Operational Digital Twin Instantiation for Edge-to-Cloud Service Platforms

Loris Belcastro, Fabrizio Marozzo, Domenico Talia, Paolo Trunfio
DIMES, University of Calabria, Rende (CS), Italy
Email: lbelcastro@dimes.unical.it, fmarozzo@dimes.unical.it, talia@dimes.unical.it, trunfio@dimes.unical.it

***Abstract*—Digital twins are increasingly used for performance evaluation in distributed service platforms such as edge-to-cloud environments, where twin models support computation offloading, resource coordination, and system-level monitoring. A key challenge is that unconstrained instantiation can result in twins that are difficult to validate with respect to simulator assumptions, hard to reproduce due to inconsistent execution conventions, or unstable under stochastic workload variability and regime changes. We address these issues by proposing a constraint-driven, schema-based methodology for the instantiation and governance of operational digital twin instances under an explicit *twin simulation contract*. The approach separates *admissible schemas*, which encode machine-checkable constraints ensuring simulability and reproducibility, from *operational schemas*, which preserve admissibility while embedding descriptor-conditioned instantiation rules induced offline. We evaluate the methodology on a canonical two-stage service pipeline and on an edge–cloud offloading scenario with network backpressure and tight resource bounds. Operational schemas reduce admission loss by up to 86% and 62% in the two scenarios, respectively, while lowering tail latency and reducing its inter-configuration variability by over an order of magnitude. The results show that admissibility alone does not ensure reliable comparative evaluation, and that contract-aware configuration governance improves robustness and predictability under bounded resources. The complete framework, including the simulator and configuration artifacts, is publicly released to enable full reproduction of the experimental results.**

***Index Terms*—Digital twins, edge computing, edge-to-cloud, constraint-driven generation, performance evaluation**

I. Introduction

Edge-to-cloud infrastructures are widely adopted to deploy latency-critical services, enabling flexible placement and computation offloading across heterogeneous resources. In such environments, operational Digital Twins (DTs) support simulation-based what-if evaluation for capacity planning and configuration assessment under stochastic workloads and regime changes. DTs originated in manufacturing and cyber-physical production, but are now increasingly applied to computing infrastructures and distributed service platforms, including edge-to-cloud systems where twin models can represent dynamic network conditions, resource usage, and computation tasks [1].

DT engineering is often framed within a lifecycle perspective spanning creation, production, operation, and end-of-life of the target system and its digital counterpart [2]. Within this broad view, a critical transition lies between the initial creation of a valid digital representation and its reliable use during operation for predictive and exploratory analyses in edge-to-cloud service platforms.

In many application settings, this lifecycle is complemented by continuous data-driven synchronization mechanisms that keep the twin aligned with an evolving operational system [3]. However, independently of synchronization, effective evaluation fundamentally requires the ability to generate twin instances that are valid, reproducible, and usable under the modeling assumptions of the chosen simulator.

For edge-to-cloud service platforms, this requirement is particularly stringent. Workload burstiness and shared-resource variability can strongly affect tail behavior in large services, where even minor configuration changes may lead to substantial differences in tail latency and loss [4]. Twin instances must therefore remain admissible, simulatable, and reproducible despite stochastic fluctuations, backpressure effects, and regime shifts that arise naturally in edge-to-cloud deployments.

Our work addresses the transition from twin creation to operational use by introducing a constraint-driven methodology for generating operational DT configurations under an explicit simulation contract. Here, the term operational does not imply modifying or optimizing the physical infrastructure online, but rather ensuring that twin instances are reliable and robust enough to support stable simulation-based evaluation across admissible variations—a setting that is especially relevant when generating twins for edge-to-cloud offloading decisions.

Unconstrained instantiation may produce twins that are difficult to validate against simulator assumptions, hard to reproduce due to inconsistent execution conventions, or unstable under workload variability [3], [1], [5]. To overcome these limitations, our methodology formalizes the separation between admissibility constraints—ensuring simulability and reproducibility—and offline operational specialization rules, enabling the single-pass generation of small, stable families of valid twin variants.

We evaluate our methodology in an edge-to-cloud offloading setting. We first study a *canonical service pattern* (a two-stage service chain) as a controlled baseline to isolate stochastic variability effects in a minimal but representative model of latency-critical service pipelines. We then evaluate the same admissible-to-operational specialization mechanism on an *edge-to-cloud offloading variant* with an explicit network transfer component that introduces additional backpressure and tighter resource bounds. Results show that operational schemas reduce tail latency by up to 27% and admission

loss by up to 86% in the canonical service pattern; in the edge-to-cloud setting, tail latency is reduced by about 12% and admission loss by about 62%, while also collapsing inter-twin variability and improving predictability across admissible variants.

The main contributions of this paper are summarized as follows:

- A constraint-driven methodology for the instantiation and governance of operational digital twins for edge-to-cloud service platforms, grounded in an explicit twin simulation contract.
- A two-layer schema formulation that separates admissibility constraints—ensuring simulability and reproducibility—from offline-induced operational specialization rules, enabling descriptor-conditioned single-pass generation of small, stable families of twin variants.
- Operability summaries based on intra-twin and inter-twin variability, complementing tail latency and admission loss to quantify stochastic sensitivity across replications and predictability across generated variants.
- An experimental evaluation organized to support edge-to-cloud offloading: a canonical service pattern baseline (Case Study I) followed by an edge-to-cloud offloading scenario with explicit network transfer (Case Study II), demonstrating improved robustness and reduced sensitivity under workload regime changes.

The discrete-event simulator used for the experimental evaluation, along with the complete twin simulation contract specification, is publicly available at https://github.com/SCAlabUnical/TWIN-OPS

The remainder of the paper is organized as follows. Section II reviews related work. Section III presents the proposed schema-based methodology and the offline induction pipeline. Sections IV and V instantiate the methodology on the canonical service-pattern baseline and on the edge-to-cloud offloading case study, respectively, and report experimental results. Section VI discusses results and Section VII concludes the paper.

II. Related Work

Digital twins (DTs) are commonly defined as digital counterparts of physical or cyber-physical systems that integrate models, data, and runtime connectivity to support monitoring, prediction, and decision making across the system lifecycle [2], [3]. In practice, DT-based solutions often combine simulation models with data synchronization pipelines and layered architectural components. However, even when the overall structure is similar, implementations may rely on heterogeneous semantic assumptions and incompatible execution conventions, undermining reproducibility and making automated instantiation difficult to validate and govern [3], [6], [7]. The lack of explicit semantic agreements and contract-level execution rules further limits interoperability and hinders systematic comparison and replication of DT configurations across projects [8], [9].

DTs have become increasingly relevant in computing infrastructures and distributed service platforms, particularly in edge-to-cloud environments where computation and networking dynamics affect operational behavior. In these domains, twins support decision making tasks such as computation offloading, resource coordination, and performance-aware service management under time-varying workloads [1], [5]. Representative approaches address edge selection and task offloading policies [10], [11], and extend to collaborative and incentive-aware mechanisms in broader IoT/IIoT settings [12], [13]. However, these contributions typically assume that suitable twin instances are already available and focus on online optimization or control, rather than on governing the instantiation layer under explicit simulation semantics.

A complementary line of work investigates the disciplined construction of simulation-ready models through explicit specifications and constraint-driven generation, highlighting the role of machine-checkable specifications and reusable structural archetypes in improving validity and reproducibility of generated models [14]. In stochastic computing infrastructures, where workload burstiness and shared-resource variability dominate tail behavior [4], reproducibility and predictability across admissible configurations become critical prerequisites for trustworthy what-if analysis. Yet, many studies leave implicit the constraints and execution contracts required to ensure that automatically generated twin instances remain simulatable, reproducible, and operationally stable.

Another related direction concerns DT-based testing and validation. Digital-twin-based testing stresses the importance of explicit correctness criteria, such as test oracles and conformance relations, to decide whether observed behavior is acceptable. Defining such criteria becomes more challenging when expected behavior shifts across operating contexts, motivating approaches that make evaluation conventions explicit and machine-checkable [15]. These perspectives reinforce the need for contract-driven governance mechanisms bridging model validity and operational usability.

Overall, while DT research in edge-to-cloud computing has explored monitoring, optimization, and control, the automated instantiation of operationally reliable DT configurations remains comparatively less specified, especially when only a small family of variants must be generated in a single pass under stochastic regime changes. Our work addresses this gap through constraint-driven schemas and offline-induced instantiation rules that ensure admissibility, reproducibility, and predictable operational behavior across generated twin families.

III. Operational Digital Twin Schemas via Offline Induction

We now formalize our constraint-driven methodology for constructing *operational digital twin schemas* from an admissible family of valid twins. Contract-level *admissibility* guarantees simulability and reproducibility, while offline-induced *operational specialization* encodes descriptor-conditioned rules for stable single-pass instantiation.

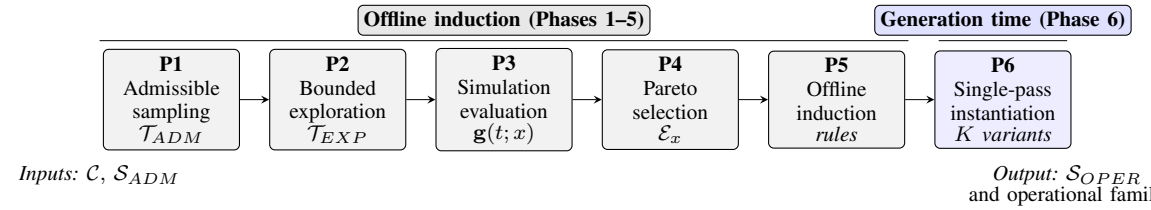


Fig. 1. Six-phase methodology: Phases 1–5 induce descriptor-conditioned rules offline; Phase 6 instantiates a small operational family in a single pass.

As summarized in Fig. 1, the methodology follows a six-phase pipeline. Phases 1–5 are executed offline to evaluate admissible candidates under descriptor-defined regimes and to induce deterministic instantiation rules, while Phase 6 generates, at run time, a small operational family in a single pass given a descriptor of the intended operating conditions. By construction, all instantiated twins remain admissible and therefore simulatable and reproducible under the shared contract semantics.

A. Methodological building blocks

Twin simulation contract: A *twin simulation contract* $\mathcal{C}$ specifies the normative semantics shared by the generator and the simulator. At a minimum, it fixes (i) the meaning of reported indicators, (ii) event semantics (e.g., queueing, loss, completion), (iii) admissible parameter domains, and (iv) reproducibility conventions (e.g., deterministic seeding and tie-breaking). The contract acts as the semantic anchor that makes admissibility machine-checkable and simulation outcomes comparable across generated twins.

Admissible schema: An *admissible schema* $\mathcal{S}_{ADM}$ defines a family of twins that are valid by construction. In practice, it encodes: (i) structural constraints (well-formed topology), (ii) semantic constraints (parameter bounds and invariants), and (iii) deterministic semantics (fixed policies and consistent execution rules). Sampling from $\mathcal{S}_{ADM}$ produces diverse twins that are all simulatable and reproducible under $\mathcal{C}$. Admissibility, however, does not by itself guarantee operational stability or predictable performance under stochastic variability.

Operational schema: An *operational schema* $\mathcal{S}_{OPER}$ preserves admissibility while biasing generation toward operationally effective behavior under explicit resource bounds. Rather than performing online optimization or returning a single offline optimum, $\mathcal{S}_{OPER}$ encodes deterministic descriptor-conditioned instantiation rules: given a descriptor $x \in \mathcal{X}$, it generates a seed configuration and a small family of admissible operational variants around that seed.

Descriptors for regime-dependent specialization: A descriptor x identifies the intended operating conditions under which an operational twin should be robust (e.g., regime label, environment class, workload/stress level). We assume x is chosen from a finite set $\mathcal{X}$ defined *a priori*; offline induction and instantiation are performed independently for each $x \in \mathcal{X}$. Descriptors provide the handle through which offline specialization is tied to regime-dependent operating conditions.

B. Six-phase pipeline

We outline the pipeline in six phases. Each phase is identified by its main operation, and the relevant artifacts are introduced in context.

Phase 1: Admissible sampling: Starting from $\mathcal{C}$ and $\mathcal{S}_{ADM}$, we sample an initial set of admissible twins, denoted $\mathcal{T}_{ADM}$. These are syntactically and semantically valid artifacts and serve as starting points for exploration.

Phase 2: Bounded admissible exploration: We expand the candidate set by generating additional admissible variants in the neighborhood of sampled twins. Exploration operates under an explicit budget (e.g., maximum variants per seed and/or maximum evaluated candidates per descriptor), producing an expanded admissible set $\mathcal{T}_{EXP}$.

Phase 3: Simulation-based evaluation: For each descriptor $x \in \mathcal{X}$, candidates in $\mathcal{T}_{EXP}$ are simulated under the scenarios associated with x, denoted $\Sigma(x)$, using R independent replications per scenario. This phase returns per-candidate estimates of a small number of *primitive* operational indicators, defined by the contract and the domain, that will be used for selection in Phase 4.

Phase 4: Multi-criteria selection with cost-aware size control: We perform selection using a multi-criteria (Pareto) filter. Let

$$\mathbf{g}(t;x) = \big(g_1(t;x), \ldots, g_m(t;x)\big)$$

be the vector of *primitive* operational indicators returned by Phase 3 for candidate t under descriptor x (each g_i is estimated by averaging over replications). We assume that all primitive indicators are to be minimized. A candidate t *dominates* t' if $g_i(t;x) \le g_i(t';x)$ for all $i \in \{1, \ldots, m\}$, with at least one strict inequality. The non-dominated set for descriptor x is denoted $\mathcal{P}_x$.

To obtain a deterministic finite elite subset suitable for offline induction, we apply a monotone resource cost function $Cost(t)$ only for size control and tie-breaking within $\mathcal{P}_x$. Specifically, if $|\mathcal{P}_x| > E$, we select an elite set $\mathcal{E}_x \subseteq \mathcal{P}_x$ consisting of the E lowest-cost candidates under a deterministic tie-breaking rule. If $|\mathcal{P}_x| \le E$, we set $\mathcal{E}_x = \mathcal{P}_x$. This phase produces $\mathcal{E}_x$ for each descriptor x.

Phase 5: Offline induction of operational rules: For each descriptor x, we induce a deterministic *seed rule* that maps x to a seed twin t_{seed}, and a deterministic *perturbation rule* that defines an admissible bounded move set around t_{seed}. We select t_{seed} as a *representative* element of the elite set $\mathcal{E}_x$ (the medoid under a normalized ℓ_1 configuration-space distance), with deterministic tie-breaking.

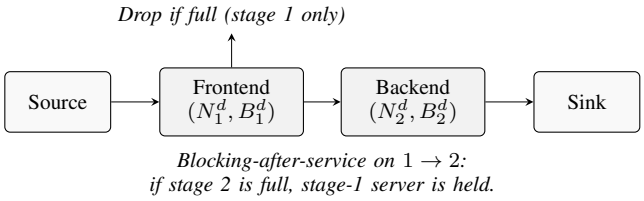


Fig. 2. Canonical service pattern baseline: two-stage service chain model.

Phase 6: Single-pass instantiation of operational families: At generation time, given a descriptor x, the operational schema applies the induced rules to produce t_{seed} and instantiate a family of K operational variants by executing bounded perturbations around the seed. If a perturbation would violate admissible bounds, it is handled deterministically via a repair (or null-move) rule, so that all generated variants remain admissible by construction.

IV. Case Study I: Canonical Service Pattern Baseline

This section instantiates the proposed methodology on a *canonical service pattern* modeled as a two-stage processing pipeline, capturing the core structure of a data-center (DC) request-processing service. The purpose of this case study is to provide a controlled baseline that isolates stochastic variability effects and makes it possible to quantify intra-/inter-twin dispersion under an explicit discrete-event contract. This baseline is then used to motivate and structure the edge-to-cloud evaluation in Case Study II (Section V).

We define an admissible schema (DC_ADM) capturing structural and semantic constraints ensuring simulability and reproducibility, and derive an operational schema (DC_OPER) via offline induction. The objective is to assess whether offline-induced instantiation rules can improve predictability under regime changes without requiring online optimization. We stress the system using deterministic burst windows and evaluate both nominal and bursty regimes through two primitive indicators: *tail latency* and *drop rate*. In addition, we report intra-/inter-twin variability of the same indicators as measures of operational stability (Section IV-H). Twins are generated either by direct sampling from DC_ADM (ADM) or by instantiation via DC_OPER (OPER), and compared under multiple replications. All numeric constants for this case study are summarized in Table I.

A. System Model

As shown in Fig. 2, we model a service chain as a canonical two-stage processing pipeline:

$$\text{Source} \to \text{Frontend} \to \text{Backend} \to \text{Sink}.$$

The two service stages are indexed by $k \in \{1, 2\}$, where $k = 1$ denotes the Frontend and $k = 2$ the Backend. A DT configuration is defined by the per-stage server counts and buffer sizes $\{N_k^d, B_k^d\}_{k=1}^2$, and by the service-time parameters. For the canonical two-stage schema, we instantiate the medoid distance over $(N_1^d, N_2^d, B_1^d, B_2^d)$, normalized by the admissible parameter ranges. Service times are exponentially distributed with means m_1^d and m_2^d (Table I). The workload enters the system at the Source and traverses the two stages sequentially until completion at the Sink. We evaluate tail latency and drop rate under steady and burst-window workloads (Section IV-H).

B. Twin Simulation Contract Instantiation

The twin simulation contract instantiates a deterministic discrete-event semantics for the pipeline, removing implementation ambiguity and ensuring reproducibility. Each stage k consists of N_k^d identical non-preemptive servers with a FIFO waiting buffer of size B_k^d, yielding total capacity $N_k^d + B_k^d$.

Admission control is enforced only at stage 1, while downstream finite capacity is handled via blocking-after-service on the $1 \to 2$ transfer. The contract further specifies deterministic event ordering, tie-breaking, and replication seeding conventions, as documented in the released reference implementation. Appendix A summarizes the essential clauses.

C. Admissible Schema (DC_ADM)

A DC twin is admissible if, for $k \in \{1, 2\}$,

$$N_k^d \in [N_{\min}^d, N_{\max}^d], \quad B_k^d \in [B_{\min}^d, B_{\max}^d],$$

where the concrete bounds are reported in Table I, and all simulator semantics follow the simulation contract.

D. Descriptor Set

For this case study we instantiate the general descriptor notion as $x = (\rho, b)$, where ρ denotes a (dimensionless) load level and $b \in \{0, 1\}$ is a stress flag distinguishing steady workload ($b = 0$) from deterministic burst windows ($b = 1$). The specific ρ values used in this paper are reported in Table I.

E. Workload Scenarios

We evaluate both steady and bursty arrival regimes.

- *Steady workload.* Arrivals follow a Poisson process with constant rate $\lambda_{base}^d(\rho)$.
- *Deterministic burst windows.* We introduce bursty workloads by periodically switching the Poisson arrival rate between a low level λ_{low}^d and a high level λ_{high}^d. Bursts occur every P time units and last for W, resulting in duty cycle $d = W/P$ and burst factor $F = \lambda_{high}^d / \lambda_{low}^d$ (Table I).

Arrival rates are calibrated by defining a nominal backend reference capacity as

$$C_{ref}^d = N_{2,ref}^d \cdot (1/m_2^d),$$

and setting the steady baseline rate to $\lambda_{base}^d(\rho) = \rho \cdot C_{ref}^d$. For burst windows, we choose λ_{low}^d and λ_{high}^d such that (i) the time-average arrival rate equals λ_{base}^d and (ii) $\lambda_{high}^d / \lambda_{low}^d = F$:

$$\lambda_{low}^d = \frac{\lambda_{base}^d}{(1-d) + dF}, \qquad \lambda_{high}^d = F\,\lambda_{low}^d. \tag{1}$$

This choice preserves the same mean load λ_{base}^d while controlling burstiness through F and d.

F. Operational Schema (DC_OPER)

DC_OPER is obtained by specializing DC_ADM through offline induction. For each descriptor x, we evaluate a bounded candidate set and select a deterministic representative seed from the elite region. Starting from this seed, the operational schema instantiates a small family of K admissible twins by applying controlled perturbations.

Operational variants are generated through small *budget-preserving* paired perturbations of service capacity and buffering allocations. These moves are designed to keep the total server budget $(N_1^d+N_2^d)$ and the total buffer budget $(B_1^d+B_2^d)$ constant whenever feasible, while still providing controlled diversity around the seed. Any perturbation that would violate admissible bounds is handled deterministically via a repair (or null-move) rule, ensuring that all generated variants remain admissible by construction.

G. Experimental Protocol

For clarity, we label as ADM the condition obtained by direct sampling from the admissible schema, and as OPER the condition obtained by instantiating the operational schema. We compare:

- ADM: K twins sampled from DC_ADM,
- OPER: K twins generated by perturbing the DC_OPER seed.

We use R independent replications per scenario and twin (Table I). For each twin and scenario, indicators are computed per replication. We then compute (i) per-twin averages (used for point estimates) and (ii) per-twin intra-variability across replications, before finally aggregating across the K twins.

H. Metrics

The following indicators are computed over the measurement window $[T_w, T]$.

Tail latency ($p99$): We quantify tail behavior using the 99th percentile of the end-to-end response time distribution. That is, RT_{p99} is the response-time threshold such that 99% of completed requests have response time at most RT_{p99} (and the slowest 1% exceed it). For each completed request, we measure the end-to-end response time RT from arrival at the Source to completion at the Sink, including all waiting and service components across the two stages. Let $\{RT_i\}_{i=1}^{C}$ be the response-time samples for completed requests in $[T_w, T]$, where C is the number of completed requests in the same window, sorted in non-decreasing order as $RT_{(1)} \le \cdots \le RT_{(C)}$. We use the nearest-rank definition:

$$RT_{p99} = RT_{(\lceil 0.99\, C \rceil)}.$$

Drop rate: Let A be the number of arrivals at the Source in $[T_w, T]$ and let D be the number of such arrivals that are dropped (i.e., rejected by stage 1 under the admission/drop rule). We define

$$DROP = \frac{D}{A}.$$

Operability summaries (intra-/inter-twin variability): Beyond point estimates, we report operability summaries derived from the same primitive indicators RT_{p99} and $DROP$. For each twin t under descriptor x, we compute replication-average indicators $\overline{RT_{p99}}(t;x)$ and $\overline{DROP}(t;x)$ by averaging over the R independent replications. We then quantify *intra-twin variability* as $\sigma_{RT}^{intra}(t;x)$ and $\sigma_{\mathrm{DROP}}^{intra}(t;x)$, i.e., the standard deviation of the indicator across replications for the same twin. To summarize intra-twin variability at the condition level, we report $\overline{\sigma_{RT}^{intra}}(x)$ and $\overline{\sigma_{\mathrm{DROP}}^{intra}}(x)$, obtained by averaging σ^{intra} over the K generated twins. Finally, we quantify *inter-twin variability* as $\sigma_{RT}^{inter}(x)$ and $\sigma_{\mathrm{DROP}}^{inter}(x)$, i.e., the standard deviation across the K twins of their replication-average indicators. Intra-twin variability captures sensitivity to stochastic noise, while inter-twin variability captures predictability across admissible variants generated under the same descriptor, which is desirable when twins are used for what-if evaluation rather than as one-off offline optima.

I. Indicators and Selection Instantiation

In this case study we instantiate the Phase 4 indicator vector with two primitive indicators:

$$\mathbf{g}(t;x) = \big(RT_{p99}(t;x),\ DROP(t;x)\big).$$

Cost is instantiated from provisioned resources (server counts and buffer sizes) through the monotone model in Eq. (2). Specifically, we define a normalized resource cost as a weighted sum of the per-stage average service capacity and buffering:

$$Cost(t) = w_N \cdot \frac{\frac{N_1^d+N_2^d}{2} - N_{\min}^d}{N_{\max}^d - N_{\min}^d} + w_B \cdot \frac{\frac{B_1^d+B_2^d}{2} - B_{\min}^d}{B_{\max}^d - B_{\min}^d}, \quad (2)$$

where lower values correspond to cheaper configurations. Cost is used only for Pareto-set size control and deterministic tie-breaking.

J. Results

All experiments were conducted using a SimPy-based discrete-event simulator implementing the contract semantics outlined in Section IV-B. The simulation parameters are detailed in Table I.

The results, reported in Table II, show that instantiation through the operational schema (OPER) consistently improves both primitive indicators with respect to direct sampling from the admissible schema (ADM). The average tail latency drops from ADM's $\overline{RT_{p99}} = 72.59$ ms to OPER's 53.15 ms (about a 27% reduction), while the average admission loss decreases sharply from $\overline{DROP} = 0.407$ to 0.056 (about an 86% reduction). This indicates that offline-induced, descriptor-conditioned rules do not merely preserve simulability, but effectively bias single-pass generation toward configurations that remain robust under the considered regimes.

Crucially, the operability summaries highlight a marked improvement in predictability. Under ADM, inter-twin variability is extremely large ($\sigma_{RT}^{inter} = 37.11$ ms and $\sigma_{\mathrm{DROP}}^{inter} = 0.273$), meaning that admissible twins sampled in the same way can

TABLE I
CANONICAL SERVICE PATTERN PARAMETERS.

Parameter	Value
Service means	$m_1^d = 5$ ms, $m_2^d = 10$ ms
Admissible bounds	$N_k^d \in [1, 12]$, $B_k^d \in [0, 50]$
Reference capacity	$N_{2,ref}^d = 6$
Horizon / warm-up	$T = 50{,}000$ ms, $T_w = 5{,}000$ ms
Burst schedule	$P = 5{,}000$ ms, $W = 1{,}000$ ms ($d = W/P$)
Burst factor	$F = 3$
Load regimes	$\rho \in \{0.75, 0.90\}$
Replications	$R = 30$
Operational family size	$K = 20$
Elite size cap	$E = 10$
Offline budgets	$M = 50$, $V = 10$ variants/seed, $B \leq 500$ eval.
Cost weights	$w_N = 0.8$, $w_B = 0.2$

TABLE II
CANONICAL SERVICE PATTERN RESULTS.

	RT_{p99} (ms)			$DROP$		
Condition	$\overline{RT_{p99}}$	σ_{RT}^{inter}	$\overline{\sigma_{RT}^{intra}}$	$\overline{DROP}$	$\sigma_{\text{DROP}}^{inter}$	$\overline{\sigma_{\text{DROP}}^{intra}}$
ADM	72.59	37.11	1.49	0.407	0.273	0.0036
OPER	53.15	0.26	0.68	0.056	0.061	0.0020

exhibit widely different behavior—an undesirable property when only a small family must be generated at run time. In contrast, OPER collapses inter-twin dispersion ($\sigma_{RT}^{inter} = 0.26$ ms; $\sigma_{\text{DROP}}^{inter} = 0.061$), producing families that are consistent across generated variants. Intra-twin variability across replications also decreases ($\overline{\sigma_{RT}^{intra}}$ from 1.49 to 0.68; $\overline{\sigma_{\text{DROP}}^{intra}}$ from 0.0036 to 0.0020), confirming reduced sensitivity to stochastic noise.

In addition to tail latency and admission loss, we report an outcome-oriented view through *goodput*, defined as the effective throughput of successfully completed requests, i.e., throughput multiplied by $(1 - DROP)$. Fig. 3 shows the distribution of goodput across the generated twin families, stratified by workload descriptor. While ADM can occasionally produce very high-goodput configurations, it also yields a wide spread with many low-goodput twins, reflecting a higher risk of poor selections when sampling from the admissible space. In contrast, OPER concentrates the instantiated family around consistently higher goodput levels across descriptors, reducing the mass of low-goodput outcomes and providing more reliable what-if evaluation when a small family of twins must be generated in a single pass.

V. CASE STUDY II: EDGE-TO-CLOUD OFFLOADING TARGET SCENARIO

Building on the canonical service-pattern baseline of Case Study I, we now consider an edge-to-cloud offloading target setting where resources are tighter and an explicit transfer component introduces additional backpressure. This scenario tests whether the same admissible-to-operational specialization mechanism remains effective under cross-domain coupling.

As shown in Fig. 4, the service logic remains a two-stage FIFO pipeline with admission at stage 1, but completed edge jobs must traverse a capacity-limited network stage before entering the cloud stage.

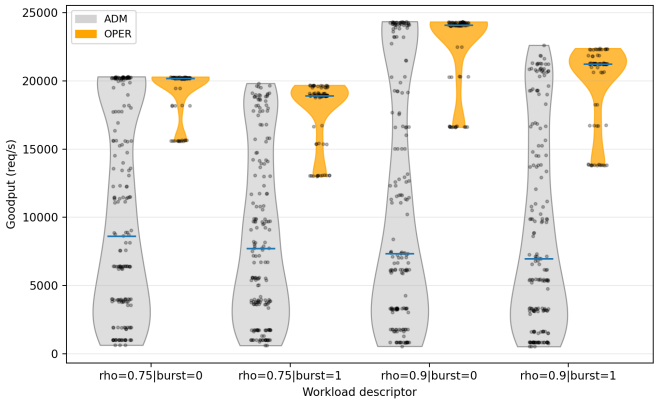


Fig. 3. Canonical service pattern baseline: goodput distributions across twins under each workload descriptor. ADM exhibits high dispersion, including many low-goodput twins, while OPER shifts the distribution toward consistently higher goodput values.

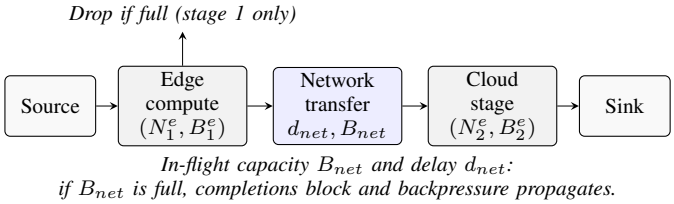


Fig. 4. Edge-to-cloud offloading target scenario with explicit network transfer.

Concretely, we insert a *Network Transfer* stage between the two service stages:

$$\text{Source} \rightarrow \text{Edge Compute} \rightarrow \text{Network Transfer} \rightarrow \text{Cloud Stage} \rightarrow \text{Sink}.$$

The network stage adds a deterministic delay d_{net} and an in-flight capacity B_{net} (Table III). When the in-flight capacity is saturated, stage-1 completions block before entering the transfer stage, propagating congestion upstream. The contract is identical to the baseline case, except for the network-transfer rule above.

EDGE_ADM uses the same structure as DC_ADM, with tighter edge-specific bounds on per-stage server counts and buffers (Table III). We keep the same descriptor set $x = (\rho, b)$, the same steady vs. burst workload regimes, and the same offline-induction and evaluation protocol used in Case Study I. The medoid-based seed selection rule is applied unchanged in this scenario. Only the reference capacity used for arrival-rate calibration changes to $C_{ref}^e = N_{2,ref}^e \cdot (1/m_2^e)$, yielding $\lambda_{base}^e(\rho) = \rho \cdot C_{ref}^e$. Indicators and operability summaries are computed as in Section IV-H, with the only difference that end-to-end response time additionally includes any blocking before network admission and the deterministic transfer delay d_{net}.

A. Results

The experiments were conducted using the same SimPy-based discrete-event simulator used in Case Study I, extended

TABLE III
EDGE-TO-CLOUD TARGET SCENARIO PARAMETERS.

Parameter	Value
Service means	$m_1^e = 8$ ms, $m_2^e = 12$ ms
Network transfer	$d_{net} = 5$ ms, $B_{net} = 20$
Admissible bounds	$N_k^e \in [1, 6]$, $B_k^e \in [0, 30]$
Reference capacity	$N_{2,ref}^e = 3$
Horizon / burst / regimes	T, T_w, P, W, F, ρ as in Table I
Replications / family size	R, K as in Table I
Elite size cap / budgets / weights	E, M, V, w_N, w_B as in Table I

TABLE IV
EDGE-TO-CLOUD TARGET SCENARIO RESULTS.

	RT_{p99} (ms)			$DROP$		
Condition	$\overline{RT_{p99}}$	σ_{RT}^{inter}	$\overline{\sigma_{RT}^{intra}}$	$\overline{DROP}$	$\sigma_{\text{DROP}}^{inter}$	$\overline{\sigma_{\text{DROP}}^{intra}}$
ADM	83.91	21.44	2.34	0.476	0.224	0.0049
OPER	73.47	0.82	1.41	0.180	0.068	0.0048

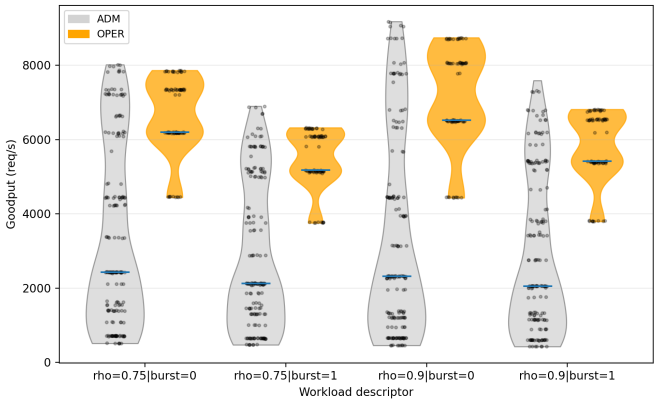


Fig. 5. Edge-to-cloud target scenario: goodput distributions across twins under each workload descriptor. ADM shows a wide spread with many low-goodput configurations, whereas OPER concentrates the operational family on higher goodput values, reducing the likelihood of poor selections under tight edge and transfer constraints.

with the network-transfer stage and implementing the contract semantics described in Section IV-B. Table III reports the edge-specific parameters.

Table IV reports point estimates and variability summaries. In the edge-to-cloud setting, the explicit network-transfer stage introduces additional backpressure and tighter resource bounds. Despite this more constrained regime, Table IV shows that OPER remains effective: $\overline{RT_{p99}}$ improves from 83.91 ms (ADM) to 73.47 ms (OPER), while $\overline{DROP}$ decreases substantially from 0.476 to 0.180.

Importantly, OPER also reduces sensitivity across generated variants, collapsing inter-twin dispersion from $\sigma_{RT}^{inter} = 21.44$ ms and $\sigma_{\text{DROP}}^{inter} = 0.224$ under ADM to 0.82 ms and 0.068, respectively. This increased consistency is reflected in Fig. 5, where OPER shifts the family toward higher goodput values and mitigates the occurrence of low-performing twins under tight edge and transfer constraints.

VI. DISCUSSION

The results support the main claim of this paper: in stochastic edge-to-cloud service platforms, generating *admissible* DTs is necessary but not sufficient for reliable evaluation. While admissible schemas guarantee contract-compatible simulability and reproducibility, direct sampling from $\mathcal{S}_{ADM}$ can still yield highly dispersed outcomes across equally valid variants, which is problematic when only a small family must be generated in a single pass.

Across both case studies, instantiation through the operational schema $\mathcal{S}_{OPER}$ improves the primitive indicators used for offline selection, RT_{p99} and $DROP$, while increasing *predictability*. In the canonical baseline, OPER reduces mean tail latency (72.59 ms $\rightarrow$ 53.15 ms) and admission loss (0.407 $\rightarrow$ 0.056). In the edge-to-cloud target scenario, where a finite-capacity network transfer stage introduces additional backpressure, OPER remains effective (83.91 ms $\rightarrow$ 73.47 ms; 0.476 $\rightarrow$ 0.180). This shows that descriptor-conditioned offline induction can bias single-pass generation toward robust configurations without online optimization.

The main operational advantage is not only in point estimates but in consistency across the generated family. Under ADM, inter-twin variability is large (e.g., 37.11 ms for RT_{p99} and 0.273 for $DROP$ in the baseline), implying that admissible twins sampled under the same descriptor can behave very differently. In contrast, OPER collapses inter-twin dispersion (e.g., 0.26 ms and 0.061), yielding families that are substantially more predictable across variants. This behavior is consistent with the goodput distributions (Figs. 3 and 5), where ADM exhibits wide spreads and many low-performing twins, whereas OPER concentrates outcomes on higher goodput values, reducing the risk of poor instantiations when only K variants are available.

Overall, operational schemas provide a lightweight governance layer: they preserve admissibility and reproducibility by construction, while encoding deterministic descriptor-conditioned rules that generate small families centered on robust seeds. In this sense, intra-/inter-twin variability summaries offer a compact assessment of stochastic sensitivity and predictability across admissible variants.

Oracle cost-matched best-K admissible selection. For reference, we also evaluated an oracle-like baseline obtained by selecting, for each descriptor, the best K twins from $\mathcal{S}_{ADM}$ under a total cost budget matched to the corresponding $\mathcal{S}_{OPER}$ family (closest feasible match; target tolerance $\approx 5\%$). Here, "best-K" denotes a Pareto-based oracle selection with respect to the same two primitive indicators $(RT_{p99}, DROP)$, i.e., twins are chosen from the non-dominated admissible set under the matched budget constraint. As expected, offline selection yields only marginal improvements in tail latency but substantially worsens admission loss. In the canonical service pattern, mean RT_{p99} improves from 53.23 ms (OPER) to 52.85 ms (best-K, -0.71%), while mean $DROP$ increases from 0.056 to 0.2175 ($+288.4\%$). In the edge-to-cloud target setting, mean RT_{p99} improves from 73.47 ms to 72.64 ms

(-1.13%), while mean $DROP$ increases from 0.1796 to 0.3553 ($+97.8\%$). This confirms that ex-post oracle selection within the admissible space may slightly reduce tail latency, but can drastically worsen admission loss; in contrast, OPER provides a more balanced and operationally stable trade-off in a single pass, without per-descriptor oracle search.

VII. Conclusions

We introduced a constraint-driven methodology for instantiating operational digital twins of edge-to-cloud service platforms under an explicit twin simulation contract. The approach relies on a principled separation between *admissible schemas*, which guarantee simulability and reproducibility through machine-checkable constraints, and *operational schemas*, which preserve admissibility while embedding offline-induced, descriptor-conditioned instantiation rules for single-pass generation.

Experimental results on a canonical service pattern baseline and on an edge-to-cloud offloading target scenario with explicit network backpressure show that operational schemas improve both tail latency and admission behavior with respect to direct admissible sampling. More importantly, they collapse inter-twin variability, yielding small operational families whose behavior is predictable across generated variants.

These findings confirm that contract-compatible validity is necessary but not sufficient for trustworthy simulation-based evaluation in stochastic edge-to-cloud infrastructures. Schema-level operational specialization provides an effective governance mechanism to generate reproducible and descriptor-aware twin families for operational what-if analysis. Future work will explore the application of this methodology to more complex service workflows and larger edge–cloud systems.

Acknowledgment

This work was funded by MUR (D.D. No. 307, PN RIC 2021–2027, Project ECHO TWIN) and co-funded by the European Union (ERDF). Azione 1.1.2_CUP: B99H26000290007; Azione 1.1.3b_CUP: B92F26000440005, Azione 1.4.3_CUP: B89J26003490005.

Appendix A
Twin Simulation Contract Essentials

We summarize here the essential clauses of the twin simulation contract $\mathcal{C}$. The full specification is available at https://github.com/SCAlabUnical/TWIN-OPS

- *Stage semantics.* Each service stage k consists of N_k identical non-preemptive servers fed by a FIFO waiting buffer of size B_k (waiting only), yielding total in-stage capacity $N_k + B_k$.
- *Admission and drop rule.* Admission control is enforced only at stage 1. After processing all same-time service completions, an arrival is admitted if capacity is available; otherwise it is dropped immediately.
- *Blocking-after-service transfer.* Downstream finite capacity is handled via blocking-after-service on the $1 \to 2$ transfer: after completing service at stage 1, a request enters stage 2 only if capacity is available; otherwise it blocks and retains the stage-1 server until the transfer becomes feasible.
- *Edge-to-cloud transfer extension.* In Case Study II, a deterministic network-transfer stage is inserted between edge and cloud processing. It introduces a fixed delay d_{net} and an in-flight capacity B_{net}. When B_{net} is saturated, completions block before entering transfer, propagating backpressure upstream.
- *Event ordering and determinism.* The contract specifies a deterministic event ordering and tie-breaking policy for simultaneous events (e.g., completions processed before arrivals), ensuring that all admissible twins are simulatable and outcomes are comparable across generated variants.
- *Reproducibility conventions.* All simulation results follow fixed replication seeding and deterministic execution conventions documented in the reference implementation.

References

[1] H. Tran-Dang and D.-S. Kim, "Digital twin-empowered intelligent computation offloading for edge computing in the era of 5g and beyond: A state-of-the-art survey," *ICT Express*, vol. 11, no. 1, pp. 167–180, 2025.

[2] M. Grieves and J. Vickers, "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems," in *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches*, F.-J. Kahlen, S. Flumerfelt, and A. Alves, Eds. Springer, 2017, pp. 85–113.

[3] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital twin: Enabling technologies, challenges and open research," *IEEE Access*, vol. 8, pp. 108 952–108 971, 2020.

[4] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013.

[5] Y. Wu, K. Zhang, and Y. Zhang, "Digital twin networks: A survey," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13 789–13 804, 2021.

[6] C. Semeraro, M. Lezoche, H. Panetto, and M. Dassisti, "Digital twin paradigm: A systematic literature review," *Computers in Industry*, vol. 130, p. 103469, 2021.

[7] A. Sharma, E. Kosasih, J. Zhang, A. Brintrup, and A. Calinescu, "Digital twins: State of the art theory and practice, challenges, and open research questions," *Journal of Industrial Information Integration*, vol. 30, p. 100383, 2022.

[8] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, "Characterising the digital twin: A systematic literature review," *CIRP Journal of Manufacturing Science and Technology*, vol. 29, pp. 36–52, 2020.

[9] K. J. Kuehner, R. Scheer, and S. Strassburger, "Digital twin: Finding common ground – a meta-review," *Procedia CIRP*, vol. 104, pp. 1227–1232, 2021.

[10] T. Do-Duy, D. V. Huynh, O. A. Dobre, B. Canberk, and T. Q. Duong, "Digital twin-aided intelligent offloading with edge selection in mobile edge computing," *IEEE Wireless Communications Letters*, vol. 11, no. 4, pp. 806–810, 2022.

[11] T. Liu, L. Tang, W. Wang, Q. Chen, and X. Zeng, "Digital-twin-assisted task offloading based on edge collaboration in the digital twin edge network," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1427–1444, 2022.

[12] K. Peng, H. Huang, M. Bilal, and X. Xu, "Distributed incentives for intelligent offloading and resource allocation in digital twin driven smart industry," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 3, pp. 3133–3143, Mar. 2023.

[13] L. Gu, M. Cui, L. Xu, and X. Xu, "Collaborative offloading method for digital twin-empowered cloud edge computing on internet of vehicles," *Tsinghua Science and Technology*, vol. 28, no. 3, pp. 433–451, 2023.

[14] P. Trunfio, "Generating simulation-ready models by construction using performance-driven archetypes," in *2026 Winter Simulation Conference (WSC)*, Glasgow, Scotland, 2026.

[15] R. J. Somers, J. A. Douthwaite, D. J. Wagg, N. Walkinshaw, and R. M. Hierons, "Digital-twin-based testing for cyber–physical systems: A systematic literature review," *Information and Software Technology*, vol. 156, p. 107145, 2023.