UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

BRIGGETTE OLENKA ROMÁN HUAYTALLA

DWT in P4: Periodicity Detection in the Data Plane

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Advisor: Prof. Dr. Lisandro Z. Granville

Porto Alegre
September 2023

CIP — CATALOGING-IN-PUBLICATION

Huaytalla, Briggette Olenka Román

DWT in P4: Periodicity Detection in the Data Plane / Briggette Olenka Román Huaytalla. – Porto Alegre: PPGC da UFRGS, 2023.

48 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2023. Advisor: Lisandro Z. Granville.

1. DWT. 2. P4. 3. Periodicity Detection. 4. Haar Wavelets. 5. Programmable Data Plane. I. Granville, Lisandro Z.. II. Título.

“Keep it simple, as simple as possible,
but not simpler.”
— Albert Einstein

ACKNOWLEDGMENT

Firstly, I would like to start by thanking my mothers, Raquel, Sara, and Rosula, because you always supported me from a distance, encouraged me, and reminded me of my way and my dreams of pursuing my master's degree in another country. Mommy, until your last days, you guided me and reminded me of what my duty was and gave me the motivation to finish my master's degree; you always will be in my mind and my heart. Raquel, besides being my mother, you are my best friend. Thank you for all the support, love, understanding, and advice you give me daily. It was a great help all this time. Mom Sara, you always encourage me and support my dreams and ideas. All of you are the best of my life. I love you so much.

I also thank my advisor, Prof. Lisandro Zambenedetti Granville, first for the opportunity and acceptance to be my advisor for believing and investing in me. I am grateful for the valuable lessons and teachings that you gave me to be a better researcher. In addition, I would like to thank my co-advisor, Prof. Ronaldo Alves Ferreira, for his guidance and time, always trying to polish my inner researcher and helping me with the most challenging problems in our research. Equally, I would like to thank Arthur Selle Jacobs for his guidance, advice, and time to help me understand our research's core.

Also, I am thankful to my professors from the Institute of Informatics at UFRGS, who taught me, guided me, and gave me their support during my master's degree, professors Jefferson Campos Nobre and Luciano Paschoal Gaspary. And thank the administrative staff of the informatics institute, Luis Otavio, Silvania, Eliane and Rossane, for the support provided.

Also, I want to thank the JEMS project of which I was a part, and it took me to meet Marcus and Tiago, with whom I had the honor of teaming up.

I thank my best friend and brother, Victor, for his support. He was always ready to help me when needed, even from a distance, and encouraged me at every step I took. During my journey toward the master's degree, I met exceptional people who always encouraged and supported me. Laura, Marcelo, Francisco, Nicolas, Ricardo, Luciano and Faezeh, thank you very much. Laura and Marcelo, you are an essential part of my life. You became my guardian angels and were my main support in the most difficult times, and Francisco joined the group. Thank you very much for all the support guys. I am very grateful to meet you. I adore you very much. All of you have my profound appreciation.

ABSTRACT

This dissertation presents an extended implementation of the (1-D) Discrete Wavelet Transform (DWT) method in the P4 programming language, enabling efficient and real-time analysis of periodic behavior in network traffic. The DWT is a mathematical tool widely used for signal analysis, allowing the division of a given signal into different frequency components and analyzing each component with a resolution tailored to its scale. By addressing the limitations of existing P4-programmable data plane devices, we develop an efficient online algorithm that performs the DWT decomposition entirely in the data plane, overcoming constraints and complexities associated with offloading computations to external devices or relying solely on centralized controllers. Our evaluation focuses on a hardware implementation of the algorithm, utilizing the Netronome NFP-4000 SmartNIC, and demonstrates minimal throughput overhead, with less than 1% impact on average-sized packets, while operating within the constraints of limited data plane resources. In addition to the implementation, we showcase a practical application of our lightweight P4 implementation by introducing a novel threshold-based approach for real-time detection of periodic behavior in signals, enabling efficient and timely identification of periodic patterns at line rate in the data plane (40 Gbps). Various examples of synthetic and real-world packet-level traffic traces, exhibiting periodic patterns of both benign and malicious origins, illustrate the effectiveness of our approach. The contributions of this dissertation extend to both the field of network traffic analysis and the practical implementation of the DWT in programmable data planes, offering opportunities for real-time analysis and detection of periodic behaviors directly in the network fabric. Our approach demonstrates scalability, efficiency, and accuracy, making it a valuable tool for applications such as anomaly detection, congestion control, and network security. This dissertation contributes to the advancement of in-network traffic analysis and provides a foundation for future research in the domain, showcasing the viability and potential of performing the DWT entirely in the data plane with minimal overhead and constraints, and highlighting the benefits of in-network traffic analysis for network management and security.

Keywords: DWT. P4. Periodicity Detection. Haar Wavelets. Programmable Data Plane.

DWT em P4: Detecção de Periodicidades em Plano de Dados

RESUMO

Esta dissertação apresenta uma implementação estendida do método da Transformada Discreta de Wavelet (DWT, na sigla em inglês) de uma dimensão na linguagem de programação P4, permitindo uma análise eficiente e em tempo real do comportamento periódico no tráfego de rede. A DWT é uma ferramenta matemática amplamente utilizada para análise de sinais, permitindo a divisão de um sinal dado em diferentes componentes de frequência e analisando cada componente com uma resolução adaptada à sua escala. Ao abordar as limitações dos dispositivos de plano de dados programáveis em P4 existentes, desenvolvemos um algoritmo online eficiente que realiza a decomposição DWT inteiramente no plano de dados, superando as restrições e complexidades associadas ao deslocamento de cálculos para dispositivos externos ou dependendo exclusivamente de controladores centralizados. Nossa avaliação concentra-se em uma implementação de hardware do algoritmo, utilizando o Netronome NFP-4000 SmartNIC, e demonstra um mínimo impacto na taxa de transferência, com menos de 1% de impacto em pacotes de tamanho médio, operando dentro das restrições dos recursos limitados do plano de dados. Além da implementação, demonstramos uma aplicação prática de nossa implementação leve em P4, introduzindo uma abordagem baseada em limiar para a detecção em tempo real do comportamento periódico em sinais, permitindo a identificação eficiente e oportuna de padrões periódicos na taxa de linha do plano de dados (40 Gbps). Vários exemplos de traços de tráfego de nível de pacote sintéticos e do mundo real, exibindo padrões periódicos de origens benignas e maliciosas, ilustram a eficácia de nossa abordagem. As contribuições desta dissertação se estendem tanto ao campo da análise de tráfego de rede quanto à implementação prática da DWT em planos de dados programáveis, oferecendo oportunidades para análise em tempo real e detecção de comportamentos periódicos diretamente no tecido da rede. Nossa abordagem demonstra escalabilidade, eficiência e precisão, tornando-se uma ferramenta valiosa para aplicações como detecção de anomalias, controle de congestionamento e segurança de rede. Esta dissertação contribui para o avanço da análise de tráfego em rede e oferece uma base para pesquisas futuras no domínio, demonstrando a viabilidade e o potencial de realizar a DWT inteiramente no plano de dados com um impacto mínimo e restrições, e destacando os benefícios da análise de tráfego em rede para o gerenciamento e a segurança da rede.

Palavras-chave: DWT, P4, Deteção de periodicidades, Haar Wavelets, Programação no Plano de Dados.

LIST OF ABBREVIATIONS AND ACRONYMS

AS	Autonomous System
AVM	Aggregated Variance Method
BGP	Border Gateway Protocol
COTS	Commercial Off-The-Shelf
DFT	Discrete Fourier Transform
DWT	Discrete Wavelet Transform
DPDK	Data Plane Development Kit
HM	Higuchi Method
HTTP	Hypertext Transfer Protocol
ICS	Industrial Control Systems
IoT	Internet of Things
P4	Programming Protocol-Independent Packet Processors
PCA	Principal Component Analysis
PBM	Periodogram-based Method
TCP	Transport Control Protocol
WBM	Wavelet-Based Method

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

The recent proliferation of Internet-connected devices (*e.g.,* notebooks, telephones, and Internet of Things (IoT) Devices), systems, services, and dramatic changes in Internet usage [Cisco 2020] are among the main reasons for the continued exponential growth in Internet traffic. These network environments generate an enormous volume of data when interacting within the network. The information generated describes the interaction between them in the Internet traffic exhibiting the behavior of their activities, which can result from benign or malign activities [Ahmed, Naser Mahmood e Hu 2016, Moustafa, Hu e Slay 2019]. Thus, network operators need to respond timely as possible to many issues that can arise and perform tasks such as detecting or distinguishing nefarious network activities from benign behavior [Ding, Savi e Siracusa 2020]. To achieve that, they must collect and analyze vast network measurement data. These large volumes of data require efficient traffic measurements to monitor the network behavior. Also, analyzing these measurements can be daunting for the network operator because storing and processing such data involves considerable work and requires computational resources not always at hand.

The analysis of measurement data may impose timing constraints (*e.g.,* non-real-time vs. real-time), determining the type of methods to apply at the operators' disposal, such as traditional statistical analysis techniques [Fu et al. 2021, Ji, Choi e Jeong 2015, Callegari et al. 2011, Mai, Yuan e Chuah 2008], information theory-based approaches [Ding, Savi e Siracusa 2020, González et al. 2021], and machine learning algorithms [Fu et al. 2021]. These methods can be further separated into time-domain [Popa e Manea 2015], frequency-domain [Fu et al. 2021, Luo, Li e Zhang 2019], and wavelet-domain techniques, and some of them (*e.g.,* mean, variance, standard deviation [Gao, Handley e Vissicchio 2021], and Haar Discrete Wavelet [Bartlett, Heidemann e Papadopoulos 2011]) can be adapted for streaming data analysis. Analyzing streaming data allows operators to consider different features (*e.g.,* packet and byte size counts) that are required for inferring certain network activities, computing them in high throughput scenarios [Mai, Yuan e Chuah 2008] and at line rate [Fu et al. 2021] without having to store the data under analysis.

One possible approach to analyze the network behavior is signal processing, which can be used for analyzing streaming data to infer regular activities within network traffic. Such activities often indicate recurring patterns in network usage and can be malicious or

benign. Inferring periodic activities of unknown origins will typically trigger a detailed post-mortem and offline forensic analysis by the network operator [Franco et al. 2021] to identify the observed periodic activities' root cause(s). Examples of such efforts include detecting anomaly behavior [Du et al. 2018, Ji, Choi e Jeong 2015], reconstructing the signal of a network communication [Jiang et al. 2020], and analyzing the energy spectrum [Yue et al. 2018]. However, traditional signal processing techniques such as the Discrete Fourier Transform (DFT) are known to have a high computational overhead that prevents them from being used for real-time periodicity detection in high throughput scenarios [Fu et al. 2021, Du et al. 2018]. Therefore, their practical use in this context is limited to performing post-mortem analysis tasks [Luo, Li e Zhang 2019, Jiang et al. 2020].

In contrast, the Discrete Wavelet Transform (DWT) method can be used to analyze time-series data with low computational overhead by leveraging their intrinsic ability for time-frequency localization, *i.e.,* dividing the data into different frequency components and used in prior works [Ji, Choi e Jeong 2015, Callegari et al. 2011, Bartlett, Heidemann e Papadopoulos 2011, Mai, Yuan e Chuah 2008] to analyze networking traffic data, the DWT method is a promising technique for analyzing streaming data where the "signal" is given in the form of packet-level network traces. Not only does it allows for the simultaneous analysis of the signal at different scales, but the method can be naturally parallelized and performed at line rate to enable real-time signal analysis [Roughan, Veitch e Abry 2000].

Despite its low computational overhead, implementing the DWT method to process high-volume traffic streams at a line rate poses significant challenges. Using Commercial Off-The-Shelf (COTS) hardware to perform the necessary time-frequency localization of the incoming traffic is, in general, inefficient as it can introduce additional overhead that offsets the benefits of using the DWT. In turn, recent advances in programmable data plane technologies (*e.g.,* Programming Protocol-Independent Packet Processors (P4) [Budiu e Dodd 2017, Consortium 2021] and Data Plane Development Kit (DPDK) [Intel e Foundation]) present unique opportunities to use techniques such as the DWT for line rate traffic processing in the data plane. However, P4's limited support for commonly used arithmetic operations (*e.g.,* addition, subtraction, and multiplication [Consortium 2021]) makes it challenging to implement the DWT method in P4 and run it on actual hardware. Therefore, there is a research gap in implementing DWT in P4 for detecting network traffic periodicities within the data plane.

This dissertation presents a P4 implementation of the DWT method to directly perform a signal's time-frequency localization in the data plane. To achieve that, we compute the energy function to analyze each decomposition level of the DWT and use a threshold-based heuristic to automatically alert the network operator of identified periodic behavior. To circumvent the limitations imposed by existing P4-enabled devices about floating point values, we rely on several mathematical modifications (*i.e.,* division, division with a floating point value ($\sqrt{2}$) and power-of-two algorithms) that reduce the need for complex arithmetic operations. Also, the solution was proposed to have a small memory footprint in the data plane, which results in only minimal throughput overhead (less than 1% for average-sized packets). Finally, several performance and security experiments were conducted considering selected flows (*e.g.,* based on the number of data packets and recurrent behavior) from different datasets. These experiments show how the proposed solution correctly identifies periodic behavior in synthetic and real-world packet traffic traces. This work has been published in IEEE GLOBECOM 2022 [Huaytalla et al. 2022].

The remainder of this dissertation is organized as follows. In Chapter 2, we introduce a review of Wavelet Transform concepts with an emphasis on DWT method and how it works. We also present an overview of related work on programmable data planes and signal processing techniques for network traffic analysis. Chapter 3 introduces the overall periodicity detection methodology, including the DWT method's signal decomposition and the reasoning behind the proposed DWT-based Energy function analysis for detecting periodic activities. In Chapter 4, we report on our experimental evaluations of the proposed approach, and we conclude, in Chapter 5, the dissertation by summarizing our main contributions and by discussing future work.

2 BACKGROUND AND RELATED WORK

In this chapter, we provide a background on Discrete Wavelet Transform (DWT), the general idea behind DWT, its properties, and how they can be useful for network traffic analysis, as well as the Haar Wavelets filters used to decompose the input signal. Then, we describe how the coefficients produced by the DWT can be used to identify periodic behavior in signals using the energy function. Also in this chapter, we review and discuss the different works in the literature to show the current efforts on the analysis of recurrent patterns in network traffic traces, applications of signal processing techniques in networking, and previous attempts at statistical analysis performed in programmable switches. The identified challenges and research gaps are further discussed as well.

2.1 Discrete Wavelet Transform

The Discrete Wavelet Transform (DWT) method uses waveforms to “localize" a signal in both frequency and time. In addition to pinpointing the specific times when each frequency occurs in the signal [Roughan, Veitch e Abry 1998], the DWT has lower computational complexity than the more traditional Fourier transform method—$O(n)$ vs. $O(n \log n)$ [Roughan, Veitch e Abry 1998, Fu et al. 2021], where n is the number of samples in the signal—making it more appealing for analyzing signals in high-throughput settings such as network traffic.

To decompose a signal, the DWT uses a low-pass filter (*a.k.a.*, *scaling function* or *father wavelet*) and a high-pass filter (*a.k.a.*, *wavelet function* or *mother wavelet*). These filters are convolved with k data points at a time (depending on the size of the filters), encoding high- and low-frequency information into two distinct levels of decomposition and effectively sub-sampling the original signal by half. The encoded data points generated by the high-pass and low-pass filters are referred to as the *detail* and *approximation* coefficients, respectively. We can apply the DWT decomposition recursively m times using the approximation coefficients at level $j-1$ as input to level j ($1 \leq j \leq m$) to analyze frequencies at a finer granularity. The original signal corresponds to level zero.

The original DWT method was proposed alongside a simple set of wavelet filters known as the Haar wavelet [Bartlett, Heidemann e Papadopoulos 2011]. Here, we also use the Haar wavelet and leave the application of other wavelets filters such as the Daubechies wavelets [Ji, Choi e Jeong 2015] for future work. We define the Haar wavelet

low-pass and high-pass filters as $(1/\sqrt{2}, 1/\sqrt{2}$ and $(1/\sqrt{2}, -1/\sqrt{2})$, respectively [Huang, Feldmann e Willinger 2001] and consider a time series $X_{0,k}, k = 0, 1, 2, \ldots$ representing the input signal. For scale one of the DWT decomposition, we multiply these values with consecutive samples in the input signal and then add the resulting products to compute the approximation and detail coefficients, respectively. More generically, we describe the approximation and detail coefficients for scale j, $j \geq 1$, at position k by Equations 2.1 and 2.2, respectively.

$$X_{j,k} = \frac{1}{\sqrt{2}}(X_{j-1,2k} + X_{j-1,2k+1}) \tag{2.1}$$

$$d_{j,k} = \frac{1}{\sqrt{2}}(X_{j-1,2k} - X_{j-1,2k+1}) \tag{2.2}$$

Most of the existing methods for analyzing recurring patterns in networking require storing a high volume of network traffic data and mining it post-mortem. Due to the high computational complexity of these methods [Roughan, Veitch e Abry 1998], this type of analysis can take a long time. However, mining traffic in today's high-speed networks to detect patterns in communication requires analyzing measurements at line rates with methods that have low computational complexity. Compared to most existing approaches, the DWT method with its low computational complexity and broad applicability to different problems is especially well suited for the high-throughput conditions and real-time requirements imposed by modern-day networks.

In particular, the energy function of the detail coefficients has been used to detect periodic signals in different scenarios (*e.g.,* for studying network congestion and its impact on TCP re- transmissions [Huang, Feldmann e Willinger 2001]). The energy function E_j is defined as

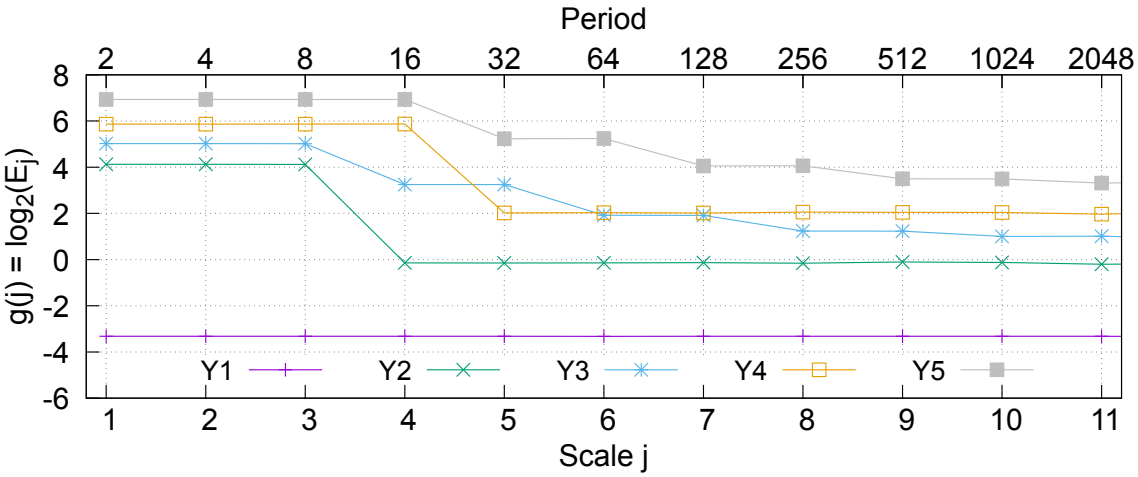
$$E_j = \frac{1}{N_j} \sum_k |d_{j,k}|^2, \quad j = 1, 2, \ldots, m \tag{2.3}$$

where j is the decomposition level, and N_j is the number of coefficients at level j. Computing the energy of the detail coefficients at each decomposition level allows us to examine the temporal properties in the signal from high to low frequencies as the level of the wavelet decomposition increases.

Feldmann *et al.* [Feldmann et al. 1999] show that plotting the function $g(j) = \log_2(E_j)$ can be used as a means to detect periodicity in a signal. As each decomposition

level filters out a specific frequency range in the original signal, a particular periodicity manifests as a sudden decrease in $g(j)$. Figure 2.1 shows an example with five different signals. Y1 is white noise with no periodicity. Y2–Y5 are mixtures of white noise with a periodic signal of period 8, 10, 16, and 20, respectively. The figure shows that while $g(j)$ decreases for Y2–Y5 near the point marking the signal's period, $g(j)$ remains flat for Y1.

Figure 2.1: Using the energy function to detect periodicity.



2.2 Periodicity Detection in Network Traces

This section presents works concerned with detecting periodic patterns in networking with several applications, such as *(i)* analysis of network traffic measurement, *(ii)* analysis of Netflow data, *(iii)* study of a specific protocol such as HTTP, and *(iv)* analysis of industrial network traffic. These applications are evaluated with several methods (*e.g.,* theory information-based [Akgül et al. 2011], machine learning [Akgül et al. 2011, Bilge et al. 2012, Eslahi et al. 2015, Barbosa, Sadre e Pras 2016]) to detect recurrent behavior using real-world network traces.

To analyze network traffic, Akgül *et al.* [Akgül et al. 2011] evaluates the flow measurements to detect periodic patterns using the Hurst parameter computed by estimation methods in different domains (*e.g.,* time, frequency, wavelet, and Eigen). Also, they conducted analytical research on theoretical concepts of the estimator methods (*e.g.,* Higuchi method (HM), Aggregated Variance Method (AVM), Periodogram-based Method (PBM), Wavelet-based Method (WBM) and Principal Component Analysis (PCA-based Method)) for studying the periodicity effects in the traffic flow. To prove these effects,

they employ synthetic uses cases and two use cases in real-world scenarios using the data from two different networks (gathered from the University of North Carolina and the Traffic Measurement and Analysis on the WIDE Internet - MAWI) where are performed in post-mortem analysis. In their results, WBM and PCA-based methods present better visualization in their regression plots for detecting periodic components than the time domain-based estimator methods.

Similarly, Bilge *et al.* [Bilge et al. 2012] proposes a botnet detection system in large-scale and wide-area using Netflow records. To achieve this detection, they apply supervised machine learning to train a subset based on the features of the aggregating data obtained from the Netflow data in half-duplex connections. Such data generates flow size-based, client access patterns-based, and temporal behavior-based features used in the trained model to identify the C&C servers. Further, they introduce an external reputation score (*i.e.,* based on Fire, Exposure, and Google Safe Browsing) to reduce false positives. The model is created using a Random Forest classifier with better results when combining all the features as input. Their proposal can perform in real-time since the feature extraction performance requires half of the time it takes to collect the data, which mean that not perform at the data rate.

Other works put their effort into studying a specific protocol. Eslahi *et al.* [Eslahi et al. 2015] explores the HTTP communication protocol between bots with their bot master in a Command and Control - C&C botnet scenario. This malicious communication takes advantage of the HTTP protocol due to the network activities generated by client and server communication. It is wide-used by end-users (*e.g.,* web browsers, web applications). They mainly strive to study the PULL style pattern of HTTP Botnet communication, which means that the bot requests the C&C for updates and brings up new commands if required [Gu, Zhang e Lee 2008]. This communication style naturally presents a recurrent behavior, where the authors perform a decision tree with a j48 classifier using three measurements (*i.e.,* Periodic Factor, Range of Absolute Frequencies, and Time Sequence Factor). The classification results in five categories: *(i)* Non-Periodic, *(ii)* Weak Periodic, *(iii)* Periodic, *(iv)* Uniform Periodic, and *(v)* Strong Periodic, where able to detect the periodic behavior and classify into four categories (*i.e., (ii) -(v)*). Evaluating offline the different datasets, they achieve to label the periodicity presence in many botnets correctly.

In an environment like Industrial Control Systems (ICS), Barbosa *et al.* [Barbosa, Sadre e Pras 2016] analyzes the network traffic periodicity produced by the commands

executed by the devices. Such an environment regularly presents recurrent activities in communication among its devices. They propose an ICS-learning approach to analyze the periodic behavior of the communication produced by the commands executed by the devices. Using the frequency generated, they detect the presence of periodicity in the commands and build a command allow-list for protecting the system against intrusion. To achieve that, the authors filter the packets into different flows for cover-up in a tokenization process, and the learner algorithm module uses this information to identify the different cycles existing in each flow. This approach has an exponential time complexity for testing all combinations to detect the presence of cycles (*i.e.,* periodicity) in each flow. The authors perform their tests in real-world industrial traffic, and the learner module has a good performance despite their exponential time complexity because they can process 30 min traces in 100 ms. Although the outstanding performance, the analysis did not manage to evaluate at line rate.

2.3 Signal Processing in Networking

This section describes works related to signal processing techniques, such as *(i)* DWT method [Mai, Yuan e Chuah 2008, Callegari et al. 2011, Ji, Choi e Jeong 2015], and *(ii)* DFT method [Fu et al. 2021] applying in networking. These works put their efforts into detecting anomalies in the network traffic where signal processing techniques are part of their process to highlight the change produced in the signal behavior, even in some cases makes to be able to detect which kind of attack was produced.

One application of signal processing techniques is applied in Autonomous Systems (AS). Mai *et al.* [Mai, Yuan e Chuah 2008] propose a BAlet framework for analyzing the BGP update messages behavior to detect anomalies. In their analysis process, they construct a matrix with the counting of BGP updates messages where each vector represents an AS origin in a time series (*i.e.,* the signal), then each vector pass through a DWT method to accentuate the possible existing abnormalities. If not exists abrupt changes in the vector, the signal is discarded. Using a clustering algorithm, they detect network-wide anomalies (*i.e.,* anomalies produced in different AS origins) for generating an alarm to the network operator. Also, they achieve to detect attacks caused by a massive amount of packets in the current Route Views Dataset (*i.e.,* slammer attacks). Even so, the authors perform a post-mortem analysis to validate their framework using the Réseaux

IP Européens Network Coordination Center (RIPE NCC) dataset due to the difficulty of testing on arbitrary BGP routing.

Another work that uses signal processing techniques is proposed in the study of Callegary *et al.* . which aim to detect anomalies in the network using Information Theory techniques (e.g., Sketches and Entropy distribution) and Wavelet Analysis (e.g., DWT). In their approach, they parser the input data originating from Netflow records, then the output is formatted into a sketches tables based on IP address, their number of bytes generated in each time bin, and the distinct hash functions. After this process, the entropy of the distribution aggregates the information by distinct time bin in a time series, since the data storage produced by the sketches tables generates huge volume of data to process and require high computational resources. The time series is analyzed by blocks for the DWT method for seeing discontinuities in the network traffic, and the euclidean distance is calculated on each analyzed block with the reference coefficients (*i.e.,* computed in the first block) for detecting anomalies if the distance surpass the threshold. The proposed work achieve to detect anomalies in the Internet2 Network Dataset adding synthetic anomalies in the data, but the process is not concerned in periodicity detection and not willing to perform at line-rate.

Ji *et al.* [Ji, Choi e Jeong 2015] analyze the network traffic to classify anomalous behavior, discerning between normal behavior depicted by three services (*i.e.,* FTP, Mail, P2P) where each service is considered as a class, and the abnormal behavior is represented by a unique class called attack (*i.e.,* virus and worm attack). They set up their datasets based on their classes and balanced the number of records to have an equilibrium between normal and abnormal behavior data. For designing their predictive model, they pre-processed the dataset with a data normalization process. The authors use a DWT method to highlight the changes in the data records to extract the relevant features for the model. To choose the most suitable features, the authors apply statistical analysis (*i.e.,* analysis of variance - ANOVA), and the selected features are used for training the predictive model using Logistic Regression (LR). Their evaluations present better results to generate a model for classifying the anomalies using the DWT method in their features than raw values. Besides, the efforts of the authors are not included in detecting periodicities; their focus is on detecting anomalies.

2.4 Statistical Analysis in Programmable Data Planes

Recent research has witnessed a growing interest in incorporating statistical analysis directly within the data plane. For instance, Lapolli *et al.* [Lapolli et al. 2019] propose a novel approach by implementing entropy estimation in P4 to detect distributed denial-of-service (DDoS) attacks. Their work demonstrates the effectiveness of integrating statistical analysis techniques within the data plane, enabling real-time detection of DDoS attacks. By leveraging P4's programmability, their implementation of entropy provides valuable insights into network behavior and enhances network security.

In a similar vein, Ding, Savi, and Siracusa [Ding, Savi e Siracusa 2020] present methods for estimating logarithmic and exponential functions in P4 to track network traffic entropy. Their approach acknowledges the importance of monitoring and understanding network traffic entropy as a key metric for identifying anomalies and potential security threats. By integrating logarithmic and exponential functions into P4, they approximate entropy values efficiently and accurately. Their work contributes to effective monitoring and detection of network anomalies, facilitating insights into network behavior and enhancing network security.

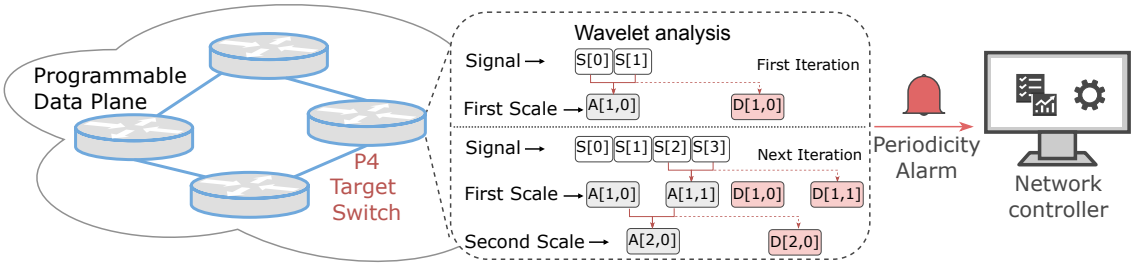
Furthermore, Gao, Handley, and Vissicchio [Gao, Handley e Vissicchio 2021] make significant contributions by implementing a library of statistical techniques directly in P4. Their work encompasses a range of statistical techniques, including mean, variance, and other measures, providing a comprehensive suite of tools for statistical analysis within the data plane. By gathering these techniques in a library, they offer a reusable resource that enables researchers and practitioners to leverage statistical analysis capabilities in P4-based networks. Their work showcases the potential of integrating statistical techniques into the data plane, promoting improved network monitoring, anomaly detection, and analysis of network behavior.

Building upon these works, our research further contributes to the field by extending the existing library of statistical techniques implemented in P4 [Huaytalla et al. 2022]. Specifically, we introduce a P4 implementation of the Discrete Wavelet Transform (DWT) and a data plane-based periodicity detection technique. These additions enhance the capabilities of the existing library, enabling more comprehensive statistical analysis within the data plane. By incorporating the DWT and periodicity detection, our work expands the range of statistical tools available in P4-based networks, facilitating improved monitoring, anomaly detection, and network behavior analysis.

3 ONLINE DWT DECOMPOSITION

We present a P4 implementation of the DWT method to perform a signal's time-frequency localization directly in the data plane [Huaytalla et al. 2022]. Figure 3.1 provides an overview of our solution, in which a P4-enabled programmable device is running an efficient algorithm that we designed to perform the DWT decomposition. We compute the energy function to analyze each decomposition level of the DWT, and use a threshold-based heuristic to automatically alert the network operator of identified periodic behavior. To circumvent the limitations imposed by existing P4-enabled devices, we rely on a number of mathematical modifications that reduce the need for complex arithmetic operations.

Figure 3.1: Wavelets analysis for periodicity detection in P4.



We consider a signal where each sample is indexed and represents the number of packets of a flow in a fixed time interval. A flow refers to all the packets that match a rule specified by the network operator, *e.g.,* all packets with a given destination port or IP address. Implementing the DWT transform for such signals in P4 presents significant challenges. First, P4 does not allow loops and does not support all the arithmetic operations required to evaluate Equations 2.1, 2.2 and 2.3. Second, processing in P4 is asynchronous, meaning that we need a packet arrival to trigger a computation. Finally, computing and storing the signal and the approximation coefficients at different levels may impose significant processing and storage overheads.

3.1 Arithmetic Operations

At first sight, Equation 2.3 presents additional hurdles for its implementation in P4, including the division by a number N_j that varies over time for each level of the decomposition and floating-point operations (division by $\sqrt{2}$). However, by expanding the equation to different levels, we can simplify it to the point where these operations are

no longer needed. First, we assume without loss of generality that the length of the signal is a power of two, *i.e.,* $N = 2^n$ for some n, so $N_j = N/2^j$, where j represents the level of the decomposition or the signal when $j = 0$. Then, we can write Equation 2.3 as

$$E_j = \frac{1}{N_j} \sum_k \left| \frac{X_{j-1,2k} - X_{j-1,2k+1}}{\sqrt{2}} \right|^2 \tag{3.1}$$

$$NE_j = 2^{j-1} \sum_k |X_{j-1,2k} - X_{j-1,2k+1}|^2 \tag{3.2}$$

For conciseness, let $A_{j,w} = X_{j-1,2w} + X_{j-1,2w+1}$, where X_j is the signal for $j = 0$ or the approximation coefficients for $j \geq 1$ as defined in Section 2. $A_{j,w}$ is a parameterized version of the expression inside the parenthesis of Equation 2.1, *i.e.,* $A_{j,w}$ is the approximation coefficient without the $\sqrt{2}$ normalization factor. For $j \geq 2$, we can expand Equation 3.1, using the definition of $A_{j,w}$ and Equation 2.1 as

$$NE_j = 2^{j-1} \sum_k \left| \frac{A_{j-1,2k}}{\sqrt{2}} - \frac{A_{j-1,2k+1}}{\sqrt{2}} \right|^2 \tag{3.3}$$

$$NE_j = 2^{j-2} \sum_k |A_{j-1,2k} - A_{j-1,2k+1}|^2 \tag{3.4}$$

Note that, since $A_{j,w}$ references the approximation coefficients $X_{j-1,2w}$ from the previous level $j-1$, we have $A_{j-1,2k} = X_{j-2,4k} + X_{j-2,4k+1}$ and $A_{j-1,2k+1} = X_{j-2,4k+2} + X_{j-2,4k+3}$. With these simplifications, we can compute Equation 3.4 using only additions, subtractions, and multiplications. Moreover, we can efficiently implement multiplication of a number by itself or by a power of two using only shift and addition operations. Finally, since we are only interested in finding points where $g(j)$ decreases, we do not need to divide the right-hand side of Equation 3.4 by N.

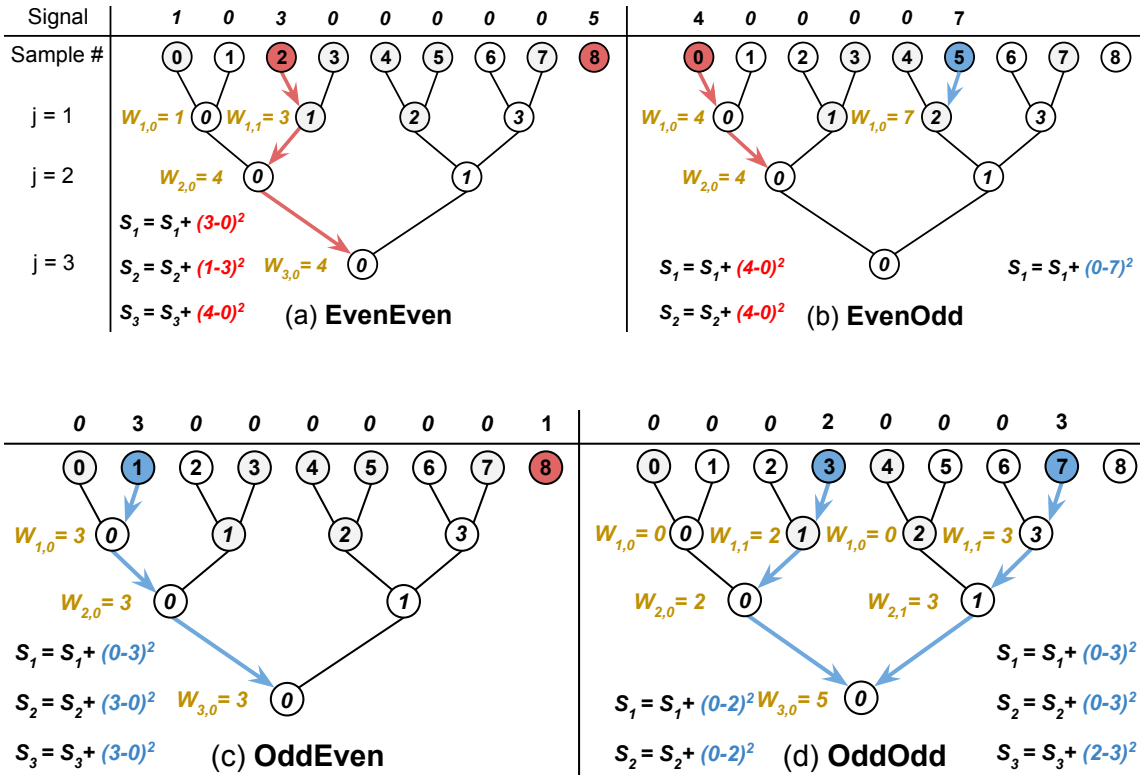
3.2 Asynchronous Computation and Resource Usage

To build a signal on a P4 switch, we need to execute an action to count the number of packets in a sampling interval. However, the switch runs an action only when a packet arrives and matches an operator-defined rule. If no packet arrives during long periods, we have a signal with multiple consecutive samples equal to zero. Since P4 does not allow loops, we cannot simply iterate over these samples to compute the approximation

and detail coefficients of the DWT transform and propagate them to calculate the other decomposition levels. Also, storing the signal's samples and the approximation and detail coefficients for each decomposition level would consume a large amount of memory in the switch and limit the length of the signal and the number of flows we could analyze.

We leverage the simplification of the energy function we developed in Section 3.1 to develop an efficient algorithm that stores for each signal only a sliding window W_j with two approximation coefficients and the cumulative sum $S_j = \sum_k |A_{j-1,2k} - A_{j-1,2k+1}|^2$ for each level j. We also store the index (L) of the sample that was last processed by the switch, totaling $3 * \text{MAX\_LEVEL} + 1$ integers per flow, where MAX_LEVEL is the maximum number of decomposition levels. To compute the k-th approximation coefficient at level j, we need the coefficients (or signal samples if $j = 1$) with indices $2k$ and $2k + 1$ at level $j - 1$ (See Equations 2.1 and 3.4). We can map this dependence of the approximation coefficients to a tree structure where signal samples are the leaves, and the coefficients are the internal nodes of a binary tree, as illustrated in Figure 3.2.

Figure 3.2: Examples of signal decompositions and computations of the energy function.



Note that to calculate the coefficient of an internal node, we always need the values of its two children. Since our algorithm performs online computations, some samples or coefficients might not be available when trying to compute the approximation coefficient and the cumulative sum of the energy function at the next level at a specific time. Also, to deal with the case where the switch does not receive any packet for a flow during several sampling intervals, we introduce samples with a value set to zero. Therefore, the arrival of a packet at the switch may trigger the computation of the coefficients at several levels as a new sample may complete an entire subtree. For example, the availability of sample 7 in Figure 3.2d allows the computation of the approximation coefficients 3 at level 1, 1 at level 2, and 0 at level 3.

Algorithm 1 shows the pseudo-code to decompose a signal using the DWT transform, compute the energy function for each decomposition level, and generate an alarm when it detects a periodicity in a signal. To simplify the explanation, we present the algorithm with recursive functions to traverse the decomposition tree. Although P4 does not support recursion, we can implement the algorithm by simply expanding the recursive functions MAX_LEVEL times. We evaluate the maximum number of levels we can support on a programmable NIC in Section 4. Line 1 shows the main function of the algorithm that is invoked every time the switch receives a packet that marks the end of a sampling interval. Figure 3.2a illustrates a signal with nine samples indexed from 0 to 8 that triggers the call to function `OnlineDWT` with $L = 2$, $R = 8$, and $s = 4$. We use L and R and the fact that all samples between them are zero to define four different cases depending on whether L and R are even or odd: EvenEven, EvenOdd, OddEven, and OddOdd (Figures 3.2a-d).

Algorithm 1 Online DWT Decomposition and Energy Function.

1: **procedure** ONLINEDWT(
   L: Index of the previously measured sample,
   R: Index of the currently measured sample,
   s: Number of packets in the current sample)
2: **if** ISEVEN(L) **and** ISEVEN(R) **then** PROPAGATEL($L, R, 1$)
3: **else if** ISEVEN(L) **and** ISODD(R) **then** PROPAGATELR($L, R, 1, s$)
4: **else if** ISODD(L) **and** ISEVEN(R) **then** PROPAGATEL($L + 1, R, 1$)
5: **else if** ISODD(L) **and** ISODD(R) **then**
6: PROPAGATELR($L + 1$, R, 1, s)
7: $W_{0,\text{ISEVEN}(R)\ ?\ 0\ :\ 1} \leftarrow s$
8: $L \leftarrow R$
9: **procedure** PROPAGATEL(L, R, j)
10: **if** $R - L < 2$ **then**
11: **if** $R - L = 1$ **and** ISODD(L) **then**
12: BRANCHODD(L, j)
13: **else**
14: DECOMPOSE($\frac{L}{2}, j$)
15: **if** $j <$ MAX_LEVEL **then** PROPAGATEL($\frac{L}{2}, \frac{R}{2}, j + 1$)
16: **procedure** PROPAGATELR(L, R, j, s)
17: **if** $R - L = 1$ **then**
18: **if** $j = 1$ **then** $W_{0,1} \leftarrow s$
19: DECOMPOSE($\frac{L}{2}, j + 1$)
20: **if** ISODD(L) **then** BRANCHODD($\frac{L}{2}, j + 1$)
21: **else**
22: BRANCHEVEN(L, R, j)
23: $W_{0,1} \leftarrow s$
24: BRANCHODD(R, j)
25: **procedure** BRANCHODD($index$, j)
26: DECOMPOSE($\frac{index}{2}, j$)
27: **if** $j <$ MAX_LEVEL **and** ISODD($\frac{index}{2}$) **then**
28: BRANCHODD($\frac{index}{2}, j + 1$)
29: **procedure** BRANCHEVEN(L, R, j)
30: **if** $R - L < 2$ **then**
31: **if** ISODD(L) **then** BRANCHODD(L, j)
32: **else**
33: DECOMPOSE($\frac{L}{2}, j$)
34: **if** $j <$ MAX_LEVEL **then** BRANCHEVEN($\frac{L}{2}, \frac{R}{2}, j + 1$)
35: **procedure** DECOMPOSE($index$, j)
36: $W_{j,\text{ISEVEN}(index)\ ?\ 0\ :\ 1} \leftarrow W_{j-1,0} + W_{j-1,1}$
37: $S_j \leftarrow S_j + (W_{j-1,0} - W_{j-1,1})^2$
38: **if** $\alpha(S_{j-1} \ll (j-3)) - \beta(S_j \ll (j-2)) > 0$ **then**
39: GENERATEALARM($j - 1$)

We only describe the EvenEven case here, but note that similar arguments apply when considering the remaining cases as detailed by Algorithm 1. We use Figure 3.2a to illustrate how the algorithm works with an example where $L = 2$ and $R = 8$. Function `OnlineDWT` calls `PropagateL` (Line 9) to propagate sample L (saved in $W_{0,0}$) to the next levels because its sibling (*e.g.,* sample with index 3 in Figure 3.2a) for computing the approximation coefficient was not available in the previous invocation of `OnlineDWT`. As R is even and its sibling is not available yet, the algorithm does not propagate sample R to the next levels and just updates $W_{0,0}$ with the current sample (Line 7) and L with R (Line 8) for a future invocation of `OnlineDWT`. The `PropagateL` function recurses with L and R divided by two (Line 15) to map them to the proper nodes of the decomposition tree at the next level. This recursion stops when j reaches MAX_LEVEL (Line 15) or when the difference between L and R becomes less than two (Line 10). In this last case, the propagation has reached a level where either L is equal to R, and there is nothing more to propagate, or we call `BranchOdd` if L is odd at that level of the decomposition. When $index$ at `BranchOdd` is odd at a level, it means we have a sample that completes a pair of children for computing an approximation coefficient, so the `BranchOdd` function recurses while $index$ is odd in the subsequent levels, computing the approximation coefficients until j reaches MAX_LEVEL or the function reaches a node with $index$ even that does not have its sibling yet to compute the next approximation coefficient.

To detect periodicity in a signal, we use a heuristic that divides the energy at level $j-1$ by the energy at level j and checks if the result is greater than a threshold. If the energy decreases suddenly from level $j-1$ to level j, then E_{j-1}/E_j will be greater than one. More specifically, we check if $\frac{E_{j-1}}{E_j} > \frac{\alpha}{\beta}$, where α and β are integers and $\alpha > \beta$. Using Equation 3.4 and the definition of S_j, we rewrite this formula as $\frac{2^{j-3}S_{j-1}}{2^{j-2}S_j} > \frac{\alpha}{\beta}$. Substituting the multiplication by powers of two with shift operations, we can rewrite the equation as $\beta(S_{j-1} \ll (j-3)) - \alpha(S_j \ll (j-2)) > 0$, which is the same as Line 38 of Algorithm 1. In Section 4, we determine α and β empirically for a set of use cases evaluated in this dissertation.

4 EVALUATION

To evaluate the performance and applicability of our approach, we first implement our algorithm in P4 with Micro-C and load it into a Netronome NFP-4000 SmartNIC to assess its overhead. Next, we determine the threshold for our periodicity heuristic and use publicly available traces of periodic traffic to analyze the energy function plot. To support reproducibility, the artifacts of our work are available in Github at https://github.com/ComputerNetworks-UFRGS/p4wavelets.

4.1 Performance Evaluation

This section focuses on assessing the performance of the proposed mechanism. It consists of two subsections: setup and results. The setup subsection describes the experimental environment, including employed hardware. The results subsection presents the results of the experiments.

4.1.1 Setup

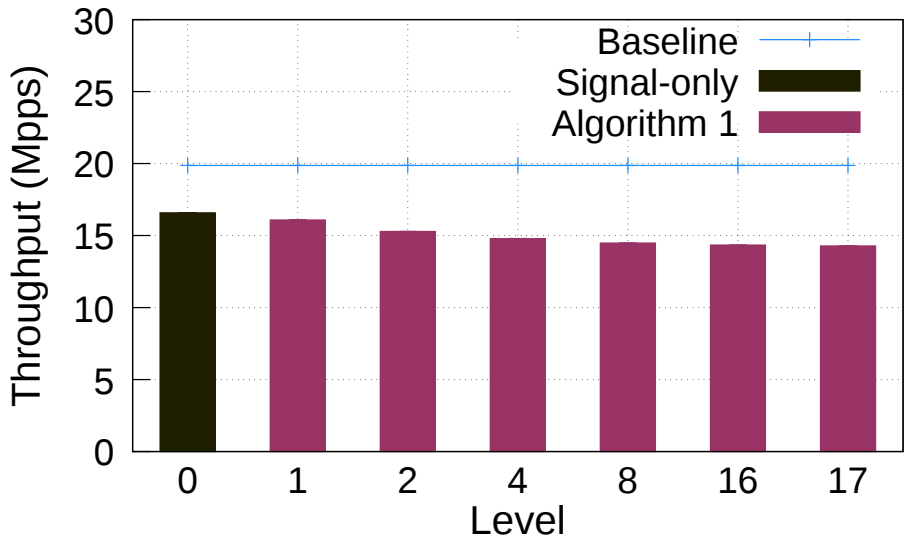
We use two servers with two Intel(R) Xeon(R) Silver 4114, each with 128 GB of RAM and a dual-port Netronome NFP-4000 40 GbE NIC directly connected on a sender-receiver configuration. The SmartNICs have a limit of 8k instructions per flow-processing core, allowing us to use at most 17 levels of decomposition. The receiver's SmartNIC receives packets, processes them according to Algorithm 1, and forwards them back to the sender. The sender sends packets and collects the throughput results. To quantify the throughout overhead of our implementation, we consider different packet sizes (from 64 to 1500 bytes) and different number of decomposition levels (from 1 to 17), and report experimental results for two different sampling intervals (250μs and 1s).

4.1.2 Results

Starting with the extreme case where all packets are of minimal size (*i.e.,* 64 bytes), Figures 4.1 and 4.2 show the SmartNIC throughput (in Mpps) for up to 17 decomposition levels (in red) for the sampling intervals 250μs and 1s, respectively. In both

cases, level 0 (in black) denotes the throughput for basic packet capture without any decomposition (*i.e.,* constructing per-sample signal), and the baseline (in blue) is measured by simply forwarding packets to their destination. The SmartNIC provides 64-bits integer packet timestamps composed of two 32-bits variables, one for seconds and the other for nanoseconds. To compute accurate sampling intervals, we implemented a set of shift and multiplication operations to perform integer divisions using constants. As a result, when using a 250μs sampling interval, the throughput overhead for construction of the signal is roughly 17% when compared to the baseline (see level 0 in Figure 4.1).

Figure 4.1: Performance results varying levels using 64b packets - 250μs

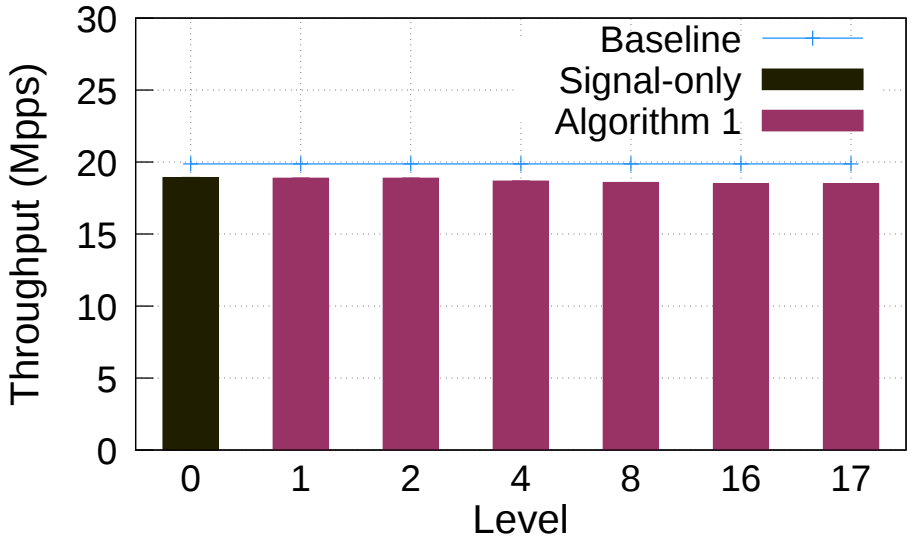


However, this overhead drops to less than 5% when the sampling interval is 1s (see level 0 in Figure 4.2), mainly because in this case, calculating accurate timestamps requires fewer operations.

Figures 4.1 and 4.2 also show that our algorithm manages to compute the DWT using very few resources. In particular, we observe that the performance overhead for the different decomposition levels (red bars) is small – roughly 12% compared to constructing the signal (black bar). This result indicates that accurately constructing the input signal (*i.e.,* placing samples in the correct interval) is responsible for most of the throughput overhead. When the sampling interval is 1s, the overhead is negligible because the SmartNIC provides timestamps already at the required granularity and there are fewer cases when the algorithm has to propagate coefficients to the subsequent levels.

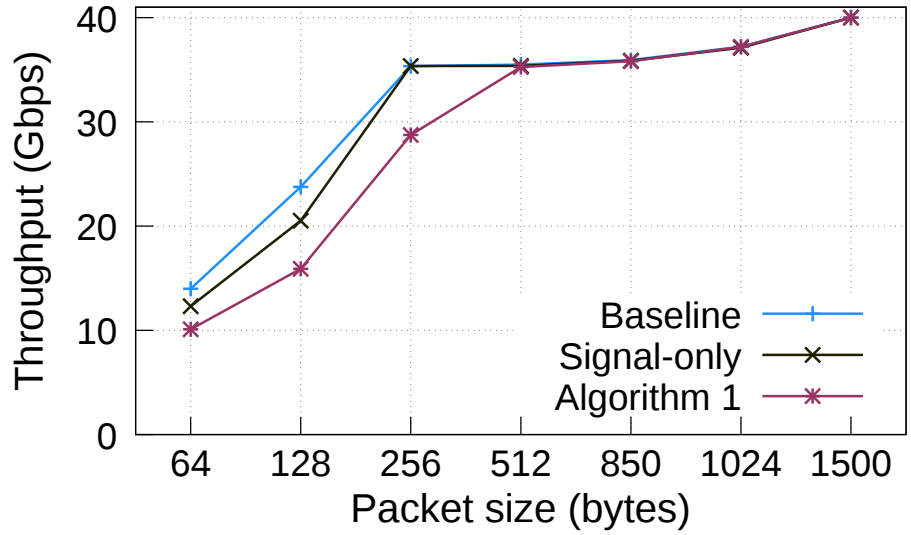
All previous results assume a worst-case scenario where all packets are of minimal size (*i.e.,* 64 bytes). In Figures 4.3 and 4.4, we consider more realistic scenarios and analyze the throughput overhead of our algorithm with 17 levels of decomposition for

Figure 4.2: Performance results varying levels using 64b packets - 1s.



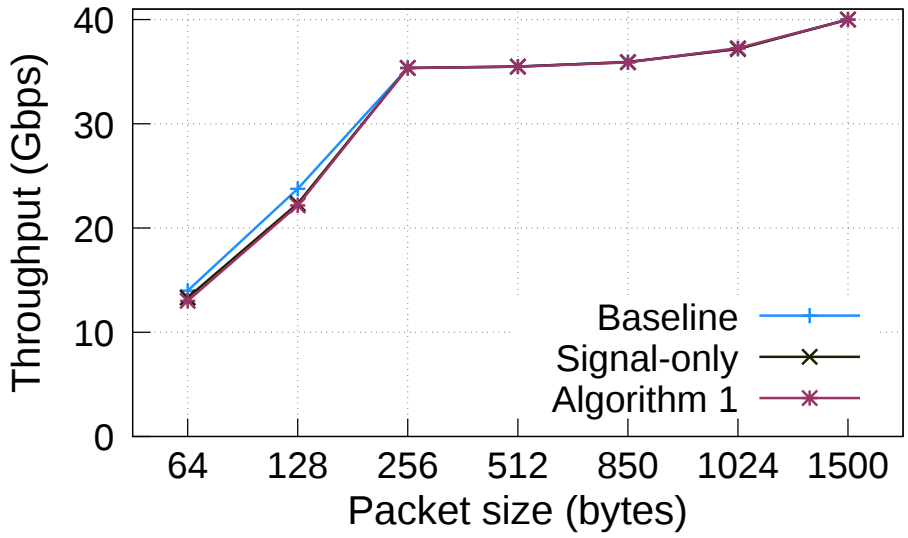
packet sizes that vary from 64 to 1500 bytes. When the sampling interval is 250μs (Figure 4.3), we notice that for packets of size 512 bytes and larger, the throughput overhead of our algorithm becomes negligible.

Figure 4.3: Performance results varying packet size using 17 levels - 250μs.



In fact, assuming averaged-sized packets in the Internet to have 900 bytes [CAIDA 2021], we observe a throughput overhead of less than 1% when compared to the baseline. For a sampling interval of 1s (Figure 4.4), the throughput overhead is practically zero when compared to the baseline behavior of the SmartNIC.

Figure 4.4: Performance results varying packet size using 17 levels - 1s.



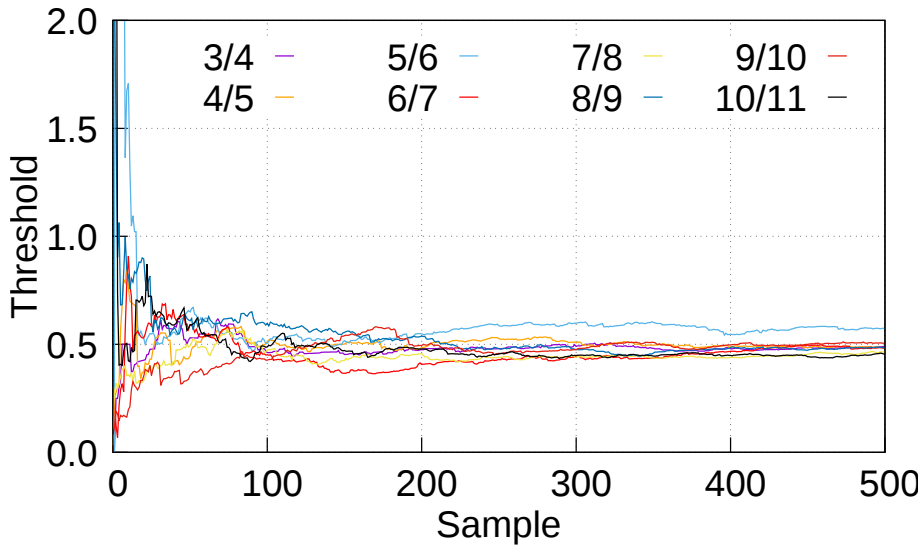
4.2 Applicability Evaluation

This section assesses the practical usability of the proposed method. It consists of two subsections: setup and results. The setup subsection describes the experimental setup and methodology, while the Results subsection presents the findings and analysis of the evaluation. This section provides insights into the applicability of the method to detect peridiocity.

4.2.1 Setup

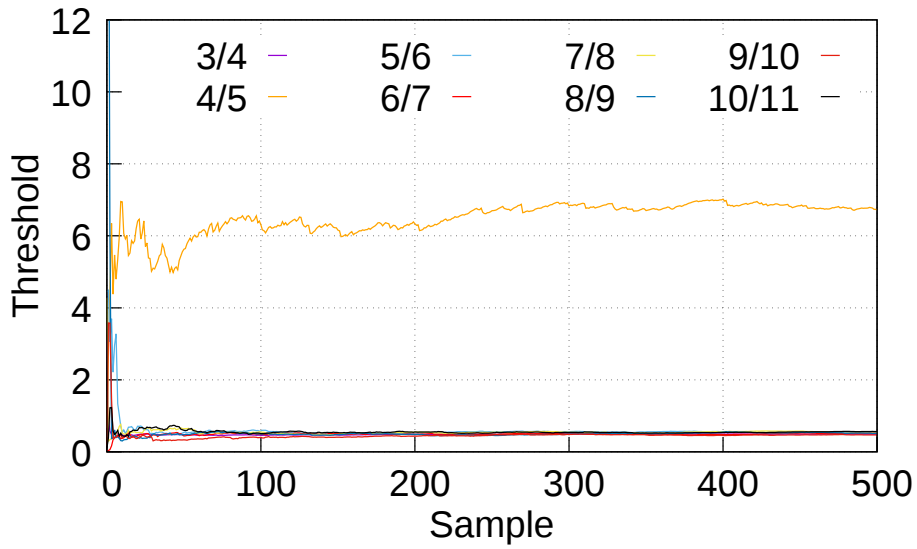
To apply our algorithm in practice, it is first necessary to define an appropriate threshold to be used in Line 38. To this end, we examine the examples considered in Figure 2.1 (see Chapter 2), dividing the energy value at one level by that at the next level. Figure 4.5 shows that for a non-periodic signal such as Y1, after some initial phase, the ratios between successive scales converge to a value of around 0.5.

Figure 4.5: Threshold analysis - Y1.



In contrast, for periodic signals such as Y4 (Figure 4.6), the ratios between levels 4/5 (dip in energy evident in Figure 2.1) converge to a value larger than 6. Based on those observations, we consider a heuristic threshold value of 1.5 ($\alpha = 3$ and $\beta = 2$ in Algorithm 1).

Figure 4.6: Threshold analysis - Y4.



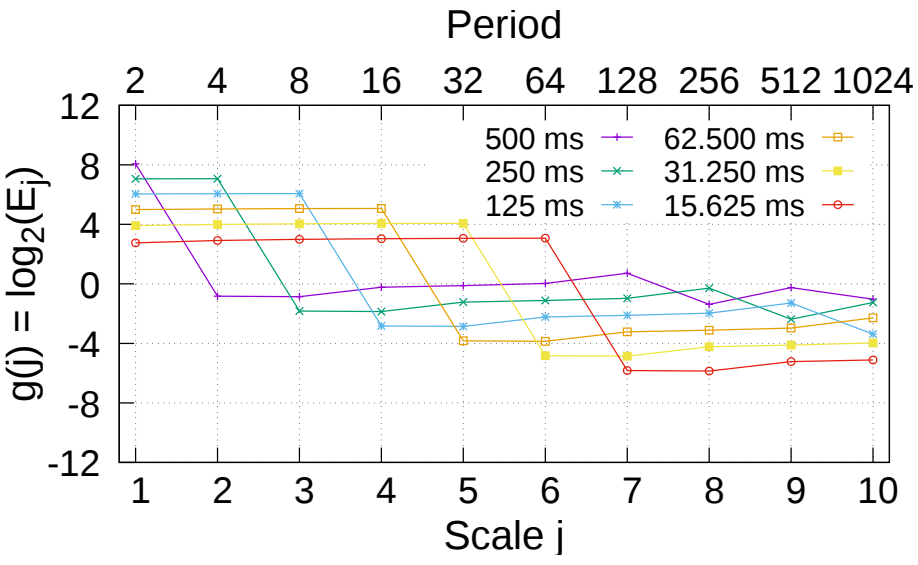
To experiment with our algorithm, we next surveyed the literature for security and performance use cases with intrinsic periodic behavior and summarize the examples we found in Table 4.1. Performance use cases (1-2) capture the RTT of packets in benign traffic over a normal link and a congested link, respectively. The security use cases (3-5) consist of different attacks that generate malicious traffic such as Heartbleed and Cobalt strike.

Table 4.1: Network traffic use cases.

#	Use Cases	Period	Trace Length [minutes]	Sampling Interval
1	Normal link RTT [Feldmann et al. 1999]	≈24ms	14	750μs
2	Congested link RTT [Feldmann et al. 1999]	≈1.3s	33	325ms
3	Heartbleed [Sharafaldin., Habibi Lashkari. e Ghorbani. 2018]	≈1s	20	125ms
4	Cobalt strike [Malware-Traffic-Analysis.net 2021]	≈60s	17	4s
5	Trickbot (a) [Malware-Traffic-Analysis.net 2021], (b) [Malware-Traffic-Analysis.net 2021]	≈300s	291, 161	20s

We analyze each use case's traffic separately. Also, for each use case we rely on a different sampling interval to facilitate presentation. While a larger sampling interval limits the applicability for fine-grained periodicity, it simply shifts the dips in the energy function to smaller levels. We illustrate this behavior for the Heartbleed use case in Figure 4.7.

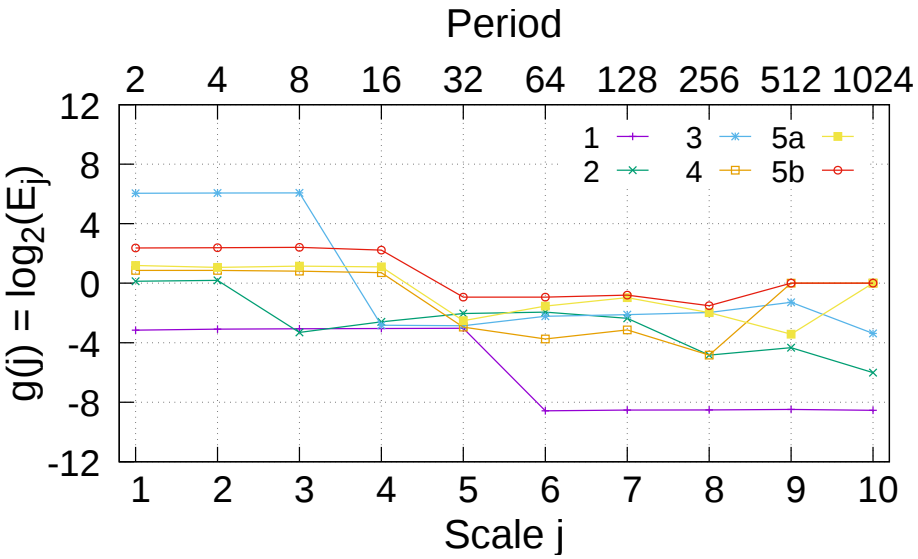
Figure 4.7: Energy function plots for use cases from Table 4.1 - Heartbleed.



4.2.2 Results

Figure 4.8 shows the energy function plots for all use cases. Each use case corresponds to a different line, and the relevant sampling interval is as specified in Table 4.1. As shown in Chapter 2, the second X-axis at the top of the plot represents the period of each decomposition level when multiplied by the sampling interval. In all use cases, the drop in the energy plot happens in the correct periodic interval where it generates an alarm. For instance, the Heartbleed attack (use case 3 in Figure 4.8) has a period of 1s and a sampling interval of 125ms. Hence, we see a decrease in the energy function plot between levels 3-4, which indicates a periodicity of $8 \times 125\text{ms} = 1\text{s}$.

Figure 4.8: Energy function plots for use cases from Table 4.1 - All use cases.



These results highlight that our algorithm is both accurate and versatile with respect to detecting periodic behavior. For example, it can capture fine-grained periodic behavior such as the 24ms RTT pattern in a regular link (use case 1) using a 750μs sampling interval as well as coarser periodicities such as the 300s cycle of the Trickbot attack (use case 5) using a 20s sampling interval. Moreover, because of its demonstrated efficiency (see Section IV.A), the algorithm is highly effective in leveraging modern data plane technologies to detect periodic patterns in real-time.

5 CONCLUSIONS AND FUTURE WORK

In this dissertation, we presented an innovative approach for performing periodicity detection in the data plane using the Discrete Wavelet Transform (DWT) in the P4 language. By leveraging the capabilities of programmable data planes, we demonstrated the feasibility and effectiveness of integrating periodicity detection directly into the network fabric. Our research contributes to the field of in-network traffic analysis and addresses critical challenges in network management, security, and resource allocation. The findings and contributions of this work has been published in IEEE GLOBECOM 2022 [Huaytalla et al. 2022].

Throughout this dissertation, we have made significant contributions to the state-of-the-art in periodicity detection. Firstly, we introduced a method for embedding DWT-based periodicity detection algorithms in P4-compatible switches, allowing for real-time analysis of network traffic without the need for offloading to external devices or relying solely on centralized controllers. Our solution empowers network operators with the ability to perform sophisticated traffic analysis at the edge of the network, providing increased visibility and control.

Secondly, we designed and implemented an efficient and scalable DWT-based periodicity detection module in the P4 language. This module is flexible and programmable, enabling network operators to adapt and customize the periodicity detection algorithms based on their specific requirements. By leveraging the programmability of P4-compatible switches, we offer a powerful and extensible platform for in-network traffic analysis, paving the way for further research and innovation in this field.

Furthermore, we conducted extensive experiments using various network topologies and traffic scenarios, ensuring a thorough evaluation of our proposed approach. Our results demonstrate the accuracy and efficiency of the DWT-based periodicity detection in the data plane. By detecting periodic patterns directly in the network fabric, we enable faster response times to network events, enhance network security by identifying anomalous periodic behaviors, and optimize network resource allocation based on real-time traffic analysis.

The potential impact of our research extends beyond the scope of this dissertation. Integrating periodicity detection in the data plane can revolutionize network operations, providing numerous benefits to both service providers and end-users. The ability to detect periodic patterns directly in the network fabric opens up new possibilities for intelligent

network management, enabling dynamic adaptation to changing traffic patterns, efficient utilization of network resources, and proactive identification of network anomalies.

While our work has achieved considerable advancements, there are promising avenues for future research and development. Firstly, we plan to explore more advanced DWT-based algorithms and investigate their applicability in the context of periodicity detection. This could involve incorporating machine learning techniques to enhance detection accuracy and adaptability to dynamic network environments. We also aim to further optimize the performance of our solution by exploring hardware acceleration techniques and leveraging the capabilities of emerging programmable switch architectures. Additionally, we envision extending our work to consider other types of temporal patterns beyond periodicity. Burstiness, seasonality, and other temporal characteristics play crucial roles in network traffic analysis. Investigating and developing techniques to detect and analyze such patterns in the data plane could significantly enhance the understanding and management of complex network behavior.

In conclusion, this dissertation presented a novel approach for integrating DWT-based periodicity detection in the data plane using the P4 language. Our research demonstrated the feasibility, effectiveness, and scalability of performing periodicity analysis directly in the network fabric. By pushing the boundaries of in-network traffic analysis, we provided valuable insights and lay the foundation for future research in the field. The potential applications of our work extend to various domains, including network management, security, and resource allocation, fostering a more efficient and resilient network infrastructure.

REFERENCES

AHMED, M.; Naser Mahmood, A.; HU, J. A survey of network anomaly detection techniques. **Journal of Network and Computer Applications**, v. 60, p. 19–31, 2016. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1084804515002891>. Acesso em: 6 set. 2023.

AKGüL, T. et al. Periodicity-based anomalies in self-similar network traffic flow measurements. **IEEE Transactions on Instrumentation and Measurement (TIM 2011)**, v. 60, n. 4, p. 1358–1366, 2011.

BARBOSA, R. R. R.; SADRE, R.; PRAS, A. Exploiting traffic periodicity in industrial control networks. **International Journal of Critical Infrastructure Protection (ijcip 2016)**, v. 13, p. 52–62, 2016.

BARTLETT, G.; HEIDEMANN, J.; PAPADOPOULOS, C. Low-rate, flow-level periodicity detection. In: **IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS 2011)**. [S.l.: s.n.], 2011. p. 804–809.

BILGE, L. et al. Disclosure: Detecting Botnet Command and Control Servers through Large-Scale NetFlow Analysis. In: **ACM Annual Computer Security Applications Conference (ACSAC 2012)**. [S.l.: s.n.], 2012. p. 129–138. ISBN 9781450313124.

BUDIU, M.; DODD, C. The P416 Programming Language. **SIGOPS Oper. Syst. Rev.**, Association for Computing Machinery, New York, NY, USA, v. 51, n. 1, p. 5–14, sep 2017.

CAIDA. **CAIDA Data Monitors**. 2021. Disponível em: <https://www.caida.org/catalog/datasets/monitors/>. Acesso em: 6 set. 2023.

CALLEGARI, C. et al. Combining Wavelet Analysis and Information Theory for Network Anomaly Detection. In: **ACM International Symposium on Applied Sciences in Biomedical and Communication Technologies (ISABEL 2011)**. New York, NY, USA: ACM, 2011. ISBN 9781450309134.

CISCO. White Paper, **Cisco Annual Internet Report (2018–2023)**. 2020. Disponível em: <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html>. Acesso em: 6 set. 2023.

CONSORTIUM, T. P. L. **P4-16 Language Specification, version 1.2.2**. 2021. 1–170 p. Disponível em: <https://p4.org/p4-spec/docs/P4-16-v1.2.2.html>. Acesso em: 6 set. 2023.

DING, D.; SAVI, M.; SIRACUSA, D. Estimating Logarithmic and Exponential Functions to Track Network Traffic Entropy in P4. In: **IEEE/IFIP NOMS**. [S.l.: s.n.], 2020. p. 1–9.

DU, Z. et al. Network Traffic Anomaly Detection Based on Wavelet Analysis. In: **IEEE SERA**. [S.l.: s.n.], 2018.

ESLAHI, M. et al. Periodicity classification of HTTP traffic to detect HTTP Botnets. In: **IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE 2015)**. [S.l.: s.n.], 2015. p. 119–123.

FELDMANN, A. et al. Dynamics of ip traffic: A study of the role of variability and the impact of control. In: **Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication**. New York, NY, USA: Association for Computing Machinery, 1999. (SIGCOMM '99), p. 301–313. ISBN 1581131356. Disponível em: <https://doi.org/10.1145/316188.316235>. Acesso em: 6 set. 2023.

FRANCO, M. et al. SecGrid: A Visual System for the Analysis and ML-Based Classification of Cyberattack Traffic. In: **IEEE 46th Conference on Local Computer Networks (LCN 2021)**. Edmonton, Canada: [s.n.], 2021. p. 1–8.

FU, C. et al. Realtime Robust Malicious Traffic Detection via Frequency Domain Analysis. In: **ACM Conference on Computer and Communications Security (CCS 2021)**. [S.l.]: Association for Computing Machinery, 2021.

GAO, S.; HANDLEY, M.; VISSICCHIO, S. Stats 101 in p4: Towards in-switch anomaly detection. In: **Proceedings of the Twentieth ACM Workshop on Hot Topics in Networks**. New York, NY, USA: Association for Computing Machinery, 2021. (HotNets '21), p. 84–90. ISBN 9781450390873. Disponível em: <https://doi.org/10.1145/3484266.3487370>. Acesso em: 6 set. 2023.

GONZáLEZ, L. A. Q. et al. BUNGEE: An Adaptive Pushback Mechanism for DDoS Detection and Mitigation in P4 Data Planes. In: **IEEE International Symposium on Integrated Network Management (IM 2021)**. [S.l.: s.n.], 2021. p. 393–401.

GU, G.; ZHANG, J.; LEE, W. Botsniffer: Detecting botnet command and control channels in network traffic. 2008.

HUANG, P.; FELDMANN, A.; WILLINGER, W. A non-instrusive, wavelet-based approach to detecting network performance problems. In: **Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement**. New York, NY, USA: Association for Computing Machinery, 2001. (IMW '01), p. 213–227. ISBN 1581134355. Disponível em: <https://doi.org/10.1145/505202.505229>. Acesso em: 6 set. 2023.

HUAYTALLA, B. R. et al. Dwt in p4: Periodicity detection in the data plane. In: **GLOBECOM 2022 - 2022 IEEE Global Communications Conference**. [S.l.: s.n.], 2022. p. 6343–6348.

INTEL; FOUNDATION, L. **Data Plane Development Kit (DPDK)**. Disponível em: <https://www.dpdk.org/>. Acesso em: 6 set. 2023.

JI, S. Y.; CHOI, S.; JEONG, D. H. Designing an Internet Traffic Predictive Model by Applying a Signal Processing Method. **Journal of Network and Systems Management**, Springer US, v. 23, p. 998–1015, 2015.

JIANG, D. et al. A Compressive Sensing-Based Approach to End-to-End Network Traffic Reconstruction. In: **IEEE Transactions on Network Science and Engineering**. [S.l.: s.n.], 2020.

LAPOLLI, A. C. et al. Offloading Real-time DDoS Attack Detection to Programmable Data Planes. In: **IFIP/IEEE IM**. [S.l.: s.n.], 2019.

LUO, X.; LI, D.; ZHANG, S. Traffic flow prediction during the holidays based on DFT and SVR. In: . [S.l.: s.n.], 2019. v. 2019.

MAI, J.; YUAN, L.; CHUAH, C.-N. Detecting BGP Anomalies with Wavelet. In: **IEEE Network Operations and Management Symposium (NOMS 2008)**. [S.l.: s.n.], 2008. p. 465–472.

MALWARE-TRAFFIC-ANALYSIS.NET. **2019-10-31 - IcedId Infection with Trickbot.** 2021. Disponível em: <https://www.malware-traffic-analysis.net/2019/10/31/index.html>. Acesso em: 6 set. 2023.

MALWARE-TRAFFIC-ANALYSIS.NET. **2019-12-26 - IcedId Infection With Trickbot**. 2021. Disponível em: <https://www.malware-traffic-analysis.net/2019/12/26/index.html>. Acesso em: 6 set. 2023.

MALWARE-TRAFFIC-ANALYSIS.NET. **2021 - 05 - 13 Hancitor with Ficker Stealer and Cobalt Strike.** 2021. Disponível em: <https://www.malware-traffic-analysis.net/2021/05/13/index.html>. Acesso em: 6 set. 2023.

MOUSTAFA, N.; HU, J.; SLAY, J. A holistic review of network anomaly detection systems: A comprehensive survey. **Journal of Network and Computer Applications**, v. 128, p. 33–55, 2019. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1084804518303886>. Acesso em: 6 set. 2023.

POPA, S. M.; MANEA, G. M. Using Traffic Self-Similarity for Network Anomalies Detection. In: **IEEE International Conference on Control Systems and Computer Science (CSCS 2015)**. [S.l.: s.n.], 2015. p. 639–644.

ROUGHAN, M.; VEITCH, D.; ABRY, P. On-line estimation of the parameters of long-range dependence. In: **IEEE Global Communications Conference (GLOBECOM 1998)**. [S.l.: s.n.], 1998. v. 6, p. 3716–3721 vol.6.

ROUGHAN, R.; VEITCH, D.; ABRY, P. Real-time estimation of the parameters of long-range dependence. **IEEE/ACM Transactions on Networking**, v. 8, n. 4, p. 467–478, 2000.

SHARAFALDIN., I.; Habibi Lashkari., A.; GHORBANI., A. A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In: **International Conference on Information Systems Security and Privacy (ICISSP 2018)**. [S.l.: s.n.], 2018. ISBN 978-989-758-282-0.

YUE, M. et al. Identifying LDoS attack traffic based on wavelet energy spectrum and combined neural network. In: **International Journal of Communication Systems**. [S.l.: s.n.], 2018.

APPENDIX A — RESUMO EXPANDIDO

Esta dissertação apresenta uma implementação estendida do método da Transformada Discreta de Wavelet (DWT) de uma dimensão na linguagem de programação P4, permitindo uma análise eficiente e em tempo real do comportamento periódico no tráfego de rede. A DWT é uma ferramenta matemática amplamente utilizada para análise de sinais, permitindo a divisão de um sinal dado em diferentes componentes de frequência e analisando cada componente com uma resolução adaptada à sua escala.

Ao abordar as limitações dos dispositivos de plano de dados programáveis em P4 existentes, desenvolvemos um algoritmo online eficiente que realiza a decomposição DWT inteiramente no plano de dados, superando as restrições e complexidades associadas ao deslocamento de cálculos para dispositivos externos ou dependendo exclusivamente de controladores centralizados.

Nossa avaliação concentra-se em uma implementação de hardware do algoritmo, utilizando o Netronome NFP-4000 SmartNIC, e demonstra um mínimo impacto na taxa de transferência, com menos de 1% de impacto em pacotes de tamanho médio, operando dentro das restrições dos recursos limitados do plano de dados.

Além da implementação, demonstramos uma aplicação prática de nossa implementação leve em P4, introduzindo uma abordagem baseada em limiar para a detecção em tempo real do comportamento periódico em sinais, permitindo a identificação eficiente e oportuna de padrões periódicos na taxa de linha do plano de dados (40 Gbps).

Vários exemplos de traços de tráfego de nível de pacote sintéticos e do mundo real, exibindo padrões periódicos de origens benignas e maliciosas, ilustram a eficácia de nossa abordagem.

As contribuições desta dissertação se estendem tanto ao campo da análise de tráfego de rede quanto à implementação prática da DWT em planos de dados programáveis, oferecendo oportunidades para análise em tempo real e detecção de comportamentos periódicos diretamente no tecido da rede.

Nossa abordagem demonstra escalabilidade, eficiência e precisão, tornando-se uma ferramenta valiosa para aplicações como detecção de anomalias, controle de congestionamento e segurança de rede.

Em conclusão, esta dissertação contribui para o avanço da análise de tráfego em rede e oferece uma base para pesquisas futuras no domínio, demonstrando a viabilidade e o potencial de realizar a DWT inteiramente no plano de dados com um impacto mínimo e

restrições, e destacando os benefícios da análise de tráfego em rede para o gerenciamento e a segurança da rede.

APPENDIX B — GLOBECOM 2022 PUBLISHED PAPER

B. R. Huaytalla, A. S. Jacobs, M. V. B. Silva, F. B. Carvalho, R. A. Ferreira, W. Willinger, L. Granville. "DWT in P4: Periodicity Detection in the Data Plane." 2022 IEEE Global Communications Conference, Rio de Janeiro, Brazil, 2022, pp. 6343-6348

- **Title:** *DWT in P4: Periodicity Detection in the Data Plane*.
- **Abstract:** This paper presents a P4 implementation of the (1-D) Discrete Wavelet Transform (DWT) method. As a mathe-matical tool for analyzing signals such as packet-level traces, the DWT divides a given signal into different frequency components and analyzes each component with a resolution matched to its scale. We develop an efficient online algorithm that circumvents various limitations of existing P4-programmable data plane devices and performs the DWT decomposition entirely in the data plane. Our evaluation of a hardware implementation (i.e., Netronome NFP-4000 SmartNIC) of the algorithm shows that it results in only minimal throughput overhead (less than 1% for average-sized packets) and operates within constraints imposed by the limited available data plane resources. As an application, we use our lightweight P4 implementation of the DWT and describe a novel threshold-based approach for detecting periodic behavior in a signal in real-time, at line rate in the data plane (40 Gbps). We illustrate our approach with different examples of synthetic and real-world packet-level traffic traces that exhibit periodic patterns of either benign or malicious origins.
- **Status:** Published.
- **Qualis:** A1.
- **Conference:** IEEE Global Communications Conference (GLOBECOM).
- **Date:** 4–8 December 2022.
- **Address:** Rio de Janeiro, Brazil.
- **URL:** <https://globecom2022.ieee-globecom.org/>.
- **Digital Object Identifier (DOI):** 10.1109/GLOBECOM48099.2022.10000755

DWT in P4: Periodicity Detection in the Data Plane

Briggette R. Huaytalla* Arthur S. Jacobs* Marcus V. B. Silva* Fabrício B. Carvalho†
Ronaldo A. Ferreira† Walter Willinger‡ Lisandro Z. Granville*
*UFRGS, Brazil †UFMS, Brazil ‡NIKSUN, USA

***Abstract*—This paper presents a P4 implementation of the (1-D) Discrete Wavelet Transform (DWT) method. As a mathematical tool for analyzing signals such as packet-level traces, the DWT divides a given signal into different frequency components and analyzes each component with a resolution matched to its scale. We develop an efficient online algorithm that circumvents various limitations of existing P4-programmable data plane devices and performs the DWT decomposition entirely in the data plane. Our evaluation of a hardware implementation (*i.e.,* Netronome NFP-4000 SmartNIC) of the algorithm shows that it results in only minimal throughput overhead (less than 1% for average-sized packets) and operates within constraints imposed by the limited available data plane resources. As an application, we use our lightweight P4 implementation of the DWT and describe a novel threshold-based approach for detecting periodic behavior in a signal in real-time, at line rate in the data plane (40 Gbps). We illustrate our approach with different examples of synthetic and real-world packet-level traffic traces that exhibit periodic patterns of either benign or malicious origins.**

I. Introduction

The recent proliferation of Internet-connected devices, systems, and services and dramatic changes in Internet usage [1] are among the main reasons for the continued exponential growth in Internet traffic. To carry out tasks such as detecting nefarious network activities or distinguishing these activities from benign behavior, network operators are required to collect and analyze enormous amounts of network measurement data. The analysis of such data may impose timing constraints (*e.g.,* non-real-time vs. real-time), determining the type of methods at the operators' disposal, such as traditional statistical analysis techniques [2]–[5], information theory-based approaches [6], [7], and machine learning algorithms [2]. These methods can be further separated into time-domain [8], frequency-domain [2], [9], and wavelet-domain techniques, and some of them can be adapted for streaming data analysis. Analyzing streaming data allows operators to consider different features (*e.g.,* packet counts in a time interval) required for inferring certain network activities, computing them in high throughput scenarios [5] and at line rate [2] without having to store the data under analysis.

One practical use of signal processing for analyzing streaming data is to infer periodic activities within network traffic. Such activities are often indications of recurring patterns in network usage and can be either malicious or benign. Inferring periodic activities of unknown origins will typically trigger a detailed post-mortem and offline forensic analysis by the network operator to identify the observed periodic activities' root cause(s). Examples of such efforts include detecting anomaly behavior [3], [10], reconstructing the signal of a network communication [11], and analyzing the energy spectrum [12]. However, traditional signal processing techniques such as the Discrete Fourier Transform (DFT) are known to have a high computational overhead that prevents them from being used for real-time periodicity detection in high throughput scenarios [2], [10]. Their practical use in this context is therefore limited to performing post-mortem analysis tasks [9], [11].

978-1-6654-3540-6/22/$31.00 © 2022 IEEE

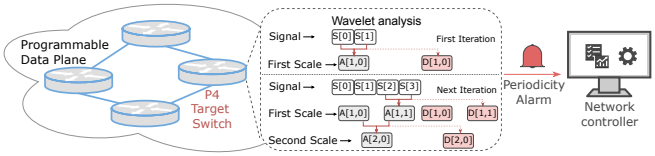


Fig. 1: Wavelets analysis for periodicity detection in P4.

In contrast, the Discrete Wavelet Transform (DWT) method can be used to analyze time-series data with low computational overhead by leveraging their intrinsic ability for time-frequency localization, *i.e.,* dividing the data into different frequency components and analyzing each component with a resolution matched to its scale. Used in prior works [3]–[5], [13] to analyze networking traffic data, the DWT method is an especially promising technique for analyzing streaming data where the "signal" is given in the form of packet-level network traces. Not only does it allow for the simultaneous analysis of the signal at different scales, but the method can be naturally parallelized and performed at line rate to enable real-time signal analysis.

Despite its low computational overhead, implementing the DWT method to process high-volume traffic streams at line rate poses significant challenges. Using off-the-shelf commodity hardware to perform the necessary time-frequency localization of the incoming traffic is, in general, inefficient as it can introduce additional overhead that offsets the benefits of using the DWT. In turn, recent advances in programmable data plane technologies (*e.g.,* P4 [14]) present unique opportunities to use techniques such as the DWT for line rate traffic processing in the data plane. However, P4's limited support for commonly used arithmetic operations makes it difficult to implement the DWT method in P4 and run it on actual hardware.

This paper presents a P4 implementation of the DWT method to perform a signal's time-frequency localization directly in the data plane. Figure 1 provides an overview of our solution, in which a P4-enabled programmable device is running an efficient algorithm that we designed to perform the DWT decomposition. We compute the energy function to analyze each decomposition level of the DWT, and use a

threshold-based heuristic to automatically alert the network operator of identified periodic behavior. To circumvent the limitations imposed by existing P4-enabled devices, we rely on a number of mathematical modifications that reduce the need for complex arithmetic operations. Our solution has a small memory footprint in the data plane and results in only minimal throughput overhead (less than 1% for average-sized packets). Finally, we show with several performance and security use cases how our proposed solution correctly identifies periodic behavior in both synthetic and real-world packet traffic traces.

II. Background

The Discrete Wavelet Transform (DWT) method uses waveforms to "localize" a signal in both frequency and time. In addition to pinpointing the specific times when each frequency occurs in the signal [15], the DWT has lower computational complexity than the more traditional Fourier transform method—$O(n)$ vs. $O(n \log n)$ [2], [15], where n is the number of samples in the signal—making it more appealing for analyzing signals in high-throughput settings such as network traffic.

To decompose a signal, the DWT uses a low-pass filter (*a.k.a.*, *scaling function* or *father wavelet*) and a high-pass filter (*a.k.a.*, *wavelet function* or *mother wavelet*). These filters are convolved with k data points at a time (depending on the size of the filters), encoding high- and low-frequency information into two distinct levels of decomposition and effectively sub-sampling the original signal by half. The encoded data points generated by the high-pass and low-pass filters are referred to as the *detail* and *approximation* coefficients, respectively. We can apply the DWT decomposition recursively m times using the approximation coefficients at level $j-1$ as input to level j ($1 \leq j \leq m$) to analyze frequencies at a finer granularity. The original signal corresponds to level zero.

The original DWT method was proposed alongside a simple set of wavelet filters known as the Haar wavelet [13]. Here, we also use the Haar wavelet and leave the application of other wavelets filters such as the Daubechies wavelets [3] for future work. We define the Haar wavelet low-pass and high-pass filters as $(1/\sqrt{2}, 1/\sqrt{2})$ and $(1/\sqrt{2}, -1/\sqrt{2})$, respectively [16] and consider a time series $X_{0,k}, k = 0, 1, 2, \ldots$ representing the input signal. For scale one of the DWT decomposition, we multiply these values with consecutive samples in the input signal and then add the resulting products to compute the approximation and detail coefficients, respectively. More generically, we describe the approximation and detail coefficients for scale j, $j \geq 1$, at position k by Equations 1 and 2, respectively.

$$X_{j,k} = \frac{1}{\sqrt{2}}(X_{j-1,2k} + X_{j-1,2k+1}) \tag{1}$$

$$d_{j,k} = \frac{1}{\sqrt{2}}(X_{j-1,2k} - X_{j-1,2k+1}) \tag{2}$$

A. Periodicity Detection: An Energy-Function Analysis

Most of the existing methods for analyzing recurring patterns in networking require storing a high volume of network traffic data and mining it post-mortem. Due to the high computational complexity of these methods [15], this type of analysis can take a long time. However, mining traffic in today's high-speed networks to detect patterns in communication requires analyzing measurements at line rates with methods that have low computational complexity. Compared to most existing approaches, the DWT method with its low computational complexity and broad applicability to different problems is especially well suited for the high-throughput conditions and real-time requirements imposed by modern-day networks.

In particular, the energy function of the detail coefficients has been used to detect periodic signals in different scenarios (*e.g.,* for studying network congestion and its impact on TCP retransmissions [16]). The energy function E_j is defined as

$$E_j = \frac{1}{N_j} \sum_k |d_{j,k}|^2, \quad j = 1, 2, \ldots, m \tag{3}$$

where j is the decomposition level, and N_j is the number of coefficients at level j. Computing the energy of the detail coefficients at each decomposition level allows us to examine the temporal properties in the signal from high to low frequencies as the level of the wavelet decomposition increases.

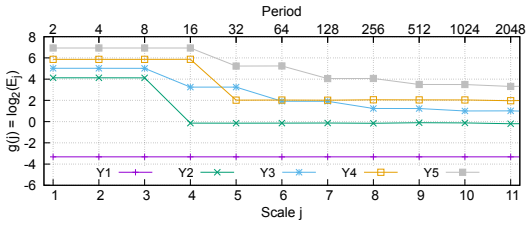


Fig. 2: Using the energy function to detect periodicity.

Feldmann *et al.* [17] show that plotting the function $g(j) = \log_2(E_j)$ can be used as a means to detect periodicities in a signal. As each decomposition level filters out a specific frequency range in the original signal, a particular periodicity manifests as a sudden decrease in $g(j)$. Figure 2 shows an example with five different signals. Y1 is white noise with no periodicity. Y2–Y5 are mixtures of white noise with a periodic signal of period 8, 10, 16, and 20, respectively. The figure shows that while $g(j)$ decreases for Y2–Y5 near the point marking the signal's period, $g(j)$ remains flat for Y1.

III. Online DWT Decomposition

We consider a signal where each sample is indexed and represents the number of packets of a flow in a fixed time interval. A flow refers to all the packets that match a rule specified by the network operator, *e.g.,* all packets with a given destination port or IP address. Implementing the DWT transform for such signals in P4 presents significant challenges. First, P4 does not allow loops and does not support all the arithmetic operations required to evaluate Equations 1, 2 and 3. Second, processing in P4 is asynchronous, meaning that we need a packet arrival to trigger a computation. Finally, computing and storing the signal and the approximation coefficients at different levels may impose significant processing and storage overheads.

A. Arithmetic Operations

At first sight, Equation 3 presents additional hurdles for its implementation in P4, including the division by a number N_j that varies over time for each level of the decomposition and floating-point operations (division by $\sqrt{2}$). However, by expanding the equation to different levels, we can simplify it to the point where these operations are no longer needed. First, we assume without loss of generality that the length of the signal is a power of two, *i.e.,* $N = 2^n$ for some n, so $N_j = N/2^j$, where j represents the level of the decomposition or the signal when $j = 0$. Then, we can write Equation 3 as

$$NE_j = 2^{j-1} \sum_k \left| X_{j-1,2k} - X_{j-1,2k+1} \right|^2 \quad (4)$$

For conciseness, let $A_{j,w} = X_{j-1,2w} + X_{j-1,2w+1}$, where X_j is the signal for $j = 0$ or the approximation coefficients for $j \geq 1$ as defined in Section II. $A_{j,w}$ is a parameterized version of the expression inside the parenthesis of Equation 1, *i.e.,* $A_{j,w}$ is the approximation coefficient without the $\sqrt{2}$ normalization factor. For $j \geq 2$, we can expand Equation 4, using the definition of $A_{j,w}$ and Equation 1 as

$$NE_j = 2^{j-2} \sum_k \left| A_{j-1,2k} - A_{j-1,2k+1} \right|^2 \quad (5)$$

Also, since we have $A_{j-1,2k} = X_{j-2,4k} + X_{j-2,4k+1}$ and $A_{j-1,2k+1} = X_{j-2,4k+2} + X_{j-2,4k+3}$, we can compute Equation 5 using only additions, subtractions, and multiplications. Moreover, we can efficiently implement multiplication of a number by itself or by a power of two using only shift and addition operations. Finally, since we are only interested in finding points where $g(j)$ decreases, we do not need to divide the right-hand side of Equation 5 by N.

B. Asynchronous Computation and Resource Usage

To build a signal on a P4 switch, we need to execute an action to count the number of packets in a sampling interval. However, the switch runs an action only when a packet arrives and matches an operator-defined rule. If no packet arrives during long periods, we have a signal with multiple consecutive samples equal to zero. Since P4 does not allow loops, we cannot simply iterate over these samples to compute the approximation and detail coefficients of the DWT transform and propagate them to calculate the other decomposition levels. Also, storing the signal's samples and the approximation and detail coefficients for each decomposition level would consume a large amount of memory in the switch and limit the length of the signal and the number of flows we could analyze.

We leverage the simplification of the energy function we developed in Section III-A to develop an efficient algorithm that stores for each signal only a sliding window W_j with two approximation coefficients and the cumulative sum $S_j = \sum_k \left| A_{j-1,2k} - A_{j-1,2k+1} \right|^2$ for each level j. We also store the index (L) of the sample that was last processed by the switch, totaling $3*$MAX_LEVEL$+1$ integers per flow, where MAX_LEVEL is the maximum number of decomposition levels. To compute the k-th approximation coefficient at level j, we need the coefficients (or signal samples if $j = 1$) with indices $2k$ and $2k+1$ at level $j-1$ (See Equations 1 and 5). We can map this dependence of the approximation coefficients to a tree structure where signal samples are the leaves, and the coefficients are the internal nodes of a binary tree, as illustrated in Figure 3. Note that to calculate the coefficient of an internal node, we always need the values of its two children. Since our algorithm performs online computations, some samples or coefficients might not be available when trying to compute the approximation coefficient and the cumulative sum of the energy function at the next level at a specific time. Also, to deal with the case where the switch does not receive any packet for a flow during several sampling intervals, we introduce samples with a value set to zero. Therefore, the arrival of a packet at the switch may trigger the computation of the coefficients at several levels as a new sample may complete an entire subtree. For example, the availability of sample 7 in Figure 3d allows the computation of the approximation coefficients 3 at level 1, 1 at level 2, and 0 at level 3.

Algorithm 1 shows the pseudocode to decompose a signal using the DWT transform, compute the energy function for

Algorithm 1 Online DWT Decomposition and Energy Function.

```
1:  procedure OnlineDWT(
        L: Index of the previously measured sample,
        R: Index of the currently measured sample,
        s: Number of packets in the current sample)
2:      if IsEven(L) and IsEven(R) then PropagateL(L, R, 1)
3:      else if IsEven(L) and IsOdd(R) then PropagateLR(L, R, 1, s)
4:      else if IsOdd(L) and IsEven(R) then PropagateL(L+1, R, 1)
5:      else if IsOdd(L) and IsOdd(R) then
6:          PropagateLR(L + 1, R, 1, s)
7:      W_{0,IsEven(R) ? 0 : 1} ← s
8:      L ← R
9:  procedure PropagateL(L, R, j)
10:     if R − L < 2 then
11:         if R − L = 1 and IsOdd(L) then
12:             BranchOdd(L, j)
13:     else
14:         Decompose(L/2, j)
15:         if j < MAX_LEVEL then PropagateL(L/2, R/2, j + 1)
16: procedure PropagateLR(L, R, j, s)
17:     if R − L = 1 then
18:         if j = 1 then W_{0,1} ← s
19:         Decompose(L/2, j + 1)
20:         if IsOdd(L) then BranchOdd(L/2, j + 1)
21:     else
22:         BranchEven(L, R, j)
23:         W_{0,1} ← s
24:         BranchOdd(R, j)
25: procedure BranchOdd(index, j)
26:     Decompose(index/2, j)
27:     if j < MAX_LEVEL and IsOdd(index/2) then
28:         BranchOdd(index/2, j + 1)
29: procedure BranchEven(L, R, j)
30:     if R − L < 2 then
31:         if IsOdd(L) then BranchOdd(L, j)
32:     else
33:         Decompose(L/2, j)
34:         if j < MAX_LEVEL then BranchEven(L/2, R/2, j + 1)
35: procedure Decompose(index, j)
36:     W_{j,IsEven(index) ? 0 : 1} ← W_{j−1,0} + W_{j−1,1}
37:     S_j ← S_j + (W_{j−1,0} − W_{j−1,1})^2
38:     if α(S_{j−1} ≪ (j − 3)) − β(S_j ≪ (j − 2)) > 0 then
39:         GenerateAlarm(j − 1)
```

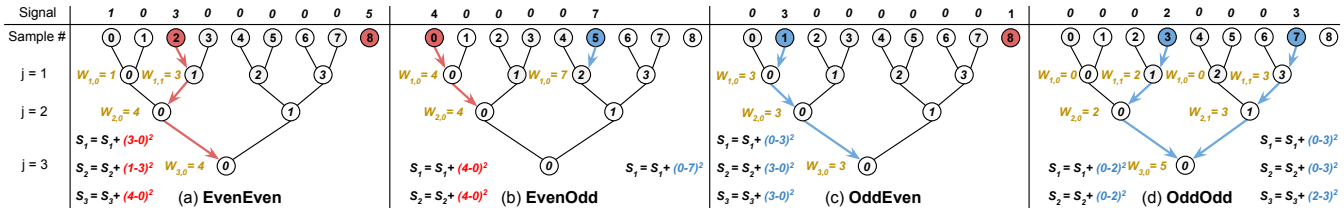


Fig. 3: Examples of signal decompositions and computations of the energy function.

each decomposition level, and generate an alarm when it detects a periodicity in a signal. To simplify the explanation, we present the algorithm with recursive functions to traverse the decomposition tree. Although P4 does not support recursion, we can implement the algorithm by simply expanding the recursive functions MAX_LEVEL times. We evaluate the maximum number of levels we can support on a programmable NIC in Section IV. Line 1 shows the main function of the algorithm that is invoked every time the switch receives a packet that marks the end of a sampling interval. Figure 3a illustrates a signal with nine samples indexed from 0 to 8 that triggers the call to function `OnlineDWT` with $L = 2$, $R = 8$, and $s = 4$. We use L and R and the fact that all samples between them are zero to define four different cases depending on whether L and R are even or odd: EvenEven, EvenOdd, OddEven, and OddOdd (Figures 3a-d).

Due to limited space, we only describe the EvenEven case, but note that similar arguments apply when considering the remaining cases as detailed by Algorithm 1. We use Figure 3a to illustrate how the algorithm works with an example where $L = 2$ and $R = 8$. Function `OnlineDWT` calls `PropagateL` (Line 9) to propagate sample L (saved in $W_{0,0}$) to the next levels because its sibling (*e.g.,* sample with index 3 in Figure 3a) for computing the approximation coefficient was not available in the previous invocation of `OnlineDWT`. As R is even and its sibling is not available yet, the algorithm does not propagate sample R to the next levels and just updates $W_{0,0}$ with the current sample (Line 7) and L with R (Line 8) for a future invocation of `OnlineDWT`. The `PropagateL` function recurses with L and R divided by two (Line 15) to map them to the proper nodes of the decomposition tree at the next level. This recursion stops when j reaches MAX_LEVEL (Line 15) or when the difference between L and R becomes less than two (Line 10). In this last case, the propagation has reached a level where either L is equal to R, and there is nothing more to propagate, or we call `BranchOdd` if L is odd at that level of the decomposition. When $index$ at `BranchOdd` is odd at a level, it means we have a sample that completes a pair of children for computing an approximation coefficient, so the `BranchOdd` function recurses while $index$ is odd in the subsequent levels, computing the approximation coefficients until j reaches MAX_LEVEL or the function reaches a node with $index$ even that does not have its sibling yet to compute the next approximation coefficient.

To detect periodicity in a signal, we use a heuristic that divides the energy at level $j-1$ by the energy at level j and checks if the result is greater than a threshold. If the energy decreases suddenly from level $j-1$ to level j, then E_{j-1}/E_j will be greater than one. More specifically, we check if $\frac{E_{j-1}}{E_j} > \frac{\alpha}{\beta}$, where α and β are integers and $\alpha > \beta$. Using Equation 5 and the definition of S_j, we rewrite this formula as $\frac{2^{j-3}S_{j-1}}{2^{j-2}S_j} > \frac{\alpha}{\beta}$. Substituting the multiplication by powers of two with shift operations, we can rewrite the equation as $\beta(S_{j-1} \ll (j-3)) - \alpha(S_j \ll (j-2)) > 0$, which is the same as Line 38 of Algorithm 1. Line 39 generates an alarm the first time the condition in Line 38 becomes true. In Section IV, we determine α and β empirically for a set of use cases we evaluate in this paper.

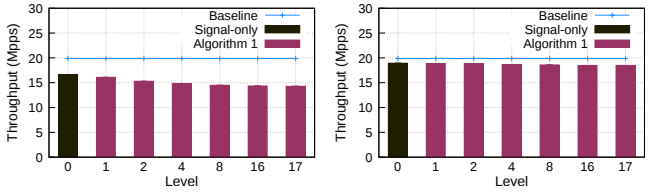
IV. Evaluation

To evaluate the performance and applicability of our approach, we first implement our algorithm in P4 with Micro-C and load it into a Netronome NFP-4000 SmartNIC to assess its overhead. Next, we determine the threshold for our periodicity heuristic and use publicly available traces of periodic traffic to analyze the energy function plot. To support reproducibility, the artifacts of our work are available at [18].

A. Performance

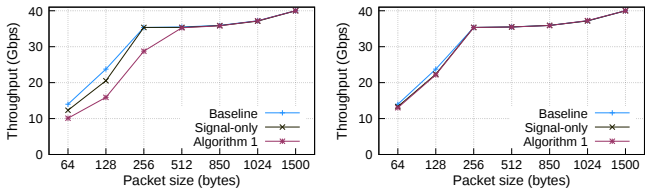
Setup. We use two servers with two Intel(R) Xeon(R) Silver 4114, each with 128 GB of RAM and a dual-port Netronome NFP-4000 40 GbE NIC directly connected on a sender-receiver configuration. The SmartNICs have a limit of 8k instructions per flow-processing core, allowing us to use at most 17 levels of decomposition. The receiver's SmartNIC receives packets, processes them according to Algorithm 1, and forwards them back to the sender. The sender sends packets and collects the throughput results. To quantify the throughout overhead of our implementation, we consider different packet sizes (from 64 to 1500 bytes) and different number of decomposition levels (from 1 to 17), and report experimental results for two different sampling intervals (250μs and 1s).

Results. Starting with the extreme case where all packets are of minimal size (*i.e.,* 64 bytes), Figures 4a and 4b show the SmartNIC throughput (in Mpps) for up to 17 decomposition levels (in red) for the sampling intervals 250μs and 1s, respectively. In both cases, level 0 (in black) denotes the throughput for basic packet capture without any decomposition (*i.e.,* constructing per-sample signal), and the baseline (in blue) is measured by simply forwarding packets to their destination. The SmartNIC provides 64-bits integer packet timestamps composed of two 32-bits variables, one for seconds and the other for nanoseconds. To compute accurate sampling intervals, we implemented a set of shift and multiplication operations to perform integer divisions using constants. As a result, when using a 250μs sampling interval, the throughput



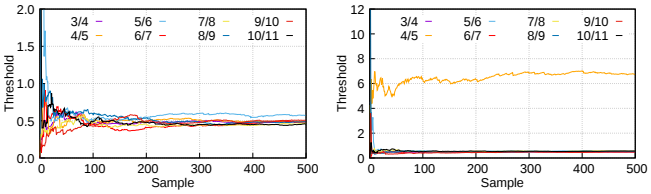
(a) 250μs. (b) 1s.

Fig. 4: Performance results varying levels using 64b packets.



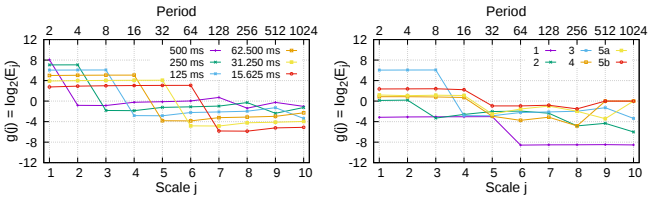
(a) 250μs. (b) 1s.

Fig. 5: Performance results varying packet size using 17 levels.



(a) Y1. (b) Y4.

Fig. 6: Threshold analysis.



(a) Heartbleed. (b) All use cases.

Fig. 7: Energy function plots for use cases from Table I.

overhead for construction of the signal is roughly 17% when compared to the baseline (see level 0 in Figure 4a). However, this overhead drops to less than 5% when the sampling interval is 1s (see level 0 in Figure 4b), mainly because in this case, calculating accurate timestamps requires fewer operations.

Figure 4 also shows that our algorithm manages to compute the DWT using very few resources. In particular, we observe that the performance overhead for the different decomposition levels (red bars) is small – roughly 12% compared to constructing the signal (black bar). This result indicates that accurately constructing the input signal (*i.e.,* placing samples in the correct interval) is responsible for most of the throughput overhead. When the sampling interval is 1s, the overhead is negligible because the SmartNIC provides timestamps already at the required granularity and there are fewer cases when the algorithm has to propagate coefficients to the subsequent levels.

All previous results assume a worst-case scenario where all packets are of minimal size (*i.e.,* 64 bytes). In Figure 5, we consider more realistic scenarios and analyze the throughput overhead of our algorithm with 17 levels of decomposition for packet sizes that vary from 64 to 1500 bytes. When the sampling interval is 250μs (Figure 5a), we notice that for packets of size 512 bytes and larger, the throughput overhead of our algorithm becomes negligible. In fact, assuming averaged-sized packets in the Internet to have 900 bytes [19], we observe a throughput overhead of less than 1% when compared to the baseline. For a sampling interval of 1s (Figure 5b), the throughput overhead is practically zero when compared to the baseline behavior of the SmartNIC.

B. Applicability

Setup. To apply our algorithm in practice, it is first necessary to define an appropriate threshold to be used in Line 38. To this end, we examine the examples considered in Figure 2 (see Section II), dividing the energy value at one level by that at the next level. Figure 6a shows that for a non-periodic signal such as Y1, after some initial phase, the ratios between successive scales converge to a value of around 0.5. In contrast, for periodic signals such as Y4 (Figure 6b), the ratios between levels 4/5 (dip in energy evident in Figure 2) converge to a value larger than 6. Based on those observations, we consider a heuristic threshold value of 1.5 ($\alpha = 3$ and $\beta = 2$ in Algorithm 1).

TABLE I: Network traffic use cases.

#	Use Cases	Period	Trace Length	Sampling Interval
1	Normal link RTT [17]	≈24ms	14’	750μs
2	Congested link RTT [17]	≈1.3s	33’	325ms
3	Heartbleed [20]	≈1s	20’	125ms
4	Cobalt strike [21]	≈60s	17’	4s
5	Trickbot (a) [22], (b) [23]	≈300s	291’, 161’	20s

To experiment with our algorithm, we next surveyed the literature for security and performance use cases with intrinsic periodic behavior and summarize the examples we found in Table I. Performance use cases (1-2) capture the RTT of packets in benign traffic over a normal link and a congested link, respectively. The security use cases (3-5) consist of different attacks that generate malicious traffic such as Heartbleed and Cobalt strike. We analyze each use case’s traffic separately. Also, for each use case we rely on a different sampling interval to facilitate presentation. While a larger sampling interval limits the applicability for fine-grained periodicity, it simply shifts the dips in the energy function to smaller levels. We illustrate this behavior for the Heartbleed use case in Figure 7a.

Results. Figure 7b shows the energy function plots for all use cases. Each use case corresponds to a different line, and the relevant sampling interval is as specified in Table I. As shown in Section II, the second X-axis at the top of the plot represents the period of each decomposition level when multiplied by the sampling interval. In all use cases, the drop in the energy plot happens in the correct periodic interval where it generates an alarm. For instance, the Heartbleed attack (use case 3 in 7b) has a period of 1s and a sampling interval of 125ms. Hence, we see a decrease in the energy function plot between levels 3-4, which indicates a periodicity of $8 \times 125\text{ms} = 1\text{s}$.

These results highlight that our algorithm is both accurate and versatile with respect to detecting periodic behavior. For

example, it can capture fine-grained periodic behavior such as the 24ms RTT pattern in a regular link (use case 1) using a 750μs sampling interval as well as coarser periodicities such as the 300s cycle of the Trickbot attack (use case 5) using a 20s sampling interval. Moreover, because of its demonstrated efficiency (see Section IV.A), the algorithm is highly effective in leveraging modern data plane technologies to detect periodic patterns in real-time.

V. Related Work

Periodicity detection in network traces. Prior works use real-world network traces with periodic components to evaluate Hurst parameter estimation methods [24], focus on detecting botnets using time- and frequency-based metrics to capture the very coarse-grained periodicity [25], and rely on the autocorrelation of their network traffic signals, producing periodic-based features for a random forest classifier [26]. Another study [27] examines periodicity in network traffic by tokenizing flows and looking for cycles in tokens from individual packets. These efforts are mainly concerned with performing post-mortem analyses of the traffic, and the proposed methods are typically not amenable to processing streaming-type data in programmable data planes.

Signal processing in networking. Several works use signal processing techniques to analyze network traffic traces. Some works use DWT coefficients to propose an anomaly detection classification model [3] and analyze changing patterns in time series with the entropy of IP addresses [4]. The BAlet [5] framework detects anomalous BGP updates by leveraging the DWT to decompose a signal of localized BGP update counts. Whisper [2] is a machine learning-based intrusion detection system that leverages frequency domain-based input features (*i.e.,* using the DFT). These efforts do not detect periodic behavior and do not consider any data plane implementation.

Statistical analysis in programmable data planes. Several recent papers propose statistical analysis approaches directly in the data plane. [28] relies on a P4 implementation of entropy to detect DDoS attacks and [6] presents methods for estimating logarithmic and exponential functions in P4 to track traffic entropy. Gao *et al.* [29] implement several statistical techniques (*e.g.,* mean, variance) in P4 and gather them in a library. Our work contributes to these ongoing efforts and adds a P4 implementation of the DWT and a data plane-based periodicity detection technique to this library.

VI. Conclusions and Future Work

This paper presents a P4 implementation of the DWT method and uses it to develop an energy function-based method for detecting periodic patterns in an incoming signal in real-time, at line rate in the data plane. Our P4 implementation of the DWT is based on an efficient online algorithm that overcomes the limitations of existing P4-programmable data plane targets by exploiting mathematical properties of the DWT decomposition. Using this new data plane capability afforded by our P4 implementation of the DWT, our novel energy function-based method for detecting periodic behavior in signals such a packet-level network traffic traces in real-time can be used to automatically alert network operators. Determining appropriate thresholds for our method and developing a dashboard for facilitating an efficient post-mortem analysis whereby operators can determine the alerts' root cause(s) so as to decide whether the automatically detected periodic patterns are malicious or benign is part of our future work.

Acknowledgements

This work was supported in part by CNPq procs. 423275/2016-0, 316662/2021-6, 142089/2018-4 and 465446/2014-0, by FAPESP procs. 2020/05152-7, 2015/24494-8, 2020/05183-0, 14/50937-1 and 15/24485-9, by FAPERGS proc. 16/2551-0000488-9, and by CAPES Finance Code 001.

References

[1] Cisco, "Cisco Annual Internet Report (2018–2023)," Cisco, White Paper.
[2] C. Fu *et al.*, "Realtime Robust Malicious Traffic Detection via Frequency Domain Analysis," in *ACM CCS*, 2021.
[3] S. Y. Ji *et al.*, "Designing an Internet Traffic Predictive Model by Applying a Signal Processing Method," *JNSM*, 2015.
[4] C. Callegari *et al.*, "Combining Wavelet Analysis and Information Theory for Network Anomaly Detection," in *ACM ISABEL*, 2011.
[5] J. Mai, L. Yuan, and C.-N. Chuah, "Detecting BGP Anomalies with Wavelet," in *IEEE/IFIP NOMS*, 2008.
[6] D. Ding *et al.*, "Estimating Logarithmic and Exponential Functions to Track Network Traffic Entropy in P4," in *IEEE/IFIP NOMS*, 2020.
[7] L. González *et al.*, "BUNGEE: An Adaptive Pushback Mechanism for DDoS Detection and Mitigation in P4 Data Planes," in *IEEE IM*, 2021.
[8] S. M. Popa and G. M. Manea, "Using Traffic Self-Similarity for Network Anomalies Detection," in *IEEE CSCS*, 2015.
[9] X. Luo, D. Li, and S. Zhang, "Traffic flow prediction during the holidays based on DFT and SVR," in *Journal of Sensors*, 2019.
[10] Z. Du *et al.*, "Network Traffic Anomaly Detection Based on Wavelet Analysis," in *IEEE SERA*, 2018.
[11] D. Jiang *et al.*, "A Compressive Sensing-Based Approach to End-to-End Network Traffic Reconstruction," in *IEEE TNSE*, 2020.
[12] M. Yue *et al.*, "Identifying LDoS attack traffic based on wavelet energy spectrum and combined neural network," in *IJCS*, 2018.
[13] G. Bartlett, J. Heidemann, and C. Papadopoulos, "Low-rate, flow-level periodicity detection," in *IEEE INFOCOM WKSHPS*, 2011.
[14] T. Dargahi, A. Caponi *et al.*, "A Survey on the Security of Stateful SDN Data Planes," in *IEEE Communications Surveys Tutorials*, 2017.
[15] M. Roughan, D. Veitch, and P. Abry, "On-line estimation of the parameters of long-range dependence," in *IEEE GLOBECOM*, 1998.
[16] P. Huang *et al.*, "A Non-Instrusive, Wavelet-Based Approach to Detecting Network Performance Problems," in *ACM IMW*, 2001.
[17] A. Feldmann *et al.*, "Dynamics of IP Traffic: A Study of the Role of Variability and the Impact of Control," *SIGCOMM CCR*, 1999.
[18] "Code," https://github.com/ComputerNetworks-UFRGS/p4wavelets.
[19] CAIDA, "CAIDA Data Monitors," 2021, https://www.caida.org/catalog/datasets/monitors/.
[20] I. Sharafaldin. *et al.*, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in *ICISSP*, 2018.
[21] "2021-05-13 - Hancitor with Ficker Stealer and Cobalt Strike." 2021, https://www.malware-traffic-analysis.net/2021/05/13/index.html.
[22] "2019-10-31 - IcedId Infection with Trickbot." 2021, https://www.malware-traffic-analysis.net/2019/10/31/index.html.
[23] "2019-12-26 - IcedId Infection With Trickbot," 2021, https://www.malware-traffic-analysis.net/2019/12/26/index.html.
[24] T. Akgül *et al.*, "Periodicity-Based Anomalies in Self-Similar Network Traffic Flow Measurements," *IEEE TIM*, 2011.
[25] M. Eslahi *et al.*, "Periodicity classification of HTTP traffic to detect HTTP Botnets," in *IEEE ISCAIE*, 2015.
[26] L. Bilge *et al.*, "Disclosure: Detecting Botnet Command and Control Servers through Large-Scale NetFlow Analysis," in *ACM ACSAC*, 2012.
[27] R. Barbosa, R. Sadre, and A. Pras, "Exploiting traffic periodicity in industrial control networks," *IJCIP*, 2016.
[28] A. C. Lapolli *et al.*, "Offloading Real-time DDoS Attack Detection to Programmable Data Planes," in *IFIP/IEEE IM*, 2019.
[29] S. Gao, M. Handley, and S. Vissicchio, "Stats 101 in P4: Towards In-Switch Anomaly Detection," in *ACM HotNets*, 2021.