Feasibility of State Space Models for Network Traffic Generation

Andrew Chu*
University of Chicago

Xi Jiang*
University of Chicago

Shinan Liu
University of Chicago

Arjun Bhagoji
University of Chicago

Francesco Bronzino
École Normale Supérieure de Lyon

Paul Schmitt
University of Hawaiʻi at Mānoa

Nick Feamster
University of Chicago

Abstract

Many problems in computer networking rely on parsing collections of network traces (e.g., traffic prioritization, intrusion detection). Unfortunately, the availability and utility of these collections is limited due to privacy concerns, data staleness, and low representativeness. While methods for generating data to augment collections exist, they often fall short in replicating the quality of real-world traffic In this paper, we i) survey the evolution of traffic simulators and generators and ii) propose the use of state space models, specifically Mamba, for packet-level, synthetic network trace generation by modeling it as an unsupervised sequence generation problem. Preliminary evaluation shows that state space models can generate synthetic network traffic with higher statistical similarity to real traffic than the state-of-the-art. Our approach thus has the potential to reliably generate realistic and informative synthetic network traces for downstream computer networking tasks.

CCS Concepts

• **Networks** → **Network simulations**; • **Computing methodologies** → *Neural networks*.

Keywords

State space models, Network trace generation

ACM Reference Format:
Andrew Chu, Xi Jiang, Shinan Liu, Arjun Bhagoji, Francesco Bronzino, Paul Schmitt, and Nick Feamster. 2024. Feasibility of State Space Models for Network Traffic Generation. In *SIGCOMM Workshop on Networks for AI Computing (NAIC '24), August 4–8, 2024, Sydney, NSW, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3672198.3673792

1 Introduction

The increase in complexity of networked environments and the number of network protocols and applications motivates robust and adaptable strategies for network management and security tasks. This requires collecting and sharing high-quality statistics, attributes or features of this data, or further, the raw data itself, which can be tedious or even impossible under certain constraints [1, 9, 32, 33, 46]. Worse, networks are becoming increasingly difficult to simulate via traditional methods (*e.g.*, NS-3 [24], Harpoon [47]) as the scale and complexity of networked systems grows [21, 23, 41]. One approach to tackle these challenges is synthetic traffic generation, which provides a controlled method to model, simulate, and test network behaviors for various traffic types, without the pitfalls of real-world data collection.

Current state-of-the-art (SOTA) synthetic traffic generation methods apply one of two approaches. The first generates concise sets of meta-attributes (*i.e.*, flow statistics) for a network trace. These attributes capture overarching characteristics of network flows, and are primarily used for session-level workloads (*e.g.*, heuristic-based analyses, ML for networking). In contrast, the second generates complete, raw network data at the per-packet level in packet capture (PCAP) form. This approach, offering a comprehensive view of network communication, can be more versatile for downstream tasks (*e.g.*, analyses of session content, classification models).

Unfortunately, both approaches face a common challenge in that the quality of generated data often falls short of expectation, reflected in low statistical similarity at the raw-byte level to real network data. Thus, generated traces may require tedious post-generation correction to ensure protocol fidelity before being usable as either a replacement for, or supplement to, real network data. Motivated to remedy these challenges, we explore how recent work in sequence modeling might help in generating high-quality, raw synthetic network traffic. In this paper, we treat the task of generating synthetic network traffic as a sequence modeling problem by applying a state-space model (SSM) built on the Mamba architecture [10] to networking data. Unlike transformer models (the predominant SOTA for sequence modeling) whose time and memory complexity scale quadratically with input length, Mamba scales linearly. This allows our model to learn from, and generate network traces of both better quality, and longer length than the existing SOTA. Specifically, we present the following contributions:

(1) **Survey of trace generation paradigms:** We provide a detailed overview of the evolution of synthetic traffic generators, their impact, and possible takeaways for motivating future trace generation work.
(2) **Apply SSMs to networking data for traffic generation:** We frame the task of synthetic network trace generation as an unsupervised sequence generation problem, and train a Mamba-based SSM model from scratch on computer networking data. We then demonstrate our model's ability to generate synthetic traces at the raw-byte (*i.e.*, PCAP) level.

*Equal contribution.

NAIC '24, August 4–8, 2024, Sydney, NSW, Australia

ACM ISBN 979-8-4007-0713-1/24/08
https://doi.org/10.1145/3672198.3673792

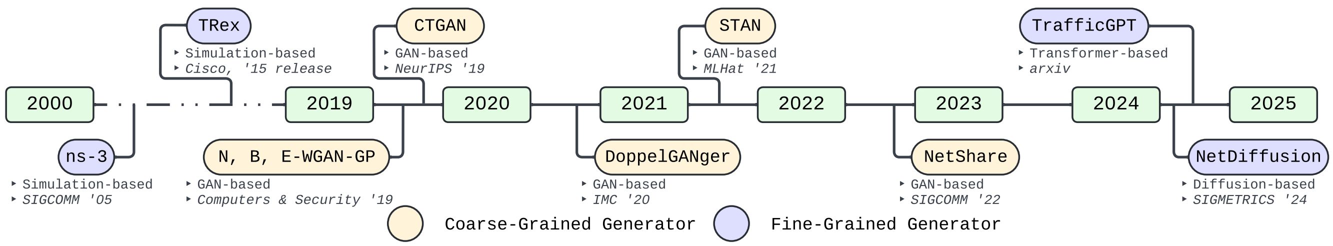

Figure 1: Timeline of synthetic network trace generation methods.

(3) **Superior trace generation quality and efficiency:** We evaluate traces generated by our model and find they better capture complex intra-packet dependencies with additionally higher fidelity, than existing SOTA approaches. Our model pipeline further has a lower training and inference resource footprint than existing, comparable approaches.

Building on our preliminary results, we provide a roadmap for improving the generation quality and flexibility of our model. We additionally discuss possible adaptations of our model for use in various downstream networking tasks.

2 Related Work

Early efforts in synthetic data generation such as TRex [7], NS-3 [14], and others [4, 6, 47, 49], are mostly simulation-based, requiring extensive human expert efforts to correctly build and configure. These tools can be limited as they depend on user-defined templates and may not necessarily capture the complex *intra-flow* (*i.e.*, relationships between header fields across multiple packets) and *intra-packet* (*i.e.*, header field dependencies within individual packets) interactions reflective of complex real-world traffic. Recent methods [16, 17, 27, 40, 55, 56, 58, 59] employ generative ML to learn such interactions, enabling the automatic generation of new synthetic traces. These approaches can generally be divided into two categories: *coarse* and *fine-grained*. Figure 1 provides a chronological overview of the approaches discussed in this section in either category, further discussed below.

2.1 Coarse-Grained Generators

Coarse-grained generators aim to synthesize meta-information or summary statistics (*e.g.*, flow duration and average packet sizes) about network traffic flows (i.e., intra-flow dependencies). Such tools [27, 40, 55, 56, 58] mainly rely on generative adversarial networks (GANs), training two competing neural networks against each other for synthesizing new network traces from a training dataset. A notable example is Lin et al.'s DoppelGANger [27], which treats modeling network metric traces as a sequential time-series generation task, conditioned on metadata like locations. This allows any single header value or intra-flow metrics to be captured for modeling. Unfortunately, this approach scales poorly, requiring additional training for each new value that is desired to be captured. Yin et al.'s NetShare [58] improves by generating aggregate flow statistics (*e.g.*, duration, packet count), or packet-level header field values (*e.g.*, time-to-live [TTL], protocol flags). While effective, it is limited to a small, fixed subset of high-level network attribute, missing granular details like intra-flow dependencies (*e.g.*, TCP options and sequence numbers) and intra-packet dependencies (*e.g.*, TCP header bits influencing data offset). Thus, coarse-grained generators often fail to capture these low-level dependencies that are necessary for accurately mimicking real data.

2.2 Fine-Grained Generators

Fine-grained generators aim to capture both intra-flow and intra-packet dependencies of network traffic by generating complete packet captures, *i.e.*, all raw packet bytes, to most accurately mimic real-world traffic for downstream use.

Diffusion-based Approaches. Jiang et al.'s NetDiffusion [16] uses image representations of network traces with a text-to-image diffusion model to perform fine-grained trace generation. Diffusion models are highly expressive, resulting in synthetic traces that more accurately mimic real network dynamics. Unfortunately, this expressiveness can produce noisy outputs that may compromise protocol compliance. The authors attempt to mitigate this concern using ControlNet [60] to impose dependencies as conditional controls during generation. This improves resulting trace protocol compliance, but overall is insufficient. While ControlNet may ensure that the distribution of packet header fields values generated largely adhere to the protocol used by the flow, it cannot guarantee the semantic accuracy of these fields, often requiring extensive post-generation manual corrections.

Transformer-based Approaches. Transformer-based models [48] leverage the attention mechanism [2] to learn a fine-grained representation of the semantics of their input data in an unsupervised fashion (encoder), and may be extended to use this representation for autoregressive generation (decoder). Many works have applied the transformer to networking tasks (e.g., traffic classification [8, 13, 26, 43, 44], networking specific Q&A [59]). However, few have used the transformer for fine-grained traffic generation. Meng et al.[34] and Wang et al. [51] train new variants of GPT-2 [37] and T5 [38] respectively, for coarse-grained generation at the header field value level (*i.e.*, similar to DoppelGANger). Qu et al.'s TrafficGPT [36] is the most related to our work, presenting a GPT-2 variant that generates trace sequences up to 12,032 tokens in length via various memory optimization strategies such as linear and local window attention [20, 54], etc [22, 35]. Our Mamba-based modeling approach differs from TrafficGPT as it provides improved training time complexity and maximum generatable context length, which is particularly critical in downstream applications where the analysis of flow-level attributes over extended durations is essential (*e.g.*, malware/intrusion detection). Additionally, traces generated by our architecture demonstrate better quality, evidenced by a higher statistical resemblance to real data.

3 State Space Models and Traffic Generation

We adapt Mamba [10], a selective structured SSM, to synthesize fine-grained network traces that capture both the intra-flow and intra-packet dependencies of network traffic. We train a new Mamba model from scratch on the raw byte values of packets from real-world flows, to generate synthetic packets of continuous traffic traces. We first provide background information on SSMs and our motivation for using SSMs to generate high-quality synthetic traces in Section 3.1 and 3.2. We then provide a technical overview of how the general, and Mamba SSM[1] operate in Section 3.3.

3.1 State Space Models

SSMs are probabilistic graphical models that build on the concept of a state space from control engineering [19]. Conceptually, SSMs are identical to Hidden Markov Models in objective (modeling discrete observations over time), but operate using continuous, instead of discrete latent variables. Instead of attention (the learning mechanism used in transformer-based models such as GPT), SSMs encode a hidden state, representative of prior observed context of an input sequence, using recurrent scans. Gu et al. [11] and Voelker et al. [50] showed that by fixing the state matrix used in SSMs, the encoded context can accurately and efficiently model long-range dependencies. Gu et al. extended these observations in the S4 convolution kernel [12] to make SSMs practical for training, and more recently in Mamba [10], where they introduce a SSM with a selection mechanism, and fixed state matrix. Mamba shows strong performance in context dependent sequence generation (*i.e.*, language modeling), and has been extended to various domains, including computer vision [29, 30, 53], DNA sequencing [42], document information retrieval [57] and speech separation [25].

Although general SSMs have long existed in the control engineering space, they have only recently been optimized and operationalized by Gu et al. [10–12] in the Mamba architecture to function as a possible alternative to transformer-based approaches for sequence modeling. We find only one recent application of Mamba to the networking domain by Wang et al.[52], who create a variant, NetMamba, to perform traffic classification. NetMamba claims better classification accuracy than the existing SOTA, with improved inference time and resource consumption. Our objective differs from NetMamba, as we explore what modifications/adaptations are needed to use Mamba for synthesizing raw data for network traces, rather than performing traffic inference tasks.

3.2 Comparison with alternative methods

We chose the Mamba SSM architecture over other fine-grained generation methods for a number of reasons. Compared to diffusion models, the tokenized input used by SSMs allows for a less processed representation of networking data. Specifically, our model processes sequences of the decimal values for the raw bytes of flows, as compared to an encoded image representation in NetDiffusion derived using two steps (nPrint [15] intermediary format, bit-based color assignment). Compared to transformer-based models, SSMs scale linearly (versus quadratically) with sequence length. This allows our models to train on inputs four times longer and generate sequences 5.5 times longer than TrafficGPT [36][2]). This is especially useful for networking data, where even a few seconds of communication between nodes can exceed the conventional context windows or capacities of other methods.

[1]For the rest of the paper, we will use "Mamba" and "Mamba SSM" interchangeably.

3.3 Selective Structured SSM (Mamba)

Mamba, and broadly, all SSMs, use the state space representation introduced in control engineering by Kalman [19]. The general SSM uses first-order ordinary linear differential equations to capture the relationship (output) between unobserved variables (state) and a series of continuous observations (input), irrespective of time (*i.e.*, is linear-time invariant). As the model observes more data, it encodes a representation of the state that captures the prior context of inputs. This state is then used to calculate an output for a given input, and can be both discretized to be calculated as a recurrent neural network in linear time, and unrolled to a convolutional neural network for efficient training. On this basis, Mamba implements two changes to the general SSM that provide *structure* and *selection*.

For structure, the state of general SSMs suffers from numerical instability similar to the vanishing gradient in vanilla recurrent neural networks, where successive compression of prior state context results in poor model performance. Mamba solves this instability by enforcing structure on the general-case SSM state matrix (typically randomly initialized), replacing it with a HiPPO matrix [11] from prior work. The HiPPO matrix introduces a probability measure that dictates how the SSM state is compressed, improving the ability of an SSM to model long-range dependencies in sequences.

For selection, the linear-time invariant nature of the general SSM lacks expressiveness, *i.e.*, all discrete inputs compressed in the state, affect the state with equal weighting. In the context of language modeling, this prevents relevant "keywords" from more heavily influencing the SSM state to develop a better semantic understanding of input. Mamba improves SSM expressiveness by removing the linear-time invariant quality from the general SSM, making the model time-*variant*. Through this, the state is calculated using learned (as compared to fixed) functions of the inputs.

Mamba's structure and selection changes result in competitive performance against transformer-based approaches for sequence modeling with regard to generation quality, while simultaneously providing better scaling (linear versus quadratic).

4 Modeling Network Data with State Space Models

We apply the Mamba architecture to model computer networking data to generate synthetic traces. Figure 2 provides an overview of our modeling pipeline. We detail our steps taken to adapt Mamba for use with networking data below.

Trace Pre-processing and Tokenization. We train our generation model on a tokenized representation of raw PCAP data at the flow level. To detail, we first split each PCAP into its comprising

[2]Training/generation hardware: NVIDIA A40, 48GB VRAM (ours), NVIDIA Tesla V100S, 32GB VRAM (theirs).

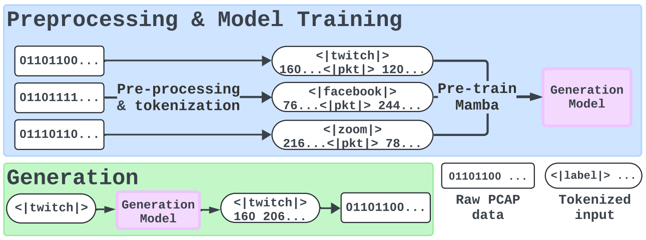

Figure 2: Model training and generation process.

flows based on connection (*i.e.*, 5-tuple of source IP, source port, destination IP, destination port, IP protocol) using `pcap-splitter` [45], resulting in a set of n flows *i.e.*, PCAP $= \{f_n\}_{n\in\mathbb{N}}$, where f denotes a flow. We then parse each packet in a flow to its sequence of raw bytes in decimal format *i.e.*, packet $= \mathrm{Seq}(f_n)_i = \{(f_n)_i[1]\} \cup \{(f_n)_i[2]\} \cup \ldots \{(f_n)_i[j]\}$, where i denotes the packet index in flow f_n, j denotes the byte-offset in packet i, and $\{(f_n)_i[j]\} \in [0, 255]$. Next, we join these packet sequences, delimiting them with a custom `<|pkt|>` special token to form a training sample. Finally, we prepend each sample with a `<|LABEL|>` special token that denotes its traffic type. Thus, sample $= \{$`<|LABEL|>`$, \mathrm{Seq}(f_n)_1,$ `<|pkt|>`$, \mathrm{Seq}(f_n)_2, \ldots, \mathrm{Seq}(f_n)_m\}$, where m denotes the number of packets in a given flow. We train a new tokenizer of the same base-type as Mamba on these representations (GPT-NeoX-20B [3]), adding the `<|pkt|>` and `<|LABEL|>` special tokens.

Model Implementation. We use the open sourced model implementation released by Gu et al. accompanying the original Mamba work [10]. Specifically, this combines the Mamba SSM described in the previous section with a gated multilayer perceptron [28], to form the Mamba block. Appendix A.1 provides further details on the architecture and its functionality. The Mamba block operates in a causal (left-to-right) manner, and trains using the common next token prediction task evaluated via cross entropy loss.

Trace Generation. Our model generates synthetic network traces by taking in two arguments: a generation *seed* and *length*. The seed is a sequence consisting of a flow's label token, and sequence of raw bytes that comprise its first packet (*e.g.*, `<|twitch|> 205 68 37...`), and is equivalent to a start prompt in NLP generative models. The generation length dictates the maximum number of tokens output by the model. The output of the model resembles the format of the training data, a sequence of raw bytes of packets in decimal format, with each packet delimited by a `<|pkt|>` special token. We then convert this raw, string-based output to binary format for use and evaluation as a PCAP file.

5 Preliminary Evaluation

We assess the effectiveness of our approach by examining the *packet-level, fine-grained generation quality* of traces generated by our model. Specifically, we perform or examine:

(1) Flexibility of our model to generate traces of varying lengths without quality degradation.
(2) Statistical resemblance of synthetic traces generated by our model to real-world network data.
(3) Empirical verification that our model learns, rather than memorizes the interactions in network data.

Content Type		Size	
Macro-Label	**# Micro-Labels**	**Raw**	**Flows**
Video Streaming [5]	4	6.36 GiB	10,032
Video Conferencing [31]	3	17.36 GiB	13,911
Social Media [18]	3	5.40 GiB	3,896

Table 1: Service-classification dataset overview.

We describe the datasets used for this preliminary evaluation, implementation specifics, training recipes, and results below.

5.1 Experiment Setup

Dataset Description. For generation quality assessment, it is essential to use network trace datasets thoroughly cleaned to reduce noise, *i.e.*, unintended traffic capture. Statistical comparisons between real and synthetic traffic are more credible and less variable with clean datasets, which also enhance the effectiveness of traffic analysis tools in downstream applications. However, many public datasets are unsuitable due to high noise levels. For our experiment, we use three representative, labeled datasets previously used in similar studies for synthetic trace generation, comprised of video streaming [5], video conferencing [31], and social media data [18], as detailed in Table 1. Each dataset contains traffic from its respective application domain, divided into flows in PCAP format. We further filter these flows to ensure their relevance to the labeled application domain by extracting relevant DNS queries, resolving them to IP addresses, and filtering the packets based on these addresses. We aggregate these datasets to form a *service-classification dataset*, which we use to both train and evaluate our model.

Pre-training configuration. We pre-train our model on this service-classification dataset using a single NVIDIA A40 48GB PCIe GPU, with gradient clip value of 1.0 and the AdamW optimizer with learning rate $5e-4$. We leave all other optimizer values as their defaults ($\beta = (0.9, 0.999)$, $\epsilon = 1e-8$, weight decay $= 1e-2$). We use the same configuration for dimension (768) and number of layers (24) as the smallest 125 million parameter pre-trained Mamba. We extend the base Mamba tokenizer to include the delimiting (`<|pkt|>`) and ten label special tokens for the traffic in the dataset. We train the generation model following the process discussed in Section 4, first splitting all PCAPs into their comprising flows, parsing them into decimal representations, and finally tokenizing these inputs. We run pre-training using batch size of one, with maximum sequence length of 50,000 tokens (the largest possible given our hardware and configuration; training samples longer than this length are truncated), for 50 epochs, until cross-entropy loss converges at 1.32 nats. Here, we minimize batch size and maximize per sample maximum sequence length, to have the model learn from as large of individual contexts of sequential packets in a flow as possible.

5.2 Evaluating generation quality

Our evaluation focuses on our model's ability to accurately reproduce intra-packet and intra-flow dependencies observed in real traces. Specifically, we aim to generate synthetic traffic that closely resembles real traffic dynamics at the packet level. The conventional empirical metric for assessing this quality, as noted in prior studies

[16, 36, 58], involves measuring the statistical similarity between the raw values of generated packet header fields and those from real traces. Below, we detail and discuss our generation process, benchmarks, and comparative results.

Conditioned Trace Generation. We generate synthetic network flows by using the first packet from each flow in the service-classification dataset as the seed to create the corresponding synthetic trace, as depicted in Figure 2. Recall that each packet begins with its corresponding label special token (*e.g.*, <|twitch|>, <|youtube|>, <|zoom|>) providing the trace type, to indicate to the model the type of traffic to generate. Therefore, a typical seed may resemble <|twitch|> 160 206 48... <|pkt|>, giving the model contextual cues for accurate synthetic trace generation. We set the generation length to be 10 tokens greater than the length of the real tokenized trace to ensure the synthetic versions have similar packet count, and to allow flexibility for slight variations and adjustments (*e.g.*, truncating malformed packets).

Benchmark Setup. We benchmark output traces generated by our model against representative SOTA network trace generators: NetShare [58] for coarse-grained and NetDiffusion [16] for fine-grained generation. Both models are trained on the service-classification dataset, producing corresponding synthetic traces. For NetDiffusion, a post-generation correction heuristic is required to ensure semantic coherence in generated traces, which our approach *does not* need. We report the NetDiffusion results after applying the correction heuristic. We also use the statistical similarity metrics reported by TrafficGPT to compare against the latest transformer-based architecture for fine-grained trace generation. Our comparison is based on their published results, as TrafficGPT is not open source and uses different data. Note that TrafficGPT focuses on a limited set of features (*e.g.*, IP addresses, ports), whereas the benchmark for NetShare, NetDiffusion, and our model consider similarity across *all* header values of the generated flow. Thus, although we cannot retrain TrafficGPT on our dataset, we acknowledge that its reported results likely showcase its optimal performance.

Generated Trace Length. We verify that our model can generate long traces while maintaining high quality. We observe the longest trace generated by our model has length of 1,081 packets (65,943 tokens), improving slightly on NetDiffusion's max trace generation length (1,024), and is 5.5 times longer than TrafficGPT's max sequence generation length (12,032 tokens). Appendix Table 1 provides more detail on the length of traces generated by our model. Recall that our trace generation process prompts our model with a generation length only 10 tokens longer than the length of the original trace from which the generation seed was taken. As such, we do not examine the maximum possible length of traces generated by our model in this evaluation. In theory, however, this value can be significantly larger (depending on traffic type) due to the linear scaling of the architecture than presented in this study. We leave a more detailed evaluation of this aspect to future work.

Statistical Similarity. We assess the statistical similarity between the generated traces and their respective ground truth representations using the average Jensen-Shannon Divergence (JSD), Total Variation Distance (TVD), and Hellinger Distance (HD) metrics across all header field values. Lower scores in these metrics represent higher statistical generation fidelity compared to the real traces. To ensure uniformity across comparisons, all synthetic and real traces are converted to a one-hot encoded binary representation using the nPrint format [15]. We include a crude upper-bound benchmark as a baseline by synthesizing all values randomly, without any model training or heuristic application.

Data Format	Generation Method	JSD	TVD	HD
NetFlow	Random Generation	0.67	0.80	0.76
	NetShare	0.16	0.16	0.18
Raw PCAP	Random Generation	0.82	0.99	0.95
	NetDiffusion†	0.04	0.04	0.05
	TrafficGPT*	0.16*	N/A	N/A
	Ours	**0.02**	**0.01**	**0.02**

† Post-generation correction applied; * Results as reported in [36].

Table 2: Statistical similarity versus real traces (packet-level, fine-grained generation quality evaluation).

Table 2 shows that our model distinguishes itself by achieving the lowest JSD, TVD, and HD scores among all evaluated network trace generators. Compared to the coarse-grained NetShare generator, we generate finer-grained raw traffic with closer statistical resemblance to real network traffic, achieving consistently lower metric scores, even though NetShare generates simpler NetFlow attributes that are more likely to achieve higher statistical similarity. Further, when comparing our model to the fine-grained NetDiffusion method, our approach still leads by a notable margin in all three statistical similarity metrics. This is particularly significant considering these results are based on output without any post-generation correction, which NetDiffusion requires to ensure semantic coherence. Considering reported results from TrafficGPT, which also treats the trace generation task as a sequence modeling problem, our approach still yields better fine-grained generation quality, despite the fact that TrafficGPT's results are calculated on fields with relatively fixed and consistent values. Overall, our model's performance suggests a high degree of statistical fidelity to the original traffic patterns, mimicking real-world behavior.

Learning Versus Memorization. To verify our model's learning capabilities, we perform one-to-one byte-wise comparison of the first 100 packets in each of our generated traces, and their corresponding real-world packets, from which the generation seed is extracted. Remarkably, on average, only one packet is identical, with the other 99 being different. The single identical packet can be explained by our trace generation process, where we prompt our model with a seed consisting of the service label, and first packet of a real-world flow. Common to other sequence generation models, our model copies the seed as part of the output sequence. All other generated packets have variances that separate them from their real-world counterparts. Further, in these packets, the average percentage of differing bytes is 6.57%. We analyze the distribution of these bytes (detailed results in Appendix Table 2) and find that they largely manifest in fields non-essential to flow state, indicating our model can discern between fields in the seed that should remain fixed (*e.g.*, source, destination IP addresses), and fields that when changed, will not disrupt communication. These results show that our model effectively captures and generalizes underlying network patterns, rather than memorizing specific data.

6 Open Challenges

Targeted generation and improving output flexibility. Training a Mamba SSM-based generation model using all contents from all flow packets allows the model to learn the important semantic relationships and underlying dependencies in general flows, evident from the high levels of statistical similarity to real traffic and observed packet byte distribution in our evaluation. One useful future direction is to fine-tune our general model to create models that can accurately generate the distinct phases or components observed in authentic network traces. As different stages of a network flow or session exhibit unique patterns and/or behaviors, more targeted models could better capture the specific intra-packet and intra-flow dynamics of these segments, improving both the quality and flexibility of generated data. Such fine-tuned models could also be extended to model the specific workloads or dynamics of particular network types (*e.g.*, internal data centers, content delivery networks). This flexibility could be particularly useful for reducing the overhead of training and data generation for downstream applications requiring only a subset of flow data. Finally, additional adjustments to the trace generation process should be explored to examine if traffic characteristic attributes (*e.g.*, mix of application types) can be more explicitly conveyed to be enforced by the model.

Temporal generation in synthetic traces. Our current model produces raw byte values for packets across different network layers, but does not generate temporal information (*i.e.*, timestamps). Because timestamps are determined by packet capturing tools at the OS level and are not embedded in the raw packet bytes, they are not currently considered in the training/generation process for our model. Temporal information provides critical details for various tasks, such as traffic classification, and attack detection. Unfortunately, directly adopting the vanilla Mamba architecture (as done in this study) for temporal generation is insufficient; it is designed for generating sequential discrete values, whereas temporal information is continuous. Thus, to improve the practicality of this approach, the underlying Mamba/SSM architecture must be modified to generate in-tandem both discrete packet-level values, and continuous timestamp values.

Better evaluation of synthetic trace quality and utility. While statistical similarity is an important quality indicator, it does not guarantee that generated synthetic traces are practical and/or adhere to protocol constraints. To illustrate, consider the IP version header, which in practice should only take on one of two values: `0100` (IPv4) or `0110` (IPv6). A synthetically generated value of `0101`, though statistically close to the valid values, would likely result in a dropped packet if transmitted in the real-world. As such, it is crucial to further inspect these synthetic traces to ensure their coherence. Next, as mentioned in Section 5.2, the maximum generatable length of traces and their resulting quality, should also be examined. Finally, we also note the importance of evaluating the utility of generated traces in practice in common downstream tasks (*e.g.*, training and augmenting machine learning models on network traffic, verifying heuristic-based analysis tools).

7 Conclusion

In this work, we presented a new approach to synthetic network trace generation, posing the task as a unsupervised sequence generation problem using the Mamba state-space model architecture. Our preliminary evaluation demonstrates that this method produces synthetic traces with significantly higher statistical similarity to real-world traffic compared to current state-of-the-art techniques. We outline future improvements and applications for our model towards improving the realism and utility of synthetic network traces, for use in different downstream tasks.

This paper does not raise any ethical concerns.

Acknowledgments

We thank the anonymous reviewers at NAIC for their feedback and suggestions. This work has been supported in part by grants from ANR Project No ANR-21-CE94-0001 (MINT), the Fédération Informatique de Lyon through the INTERFERE project, the France and Chicago Collaborating in the Sciences program, and NSF CNS-2124393.

References

[1] Sebastian Abt and Harald Baier. 2014. Are we missing labels? A study of the availability of ground-truth in network security research. In *2014 third international workshop on building analysis datasets and gathering experience returns for security (badgers)*. IEEE, 40–55.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1409.0473

[3] Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, Usvsn Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. GPT-NeoX-20B: An Open-Source Autoregressive Language Model. In *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*. Association for Computational Linguistics, virtual+Dublin, 95–136. https://doi.org/10.18653/v1/2022.bigscience-1.9

[4] Alessio Botta, Alberto Dainotti, and Antonio Pescapé. 2012. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks* 56, 15 (2012), 3531–3547.

[5] Francesco Bronzino, Paul Schmitt, Sara Ayoubi, Guilherme Martins, Renata Teixeira, and Nick Feamster. 2019. Inferring streaming video quality from encrypted traffic: Practical models and deployment experience. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 3, 3 (2019), 1–25.

[6] Tobias Bühler, Roland Schmid, Sandro Lutz, and Laurent Vanbever. 2022. Generating representative, live network traffic out of millions of code repositories. In *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*. 1–7.

[7] ciscotrex2023 2024. The CISCO TRex Tool. https://trex-tgn.cisco.com/. [Online; accessed 31-May-2024].

[8] Jianbang Dai, Xiaolong Xu, Honghao Gao, Xinheng Wang, and Fu Xiao. 2022. SHAPE: A Simultaneous Header and Payload Encoding Model for Encrypted Traffic Classification. *IEEE Transactions on Network and Service Management* (2022).

[9] François De Keersmaeker, Yinan Cao, Gorby Kabasele Ndonda, and Ramin Sadre. 2023. A Survey of Public IoT Datasets for Network Security Research. *IEEE Communications Surveys & Tutorials* (2023).

[10] Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *ArXiv preprint* abs/2312.00752 (2023). https://arxiv.org/abs/2312.00752

[11] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. 2020. HiPPO: Recurrent Memory with Optimal Polynomial Projections. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/102f0bb6efb3a6128a3c750dd16729be-Abstract.html

[12] Albert Gu, Karan Goel, and Christopher Ré. 2022. Efficiently Modeling Long Sequences with Structured State Spaces. In *The Tenth International Conference*

on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net. https://openreview.net/forum?id=uYLFoz1vlAC

[13] Hong Ye He, Zhi Guo Yang, and Xiang Ning Chen. 2020. PERT: Payload encoding representation from transformer for encrypted traffic classification. In *2020 ITU Kaleidoscope: Industry-Driven Digital Transformation (ITU K)*. IEEE, 1–8.

[14] Thomas R Henderson, Mathieu Lacage, George F Riley, Craig Dowell, and Joseph Kopena. 2008. Network simulations with the ns-3 simulator. *SIGCOMM demonstration* 14, 14 (2008), 527.

[15] Jordan Holland, Paul Schmitt, Nick Feamster, and Prateek Mittal. 2021. New Directions in Automated Traffic Analysis *(CCS '21)*. Association for Computing Machinery, New York, NY, USA, 3366–3383. https://doi.org/10.1145/3460120.3484758

[16] Xi Jiang, Shinan Liu, Aaron Gember-Jacobson, Arjun Nitin Bhagoji, Paul Schmitt, Francesco Bronzino, and Nick Feamster. 2024. NetDiffusion: Network Data Augmentation Through Protocol-Constrained Traffic Generation. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 8, 1 (2024), 1–32.

[17] Xi Jiang, Shinan Liu, Aaron Gember-Jacobson, Paul Schmitt, Francesco Bronzino, and Nick Feamster. 2023. Generative, high-fidelity network traces. In *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks*. 131–138.

[18] Xi Jiang, Shinan Liu, Saloua Naama, Francesco Bronzino, Paul Schmitt, and Nick Feamster. 2023. AC-DC: Adaptive Ensemble Classification for Network Traffic Identification. *ArXiv preprint* abs/2302.11718 (2023). https://arxiv.org/abs/2302.11718

[19] Rudolph Emil Kalman. 1960. A new approach to linear filtering and prediction problems. (1960).

[20] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 5156–5165. http://proceedings.mlr.press/v119/katharopoulos20a.html

[21] Anthony Kenyon, Lipika Deka, and David Elizondo. 2020. Are public intrusion datasets fit for purpose characterising the state of the art in intrusion event datasets. *Computers & Security* 99 (2020), 102022.

[22] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=rkgNKkHtvB

[23] Craig Labovitz, S Iekel-Johnson, D McPherson, J Oberheide, and F Jahanian. 2010. Internet traffic and content consolidation. *77th Internet Engineering Task Force* (2010).

[24] Mathieu Lacage and Thomas R Henderson. 2006. Yet another network simulator. In *Proceedings of the 2006 Workshop on ns-3*. 12–es.

[25] Kai Li and Guo Chen. 2024. SPMamba: State-space model is all you need in speech separation. *ArXiv preprint* abs/2404.02063 (2024). https://arxiv.org/abs/2404.02063

[26] Xinjie Lin, Gang Xiong, Gaopeng Gou, Zhen Li, Junzheng Shi, and Jing Yu. 2022. ET-BERT: A Contextualized Datagram Representation with Pre-training Transformers for Encrypted Traffic Classification. In *Proceedings of the ACM Web Conference 2022 (WWW '22)*. ACM. https://doi.org/10.1145/3485447.3512217

[27] Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, and Vyas Sekar. 2020. Using gans for sharing networked time series data: Challenges, initial promise, and open questions. In *Proceedings of the ACM Internet Measurement Conference*. 464–483.

[28] Hanxiao Liu, Zihang Dai, David R. So, and Quoc V. Le. 2021. Pay Attention to MLPs. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.). 9204–9215. https://proceedings.neurips.cc/paper/2021/hash/4cc05b35c2f937c5bd9e7d41d3686fff-Abstract.html

[29] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, and Yunfan Liu. 2024. Vmamba: Visual state space model. *ArXiv preprint* abs/2401.10166 (2024). https://arxiv.org/abs/2401.10166

[30] Jun Ma, Feifei Li, and Bo Wang. 2024. U-mamba: Enhancing long-range dependency for biomedical image segmentation. *ArXiv preprint* abs/2401.04722 (2024). https://arxiv.org/abs/2401.04722

[31] Kyle MacMillan, Tarun Mangla, James Saxon, and Nick Feamster. 2021. Measuring the performance and network utilization of popular video conferencing applications. In *Proceedings of the 21st ACM Internet Measurement Conference*. 229–244.

[32] Matthew V Mahoney. 2003. Network traffic anomaly detection based on packet bytes. In *Proceedings of the 2003 ACM symposium on Applied computing*. 346–350.

[33] John McHugh. 2000. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information and System Security (TISSEC)* 3, 4 (2000), 262–294.

[34] Xuying Meng, Chungang Lin, Yequan Wang, and Yujun Zhang. 2023. Netgpt: Generative pretrained transformer for network traffic. *ArXiv preprint* abs/2304.09513 (2023). https://arxiv.org/abs/2304.09513

[35] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, et al. 2023. Rwkv: Reinventing rnns for the transformer era. *ArXiv preprint* abs/2305.13048 (2023). https://arxiv.org/abs/2305.13048

[36] Jian Qu, Xiaobo Ma, and Jianfeng Li. 2024. TrafficGPT: Breaking the Token Barrier for Efficient Long Traffic Analysis and Generation. *ArXiv preprint* abs/2403.05822 (2024). https://arxiv.org/abs/2403.05822

[37] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[38] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67. http://jmlr.org/papers/v21/20-074.html

[39] Prajit Ramachandran, Barret Zoph, and Quoc V Le. 2017. Searching for activation functions. *ArXiv preprint* abs/1710.05941 (2017). https://arxiv.org/abs/1710.05941

[40] Markus Ring, Daniel Schlör, Dieter Landes, and Andreas Hotho. 2019. Flow-based network traffic generation using generative adversarial networks. *Computers & Security* 82 (2019), 156–172.

[41] Markus Ring, Sarah Wunderlich, Deniz Scheuring, Dieter Landes, and Andreas Hotho. 2019. A survey of network-based intrusion detection data sets. *Computers & Security* 86 (2019), 147–167.

[42] Yair Schiff, Chia-Hsiang Kao, Aaron Gokaslan, Tri Dao, Albert Gu, and Volodymyr Kuleshov. 2024. Caduceus: Bi-directional equivariant long-range dna sequence modeling. *ArXiv preprint* abs/2403.03234 (2024). https://arxiv.org/abs/2403.03234

[43] Zhaolei Shi, Nurbol Luktarhan, Yangyang Song, and Gaoqi Tian. 2023. BFCN: A Novel Classification Method of Encrypted Traffic Based on BERT and CNN. *Electronics* 12, 3 (2023), 516.

[44] Zhaolei Shi, Nurbol Luktarhan, Yangyang Song, and Huixin Yin. 2023. TSFN: A Novel Malicious Traffic Classification Method Using BERT and LSTM. *Entropy* 25, 5 (2023), 821.

[45] shramos. 2019. shramos/pcap-splitter. https://github.com/shramos/pcap-splitter.

[46] Robin Sommer and Vern Paxson. 2010. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE symposium on security and privacy*. IEEE, 305–316.

[47] Joel Sommers, Hyungsuk Kim, and Paul Barford. 2004. Harpoon: a flow-level traffic generator for router and network tests. *ACM SIGMETRICS Performance Evaluation Review* 32, 1 (2004), 392–392.

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[49] Kashi Venkatesh Vishwanath and Amin Vahdat. 2009. Swing: Realistic and responsive network traffic generation. *IEEE/ACM Transactions on Networking* 17, 3 (2009), 712–725.

[50] Aaron Voelker, Ivana Kajic, and Chris Eliasmith. 2019. Legendre Memory Units: Continuous-Time Representation in Recurrent Neural Networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 15544–15553. https://proceedings.neurips.cc/paper/2019/hash/952285b9b7e7a1be5aa7849f32ffff05-Abstract.html

[51] Qineng Wang, Chen Qian, Xiaochang Li, Ziyu Yao, and Huajie Shao. 2024. Lens: A Foundation Model for Network Traffic. *ArXiv preprint* abs/2402.03646 (2024). https://arxiv.org/abs/2402.03646

[52] Tongze Wang, Xiaohui Xie, Wenduo Wang, Chuyi Wang, Youjian Zhao, and Yong Cui. 2024. NetMamba: Efficient Network Traffic Classification via Pre-training Unidirectional Mamba. *ArXiv preprint* abs/2405.11449 (2024). https://arxiv.org/abs/2405.11449

[53] Zhaohu Xing, Tian Ye, Yijun Yang, Guang Liu, and Lei Zhu. 2024. Segmamba: Long-range sequential modeling mamba for 3d medical image segmentation. *ArXiv preprint* abs/2401.13560 (2024). https://arxiv.org/abs/2401.13560

[54] Wenhan Xiong, Barlas Oguz, Anchit Gupta, Xilun Chen, Diana Liskovich, Omer Levy, Scott Yih, and Yashar Mehdad. 2022. Simple Local Attentions Remain Competitive for Long-Context Tasks. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Seattle, United States, 1975–1986. https://doi.org/10.18653/v1/2022.naacl-main.144

[55] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling Tabular data using Conditional GAN. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc,

Emily B. Fox, and Roman Garnett (Eds.). 7333–7343. https://proceedings.neurips.cc/paper/2019/hash/254ed7d2de3b23ab10936522dd547b78-Abstract.html

[56] Shengzhe Xu, Manish Marwah, Martin Arlitt, and Naren Ramakrishnan. 2021. Stan: Synthetic network traffic generation with generative neural models. In *Deployable Machine Learning for Security Defense: Second International Workshop, MLHat 2021, Virtual Event, August 15, 2021, Proceedings 2*. Springer, 3–29.

[57] Zhichao Xu. 2024. RankMamba, Benchmarking Mamba's Document Ranking Performance in the Era of Transformers. *ArXiv preprint* abs/2403.18276 (2024). https://arxiv.org/abs/2403.18276

[58] Yucheng Yin, Zinan Lin, Minhao Jin, Giulia Fanti, and Vyas Sekar. 2022. Practical gan-based synthetic ip header trace generation using netshare. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 458–472.

[59] Siyao Zhang, Daocheng Fu, Wenzhe Liang, Zhao Zhang, Bin Yu, Pinlong Cai, and Baozhen Yao. 2024. Trafficgpt: Viewing, processing and interacting with traffic foundation models. *Transport Policy* (2024).

[60] Shihao Zhao, Dongdong Chen, Yen-Chun Chen, Jianmin Bao, Shaozhe Hao, Lu Yuan, and Kwan-Yee K. Wong. 2023. Uni-ControlNet: All-in-One Control to Text-to-Image Diffusion Models. https://arxiv.org/abs/2305.16322

Appendices

Appendices are supporting material that has not been peer-reviewed.

A Configuration

The following section provides details referenced in the paper regarding model architecture/training.

A.1 Mamba Block

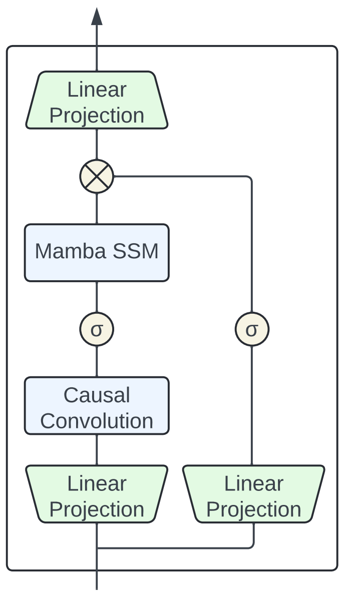

Figure 1: Architecture for the Mamba block, described in Section 4. σ denotes the SiLU/Swish non-linear activation. $\otimes$ denotes element-wise multiply.

Figure 1 depicts the Mamba block. Here, an input sequence is linearly projected twice to the input dimension. One copy is passed through a FFT-based causal convolution and SiLu non-linear activation [39], before being used as input to the Mamba SSM. This allows for efficient training in parallel via convolution, and improves approximation of the true input distribution. The second copy passes only through a SiLu activation, and "gates" the output of the Mamba SSM from the first copy, *i.e.*, using element-wise multiplication, enhances or filters the Mamba SSM output. The gated output is then linearly projected back to the original input dimension.

B Evaluation

The following section provides additional details referenced in the paper regarding model evaluation.

B.1 Generation Length

Metric	Max	Min	Mean	Std Dev	Var
Packets	1081	4	123.17	230.57	53161.55

Table 1: Packet count statistics of generated flows.

Table 1 provides statistics for packet length (in bytes) of packets in our generated synthetic traces.

B.2 Comparing synthetic and real-world packet data

Field	Average Change
TCP_ack	1982981477.85
TCP_seq	1475508827.26
TCP_chksum	24859.46
TCP_sport	23406.24
TCP_dport	23406.24
IP_chksum	6369.34
IP_id	2924.89
TCP_window	1136.37
IP_len	507.86
IP_ttl	11.52
TCP_dataofs	2.16
Raw_load	1.00
TCP_options	0.99
TCP_flags	0.86
Ether_dst	0.48
Ether_src	0.48
IP_src	0.48
IP_dst	0.48
Ether_type	0.00
IP_version	0.00
IP_ihl	0.00
IP_tos	0.00
IP_flags	0.00
IP_frag	0.00
IP_proto	0.00
IP_options	0.00
TCP_reserved	0.00
TCP_urgptr	0.00

Table 2: Average change of representative header fields

Table 2 shows the average change in packet bytes of synthetic traces and real-world data from our service-classification dataset. Average change is calculated by comparing the fields of real and synthetic packets, computing the absolute differences for each field, and averaging those differences across all compared packets.