, Y., Chen, G., Jin, D.: Industrial network protocol security enhancement using programmable switches. In: 2023 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), pp. 1–7 (2023)

[38] Hu, Z., Lin, H., Qu, Y., Jin, D (2024) Leveraging compact data accumulator to enable in-network anomaly detection in programmable switches for power grids. In: 2024 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), pp. 548–554

[39] IEEE Standard for Electric Power Systems Communications—Distributed Network Protocol (DNP3) (2012), IEEE Std 1815™-2012

[40] Samson Raj, R., Jin, D (2026). Dynamic Data Driven Security Framework for Industrial Control Networks Using Programmable Switches. In: Blasch, E., Darema, F., Metaxas, D. (eds) Dynamic Data Driven Applications Systems. DDDAS/Infosymbiotics for Reliable AI 2024. Lecture Notes in Computer Science, vol 15514. Springer, Cham. https://doi.org/10.1007/978-3-031-94895-4_16

[41] Samson Raj, R., & Jin, D (2025) Leveraging Data Plane Programmability Towards a Policy-driven In-Network Security Framework for Industrial Control Systems. In Proceedings of the 16th ACM International Conference on Future and Sustainable Energy Systems (pp. 152-163).