Data-Driven Stochastic Modeling Using Autoregressive Sequence Models: Translating Event Tables to Queueing Dynamics

Daksh Mittal, Shunri Zheng, Jing Dong, Hongseok Namkoong
Columbia University
{dm3766, sz3091, jd2736, hn2369}@columbia.edu

Abstract

While queueing network models are powerful tools for analyzing service systems, they traditionally require substantial human effort and domain expertise to construct. To make this modeling approach more scalable and accessible, we propose a data-driven framework for queueing network modeling and simulation based on autoregressive sequence models trained on event-stream data. Instead of explicitly specifying arrival processes, service mechanisms, or routing logic, our approach learns the conditional distributions of event types and event times, recasting the modeling task as a problem of sequence distribution learning. We show that Transformer-style architectures can effectively parameterize these distributions, enabling automated construction of high-fidelity simulators. As a proof of concept, we validate our framework on event tables generated from diverse queueing networks, showcasing its utility in simulation, uncertainty quantification, and counterfactual evaluation. Leveraging advances in artificial intelligence and the growing availability of data, our framework takes a step toward more automated, data-driven modeling pipelines to support broader adoption of queueing network models across service domains.

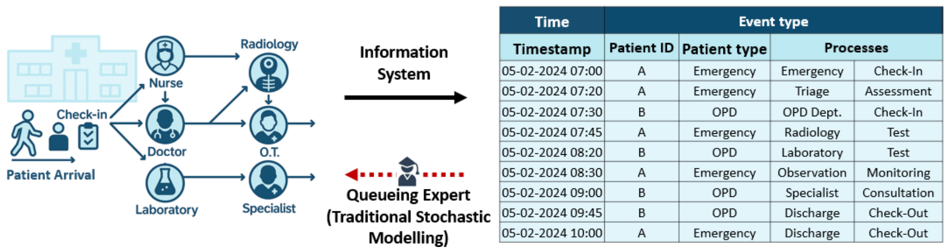
1 Introduction

Queueing network models are powerful mathematical frameworks for understanding and managing congestion in a wide range of service systems, such as call centers, hospitals, and ride-sharing platforms [6, 9, 7, 10, 2]. By capturing the stochastic nature of demand and service processes in resource-constrained environments, they enable performance analysis under uncertainty, evaluate trade-offs between service quality and resource utilization, and support policy evaluation. When closed-form analysis is intractable, stochastic simulation becomes the workhorse for estimating key metrics and optimizing operations; in practice, these simulators often serve as the computational core of *digital twins* that provide monitoring, forecasting, and what-if experimentation in congestion-prone environments [12].

Despite these advantages, their adoption at scale is limited by the expertise required to select and calibrate appropriate networks (e.g., service disciplines, routing rules, and network topologies). For instance, modeling patient flow in hospitals has been the focus of many studies [1, 11, 5], each grappling with the intricacies of capturing system-specific dynamics. This work investigates how modern machine-learning (ML) infrastructure can lower these technical barriers across domains.

Modern information systems nowadays produce rich, high-resolution *event tables* that log sequences of discrete events (transitions) along with their timestamps [6, 3]. For example, electronic health records routinely encode full patient trajectories across care stages (Figure 1). These high-resolution

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: MLxOR: Mathematical Foundations and Operational Integration of Machine Learning for Uncertainty-Aware Decision-Making.



Time	Event type			
Timestamp	Patient ID	Patient type	Processes	
05-02-2024 07:00	A	Emergency	Emergency	Check-In
05-02-2024 07:20	A	Emergency	Triage	Assessment
05-02-2024 07:30	B	OPD	OPD Dept.	Check-In
05-02-2024 07:45	A	Emergency	Radiology	Test
05-02-2024 08:20	B	OPD	Laboratory	Test
05-02-2024 08:30	A	Emergency	Observation	Monitoring
05-02-2024 09:00	B	OPD	Specialist	Consultation
05-02-2024 09:45	B	OPD	Discharge	Check-Out
05-02-2024 10:00	A	Emergency	Discharge	Check-Out

Figure 1: Operational data recorded as event tables: discrete transitions with timestamps.

Table 1: Traditional versus data-driven modeling pipelines.

Traditional Pipeline	**Data-Driven Pipeline (Ours)**
1. Select a queueing network model, e.g., $M/M/n$ or $M/M/n$+M.	1. Frame a single learning problem: predict the next event and its time.
2. Calibrate parameters (arrival rate, service rate, abandonment rate).	2. Fit a high-capacity sequence model via stochastic gradient descent.
3. Code up a discrete-event simulator.	3. Generate trajectories by autoregressive sampling.

data streams invite a fundamental rethinking—instead of hand-crafting a structural model and coding a discrete-event simulator, we can now learn a system's dynamics directly and entirely from the data.

We propose a new approach termed *data-driven stochastic modeling with autoregressive sequence models*: a generative framework that maps an event history to a distribution over the next event type and its occurrence time (Table 1). In particular, inspired by the success of autoregressive sequence models such as Transformers in modeling sequential data like natural language [13, 4], we investigate how these architectures can be adapted to the event-stream setting that dominates operations management. Once trained, the model acts as a black-box simulator, generating realistic trajectories without requiring explicit knowledge of queues, service disciplines, or routing rules.

This paradigm offers three practical benefits: (i) **Expressivity**—it naturally captures non-Markovian and cross-resource dependencies that are hard to encode analytically; (ii) **Built-in uncertainty quantification (UQ)**—autoregressive sampling yields full predictive distributions enabling UQ; and (iii) **Versatility**—a single trained model can support what-if analysis, policy evaluation, and policy optimization without retraining. (The full version of the paper is available here.)

2 Conceptual Framework and Methodology

Operational data are typically available in the form of event tables, which record a sequence of events along with their corresponding timestamps. Let the observed sequence of events and inter-event times be denoted by $\{(E_i, T_i), i = 1, \ldots\}$, where $E_i \in \mathcal{E}$ denotes the type of the ith event, and T_i is the time elapsed between event $i-1$ and event i. We assume that this data is generated by an *underlying stochastic process* $X_{0:\infty} = \{X(t) : t \geq 0\}$ defined on the probability space $(\Omega, \mathcal{F}, \mathbb{P})$ with $X(t)$ taking values in $(\mathcal{S}, \Sigma)$. The law on the path space $(\mathcal{S}^{[0,\infty)}, \mathcal{G})$ is denoted by $\mathbb{P}$. For any observed path $x_{0:t} = \{x(s) : 0 \leq s \leq t\}$, the conditional law of the future trajectory is denoted by $\mathbb{P}(\cdot \mid x_{0:t})$. In many service systems, the underlying stochastic process takes the form of a jump process, where the system visits discrete states for random durations: $\{(S_i, T_i), i = 1, 2, \cdots\}$ with transition primitives:

$$\mathbb{P}(T_n = t \,|\, S_1, T_1, \ldots, S_n)\,, \qquad \mathbb{P}(S_{n+1} = s \,|\, S_1, T_1, \ldots, S_n, T_n)\,. \tag{1}$$

Importantly, we do not assume any structural properties such as the Markov property and the process is allowed to exhibit full path dependence. Event tables can be viewed as a simplified encoding of the jump process, where states can be deduced from discrete events: given S_1 and $(E_1, \ldots, E_n)$, one can deterministically determine S_{n+1}, yielding dynamics directly in event space,

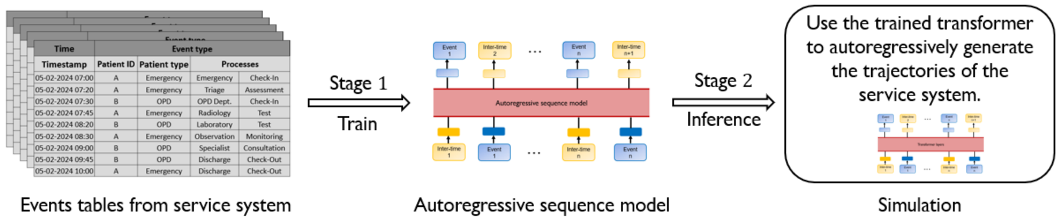
$$\mathbb{P}_n^{\text{time}}(T_n) = \mathbb{P}(T_n \mid S_1, T_{1:n-1}, E_{1:n-1}), \quad \mathbb{P}_n^{\text{event}}(E_n) = \mathbb{P}(E_n \mid S_1, T_{1:n-1}, E_{1:n-1}, T_n), \tag{2}$$

Thus the continuous-time trajectory $X_{0:\infty}$, the jump process $\{(S_i, T_i)\}$, and the event table $\{S_1, T_1, E_1, T_2, E_2, \ldots\}$ are mathematically equivalent. We adopt the event-table view because typically $|\mathcal{E}| \ll |\mathcal{S}|$ and logging is natural in this format (e.g. $M/M/1$ queue has only two events: arrival, departure). The joint distribution over the inter-event times, and event types can be written as a product of the autoregressive conditional distributions: $\mathbb{P}(S_1, T_{1:n}, E_{1:n}) = \mathbb{P}_0(S_1) \prod_{i=1}^{n} \mathbb{P}_i^{\text{time}}(T_i)\, \mathbb{P}_i^{\text{event}}(E_i)$.

The primary goal of a stochastic modeler is to simulate plausible trajectories of the service system $(S_1, E_{1:\infty}, T_{1:\infty}) \sim \mathbb{P}$, enabling prediction or other downstream analyses (e.g., estimating wait times, counterfactual analysis). In traditional framework, an expert specifies a structural model $\widetilde{\mathbb{P}}$ and then builds a discrete-event simulator. We instead *learn the conditionals* $\mathbb{P}_n^{time}(t)$ *and* $\mathbb{P}_n^{event}(e)$ for all $n \geq 1$ directly from data. Let the conditional distributions be parameterized by ϕ: $\widehat{P}_{\phi,n}^{time}(T_n)$ and $\widehat{P}_{\phi,n}^{event}(E_n)$. The resulting joint predictive model is given by $\widehat{P}_\phi(S_1, T_{1:n}, E_{1:n}) = \widehat{P}_{\phi,0}(S_1) \prod_{i=1}^n \widehat{P}_{\phi,i}^{\text{time}}(T_i)\, \widehat{P}_{\phi,i}^{\text{event}}(E_i)$. The model $\widehat{P}_\phi$ is trained to approximate the true data-generating distribution $\mathbb{P}$ by minimizing $D_{\text{kl}}\left(\mathbb{P}(\cdot) \| \widehat{P}_\phi(\cdot)\right) = -\mathbb{E}_{(S_1, E_{1:\infty}, T_{1:\infty}) \sim \mathbb{P}}[\log \widehat{P}_\phi(S_1, E_{1:\infty}, T_{1:\infty})] + \text{const}$. We can further decompose this objective as:

$$-\mathbb{E}_{S_1 \sim \mathbb{P}_0}\big[\log \widehat{P}_{\phi,0}(S_1)\big] + \sum_{i \geq 1} -\mathbb{E}_{T_i \sim \mathbb{P}_i^{\text{time}}}\big[\log \widehat{P}_{\phi,i}^{\text{time}}(T_i)\big] + \sum_{i \geq 1} -\mathbb{E}_{E_i \sim \mathbb{P}_i^{\text{event}}}\big[\log \widehat{P}_{\phi,i}^{\text{event}}(E_i)\big].$$

Training and simulation: Given K event tables, each of length N, generated from a service system $\mathbb{P}(\cdot)$, we train our model by minimizing the objective $\mathcal{L}(\phi) = \frac{1}{K} \sum_{j=1}^K \{-\log \widehat{P}_{\phi,0}(S_1^{(j)}) - \sum_{i=1}^N \log \widehat{P}_{\phi,i}^{\text{time}}(T_i^{(j)}) - \sum_{i=1}^N \log \widehat{P}_{\phi,i}^{\text{event}}(E_i^{(j)})\}$, using (stochastic) gradient descent. Once trained, the sequence model can be used to generate system trajectories in an autoregressive manner (Fig. 2) from $\widehat{P}_\phi(\cdot \mid S_1, T_{1:n}, E_{1:n})$. We adopt Transformer [13] as the backbone of our sequence model, motivated by its scalability and accessibility [4, 8]. Crucially, our methodology is model-agnostic and can be adapted to any future deep-learning architecture that offers greater computational efficiency.


Stage 1
Train
Event 1
Inter-time 2
Event n
Inter-time n+1
Autoregressive sequence model
Inter-time 1
Event 1
Inter-time n
Event n
Stage 2
Inference
Use the trained transformer to autoregressively generate the trajectories of the service system.
Events tables from service system
Autoregressive sequence model
Simulation

Time	Event type			
Timestamp	Patient ID	Patient type	Processes	
05-02-2024 07:00	A	Emergency	Emergency	Check-In
05-02-2024 07:20	A	Emergency	Triage	Assessment
05-02-2024 07:30	B	OPD	OPD Dept.	Check-In
05-02-2024 07:45	A	Emergency	Radiology	Test
05-02-2024 08:20	B	OPD	Laboratory	Test
05-02-2024 08:30	A	Emergency	Observation	Monitoring
05-02-2024 09:00	B	OPD	Specialist	Consultation
05-02-2024 09:45	B	OPD	Discharge	Check-Out
05-02-2024 10:00	A	Emergency	Discharge	Check-Out

Figure 2: Two stages of implementing our approach.

Generalizing the formulation: Our formulation naturally extends to incorporate parameter uncertainty $\theta \sim \mu$, as well as the control policy π by defining the process law as $\mathbb{P}(X_{0:\infty} \mid \pi) = \int P_{\pi,\theta}(X_{0:\infty})\, d\mu(\theta)$. Under this setting, the conditional distribution is given by $\mathbb{P}_n^{\text{time}}(T_n \mid \pi) = \int P_{\pi,\theta}(T_n \mid S_1, T_{1:n-1}, E_{1:n-1})\, d\mu(\theta \mid S_1, T_{1:n-1}, E_{1:n-1}, \pi)$. Similarly we can define $\mathbb{P}_n^{\text{event}}(E_n \mid \pi)$.

3 Experiments and Theoretical Insight

In this section we present a rigorous empirical evaluation of our approach across a range of downstream tasks, and a key theoretical insight into the performance of a trained sequence model.

Empirical validation. We begin by validating our methodology on canonical Markovian parallel server queues, such as multi-class $M/M/n$ queues, under different scheduling policies. We then demonstrate its scalability to more complex settings, including non-Markovian queues such as $G/G/1$ queue and non-stationary systems such as $M_t/M/n$ queue. Additionally, we conduct a case study using real-world call center data, involving a tandem queueing network with customer abandonment and multiple customer classes. In all experiments, we train a Transformer on event–time sequences $\mathcal{D}^{\text{train}} = \{(S_1^{(j)}, T_1^{(j)}, E_1^{(j)}, \ldots, T_N^{(j)}, E_N^{(j)}) : 1 \leq j \leq K\}$, generated by a discrete-event simulator from the target system. To evaluate the Transformer's performance, we generate trajectories using the trained Transformer and compare the resulting distributions of key performance metrics (e.g. inter-arrival times, service times, and waiting times) with those obtained from the ground-truth simulator. As shown in Figure 3, the results demonstrate strong alignment. These experiments demonstrate the flexibility of our Transformer-based approach in modeling a broad range of queueing dynamics.

Uncertainty quantification. We further evaluate our method's ability to quantify uncertainty in performance metrics when the system's transition dynamics are uncertain. To this end, we consider a $M/M/1$ queue where the parameters $\theta :=$ (arrival rate, service rate) are sampled from a prior. We train the transformer-based model on data generated under these priors. For evaluation, we compare the posterior distributions of trajectory-level averages (e.g. waiting times) produced by our model to those obtained by traditional Bayesian inference approach (Oracle). We observe a strong agreement between the two (Figure 3), confirming our method's natural ability to quantify uncertainty.

Counterfactual simulation for staffing ($M_t/M/N$). We also showcase our method's capability for counterfactual analysis, a key prerequisite for policy optimization in service operations. We consider an $M_t/M/N$ service system, where the arrival rate varies each day, characterized by a day-specific baseline c. The objective is to determine how many servers $N \in \{2, 3, \ldots, 20\}$ to employ. Conditioned on the baseline c and each candidate server N, the trained transformer model generates counterfactual trajectories to estimate time-varying waiting times. As illustrated in Figure 3, the model's estimates closely align with those from a discrete-event simulator, enabling efficient and data-driven staffing decisions.

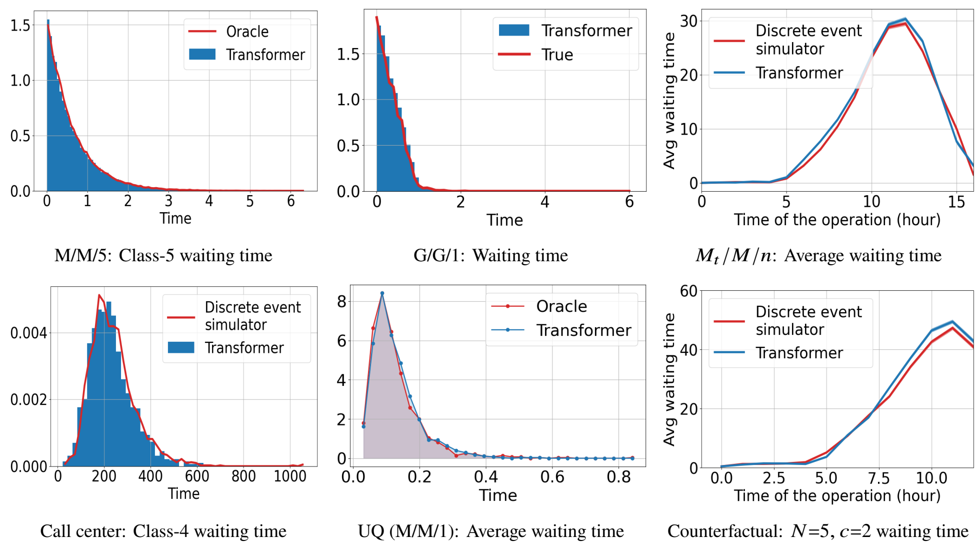


Figure 3: Representative results from experiments

Data requirements and towards foundation models: Naturally, the flexibility of our method comes at a cost of increased data demand, a condition increasingly satisfied in modern service systems. For M/M/1 we require ∼1k–2k event tables, M/M/5 needs ∼3k–4k for comparable convergence. As expected, required data increases with structural/event-type complexity. When real-world data are scarce, a promising alternative is to generate synthetic event data using high-fidelity simulators for pretraining foundation models of service systems, which can then be fine-tuned to specific environments with smaller amounts of domain-specific data. We conduct a preliminary experiment with a Transformer pretrained across four three-node networks and observe effective in-context adaptation: with only 10–15 context events it typically infers the correct network and generates valid continuations. This suggests scalable, general-purpose queueing simulators are feasible.

Key theoretical insight: We study how accurately a trained sequence model $\widehat{P}_\phi$ ($\widehat{X}_{0:t} \sim \widehat{P}_{0:t}$) estimates a performance functional f (e.g., average waiting time) relative to the true process $\mathbb{P}$. To enable theoretical analysis, we assume that both $\mathbb{P}$ and $\widehat{P}_\phi$ are mixtures over a family of positive Harris recurrent Markov processes $\{P_\theta\}$, satisfying standard regeneration and moment conditions, but differing in their respective mixing measures $\mu(\theta)$ and $\widehat{\mu}(\theta)$. For time-averaged metrics $f(X_{0:t}) = \frac{1}{t}\int_0^t h(X_s)\,ds$, the limit $f(\theta)$ exists almost surely under P_θ. Our main result establishes,

$$W_1\big(f(\widehat{X}_{0:t}),\ f(\theta)\big) \ \le\ c_1\sqrt{D_{\mathrm{kl}}\,(\mu\|\widehat{\mu})}\ +\ c_2/\sqrt{t} \qquad \text{for all } t > 0.$$

The first term on right-hand side, quantifies the error arising from imperfect learning; and we note that $D_{\mathrm{kl}}\left(\mathbb{P}_{0:t}\|\widehat{P}_{0:t}\right) \rightarrow D_{\mathrm{kl}}\,(\mu\|\widehat{\mu})$. The second term (order $t^{-1/2}$) captures approximation error induced by the finite prediction horizon which persists even with a perfect model for finite t. In essence, model's utility is governed by our ability to minimize its training loss. Encouragingly, modern machine learning practices are well-suited to minimizing such losses effectively at scale.

Conclusion: In this work, we propose a novel framework for stochastic modeling that leverages autoregressive sequence models to learn system dynamics directly from event-stream data. This approach significantly lowers the barrier to access for sophisticated stochastic modeling, enabling the development of high-fidelity simulators using modern AI tools and readily available operational data. Promising directions for future research include the development of queueing foundation models and the integration of learned simulators with policy optimization for automated decision-making.

References

[1] M. Armony, S. Israelit, A. Mandelbaum, Y. N. Marmor, Y. Tseytlin, and G. B. Yom-Tov. On patient flow in hospitals: A data-based queueing-science perspective. *Stochastic systems*, 5(1): 146–194, 2015.

[2] S. Banerjee, C. Riquelme, and R. Johari. Pricing in ride-share platforms: A queueing-theoretic approach. *Available at SSRN 2568258*, 2015.

[3] D. Bertsimas and V. Misic. Robust queueing network models for service system optimization. *Operations Research*, 55(6):1022–1041, 2007. doi: 10.1287/opre.1070.0440. Optimization techniques using queueing networks to improve resource allocation.

[4] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, and A. Askell. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33*, 2020.

[5] J. Dong and O. Perry. Queueing models for patient-flow dynamics in inpatient wards. *Operations research*, 68(1):250–275, 2020.

[6] N. Gans, G. Koole, and A. Mandelbaum. Telephone call centers: Tutorial, review, and research prospects. *Manufacturing & Service Operations Management*, 5(2):79–141, 2003.

[7] L. V. Green, S. Savin, and B. Wang. Managing patient flow in hospitals: A queueing perspective. *Operations Research*, 54(1):1–13, 2006. doi: 10.1287/opre.1060.0268. Application of queueing theory to hospital resource management and patient flow.

[8] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models, 2020. URL `https://arxiv.org/abs/2001.08361`.

[9] G. Koole. Call center mathematics: A scientific method for service management. *Queueing Systems*, 66(1):65–90, 2010. doi: 10.1007/s11134-010-9190-y. Mathematical models for queueing networks and decision-making in service operations.

[10] N. Liu and S. Ziya. Panel size and overbooking decisions for appointment-based services under patient no-shows. *Production and Operations Management*, 23(12):2209–2223, 2014.

[11] P. Shi, M. C. Chou, J. G. Dai, D. Ding, and J. Sim. Models and insights for hospital inpatient operations: Time-dependent ED boarding time. *Management Science*, 62(1):1–28, 2016.

[12] F. Tao, H. Zhang, A. Liu, and A. Y. Nee. Digital twin in industry: State-of-the-art. *IEEE Transactions on industrial informatics*, 15(4):2405–2415, 2018.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, 2017.