FTG-Net: Hierarchical Flow-to-Traffic Graph Neural Network for DDoS Attack Detection

Luca Barsellotti*
Università di Modena e Reggio Emilia
Modena, Italy
luca.barsellotti@unimore.it

Lorenzo De Marinis
Scuola Superiore Sant'Anna
Pisa, Italy
lorenzo.demarinis@santannapisa.it

Filippo Cugini, Francesco Paolucci
CNIT
Pisa, Italy
francesco.paolucci@cnit.it

***Abstract*—Distributed Denial of Service (DDoS) is one of the most common cyber-attacks and caused several damages in recent years. Such attacks can be executed either through the orchestration of multiple devices that synchronously send requests or through specific patterns followed by a single device to force the victim to keep resources overrun. It becomes crucial to develop robust techniques to promptly detect those two kinds of DDoS attacks and mitigate their consequences. Most of the existing Machine Learning (ML) methods are based on flow and traffic information aggregations expressed in the form of independent vectors of statistical data, ignoring topological connections. Few recent solutions try to exploit the structural information of the network to improve the classification results. In particular, Graph Neural Network (GNN) based models can process traffic-level or flow-level relationships, represented as graphs, to detect malicious patterns.**

The objective of this paper is to combine the relationships at both the traffic-level and the flow-level by developing a two-level hierarchical graph representation and a GNN model able to process it, maximizing the information brought by the traffic structure and removing the necessity of stateful features. Experiments on the CIC-IDS2017 dataset show that the performances are comparable to the state-of-the-art solutions even using only the traffic structure. The code can be accessed at https://github.com/lucabarsellotti/FTG-Net.

***Index Terms*—cybersecurity, DDoS attack detection, machine learning, graph neural networks.**

I. Introduction

In recent years, with the rise of digital technologies, cyber-attacks have become increasingly frequent causing various disruptions with economic, political, military, and privacy-related damages. According to the FBI's Internet Crime Complaint Center (IC3) report [1], 847,376 complaints were reported in 2021, leading to a 7% increase compared to 2020 and a 181% increase compared to 2017, with a potential loss exceeding 6.9$ billion. Distributed Denial of Service (DDoS) attacks are one of the most common and take advantage of the capacity limits of a network resource, orchestrating multiple requests from several devices that congest the access to the resource for ordinary usage. An example of a DDoS attack for political reasons is the one received by the public transportation websites in Israel and the United Kingdom during Q2 2022 by the ALtahrea Team [2].

There are three main types of DDoS attacks:

- Volumetric attacks: A botnet attack floods the network with a massive volume of legitimate traffic that saturates the bandwidth.
- Protocol attacks: The resources of the servers or the intermediate communication equipments, such as firewalls and load balancers, are consumed by targeting Layer 3 and Layer 4 protocol communications.
- Application attacks: The application layer (Layer 7) is attacked by legitimate requests that target vulnerabilities to consume specific resources, such as database queries or file reads.

An example of a Volumetric Attack is the ICMP Floods that overwhelms a target device with ICMP echo-requests, forcing it to respond with an equal number of echo-replies. Instead, an example of a Protocol Attack is the SYN Floods, which exploits the TCP handshake mechanism by opening several connections with SYN packets destined to the server. It will send SYN/ACK packets until the client responds with an ACK, that the attacker will never send, or a connection timeout, uselessly consuming bandwidth. To build a flexible DDoS Detection System (DDS) able to face the variety of attacks, it is necessary to observe both the aggregated traffic between servers and hosts and the specific traffic matching the communication flow (e.g., protocol, ports) between two endpoints. The DDS must reach a trade-off between the delay due to the amount of incoming traffic under analysis (to provide a sufficient overview to observe possible malicious patterns) and the reactivity to enforce proper mitigation and security rules promptly before significant damages. Moreover, this control mechanism should not overload the computation capabilities of the network. For example, in [3], the DDS is completely offloaded to a Data Processing Units (DPU) that is used to efficiently extract stateful features using hardware accelerations, classify them and generate mitigation rules online.

DDSs based on traditional Machine Learning (ML) and Deep Learning (DL) techniques exploit meaningful either flow-level or traffic-level features that are statistical aggregations of the information related to the exchanged packets in a flow or in a time window, respectively. Those approaches show good performance when trained on popular Intrusion Detection System (IDS) datasets, but lack adoption in real-

* The research for this paper was conducted during the author's time at CNIT.

world scenarios due to their inability to generalize and be flexible to different networks and traffic profiles. This problem can be addressed by replacing the statistical aggregation with the investigation of the structure of flows, given by sequences of exchanged packets between two endpoints, and aggregated traffics, intended as the set of flows established among endpoints in a certain time window. This augmented topological knowledge provides the possibility to detect common structural patterns that appear in specific types of DDoS attacks. A turning point has been reached recently with the application of Graph Neural Network (GNN) to the DDS problem [4], leveraging topological information to improve robustness and detection accuracy. However, it is necessary to analyze the intra- and inter-entity flow level interactions to completely capture all the possible attack patterns.

In this paper we propose a novel graph structure Flow-to-Traffic Graph (FTG) that incorporates both flow-level and aggregated traffic-level structures in a two-level hierarchical representation, and a GNN model (FTG-Net) able to process those fine-to-coarse graphs and classify the flows as legitimate or malicious. Our approach can represent and embed the structure of a flow between a host and a server, and combine this representation with the structure of the whole traffic, fulfilling the requirement of including each type of DDoS attack in the possibly recognized patterns. This solution is only based on the traffic topology and does not require stateful features, that can be expensive to compute in real-time scenarios and often overfit the characteristics of a specific training dataset, without the ability to generalize. When FTG-Net is deployed in real-world environments, significant stateful features, depending on the network requirements and capabilities, can be added to the graph nodes to enrich the representation and improve the performance.

We summarize the contributions of this paper as follows:

- We convert traffic data into a novel hierarchical graph structure called FTG that brings information about flows and aggregated traffics.
- We propose a GNN model, FTG-Net, able to detect DDoS attacks exploiting the introduced FTG structures.
- We conduct experiments on CIC-IDS2017 [5] to show that the information contained in the network traffic structure is sufficient to obtain results comparable with other state-of-the-art methods.

II. Related Work

Previous works based on traditional ML and DL are focused on the trade-off between detection performance and introduced latency due to features collection, processing, and classification in a real-time scenario.

Musumeci et al. [6] introduce a Software Defined Networking (SDN) system that offloads part of the DDoS detection in the data plane without the involvement of the SDN controllers by combining ML and P4-enabled switch. In particular, the P4 switches periodically provide stateful traffic information on a sliding-window basis to the DDoS attack detection module. The classification outputs are used to communicate to the switches operator-specific actions to mitigate the attacks. This approach shows significant latency reduction without compromising classification performance.

Doriguzzi et al. [7] propose LUCID, a DDoS detection architecture based on Convolutional Neural Networks (CNN) that is suitable for online resource-constrained environments. Network traffic is seen as data flows between endpoints and each input sample for the CNN is composed of the f features of packets belonging to the same flow gathered in a time window t for a maximum of n packets. If a flow is shorter than n packets, the input sample is zero-padded. The 1D convolutional kernel, at each iteration, inspects all the features of h (kernel size) sequential packets.

A significant limitation of traditional ML and DL solutions is that they analyze and classify flows independently, ignoring their relationships within the network that are crucial to adapt the system to detect several DDoS attacks. This problem is introduced by Pujol Perich et al. [8], that propose the usage of GNN to address it. In particular, given a set of flows F, they build a host-connection graph, in which hosts and flows are represented as nodes. Given a flow f with source host S and destination host D, two undirected edges are created: one between S and f, and one between f and D. The GNN model is intended to manage the heterogeneity of the introduced graph structure through different initial hidden states, message functions, and aggregation functions between host nodes and flow nodes. Finally, the flow nodes are classified as a specific attack or benign traffic.

Guo et al. [9] propose GLD-Net to fuse topological structure and traffic features. Traffic data is divided into time slots and for each of them a subgraph is built, inserting topology information (degree centrality and betweenness centrality) as node features and flow statistics as edge features. The subgraphs are processed using Graph Attention Network (GAT) Layers, which simultaneously analyze traffic and topological features, and the outputs are interpreted as a time series in input to an LSTM network.

Li et al. [10] discuss the importance of observing packet relationships to robustly detect DDoS attacks. Their proposed solution GraphDDoS exploits endpoint traffic graphs in which nodes represent packets that belong to the communication between two endpoints, including multiple flows in the same graphs. This approach allows catching peculiar patterns that appear in specific DDoS attacks such as HTTP GET and SYN Flood attacks. Our flow-level graphs are inspired by the ones used in GraphDDoS, replacing the N nodes limit with the introduction of time slots.

In section I we discussed the importance of analyzing topology from a flow-level and a traffic-level. The usage of a fine-to-coarse graph structure as input for a GNN has been exploited by Pati et. al [11] [12] in their HACT-Net, designed for breast cancer classification. The low-level Cell Graph is processed by a Cell GNN, and the final node representations are summed and concatenated in the Tissue Graph nodes. Finally, the Tissue Graph is processed by a Tissue GNN to provide predictions. Even if the use case is

completely different from DDoS Detection, this structure fits the problem. In our solution, Cell Graphs and Tissue Graphs are, respectively, replaced by Flow Graphs and Traffic Graphs.

III. Background

GNN is a Neural Network family introduced by Scarselli et al. [13] to learn over data structured as graphs. This approach has shown remarkable performance in many applications that can be represented using graphs and in which patterns are significant to solve the problem. The most common architecture is the Message Passing Neural Network (MPNN). Given a graph $G = (V, E)$, every node $v \in V$ is represented through a hidden state initialized as h_v^0 (a n-dimensional vector). For each iteration t of the process, a message-passing operation is performed, in which the hidden states of the nodes are combined with the hidden state of their neighbors. In particular, the message-passing operation is composed of two functions: an aggregation function $a(\cdot)$ that combines the received hidden states and an update function $u(\cdot)$ that combines the current hidden state of the node and the result of the aggregation.

Formally, the message-passing operation at iteration t is defined as follows:

$$m_{v,w} = m(h_v^t, h_w^t, e_{v,w}) \tag{1}$$

$$M_v^{t+1} = a(\{m_{v,w} | w \in N(v)\}) \tag{2}$$

$$h_v^{t+1} = u(h_v^t, M_v^{t+1}) \tag{3}$$

where h_v^t is the hidden state of node v at iteration t, $e_{v,w}$ is the edge that connects nodes v and w, $m_{v,w}$ is the message sent by w to v, M_v^t is the aggregated message received by v and $N(v)$ is the neighborhood of v.

After T iterations, it is possible to convert the whole graph into a single representation vector through a readout phase, that corresponds to a permutation invariant function applied to all the hidden states of the nodes:

$$y = r(h_v^T | v \in V) \tag{4}$$

Kipf and Welling [14] formulate a spectral-based multi-layer Graph Convolutional Network (GCN) represented by the following layer-wise propagation rule:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}) \tag{5}$$

where $\tilde{A} = A + I_N$ is the adjacency matrix of the undirected graph G with added self-connections, I_N is the identity matrix, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, $W^{(l)}$ is a layer-specific trainable weight matrix, $\sigma(\cdot)$ denotes an activation function and $H^{(l)} \in \mathbb{R}^{N \times D}$ is the matrix of activations in the l^{th} layer (starting with $H^0 = X$, where X is the input signal). This GCN formulation can be interpreted as a differentiable and parameterized generalization of the 1-dim Weisfeiler-Lehman algorithm on graphs [15].

IV. Graph Structure

Networking traffic can be considered as hierarchical organizations of flow entities, ranging from fine-level (packets) to coarse-level (communications among endpoints). Our proposed graph structure FTG is built from traffic data divided into time slots of size t_s and consists of a high-level Traffic Graph for each time slot in which each node has a corresponding low-level Flow Graph. Each of the two levels has its own GNN model to be processed in the FTG-Net.

A. Flow Graph

Flow Graphs are created drawing inspiration from the graph structure proposed in GraphDDoS [10]. All the packets exchanged between two endpoints in a time slot form a group, even if they belong to different networking flows (e.g., TCP flows). In each group, packets are sorted in ascending order of time. Each packet is converted into a node that has only the packet length as a feature. The upstream traffic, from client to server, is distinguished from the downstream traffic, from server to client, by setting the length of upstream as positive and the length of downstream as negative. The packets sent consecutively by one of the endpoints form a mini-group. Edges are added to adjacent nodes corresponding to packets of the same mini-group. The first packet of a mini-group is connected, using edges, to the first packet of the previous mini-group and to the one of the subsequent mini-group, as it happens among the last packets of mini-groups. An example is shown in Figure 1.

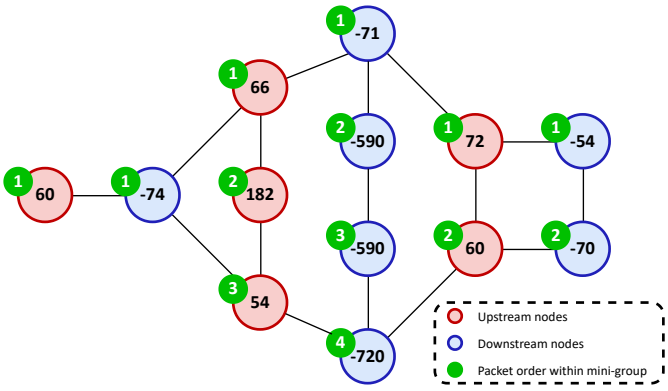

Fig. 1. Example of a Flow Graph. Upstream packets have positive packet length features, whereas downstream packets have negative packet length features. The mini-groups sequence is sorted from left to right.

This structure allows observing packet relationships of a single flow, that in certain DDoS attacks present a behavior different from legitimate traffic, and relationships of multiple flows between the same endpoints, that allow detecting Burst Information and Periodic Information. Burst Information corresponds to the scenario in which the attacker sends a large number of packets to establish many connections with the victim on different ports. Periodic Information, instead, refers to low-rate attacks in which the attacker mimics periodically a normal client to occupy the resources of the victim as long as possible.

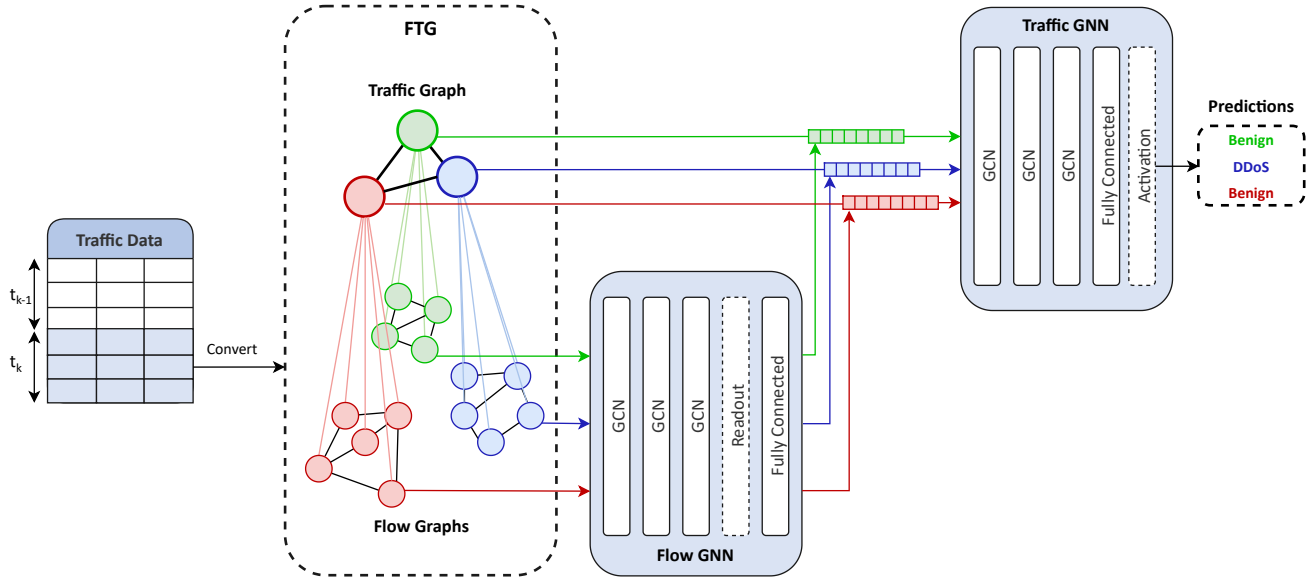

Fig. 2. Hierarchical model architecture of the FTG-Net solution.

B. Traffic Graph

A single Traffic Graph is built for each time slot. Each couple of endpoints that appears at least in a packet from traffic data of the corresponding time slot is converted in a graph node, whose features are given by the Flow GNN output vector that embeds the related low-level Flow Graph. An edge is added between two nodes when they have the source IP address or the destination IP address in common. The set of features that describes the communication between two endpoints can be enriched by concatenating statistics that provide further information according to the requirements and capabilities of a real-world scenario. The objective of this work is to show that the traffic topology is enough to detect DDoS attacks with performance in line with state-of-the-art solutions.

The Traffic Graph structure is designed to include and combine in a single graph, that can be processed by traditional GNN Layers, interactions among different flows that show the distribution of devices sending requests to servers, and flow-specific behaviors that contain malicious attack patterns. How to encode those flow-level patterns is directly coupled with the traffic-level topology because, during the training phase, the weights of the Flow GNN are optimized in the same backpropagation steps as the weights of the Traffic GNN.

V. Model Architecture

The architecture of FTG-Net, shown in Figure 2, is designed to process the multi-level graph representation FTG. It is divided into three steps. In the first step, traffic data is converted into FTG structures. In the second step, the Flow-level Graphs are processed by the Flow GNN producing an embedded representation vector for each graph. Those vectors are used as node features in the Traffic-level Graphs, which are processed in the third step by the Traffic GNN, outputting the final predictions for each flow (i.e., communication between two endpoints in a time slot).

A. Traffic Converting

In this step, traffic data is divided into time slots and for each time slot a Traffic Graph Gt and a list of Flow Graphs $F = \{Gf_1, Gf_2, ..., Gf_N\}$, whose indexes correspond to node indexes in the Traffic Graph and in which N depends by the time slot, are extracted according to section IV. During the training phase, a list of labels $Y = \{y_1, y_2, ..., y_N\}$ (one label for each Flow Graph) is coupled to each time slot data to perform supervised learning.

To extract the graphs, traffic data is analyzed in ascending order according to time. At the beginning of a new time slot, an empty Traffic Graph is created and an empty sorted list of encountered couples of endpoints is initialized. For each packet, if it is the first time that its combination of endpoints is encountered, then a new node is added to the Traffic Graph and a new Flow Graph containing only the current packet is created. During this phase, the direction of the packet is kept because it will be used to divide the packets into mini-group. If the combination of endpoints is already present in the list, then only the corresponding Flow Graph is updated with the current packet. If its direction is different from the last added packet, then the mini-group is closed and connected through edges as explained in section IV-A. At the end of the time slot, the edges in the Traffic Graph are added according to section IV-B.

B. Flow GNN

The Flow GNN is used to process Flow Graphs and build embeddings that will be used as node features in the Traffic Graph. First, three GCN Layers, defined as described in section III, are applied to the Flow Graph, spreading node

information in the three-hop neighborhood of each node. Each GCN Layer is followed by a Rectified Linear Unit (ReLU) activation function.

Then, a readout function is applied to transform the graph into a single vector. In particular, the applied readout function is the global mean pooling, that averages node features across the node dimension. It is defined as

$$h_G = \frac{1}{N} \sum_{n=1}^{N} h_n^{(3)} \tag{6}$$

where N is the number of nodes in the Flow Graph and $h_n^{(3)}$ is the n-th node feature vector after three iterations of GCN. The readout output vector is passed to a Fully Connected Layer, producing the final output.

C. Traffic GNN

The Traffic GNN takes as input a Traffic Graph to output a prediction (legitimate or malicious traffic) for each node. It is composed of three GCN Layers followed by a Fully Connected Layer, with a 50% dropout, that is individually applied to the features of each node. The single outputs are passed to a sigmoid activation function that provides the final scores in the range $[0, 1]$.

VI. Experimental results

In this section the experimental results are presented to assess the performance of the proposed FTG-Net model.

A. Experimental Setup

- Running environment: The experiments were run on a Ubuntu 18.04.5 LTS workstation with Intel Xeon Gold 6244 3.60GHz processor and NVIDIA Tesla v100 32GB graphics card. The main libraries used to implement FTG-Net are PyTorch, PyTorch Geometric, NumPy, Scapy and Pandas.
- Dataset: Our experiments are performed on the Canadian Institute of Cybersecurity Intrusion Detection System Dataset (CIC-IDS2017) [5]. It covers common attacks such as DoS, DDoS, Brute Force, XSS, SQL Injection, Infiltration, Port Scan and Botnet. The dataset is labelled with more than 80 traffic features collected using CICFlowMeter. The capturing period started at 09:00 on Monday and ended at 17:00 on Friday. The DDoS attacks were performed on Friday afternoon and are the only ones that are significant for this work. Traffic data have been converted according to section V-A considering only time slots with at least 20% of labelled data.
- Evaluation Metrics: The proposed method consists of a binary classification task, in which indicators are established through a confusion matrix and in which it is possible to formulate the following metrics:

$$accuracy = \frac{x_{correct}}{x_{total}} = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

$$recall = \frac{TP}{TP + FN} \tag{8}$$

$$precision = \frac{TP}{TP + FP} \tag{9}$$

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{10}$$

Often, the malignant class is dangerous and recall becomes the main metric. However, there are classification problems in which it is not possible to state that one class is more important than another one. This is the case for DDoS Detection, in which legitimate traffic should not be blocked (also depending on use case) and a few packets from attacks may not consume all the server resources. In those scenarios, it is common to combine a metric computed considering each class as positive and averaging the results. Weighted F1-Score is a metric in which F1-Scores are combined by weighted-averaging using the number of samples of the positive classes as weights.

B. Results

To evaluate the performance of FTG-Net, we used a 5-fold cross-validation that randomly divides the dataset into five chunks and, for each chunk, performs a training phase considering the current chunk as test set and the others as training set. We set the time slot size to 5 seconds, a 8-dimensional vector as output of the Flow GNN and as feature vector for Traffic Graph nodes, and 64 hidden channels in output to the GCN Layers. The nodes of the Flow Graphs are initially composed only by one element, corresponding to the packet length. The results obtained by combining the best model for each training phase of the cross-validation can be seen in table I. They are in line with the other state-of-the-art results, but with the advantage of considering just the network structure instead of collecting several stateful features.

TABLE I
Comparison of Results

Method	Accuracy	F1-score
LUCID	0.9967	0.9966
GLD-Net	0.9940	0.9920
GraphDDoS	0.9959	0.9959
FTG-Net	0.9914	0.9913

In our approach, the time slot size requires to be carefully configured. A large time slot allows to have a wider view of the traffic topology, that contains more information to detect potential DDoS attacks. However, the time slot size is inversely proportional to the responsiveness of the system, that could not be enough to promptly detect attacks before damages. Moreover, with large Traffic Graphs the inference time increases. A small time slot may not carry enough information to robustly classify traffic. We evaluate the performance in terms of average inference time and weighted F1-Score of four different time slot sizes, using a random split that inserts 70% of data in the training set and the remaining 30% in the test set. The results can be observed in Figure 3 and confirm the previous reasoning.

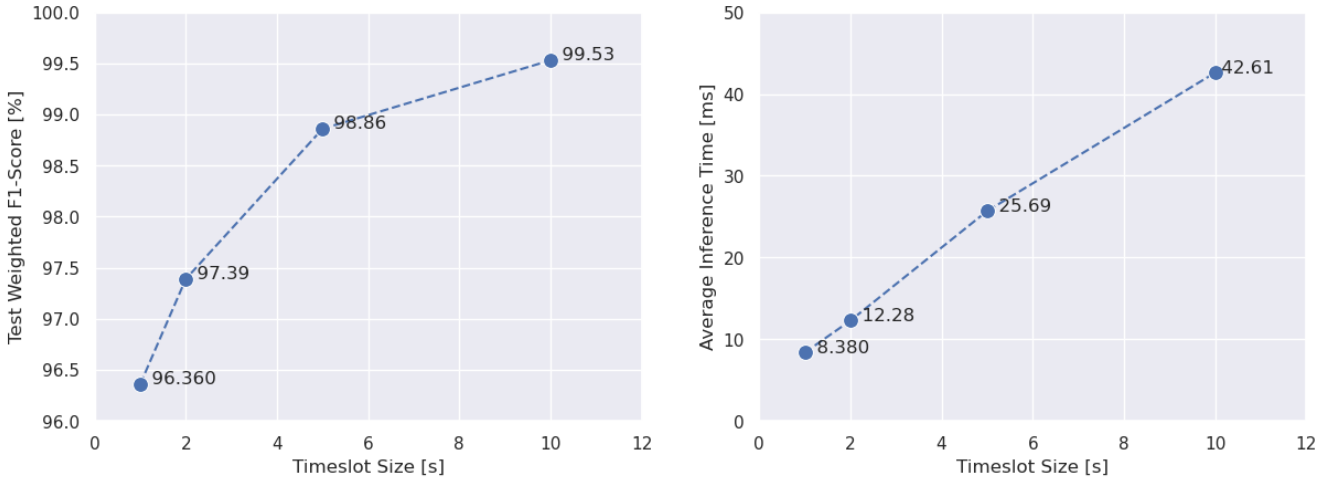

Fig. 3. Weighted F1-Score and Average Inference Time results using different time slot sizes.

VII. Conclusions

In this paper we proposed FTG-Net, composed of a novel hierarchical traffic representation and a hierarchical Graph Neural Network model to robustly detect DDoS attacks, leveraging topological information at both traffic-level and flow-level and combining them, which is a necessary process to capture significant structural patterns. The main advantage of this approach, compared to solutions that fuse traffic high-level topology with statistical flow features, is that the Flow GNN and Traffic GNN architectures are strictly coupled by sharing the same training phase. We evaluated the performance of this approach on the CIC-IDS2017 Dataset to prove that traffic structure is sufficient to obtain results comparable with state-of-the-art approaches and stateful features can be avoided. The success of our methodology may inspire the exploration of different data representations based on the same principles, considering also resource- and time-constrained real-world scenarios that may take advantage of distributed and quantized lightweight solutions.

Acknowledgment

This work has been partially supported by the Horizion Europe SMARTEDGE Project, funded by the European Commission under grant agreement No. 101092908. The work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of the NextGenerationEU partnership on "Telecommunications of the Future" (PE00000001, program "RESTART").

References

[1] Internet Crime Complaint Center IC3, "FBI Internet Crime Report 2021," 2021.

[2] Kaspersky Lab ZAO, "DDoS attacks in Q2 2022," August 2022.

[3] L. Barsellotti, F. Alhamed, J. J. Vegas Olmos, F. Paolucci, P. Castoldi, and F. Cugini, "Introducing data processing units (DPU) at the Edge," in *2022 International Conference on Computer Communications and Networks (ICCCN)*, 2022, pp. 1–6.

[4] K. Rusek, J. Suárez-Varela, A. Mestres, P. Barlet-Ros, and A. Cabellos-Aparicio, "Unveiling the potential of Graph Neural Networks for network modeling and optimization in SDN," in *Proceedings of the 2019 ACM Symposium on SDN Research*. ACM, apr 2019.

[5] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *4th International Conference on Information Systems Security and Privacy (ICISSP)*, vol. 1, pp. 108–116, 2018.

[6] F. Musumeci, A. C. Fidanci, F. Paolucci, F. Cugini, and M. Tornatore, "Machine-learning-enabled ddos attacks detection in p4 programmable networks," *Journal of Network and Systems Management*, vol. 30, pp. 1–27, 2022.

[7] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martínez-del Rincón, and D. Siracusa, "Lucid: A practical, lightweight deep learning solution for ddos attack detection," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 876–889, 2020.

[8] D. Pujol Perich, J. R. Suárez-Varela Maciá, A. Cabellos Aparicio, and P. Barlet Ros, "Unveiling the potential of graph neural networks for robust intrusion detection," in *3rd International Workshop on AI in Networks and Distributed Systems*, 2021, pp. 1–7.

[9] W. Guo, H. Qiu, Z. Liu, J. Zhu, and Q. Wang, "GLD-Net: Deep Learning to Detect DDoS Attack via Topological and Traffic Feature Fusion," *Computational Intelligence and Neuroscience*, vol. 2022.

[10] Y. Li, R. Li, Z. Zhou, J. Guo, W. Yang, M. Du, and Q. Liu, "GraphDDoS: Effective DDoS Attack Detection Using Graph Neural Networks," in *2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, 2022, pp. 1275–1280.

[11] P. Pati, G. Jaume, L. A. Fernandes, A. Foncubierta-Rodríguez, F. Feroce, A. M. Anniciello, G. Scognamiglio, N. Brancati, D. Riccio, M. Di Bonito, G. De Pietro, G. Botti, O. Goksel, J.-P. Thiran, M. Frucci, and M. Gabrani, "Hact-net: A hierarchical cell-to-tissue graph neural network for histopathological image classification," in *Uncertainty for Safe Utilization of Machine Learning in Medical Imaging, and Graphs in Biomedical Image Analysis*. Cham: Springer International Publishing, 2020, pp. 208–219.

[12] P. Pati, G. Jaume, A. Foncubierta-Rodríguez, F. Feroce, A. M. Anniciello, G. Scognamiglio, N. Brancati, M. Fiche, E. Dubruc, D. Riccio, M. Di Bonito, G. De Pietro, G. Botti, J.-P. Thiran, M. Frucci, O. Goksel, and M. Gabrani, "Hierarchical graph representations in digital pathology," *Medical Image Analysis*, vol. 75, p. 102264, 2022.

[13] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[14] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[15] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels," *Journal of Machine Learning Research*, vol. 12, no. 9, 2011.