MINT: Modeling GenAI Impact on Network Traffic

Andrew Nguyen[1], Samson Kempiak[1], Agrim Gupta[2], Koushik Kar[1], Ish Kumar Jain[1]
[1]Rensselaer Polytechnic Institute, Troy, NY, USA, [2]Samsung Research America, Plano, TX, USA

ABSTRACT

Generative AI (GenAI) is becoming a mainstream network workload, yet packet-level simulators lack measurement-driven GenAI traffic models. Currently researchers must approximate GenAI services using traditional sources such as file transfer and video streaming, limiting realistic network evaluation of scheduling and capacity planning. We present MINT, a measurement and modeling framework for GenAI network traffic. Using an isolated network-namespace capture pipeline, we collect client-side traces from three LLM providers across four modalities, cloud and edge servers, and wired and wireless network access points. We find that GenAI modalities exhibit distinct upload/download asymmetry and burst structures that differ from traditional applications. MINT clusters and models these burst regimes and reproduces empirical behavior in ns-3 with normalized Wasserstein distances of 2–25%. Our results also reveal that constant token generator models fail to capture realistic packet burst variability. MINT open-sources the first measurement-driven GenAI traffic model for packet-level network simulation.

1 INTRODUCTION

Generative AI (GenAI) traffic is becoming a mainstream network workload, with adoption exceeding 50% in measured markets for everyday interactions such as chatbots, code generation, and image generation [7, 9]. Unlike video streaming, whose download-heavy ON–OFF buffer refills make its traffic predictable, GenAI traffic has no predictable structure: prompts vary, server computation times vary, and response generation is nondeterministic. Network expectations remain limited to coarse aggregate assumptions, such as a shift toward upload-heavy traffic [4]. GenAI therefore needs the network research foundation existing for video streaming: measurement, modeling, and packet-level simulation.

WiNTECH '26, October 26–30, 2026, Austin, TX, USA

ACM ISBN 979-8-4007-2879-2/26/10
https://doi.org/10.1145/3831662.3844175

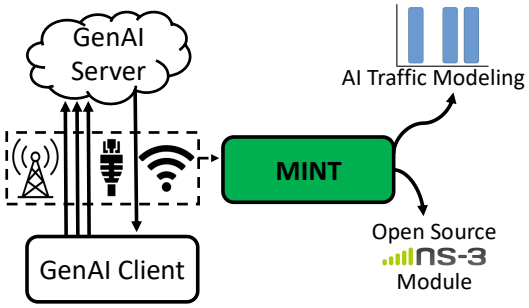

Figure 1: Modeling GenAI Impact on Network Traffic (MINT) through measurements, modeling, and simulation.

This paper asks how does GenAI network traffic differs from traditional applications, and how can we develop packet-level network simulator. Simulators evaluate scheduling, congestion control, and capacity planning using application-level traffic models that reproduce how each application generates packets and bursts [12, 20, 31]. Traditional applications have these models, but GenAI has none today.

Building a measurement-driven GenAI traffic model is challenging for three reasons. First, token streaming is invisible in network traces. Payloads are TLS-encrypted, and providers do not expose how tokens are batched into packets. A machine-in-the-middle (MITM) proxy can recover the text stream over time but not how it appears as packets on the network. Second, client-side captures are easily contaminated by browser and background traffic and by transport artifacts such as NIC offloading, which distorts packet sizes and interarrival times. Third, application burst timing must be separated from the access network, since Wi-Fi and 5G add delays that mask application behavior. Prior work does not address these problems. *GenAI-centric* studies measure server-side token metrics (time-to-first-token, inter-token latency) [23, 27] that do not translate into client-side packet bursts, while *client-side* studies pursue other goals, such as token-streaming design or traffic classification, with a single access network, few modalities or prompts, and summary statistics that cannot regenerate packet-level dynamics [1, 11, 21]. No existing GenAI measurement system can drive a packet-level simulator with realistic GenAI traffic.

We address these challenges with four key ideas. First, we identified the stages of a GenAI session using unencrypted traffic from an edge server. Second, we revealed a

multi-timescale burst structure in packet interarrival times—packet serialization, sub-bursts, bursts, and initial server delay—whose timescales can be cleanly separated and modeled independently. Third, contamination is eliminated at the source by capturing traffic inside a dedicated network namespace that carries only GenAI traffic. Fourth, the burst structure is a property of the application rather than the access network. Burst interarrival times are consistent across Ethernet, Wi-Fi, and 5G, so a model can be derived from one access technology.

MINT. We present MINT, a measurement and modeling framework for GenAI network traffic, shown in Figure 1. MINT isolates direct GenAI API traffic in network namespaces and collects a diverse dataset of nearly 10,000 client-side session traces spanning four modalities, three providers, cloud and edge servers, and Ethernet, Wi-Fi, and 5G access networks. From these measurements, we characterize how upload/download asymmetry and burst structure differ from GenAI differ from traditional traffic. We then model the burst behavior probabilistically at each timescale and implement the model in the open-source Network Simulator 3 (ns-3) [13].

Contributions:

- A diverse, isolated experimental setup spanning Ethernet, Wi-Fi, and a 5G modem, capturing client-side GenAI network traffic across three providers, four modalities, and cloud and edge servers, released as an open trace dataset[1] and measurement framework[2].
- A study of GenAI modalities revealing distinct burst timing structures and a dynamic UL/DL asymmetry: whereas macro-level reports cite a 74%/26% downlink/uplink split [6], we show that the split depends heavily on modality and application phase, ranging from 0% to 100% downlink.
- The first open-source, measurement-driven GenAI network burst model[3], achieving normalized Wasserstein distances of 2–25% for burst interarrival times.

2 BACKGROUND AND RELATED WORK

2.1 Background

We review how LLMs generate traffic, how it is measured, and how application traffic is modeled.

How do LLM clients and servers interact? A user prompt is sent to the LLM server, tokenized, and processed in parallel during a prefill phase; the model then generates output tokens iteratively until an end-of-sequence token. Tokens are either streamed to the client in chunks (streaming mode) or returned all at once (non-streaming mode).

How is GenAI network traffic measured? Tools such as `tcpdump` and `Wireshark/tshark` capture packets at points such as Ethernet and Wi-Fi interfaces [1, 11]; because payloads are encrypted, analysis relies on metadata such as per-packet size and interarrival time. Traffic is isolated either by attributing packets by IP address and metadata or through controlled experiments with network namespaces, then processed to expose transport-agnostic application behavior [17, 25, 30].

How are network bursts modeled? GenAI has no application layer burst traffic model today; a good model must capture the mechanisms that generate the application's traffic. Video streaming follows an ON–OFF buffer-refill model governed by bit rate and buffer state [2, 13]; file transfer is bulk delivery at link capacity [26]; aggregate models such as Poisson Pareto burst processes reproduce composite traffic but discard per-application structure [20]. By analogy, streamed GenAI calls for server delay plus token-stream bursts, and non-streamed GenAI for server delay plus a file-transfer-like bulk phase.

2.2 Related Work

We group prior work into GenAI-centric measurements and network measurements of GenAI traffic, and show that neither yields a packet-level GenAI traffic model.

GenAI-centric measurements. A first line of work measures token-level behavior generated at the GenAI server, characterizing application-layer generation dynamics [1, 18] or designing improved token-streaming algorithms [11, 22]. These studies establish how tokens are generated and delivered at the application layer. However, token-level metrics do not capture how a provider converts tokens into the packet bursts a client actually receives, so they cannot drive a packet-level network model.

Network measurements of GenAI traffic. A second line of work measures GenAI traffic on the network itself, building measurement systems for goals such as token streaming over lossy networks, GenAI traffic classification, and high-level impact studies [1, 3, 8, 11, 14, 21]. These efforts include the first multi-modality measurements of GenAI traffic [21] and the first measurements across multiple access networks [11], both of which we build on. For traffic modeling, however, three gaps remain. First, no existing measurement system combines the prompt diversity, modality diversity, and access-network isolation that a comprehensive burst model requires. Second, studies that derive statistical metrics to distinguish GenAI providers from each other and from traditional traffic [1, 21] discard the reproducible packet mechanisms that modeling needs. Third, the models that do exist—for token streaming over lossy networks and

[1] https://huggingface.co/datasets/wayslab/llm-network-study-data
[2] https://github.com/wayslab/LLM-Network-Study
[3] https://github.com/wayslab/ns-3-dev

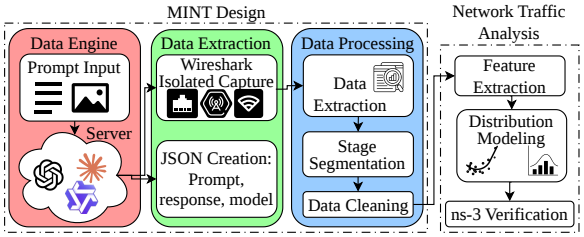

Figure 2: End-to-end GenAI traffic measurement processing, modeling, and simulation pipeline

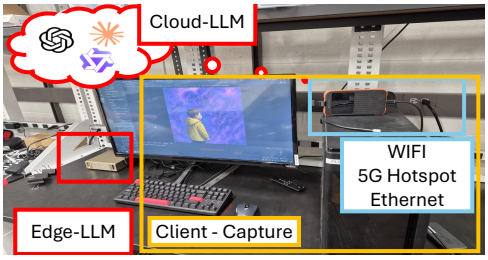

Figure 3: Hardware across access types and models.

for packet-size flows [11, 14]—omit the variation in interarrival times that drives packet bursts. No measurement-driven model has yet reproduced how time-varying bursts of packets arrive at the client.

3 MINT DESIGN

MINT has three components, shown in Figure 2: a data engine that replays real-user prompts against cloud and edge LLMs across modalities, a data extraction tool that captures each session's traffic in isolation, and a processing stage that segments and cleans traces for modeling. For each prompt, the client starts a capture, sends the prompt to the designated LLM server over one of three access networks—Ethernet, Wi-Fi, or a 5G modem—and stops the capture when the response completes.

Hardware Setup. The experimental setup is shown in Figure 3. MINT measures proprietary cloud-hosted LLMs, uses an edge-hosted LLM to validate the client-side measurements against ground truth, and repeats captures over wired Ethernet, Wi-Fi, and a SIMO 5G hotspot to validate that our application-layer findings are access-independent.

3.1 MINT Data Engine

The MINT data engine automates prompt testing across the model providers in Table 1 and the modalities in Table 2, drawing prompts from open-source real-world user prompt datasets.

Large Language Models. We measure three major LLM providers: OpenAI, Anthropic, and open-source Qwen [28], with specific models outlined in Table 1. Whereas prior work measured through browsers or apps, we remove that abstraction and access each provider's API directly from Python, eliminating browser and background traffic at the source. We further host an LLM on an edge server to validate our measurements without network artifacts. We measure all three providers to establish that their traffic behavior is repeatable with minor discrepancies. We then focus our modeling on ChatGPT and leave the remaining providers to future work.

Table 1: Measured models with capture date (MM/DD, 2026), access path, and modality: text-to-text (T2T), image-to-text (I2T), text-to-image (T2I), and code generation.

Model	Date	Access	Modality
ChatGPT 5.4	07/03	Browser	T2T
ChatGPT 5.4	04/26	Cloud API	T2T & I2T
ChatGPT Image 1	04/28	Cloud API	T2I
Claude Opus 4.6	04/25	Cloud API	T2T & I2T
Qwen 3.5 122B A10B	06/16	Cloud API	T2T & I2T
Z-Image-Turbo	06/16	Cloud API	T2I
Qwen 3.5 122B A10B Int4	06/11	Edge vLLM	T2T & I2T
Z-Image-Turbo	06/15	Edge vLLM	T2I
Claude Opus 4.8	06/29	Claude Code	Code Gen.
ChatGPT 5.5	07/03	Codex	Code Gen.

Prompt Datasets & Inputs. Table 2 summarizes the prompt datasets, chosen to cover the modalities that dominate real GenAI usage [9]—text-to-text (T2T), text-to-image (T2I), image-to-text (I2T), and code generation—with 12 short- and long-form YouTube videos for traditional-streaming comparison. These reflect real-user prompts in a reproducible framework: T2T with Berkeley Function Calling Leaderboard (BFCL), T2I and I2T with DiffusionDB image generation prompts. While we do not focus on analyzing the GenAI result, we had to filter the image generation prompts to avoid policy-related restrictions that stopped network traffic abruptly, reducing 400 prompts down to 229. We reused the generated images as image-to-text input. Text and image made up straightforward single-session prompts, so we measured a complex code generation example to show a multi-turn, back-and-forth agentic workflow spawning from one human prompt: 24 agentic prompts over 900 seconds. While these datasets do not reproduce human think-time delays, they represent realistic one-shot and multi-turn prompts reproducibly. Modeling multi-turn prompting is out of scope for this work.

3.2 MINT Data Extraction Tool

Our framework uses a Python client that invokes proprietary and open-source GenAI APIs, captures traffic in `PCAPNG` format, and logs prompt, response, and model metadata as `JSON`; we record average round-trip time before and after each run

Table 2: Prompt dataset summary: text-to-text, text-to-image, image-to-text, and code generation (qualitative comparison only*).

Source	Name	Subcategory	# Prompts / Duration
[16]	BFCL	Text-to-text	1677
[24]	DiffusionDB	Text-to-Image	229
[24]	DiffusionDB	Image-to-Text	229
[19]	Kaggle	Code Gen.*	One, 900 seconds

to confirm connection stability. We extract traffic three ways depending on cloud, browser, and edge server access. For Cloud APIs for ChatGPT, Claude, and Qwen, we route the client through a dedicated Linux network namespace joined to the host by a virtual Ethernet (`veth`) pair. Wireshark captures at the `veth` interface, isolating traffic from the GenAI flow while retaining LAN access via a consistent local IP. For consumer-facing apps used for comparison, such as ChatGPT Browser and YouTube, we capture from a browser client using Selenium and dedicated Chrome profiles to suppress ads and unrelated browser activity. For the edge server, we host an open-source LLM on an NVIDIA DGX Spark with a capture daemon the client starts/stops remotely, yielding synchronized client- and server-side traces. All three paths produce the same `PCAPNG` traces and `JSON` metadata.

API versus Browser: Prior studies measured GenAI traffic through browser/app interfaces, while API calls provide cleaner network isolation and directly observable client–server behavior. Both reflect realistic usage: APIs increasingly support application backends [15], while browser/app traffic captures direct consumer use. In our comparison, in Appendix, API traffic used a single provider connection with clearly separated interaction stages, whereas browser traffic involved multiple concurrent flows and additional exchanges for the same prompt. These structural differences motivate future decrypted-traffic studies, such as MITM studies, to attribute and model browser GenAI behavior. We focus MINT API-based traffic.

3.3 MINT Data Processing

After each capture, MINT processes the trace in three steps: data extraction, stage segmentation, and data cleaning.

Data Extraction. From each `PCAPNG` capture, we extract per-packet size, interarrival time (IAT), and direction: with the client fixed at `10.0.0.2`, packets are upload (UL) or download (DL) when the client IP is the source or destination, respectively. These fields drive our burst analysis.

Stage Segmentation. Using unencrypted traffic from a local edge server, we identified four stages in a one-shot conversation: handshake, prompt, processing, and response. The handshake begins with a `Client Hello` and concludes with an ACK packet. The prompt stage follows as sequences of uploaded application data from the client, concluding with an ACK packet from the server. The processing stage is a long period of almost no throughput, measured as the time-to-first-response (TTFR) packet. The response stage then starts with the first server download packet and continues for the rest of the capture. We observed the same structure in encrypted TLSv1.3 API traffic from cloud servers.

Data Cleaning. We cleaned transport-layer artifacts that impacted TTFR and IAT measurements to ensure we properly model application-layer behavior. TTFR cannot be measured from the first download packet alone because many traces begin with a small 24-byte control packet before the actual response. This would underestimate the true response time. We therefore ignore packets below 50 bytes and measure TTFR from the first `Application Data` packet. For IAT measurements, we found 11.5% of client-side packets were distorted, including oversized superpackets and unusually long packet IATs. Disabling NIC offloading removed the oversized packets, but the timing anomalies remained. We therefore detect these anomalous IATs and replace them using neighboring packet-serialization-scale timing, preventing capture artifacts from contaminating burst-level IAT measurements.

Provider Comparison and Repeatability: These metrics show that measurements are repeatable within a model and similar across providers, supporting our choice to focus modeling on ChatGPT. In multiple trials of each model, the cumulative distribution functions (CDFs) of response packet size and packet IAT overlapped, indicating repeatable experiments, shown in Appendix. Between providers, non-streaming mode showed similar response packet sizes and packet IATs, while streaming mode varied slightly. For example, Qwen produced a longer TTFR than ChatGPT, with similar mean packet IAT but smaller variance.

4 MINT NETWORK TRAFFIC ANALYSIS

Using the cleaned packet traces, we answer three questions. First, how does upload (UL) / download (DL) asymmetry change with GenAI applications? Second, how do packet bursts change with GenAI? And third, how can we model and simulate GenAI network traffic?

4.1 Upload/Download Asymmetry

GenAI exchanges involve diversity in both uploaded (UL) and downloaded (DL) modality, producing modality-dependent UL/DL behavior. Typically, the fraction of DL is measured across total load, producing one aggregate number such as 74%/26% downlink/uplink [6]. We show the fraction goes through different stages of UL-heavy and DL-heavy, with different throughput and frequency at each stage.

Feature Extraction. Since aggregate UL/DL ratios do not highlight the UL/DL asymmetry throughout various stages of

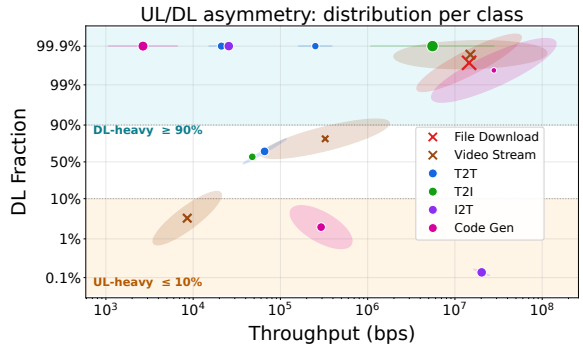

Figure 4: Gaussian mixture clusters of 1s windows for Download (DL) proportion (y-axis) highlighting changes in DL-heaviness relative to total throughput (x-axis) and packet occurrence (dot size).

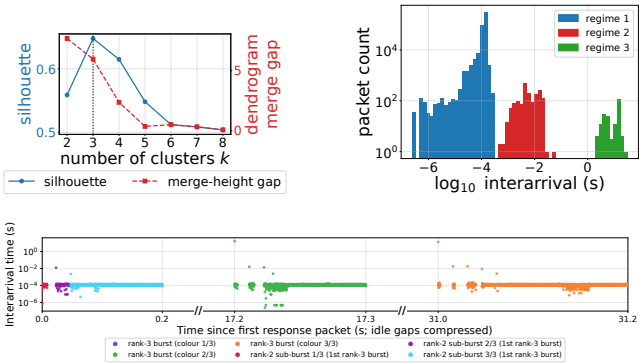

Figure 5: *Agglomerative hierarchical clustering* (AHC) on text-to-image streaming example. AHC clusters (top left), AHC time-scale burst regimes (top right), example burst classification (bottom).

traffic, we cluster DL fraction and throughput to show 2 to 3 distinct stages for each application. Across 1 second windows, we compute the total UL and DL bytes, used to compute the ratio of DL to total traffic and the total throughput in bytes per second. We present on a logit scale to highlight the tails of DL-heavy and UL-heavy periods. We cluster many data points into 2 to 3 clusters to represent different stages. The size of each dot represents how many windows fall in the cluster.

Analysis. GenAI has modality-dependent UL/DL stages flowing through UL-heavy and DL-heavy stages, adding specific context to the prior single 74/26 aggregate metric, shown in Figure 4.

Traditional application behavior is as expected. File download transfers data very quickly with very little UL occurring. Video streaming has its two largest clusters at DL-heavy and UL-heavy. During DL-heavy, the buffer is filled as a high-throughput download stage. The UL-heavy cluster has very low throughput, representing the periods between buffer refills, with periodic QUIC exchange for YouTube's playback heartbeat. The mixed usage only appears due to window sampling boundaries.

On the other hand, GenAI sessions show diverse UL/DL asymmetry when comparing stages and modality. Text modalities transfer far less data than image modalities. The T2T scenario highlights the handshake+prompt stages, resulting in moderate throughput, mixed usage, followed by comparably moderate DL-heavy throughput responses. The two DL-heavy clusters represent non-streamed and streamed responses. T2I shows a similar prompt stage, but a much higher DL-heavy throughput in the response stage. Conversely, I2T shows the image upload, much larger than the handshake, with small text responses. Interestingly, the more complex code generation spawns multiple prompt–response exchanges. The prompt stage had moderate throughput, comparable to the text uploads seen in T2T and T2I. However, outside of a single large file download by the agent, the response stage was 100x lower throughput, even lower than the T2T streaming scenarios. Overall, GenAI prompts make traffic more UL-heavy, its DL-heavy responses are smaller than traditional downloads, and both vary greatly with modality.

4.2 Time-scale Burst Modeling Analysis

Network capture shows irregular packet bursts within the same GenAI session, originating from server processing delay and token streaming behavior. We analyzed and validated the GenAI burst structure across network access, and we compared it with traditional file download and video streaming behaviors.

Feature Extraction. The burst timing structure can be extracted from longer interarrival times that start packet bursts, shown in the IAT-versus-time plot in Figure 5. We want to validate application-level packet burst behavior across network access types. Since the Ethernet packet serialization time, $\approx 10^{-4}$ s, is clearly shown in the shortest IAT cluster in the above figure, the other clusters represent application-layer burst timings. Thus, we can use *agglomerative hierarchical clustering (AHC)* [5] for data-driven thresholds for clusters. Under Wi-Fi and 5G networks, the separability is less clear, making AHC ineffective and requiring a new method, *throughput burst extraction.*

Agglomerative Hierarchical Clustering: Unlike simple thresholding, which relies on manually selected cutoffs, the AHC method exploits the density and proximity of neighboring log-IAT histogram bins to uncover natural groupings, shown in Figure 5. The number of clusters with the largest silhouette score—a measure of how well each point fits its assigned cluster versus the nearest alternative—sets the number of burst regimes. We identified distinct burst transmission regimes

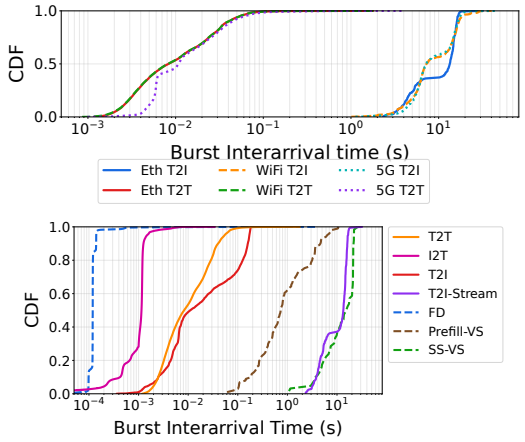

Figure 6: Burst Interarrival Time (B-IAT) across access networks (top). B-IAT across GenAI and traditional applications (bottom), including file download (FD), and the prefill and steady-state (SS) video stream (VS).

from sub-ms packet serialization to over ten-second server-thinking bursts, ranked by time-scale: R1 near packet serialization, R2 sub-bursts, and R3 bursts. These give data-driven boundaries for packet IAT clusters, where the longer time-scale regimes define the burst IAT (B-IAT) distribution.

Throughput Burst Extraction: Wi-Fi and 5G exhibited longer transmission intervals due to additional wireless access delays, requiring a new *throughput burst extraction* method to extract burst timings. We identify 1 ms windows with significant throughput and iteratively construct larger bursts from adjacent windows. We uncovered a B-IAT bimodality with a clear separation at 1 second, revealing the largest burst structure above the threshold. We use this method to validate the largest application-layer bursts found by AHC; however, it cannot identify sub-burst structure.

Burst Behavior Validation across Wi-Fi and 5G: We validated B-IAT extraction using AHC on Ethernet IAT with the throughput-based burst extraction used on Wi-Fi and 5G data, shown in Figure 10. Despite Wi-Fi and 5G introducing IAT artifacts in the sub-burst (R2) and serialization (R1) regimes, the throughput-based extraction shows B-IAT distributions aligning closely with AHC on Ethernet data. This validates that the largest-regime IAT distribution is the application B-IAT, independent of network access.

Burst Analysis. GenAI traffic exhibits five types of burstiness, only some of which resemble traditional applications, shown in Figure 10. First, image upload and non-streaming text (not shown) both send data at a near-constant rate, resembling file download. Second, text streaming has its own B-IAT distribution with a median of 10 ms, a fraction of the 100 ms token streaming rate cited in the simple token generator for [11]. Third, partial image streaming in ChatGPT

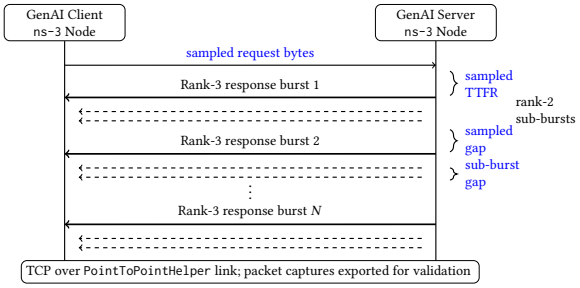

Figure 7: ns-3 simulation burst structure, setting distributions for time-to-first-response (TTFR), and burst parameters per modality.

is separated by long server-thinking delays of nondeterministic image generation, which surprisingly arrive at burst intervals similar to deterministic steady-state video buffering. These partial images also contain sub-bursts (R2), shown in Figure 5. Fourth, non-streaming images arrived in bursts spaced by a distribution centered around 10 ms, rather than the bulk transfer expected for an already generated image. Lastly, code generation demonstrated diverse burstiness in both upload and download within one capture: periods of file-download bursts, structured back-and-forth exchanges, and highly variable burstiness at the end. We keep code generation as a qualitative comparison and leave modeling such complex usage to future work.

4.3 Modeling and ns-3 Simulation

Although some GenAI burst behavior resembles traditional applications, no comprehensive GenAI model exists. We build the first mechanistic model of single-session GenAI traffic—fitting distributions to prompt size, TTFR, and the response burst structure—and implement it as a client/server application pair in ns-3. Simulated traffic reproduces empirical burst interarrival times within 2–25% normalized Wasserstein distance, whereas a constant token-streaming baseline deviates by up to 491%. We describe the model, its ns-3 implementation, and its validation in turn.

Modeling: We probabilistically model each burst timing regime identified by AHC on Ethernet data. A burst of rank R comprises N_R sub-bursts, each of byte size B_R, spaced by B-IAT $\Delta t_{B,R}$ seconds. For bursts (rank-3) and sub-bursts (rank-2), we fit each variable against candidate distributions common in networking, such as log-normal and Pareto (Table 3), selecting the distribution that maximizes goodness-of-fit by Akaike Information Criterion (AIC) while penalizing model complexity by Bayesian Information Criterion (BIC) to prevent overfitting. TTFR fits a 2- or 3-component log-Gaussian mixture model in all cases.

ns-3 Implementation. We implement the burst model in the widely used open-source ns-3 simulator [13] as a client/server application pair, `GenAIUser` and `GenAIServer`,

Table 3: Selected distribution models for different GenAI traffic burst structure, compared by Akaike/Bayesian Information Criterion (AIC/BIC) across candidate distributions. Best-fit, least-complex model minimizing AIC/BIC for different R burst regimes. (NBN: negative binomial; Pois.: Poisson; LN: lognormal; Par.: Pareto; K-logGMM: K-component log Gaussian mixture model.)

Modality	Prompt	Response	TTFR	# R3	R3 IAT	# R2	R2 IAT	R2 Bytes
Text-to-Text Streaming	LN	Burst	2-logGMM	–	–	NBN	3-logGMM	2-logGMM
Text-to-Text Non-Streaming	LN	Par.	2-logGMM	–	–	–	–	–
Image-to-Text	LN	LN	2-logGMM	–	–	–	–	–
Text-to-Image Streaming	Par.	Burst	2-logGMM	Constant	3-logGMM	Pois.	3-logGMM	1-logGMM
Text-to-Image Non-Streaming	Par.	Burst	2-logGMM	–	–	NBN	3-logGMM	LN

shown in Figure 7. The applications open a single TCP connection, over which the client sends a prompt request. The server reassembles the request, samples a processing delay for the TTFR packet, and then generates N_3 rank-3 bursts; each burst may comprise rank-2 sub-bursts, sampling N_2 sub-bursts of B_2 bytes spaced by $\Delta t_{B,2}$. A `JSON` configuration file exposes knobs for every random-variable distribution, and each simulation is deterministic and reproducible from its ns-3 RNG seed. The model represents burst reception as seen by a single user; the application pair enables future research such as multi-user network sessions. As a baseline, we also implement the token-streaming model of Eloquent [11], which sends one sub-MTU token every 100 ms.

Simulation Setup. We run both MINT's GenAI model and the baseline on two ns-3 nodes, client and server, joined by a point-to-point link mimicking the empirical test setup: 1500-byte MTU, 100 Mbps data rate, and 5 ms round-trip time. We capture 200 client-side simulated PCAP traces.

Evaluation Method. We evaluate whether MINT's simulated traffic preserves the empirical response-size and B-IAT distributions, using normalized Wasserstein distance (WS), a metric used to compare network simulators [10, 29]. WS measures the average distribution shift needed to match cumulative distribution functions, expressed as a percentage of the mean; lower is better. We divide the empirical PCAPs in half into training and validation sets, then compare the training set against 1) the empirical validation set, 2) samples from the fitted distributions, 3) MINT ns-3 simulated PCAPs, and 4) baseline token-generator PCAPs.

Burst Distribution Analysis. MINT's simulated traffic matches the empirical burst behavior closely—within 2.3–9% WS for B-IAT across modalities and under 5% for total response size, versus up to 491% for the baseline—with one exception we examine below (Table 4). All comparisons are made against the empirical validation set, itself within 6% WS of training data. First, the constant token generator [11], designed to validate token transmission rather than packet-level bursts, cannot reflect the variation in empirical T2T burst timing: its B-IAT WS reaches 491%, whereas MINT's simulation and fitted distributions stay within 4.6%. Second,

Table 4: Normalized Wasserstein distance relative to the mean (WS) between the empirical training set (Emp. A) and the validation set (Emp. B), fitted distributions (Dist.), MINT ns-3 simulation, and the Eloquent baseline [11].

Modality	Metric	WS (%)			
	Emp. A vs.	Emp. B	Dist.	MINT	Eloquent
T2T-S	Burst IAT	2.36	4.55	4.27	491
T2I-S	Rank-2 Burst IAT	4.23	5.56	24.3	–
T2I	Burst IAT	5.41	7.52	9.04	–

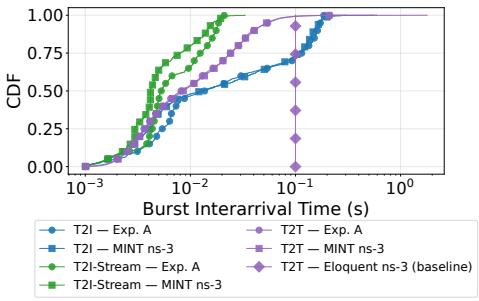

Figure 8: Burst interarrival times comparing training with MINT ns-3 and baseline token streaming in ns-3 (Eloquent [11]).

the exception: the text-to-image streaming model for rank-2 sub-bursts reaches 24.3% WS; Figure 8 shows we capture the bimodal structure of sub-bursts but not their full distribution. Overall, the ns-3 simulator reproduces empirical burst behavior across modalities and improves on the constant token-generation model by two orders of magnitude.

5 CONCLUSION

This paper presented MINT, the first measurement-driven network-simulator model of GenAI traffic, which we open-source[3] together with its dataset[1]. Across three providers and four modalities, we showed that GenAI traffic is not

simply download-heavy, but modality-dependent and characterized by multi-timescale burst structure distinct from bulk transfer and video streaming. We reproduced this behavior with an ns-3 application pair driven by fitted empirical distributions, capturing the variability of token streaming that the constant-rate baseline in [11] does not model. Across modalities, simulated burst interarrival times matched empirical measurements within 2–25% normalized Wasserstein distance.

Limitations. MINT currently models single-shot text and image API interactions. It does not capture browser/app traffic, sequential prompting with human think time, or automated agentic workloads. Because our measurements are packet-level and encrypted, application-stage boundaries are inferred from traffic structure rather than directly observed from decrypted traffic, such as MITM.

Future work. We will extend MINT to browser/app and multi-turn workloads, including code-generation and agentic traffic, and use decrypted measurements, such as controlled MITM interception, to validate application-stage boundaries. We also plan to evaluate network optimization techniques under realistic simulated GenAI workloads.

REFERENCES

[1] Alhazbi et al. 2025. LLMs Have Rhythm: Fingerprinting Large Language Models Using Inter-Token Times and Network Traffic Analysis. *IEEE Open Journal of the Communications Society* (2025).

[2] Bojovic et al. 2022. Enabling NGMN Mixed Traffic Models for ns-3. In *Proceedings of the 2022 Workshop on ns-3*. 127–134.

[3] Cheng et al. 2025. Hello, GenAI? Dissecting Human to Generative AI Calling. In *Proceedings of the 2025 ACM Internet Measurement Conference*. 308–324.

[4] Cisco. 2018. Cisco Visual Networking Index: Forecast and Trends, 2017–2022. https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_Device_Growth_Traffic_Profiles.pdf. Accessed April 24, 2026.

[5] Deart et al. 2021. Agglomerative Clustering of Network Traffic Based on Various Approaches to Determining the Distance Matrix. In *2021 28th Conference of Open Innovations Association (FRUCT)*. IEEE, 81–88.

[6] Ericsson. 2025. GenAI Data Traffic Today. https://www.ericsson.com/en/reports-and-papers/mobility-report/articles/genai-data-traffic-today-june-2025. Ericsson Mobility Report, accessed April 24, 2026.

[7] Jin et al. 2025. End-to-End Coordination of RAN and Edge Server for Latency-Critical Inference Serving over Cellular Networks. *Proceedings of the ACM on Networking* 3, CoNEXT4 (2025), 1–23.

[8] Koneva et al. 2025. Introducing Large Language Models as the Next Challenging Internet Traffic Source. *arXiv preprint arXiv:2504.10688* (2025).

[9] Landay et al. 2026. *The AI Index 2026 Annual Report*. Technical Report. AI Index Steering Committee, Institute for Human-Centered AI, Stanford University, Stanford, CA. https://hai.stanford.edu/ai-index/2026-ai-index-report

[10] Lee et al. 2007. Improving Wireless Simulation through Noise Modeling. In *Proceedings of the 6th international conference on Information processing in sensor networks*. 21–30.

[11] Li et al. 2024. Eloquent: A More Robust Transmission Scheme for LLM Token Streaming. In *Proceedings of the 2024 SIGCOMM Workshop on Networks for AI Computing*. 34–40.

[12] Loh et al. 2022. YouTube Dataset on Mobile Streaming for Internet Traffic Modeling and Streaming Analysis. *Scientific Data* 9, 1 (2022), 293.

[13] William David Diego Maza. 2016. A Framework for Generating HTTP Adaptive Streaming Traffic in ns-3. In *SIMUTools-9th EAI International Conference on Simulation Tools and Techniques-2016*.

[14] Montieri et al. 2026. From Prompts to Packets: A View from the Network on ChatGPT, Copilot, and Gemini. *Computer Networks* (2026), 112237.

[15] OpenAI. 2025. The State of Enterprise AI 2025. https://openai.com/business/guides-and-resources/the-state-of-enterprise-ai-2025-report/. Accessed: 2026-08-29.

[16] Patil et al. 2024. The Berkeley Function Calling Leaderboard (BFCL): From Tool Use to Agentic Evaluation of Large Language Models. In *Advances in Neural Information Processing Systems*.

[17] Paxson et al. 2002. Wide-Area Traffic: The Failure of Poisson Modeling. *IEEE/ACM Transactions on networking* 3, 3 (2002), 226–244.

[18] Qu et al. 2025. TokenFlow: Unified Image Tokenizer for Multimodal Understanding and Generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 2545–2555.

[19] Meg Risdal. 2017. New York City Taxi Trip Duration. https://kaggle.com/competitions/nyc-taxi-trip-duration. Kaggle.

[20] Sivaroopan et al. 2025. A Comprehensive Survey on Network Traffic Synthesis: From Statistical Models to Deep Learning. *arXiv preprint arXiv:2507.01976* (2025).

[21] Tagami et al. 2026. Understanding Network Impact of Generative AI: Application-Level Traffic Measurement. In *Proceedings of IEEE INFOCOM 2026 Workshop on 6G AI-RAN*. IEEE.

[22] Tian et al. 2025. ChatterBox: Multimodal Referring and Grounding with Chain-of-Questions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 7401–7409.

[23] Wang et al. 2024. Revisiting Service Level Objectives and System Level Metrics in Large Language Model Serving. *arXiv preprint arXiv:2410.14257* (2024).

[24] Wang et al. 2023. DiffusionDB: A Large-scale Prompt Gallery Dataset for Text-to-Image Generative Models. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*. 893–911.

[25] Willinger et al. 2019. Lessons from "On the Self-Similar Nature of Ethernet Traffic". *ACM SIGCOMM Computer Communication Review* 49, 5 (2019), 56–62.

[26] Wong et al. 2006. *Comments and Proposal to Replace Traffic Models in IEEE 802.16j-06/013*. IEEE 802.16 Contribution IEEE C802.16j-06/093r3. IEEE 802.16 Broadband Wireless Access Working Group.

[27] Xiao et al. 2025. Streaming, Fast and Slow: Cognitive Load-Aware Streaming for Efficient LLM Serving. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*. 1–13.

[28] Yang et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388* (2025).

[29] Yang et al. 2022. DeepQueueNet: Towards Scalable and Generalized Network Performance Estimation with Packet-Level Visibility. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 441–457.

[30] Zhang et al. 2024. Quic is not quick enough over fast internet. In *Proceedings of the ACM Web Conference 2024*. 2713–2722.

[31] Zink et al. 2009. Characteristics of YouTube Network Traffic at a Campus Network–Measurements, Models, and Implications. *Computer networks* 53, 4 (2009), 501–514.

6 APPENDIX

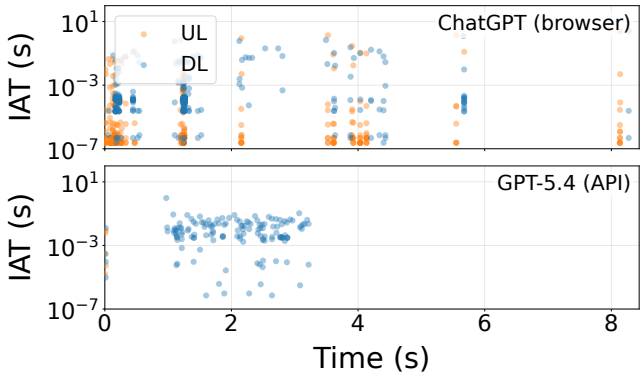

Figure 9: Browser vs API Interarrival Time (IAT) Bursts

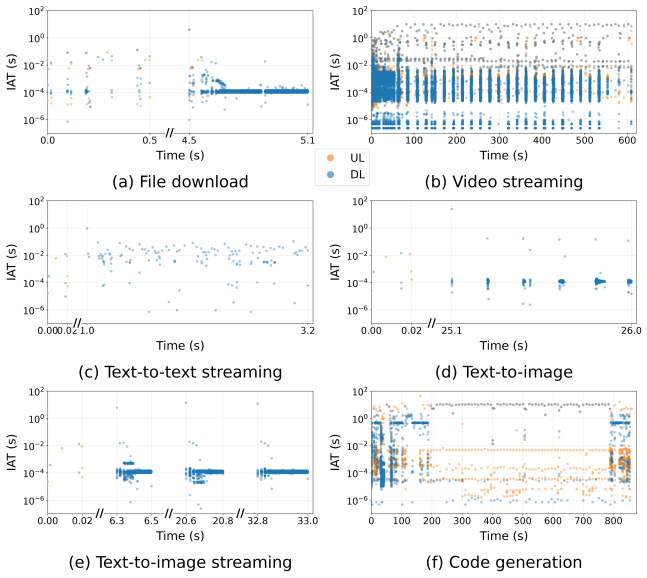

Figure 10: Interarrival Time Comparison between traditional and GenAI applications.

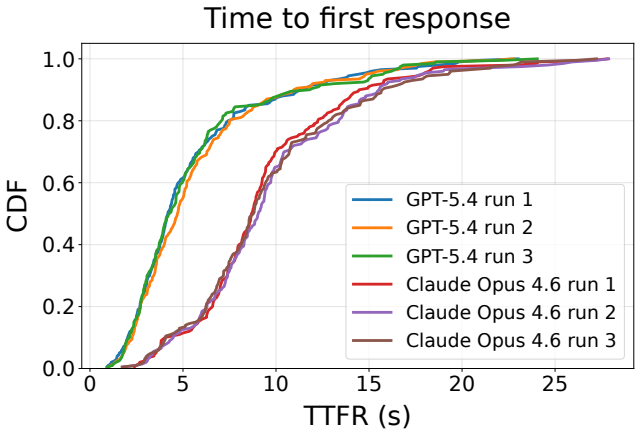

Figure 11: Distribution of time to first response across repeated runs.

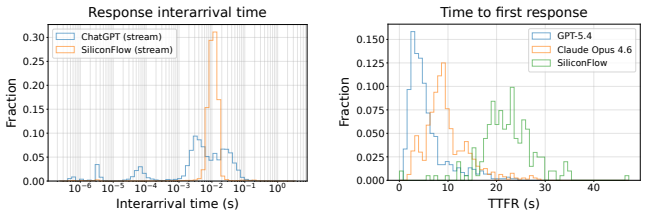

Figure 12: Distribution comparison of different models in (a) T2T-Stream response interarrival time and (b) T2T-Nonstream time to first response.