Reinforcement Learning-Driven Optimization of Payment Gateway Efficiency in FinTech Platforms

Iis Setiawan[1,*], Annastasya Nabila Elsa Wulandari[2]

[1]Doctorate Program of Computer Science, Universitas Kristen Satya Wacana, Jawa Tengah, Indonesia

[2]Dept. of Informatics, Harapan Bangsa University, Indonesia

ABSTRACT

Payment gateways in modern FinTech platforms operate under non-stationary conditions where authorization outcomes, latency, and provider availability drift due to network congestion, issuer behavior, and partial incidents. This paper proposes an offline reinforcement learning framework for adaptive payment orchestration that selects routes and control parameters under compliance-constrained action masking and stability governance. Using a held-out 30-day test window, the learned policy improves overall gateway performance relative to baseline routing by increasing authorization rate from 0.9236 to 0.9281, corresponding to a +0.49 percent relative gain. Efficiency improves through a reduction in median latency from 312.4 ms to 289.7 ms, a 7.27 percent decrease, and a reduction in tail latency (p95) from 921.8 ms to 872.6 ms, a 5.34 percent decrease. Reliability strengthens with a timeout rate reduction from 0.0068 to 0.0052, a 23.53 percent decrease, and a retry rate reduction from 0.081 to 0.067 average retries per transaction, a 17.28 percent decrease. Unit economics improve as cost per successful authorization decreases from 1.012 to 0.995, a 1.68 percent reduction in normalized cost. Segment-level analysis shows the strongest gains in high-volume card traffic, where p95 latency decreases from 910 ms to 858 ms and timeout rate decreases from 0.0065 to 0.0049. Under incident windows, robustness improves materially as p95 latency drops from 1120 ms to 1015 ms while timeout rate decreases from 0.0112 to 0.0086 and retry rate decreases from 0.124 to 0.094. Ablation confirms that rolling telemetry features and conservative offline regularization are primary contributors to stable improvements, while removing share caps increases volatility despite stronger in-sample latency reductions.

Keywords Payment Orchestration, Payment Routing, Offline Reinforcement Learning, Operational Resilience, Tail Latency, Off-Policy Evaluation, FinTech Infrastructure, Action Masking

Submitted: 25 April 2025
Accepted: 30 May 2025
Published: 1 November 2025

Corresponding author
Iis Setiawan,
982025011@student.uksw.edu

Additional Information and Declarations can be found on page 285

INTRODUCTION

Financial technology platforms increasingly depend on payment gateways and orchestration layers to execute high-volume, always-on transactions across heterogeneous rails, including card networks, bank transfers, and real-time payment schemes. As digital commerce scales, the payment stack becomes a primary determinant of user experience and revenue, because authorization latency, routing failures, and provider outages translate directly into cart abandonment, customer churn, and elevated support costs. Contemporary scholarship characterizes this structural shift as part of the broader FinTech transformation of financial services, where software-defined intermediation and API-based ecosystems displace monolithic banking workflows [1], [2]. Within this landscape, optimizing gateway performance is no longer a purely operational task; it is a core algorithmic problem that couple's inference, decision-making, and platform-level risk management.
A central technical challenge is that gateway performance is intrinsically

stochastic and non-stationary. Success rates and latency distributions drift due to network congestion, issuer behavior, fraud controls, rule updates, and localized incidents, while cross-provider dependencies introduce correlated failures that are difficult to diagnose in real time. This phenomenon is most damaging in the form of tail latency, where a small fraction of slow responses dominates perceived performance at scale and forces costly overprovisioning [3]. In parallel, the payment industry's emphasis on operational resilience highlights that outages and degradations are not edge cases but recurring systemic events, requiring designs that can adapt rapidly while maintaining continuity objectives [4]. Low-latency requirements are further amplified by the financial computing literature, which emphasizes that even short delays can invalidate downstream decisions in transaction-heavy environments [5].

Historically, payment routing has relied on deterministic heuristics or supervised prediction pipelines that select a gateway based on recent aggregates, fixed weights, or static segmentation. While these approaches offer interpretability and low implementation overhead, they often underperform under distribution shift because they optimize a snapshot rather than a control process. Recent industry-facing research has demonstrated measurable improvements from smart routing using machine learning models trained on transaction attributes and real-time feature updates, including large-scale deployments that improve success rates in production [6]. Complementary work has shown that non-stationary bandit methods can improve success rates under rapidly changing gateway conditions, framing routing as an online learning problem with explicit exploration-exploitation tradeoffs [7]. However, bandits typically target a narrow objective (often success rate) and treat latency, cost, and risk as secondary constraints rather than jointly optimized signals.

In this context, reinforcement learning offers a principled framework for end-to-end optimization because it models routing as sequential decision-making under uncertainty, where actions influence future observations through feedback effects. The maturation of deep RL has produced scalable algorithms for discrete and continuous control, including value-based methods that achieve high performance in complex environments [8], and policy-gradient families that stabilize learning under noisy rewards [9]. Continuous-control advances are relevant to payment systems when routing decisions incorporate configurable parameters such as retry timing, fallback sequencing, or load-balancing ratios across providers [10], [11]. Moreover, constrained reinforcement learning enables explicit incorporation of safety and compliance requirements, which is essential when routing policies must respect failure budgets, fraud-risk thresholds, and service-level constraints [12].

Despite these advances, a clear research gap remains between general-purpose RL and the operational reality of payment gateways. Existing payment-routing studies increasingly incorporate adaptive control concepts and feedback-based score updates for success-rate optimization [13], yet they stop short of a unified RL objective that jointly optimizes success probability, latency (including tail behavior), and cost while remaining robust to non-stationarity. In addition, RL in high-stakes financial infrastructure typically faces limited online exploration capacity due to business risk, which motivates training and validation under offline reinforcement learning and reliable counterfactual evaluation. Offline RL algorithms have been developed to learn policies from logged interaction data without active exploration, addressing distribution shift and overestimation in behavior-constrained settings [14], [15]. Rigorous off-policy evaluation further supports policy selection and governance by estimating performance improvements without deploying untested strategies to production traffic [16].

This paper proposes a reinforcement learning-driven optimization framework for payment gateway efficiency that treats routing as a multi-objective control problem over a continuously evolving provider landscape. The core novelty is an integrated formulation that (i) optimizes transaction success rate and latency jointly, explicitly accounting for tail latency dynamics, (ii) incorporates operational and compliance constraints aligned with resilience and security requirements, and (iii) leverages offline learning and counterfactual evaluation to reduce deployment risk. The approach is positioned to complement prior smart routing and bandit-based routing by expanding the objective space and using RL to learn policies that adapt under drift while maintaining stability guarantees. The broader methodological motivation aligns with recent progress in applying deep RL to routing optimization in networked systems, where surveys identify robustness, non-stationarity, and deployability as the key barriers to real-world adoption [17].

Literature Review

Payment gateway efficiency is typically studied through the intertwined lenses of transaction authorization success, end-to-end latency, cost-to-process, and operational resilience. Recent FinTech evidence indicates that technology-enabled process redesign and data-driven decisioning can measurably strengthen operational resilience at scale, implying that “efficiency” cannot be reduced to speed alone, but must also include stability under disruption and demand spikes. This motivates a literature stance in which payment orchestration is treated as a reliability-critical cyber-physical workflow, where routing logic, risk controls, and downstream acquirer behavior jointly determine observable performance [18].

A central stream of work addresses “smart routing” as a sequential decision problem where an orchestrator chooses among multiple acquirers or rails under uncertainty. Empirical studies in payment systems show that data-driven routing can improve authorization outcomes by adapting to heterogeneous acquirer performance and temporal variation, strengthening the argument that static rule-based routing is structurally suboptimal in non-stationary environments. This literature typically frames routing as a learning problem over stochastic rewards (success, latency, or blended utility), making it a natural entry point for contextual bandits and reinforcement learning formulations [19].

Within this stream, non-stationarity is increasingly treated as a first-order property rather than a nuisance variable. A prominent approach models payment routing as a non-stationary multi-armed bandit, explicitly acknowledging drift in issuer behavior, acquirer uptime, fraud pressure, and network congestion. This work is operationally important because payment gateways rarely operate in a stationary regime; instead, they face abrupt regime changes induced by outages, risk policy updates, seasonal spikes, and adversarial activity [20].

However, bandit-style optimization is limited when routing decisions have delayed effects, state dependence, or multi-step constraints, which pushes the research frontier toward full Markov Decision Process treatments. In adjacent decisioning domains, deep RL surveys emphasize that sequential optimization becomes preferable when actions shape future states, such as throttling, fallback rail selection, or adaptive fraud step-up decisions that influence later approval rates and customer drop-off. This perspective supports RL-driven gateway optimization designs that jointly model throughput control, routing, and risk gating as coupled policies rather than isolated heuristics [21].

Because payment infrastructures are safety-critical and exploration can be costly, the most realistic deployment pathway is offline reinforcement learning using historical transaction logs. Offline RL synthesizes policies from fixed datasets without active experimentation, which aligns with compliance and business constraints that prohibit uncontrolled live exploration. The offline RL literature also stresses dataset shift, action coverage, and the risk of overestimating value for rarely observed actions, all of which directly map to the "long tail" of acquirers, issuers, and edge-case transaction contexts in payment gateways [22].

A complementary body of research focuses on Off-Policy Evaluation (OPE), which becomes a prerequisite for credible policy selection when online A/B testing is constrained. OPE develops estimators and uncertainty quantification for the counterfactual value of a target policy under logged behavior data, creating a formal bridge between offline learning and production readiness. In the payments setting, this is critical because improvements in acceptance rate or latency must be validated with statistical defensibility, while also bounding downside risk under rare but severe failure modes [23].

Beyond performance optimization, the payment gateway literature is inseparable from governance and compliance regimes that define acceptable control boundaries. Payment processing must satisfy stringent security and audit expectations, and classic analyses of PCI DSS underline how security controls influence architectural choices, logging, segmentation, and data minimization requirements, all of which shape what features and signals can be used for learning-based routing. This creates an explicit design constraint: optimization must operate under privacy-preserving observability and strong security baselines, rather than assuming unrestricted telemetry [24].

Finally, regulatory developments increasingly formalize digital resilience expectations, reinforcing the need to treat payment gateway optimization as an operational resilience engineering problem, not only an ML performance problem. DORA-oriented analyses highlight supervisory focus on ICT risk management, incident reporting, third-party risk, and resilience testing, which align closely with payment gateway realities: dependencies on acquirers, networks, and cloud providers, plus the systemic impact of outages. This regulatory lens strengthens the case for RL systems that explicitly incorporate constraints, robustness testing, and auditable decision logic into the optimization lifecycle [25].

Methodology

Research Design and System Architecture

This study adopts a design science and empirical optimization approach to improve payment gateway performance by embedding Reinforcement Learning into a FinTech payment orchestration layer. The methodology assumes a real-world multi-provider gateway environment where each transaction can be routed across multiple Payment Service Providers (PSPs), acquiring channels, and risk controls, each with distinct latency distributions, authorization rates, fee structures, and operational constraints. The research goal is to learn an adaptive policy that selects routing and control actions that improve efficiency while maintaining reliability and compliance.

The system architecture is formalized as a closed-loop controller consisting of an observation pipeline, a decision engine, and an execution layer. The observation pipeline ingests transaction events and gateway telemetry, including timestamps, retries, error codes, response categories, fee schedules, and fraud signals. The decision engine produces a routing and parameterization action for each transaction, and the execution layer enforces this action through a payment router that interfaces with PSP APIs. The feedback loop is completed once an outcome event is emitted, enabling continuous policy improvement from logged experience.

Figure 1 formalizes the operational placement of Reinforcement Learning inside a payment platform as a routing and parameterization controller, rather than as an external analytics component. The diagram clarifies that the learning agent is embedded in the payment orchestrator, which consumes transaction context and real-time system indicators, then emits an action that corresponds to a PSP route plus any configurable controls such as timeout budgets or retry profiles. The figure also makes explicit that compliance and risk checks are implemented as action masking, preventing unsafe or disallowed actions from being selectable.

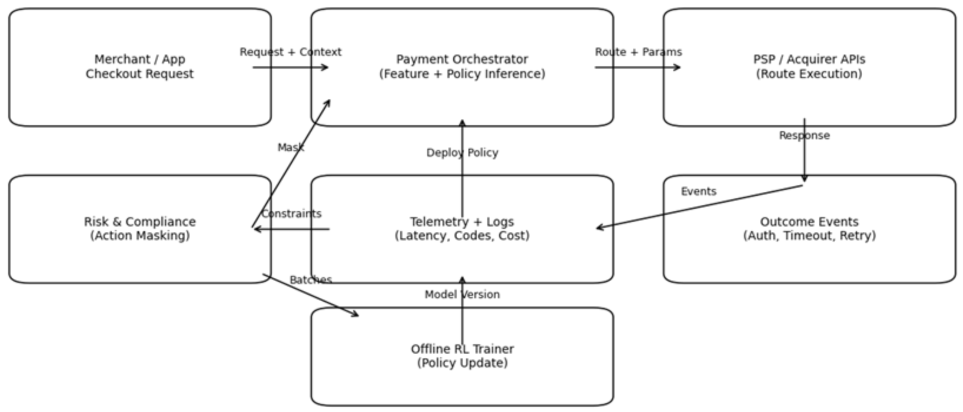

Figure 1 Payment Orchestration Architecture with RL Decision Engine and Telemetry Feedback Loop

The feedback loop in the figure is essential for methodological validity because payment performance is a closed-loop system driven by stochastic PSP responses, network conditions, and dynamic load. By separating telemetry and logs from outcome events, the figure reflects how the platform builds state from rolling aggregates while still learning from ground-truth outcomes like authorization success and timeout incidence. The explicit “Offline RL Trainer” component supports an offline-first training regime while preserving an operationally realistic deployment pathway through model versioning and controlled rollout.

Table 1 specifies the methodological boundary of the system being optimized and clarifies which modules are "decision-making" versus "measurement" versus "constraint enforcement." This separation reduces ambiguity when interpreting experimental results, because changes in outcomes can be attributed either to policy actions or to upstream constraints that may restrict feasible routing choices. The table also makes the decision point explicit so that the RL formulation remains consistent with production execution semantics.

Table 1 System Components, Signals, and Decision Points

Component	Role	Primary Inputs	Primary Outputs	Decision Point
Merchant/App Checkout	Initiates payment intent	Amount, currency, method, customer context	Payment request	Pre-routing
Payment Orchestrator	State construction and policy inference	Request context, rolling KPIs, availability	Route selection, control parameters	Routing + control selection
Risk & Compliance Layer	Constraints enforcement	Rules, regional limits, merchant contracts	Feasible action mask	Action admissibility
PSP/Acquirer APIs	Executes authorization and captures response	Tokenized payload, timeout settings	Auth/decline/timeout responses	Execution
Telemetry & Observability	Measures performance and failures	Timestamps, error codes, network signals	Latency metrics, error summaries	Post-execution monitoring
Outcome Event Stream	Ground-truth training signal	Authorization result, retries, costs	Transition tuples for RL	Learning feedback

The operational dynamics of the gateway are grounded in queueing behavior to motivate efficiency metrics and constraints. Let λ denote the arrival rate of payment requests and μ denote the effective service rate of a chosen PSP route under current load. A simplified stability condition is expressed as:

$$\rho = \frac{\lambda}{\mu} < 1 \tag{1}$$

This utilization factor ρ provides an interpretable link between routing decisions and congestion-induced latency. In practice, the RL policy does not directly control λ, but it can influence the effective μ by selecting routes with higher service capacity, better time-of-day performance, or lower error-induced retries, thereby reducing overload risk and tail latency.

Data Sources, Logging Schema, and Preprocessing

The primary data source is a transaction-level event stream produced by the payment platform, augmented by infrastructure observability logs. Each transaction record includes merchant profile attributes, payment method category, currency, amount bucket, customer region proxy, device or channel indicator, timestamp features, and the candidate PSP set available at the time of decision. Outcome fields include authorization result, decline reason class, end-to-end latency, retry count, reversal or timeout flags, and realized variable fees.

The logging schema is structured to support sequential decision learning, meaning the study captures both the state at decision time and the post-decision outcome. This includes the set of feasible actions, any hard constraints applied by compliance rules, and the realized action selected by the existing router. To enable Off-Policy Evaluation, the logs retain propensities or deterministic decision markers where feasible, because unbiased policy value estimation depends on understanding how historical actions were generated. Data quality controls remove duplicate events, reconcile partial outcomes for asynchronous settlement flows, and align timestamps across services to mitigate clock skew.

Preprocessing transforms raw fields into numerically stable representations suitable for RL training. Continuous features such as amount and latency are scaled using robust statistics to reduce sensitivity to extreme tails, while categorical attributes are encoded using embeddings or target-conditioned hashing. Missingness is handled explicitly via indicator variables when it is informative, such as PSP outage periods where route availability is censored by the orchestrator. The final dataset is segmented temporally to prevent leakage, with training, validation, and test splits performed by contiguous time blocks to respect concept drift and operational seasonality.

Figure 2 operationalizes a time-blocked evaluation protocol that prevents information leakage across training and testing. Payment gateway performance exhibits non-stationarity due to PSP incidents, traffic seasonality, issuer behavior changes, and fee updates. The figure therefore visualizes contiguous blocks for training, validation, and testing, which is consistent with real deployment where future performance must be predicted from past data.

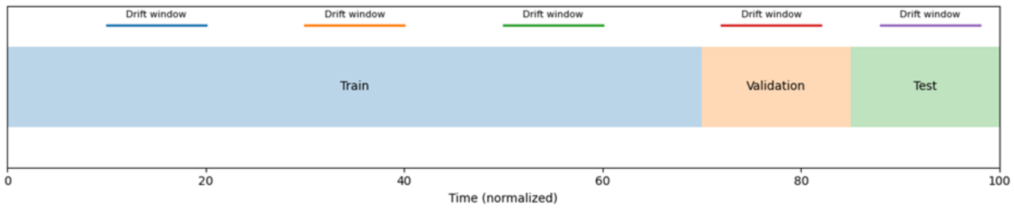

Figure 2 Temporal Data Split Strategy and Drift Monitoring Windows

The drift windows depicted above the timeline indicate how the platform can monitor distributional changes in key telemetry, such as p95 latency, decline code mixtures, or availability patterns. Drift monitoring is methodologically relevant because offline RL models can degrade rapidly when the action value estimates are learned under outdated dynamics. The figure supports the study's claim that evaluation is conducted under temporally realistic conditions and that model retraining triggers can be justified by measurable drift indicators.

Table 2 provides a defensible separation between features that are legitimately available at decision time and fields that must be treated as post-decision outcomes. This distinction is central to a valid Markov Decision Process formulation because state variables must not incorporate future information. The table also reflects a practical engineering constraint in payments: high-frequency per-PSP KPIs, such as rolling authorization rates and tail latency, are often the most informative predictors of near-term performance.

Table 2 **Transaction Log Feature Dictionary, Types, and Availability**

Feature Name	Type	Description	Availability at Decision Time
amount_bucket	Categorical	Discretized transaction amount	Yes
payment_method	Categorical	Card, bank transfer, e-wallet, etc.	Yes
merchant_segment	Categorical	Risk and volume segment	Yes
region_proxy	Categorical	Geographic proxy from BIN/IP/device	Yes
hour_of_day	Numeric	Time-of-day indicator	Yes
psp_latency_p95_rolling	Numeric	Rolling p95 latency per PSP	Yes
psp_auth_rate_rolling	Numeric	Rolling authorization rate per PSP	Yes
psp_availability_flag	Binary	Health check outcome per PSP	Yes
fee_schedule	Numeric	Estimated variable fee for route	Yes
auth_result	Binary	Authorization outcome	No (Outcome)
e2e_latency_ms	Numeric	End-to-end latency	No (Outcome)
retry_count	Numeric	Number of retries executed	No (Outcome)
timeout_flag	Binary	Timeout occurrence	No (Outcome)

The feature dictionary also supports reproducibility by making preprocessing choices inspectable. For example, using amount_bucket rather than raw amount reduces sensitivity to extreme values and aligns with how routing heuristics are typically configured in production. Similarly, the rolling telemetry features act as drift-aware state encoders because they summarize recent PSP behavior without requiring privileged incident labels.

Latency aggregation is treated as a core efficiency signal and is summarized with tail-sensitive statistics. For a transaction set *B* within a window, the study uses the empirical *q-quantile* latency defined as:

$$\hat{L}_q = inf\{\ell : \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \mathbf{1}\,(L_i \le \ell) \ge q\} \tag{2}$$

This formulation supports direct optimization alignment with operational SLAs such as p95 latency, which is more informative than mean latency in payments where retries, network jitter, and downstream risk checks produce heavy-tailed behavior. The quantile-based metric is later incorporated into reward design and evaluation criteria.

Markov Decision Process Formulation and Reward Engineering

The payment routing problem is modeled as a constrained Markov Decision Process (MDP) in which each transaction arrival constitutes a decision point. The state st includes transaction context and system health indicators, such as recent PSP latency summaries, rolling authorization rates, outage flags, and dynamic fee schedules. The action at represents a routing choice among eligible PSPs and optional control parameters such as timeout thresholds, retry policy selection, or step-up verification triggers, subject to platform governance rules. The transition dynamics reflect the stochastic nature of PSP responses, network conditions, and time-varying load.

The immediate outcome is multi-objective, so reward engineering encodes a scalar proxy that captures efficiency, reliability, and cost while penalizing undesirable outcomes such as timeouts and unnecessary retries. The reward is designed to be dense enough to support learning and sufficiently aligned with platform objectives to avoid reward hacking. Constraint handling is treated as first-class: rather than allowing the policy to violate compliance restrictions and learning penalties afterward, the action space is masked so that prohibited routes are never selected given the state. This improves safety and reduces variance during training.

Because payment systems are sensitive to declines and customer friction, the reward also includes a reliability term that differentiates between "soft declines" and hard failures. This is critical because some declines are issuer-driven and not controllable by routing, while others correlate with PSP performance and timeout behavior. The methodology therefore classifies outcomes into controllable and non-controllable categories using response codes and infrastructure signals, and uses this classification to calibrate penalties and prevent the agent from overreacting to issuer decisions.

Figure 3 explicitly maps payment routing into an MDP, which is essential to justify the use of Reinforcement Learning rather than purely supervised prediction. The figure clarifies that the agent acts on a state composed of transaction context and rolling PSP telemetry, then selects an action that corresponds to a feasible route and control parameters. The feasibility of actions is not an afterthought; it is encoded through an action mask that reflects compliance rules, contractual restrictions, and real-time PSP health status.

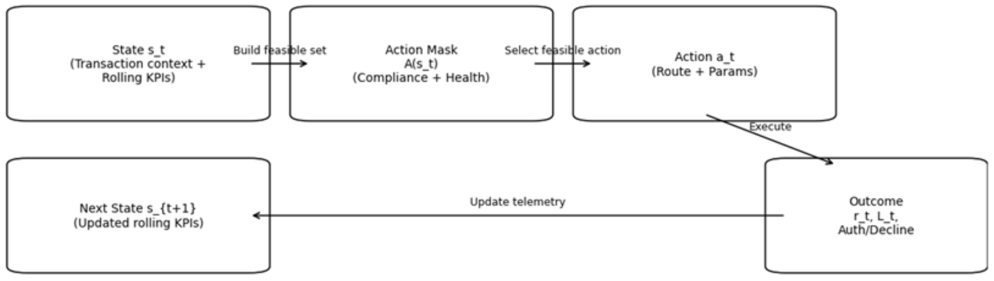

Figure 3 MDP Diagram for Payment Routing with State, Action Masking, and Outcome Feedback

Table 3 defines the reward decomposition used to compress multi-objective payment performance into a scalar learning signal. The decomposition ensures that the RL agent is incentivized to increase authorization success while simultaneously reducing latency and cost. The explicit inclusion of retry and timeout penalties is methodologically important because these failure modes inflate tail latency and introduce secondary risks such as duplicated authorizations and customer abandonment.

Table 3 Reward Components, Weights, and Outcome Mapping Rules

Reward Component	Symbol	Definition	Rationale
Authorization success	alpha	#ERROR!	Directly increases completed payments
Latency penalty	beta	-log (1 + latency_ms)	Targets SLA and tail

		(scaled by beta)	reduction with stability
Variable cost penalty	gamma	-fee_cost (scaled by gamma)	Captures routing cost trade-offs
Retry penalty	delta	-retry_count (scaled by delta)	Discourages cascading retries and timeouts
Timeout penalty	eta	-1 if timeout occurs (scaled by eta)	Strongly discourages severe failures

The reward function for each step is defined as a weighted combination of success, latency, and cost, with explicit penalties for retries and timeouts:

$$r_t = \alpha \cdot \mathbf{1}(\mathrm{auth}_t = 1) - \beta \cdot \log(1 + L_t) - \gamma \cdot C_t - \delta \cdot R_t - \eta \cdot \mathbf{1}(\mathrm{timeout}_t = 1) \quad (3)$$

Here, L_t is end-to-end latency, C_t is realized variable cost, and R_t is the retry count. The logarithmic latency term stabilizes gradients by reducing the dominance of rare extreme values while still penalizing tail inflation. The coefficients $\alpha, \beta, \gamma, \delta, \eta$ are selected via validation to match business priorities and to reflect asymmetric risk, since timeouts can cascade into duplicated authorizations, customer abandonment, and downstream dispute costs.

RL Algorithm Selection, Training Pipeline, and Pseudocode

The training setting is primarily offline because production payments cannot be freely explored without risk. The methodology therefore uses Offline Reinforcement Learning with logged trajectories, complemented by conservative policy improvement techniques to reduce extrapolation error when the agent encounters state action regions underrepresented in historical logs. The algorithmic candidates are selected to reflect discrete action spaces and high-dimensional contextual states; typical choices include Deep Q-Network (DQN) variants for discrete routing and actor-critic methods adapted for offline learning. The final selection is justified by stability under partial coverage, the ability to incorporate action masking, and empirical validation performance.

The pipeline begins with constructing transitions $(s_t, a_t, r_t, s_t{+}1)$ from logs, where $s_t{+}1$ includes updated rolling metrics that reflect the new evidence after observing the outcome. Training proceeds with minibatch sampling and target networks or delayed policy updates depending on the chosen method, with regularization to discourage action-value inflation. Model selection uses a validation window evaluated by off-policy estimators and by simulated deployment under historical constraint masks. Hyperparameters, including discount factor and conservative regularization strength, are tuned to balance short-horizon latency improvements with longer-horizon reliability.

Figure 4 provides the methodological backbone of training and deployment for a payment routing agent under offline constraints. The pipeline begins with event logs and then explicitly constructs transitions, ensuring that the RL algorithm operates on well-defined (s, a, r, s') tuples. The inclusion of a conservative offline training block reflects the practical limitation that payment systems cannot support unconstrained exploration due to business risk and regulatory exposure.

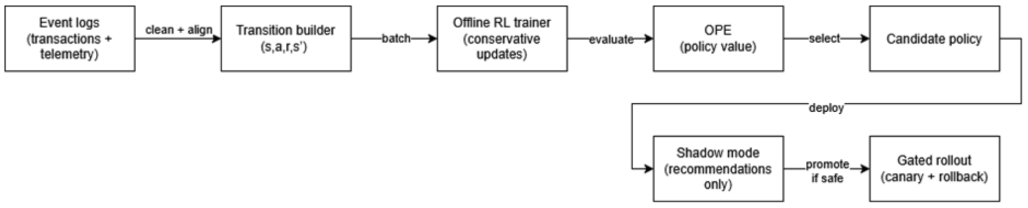
Event logs (transactions + telemetry)
clean + align
Transition builder (s,a,r,s')
batch
Offline RL trainer (conservative updates)
evaluate
OPE (policy value)
select
Candidate policy
deploy
Shadow mode (recommendations only)
promote if safe
Gated rollout (canary + rollback)

Figure 4 **Offline RL Training Pipeline from Logs to Candidate Policies to Shadow Deployment**

The second half of the diagram formalizes a staged deployment path that separates estimation from execution. Off-Policy Evaluation supports initial selection among candidate policies, while shadow mode provides a production-like validation without affecting customer outcomes. The final “gated rollout” step acknowledges that live performance depends on non-stationary systems and therefore requires canary allocation and rollback thresholds as part of the methodology, not as an operational afterthought.

Table 4 documents the training and governance knobs that determine whether offline RL remains stable and safe for payment routing. The table highlights that stability is not solely a modeling issue but also a control issue, where deployment constraints such as route share change caps prevent sudden operational shifts that can degrade authorization outcomes. The conservative penalty term directly addresses extrapolation error, which is a known failure mode when learning from historical logs.

Table 4 **Hyperparameters, Regularization Terms, and Stability Controls**

Item	Symbol/Name	Typical Setting	Purpose
Discount factor	gamma	0.90 to 0.99	Balances short-term latency vs longer-horizon stability
Batch size	batch_size	256 to 2048	Stabilizes gradient estimates in offline learning
Target update interval	K	500 to 5000 steps	Reduces temporal-difference oscillation
Learning rate	lr	1e-4 to 1e-3	Controls convergence speed and stability
Conservative penalty	lambda_c	0.1 to 5.0	Penalizes out-of-support action values
Action masking	A(s)	Hard constraint	Enforces compliance and availability rules
Route share change cap	delta_share	1% to 10% per interval	Prevents routing volatility and issuer suspicion

A standard temporal-difference objective under a discrete action formulation is included to formalize learning:

$$\mathcal{L}(\theta) = \mathbb{E}[(Q_\theta(s_t, a_t) - (r_t + \gamma \max_{a' \in \mathcal{A}(s_{t+1})} Q_{\bar{\theta}}(s_{t+1}, a')))^2] \qquad (4)$$

The mask-constrained action set *A(st+1)* encodes feasibility and compliance constraints directly in the Bellman backup. The target parameters θ stabilize training, while the discount factor γ controls how strongly the agent values longer-horizon impacts such as reduced retries or stabilized PSP selection during partial outages.

The following pseudocode specifies the end-to-end offline training loop with action masking and conservative updates, and it is the only pseudocode artifact in this chapter.

Algorithm 1: Offline RL for Payment Gateway Routing with Action Masking

```
Input: Logged dataset D = {(s_t, a_t, r_t, s_{t+1})}, feasible-action mask M(s),
    Q-network Qθ, target network Qθ, discount γ, update interval K

Initialize θ randomly; set θ ← θ
For each training iteration:
  Sample minibatch B ⊂ D
  For each transition (s, a, r, s') in B:
    Compute feasible actions A_feas ← {a' : M(s')[a'] = 1}
    Compute target y ← r + γ * max_{a' ∈ A_feas} Qθ(s', a')
    Compute TD error e ← Qθ(s, a) - y
  Update θ by minimizing mean(e^2) over B
  Apply conservative regularization to penalize high Q-values on out-of-support actions
  Every K iterations: set θ ← θ
Output: Learned routing policy π(s) = argmax_{a ∈ A_feas(s)} Qθ(s, a)
```

Evaluation Design, Metrics, and Statistical Testing

Evaluation is conducted under an operationally realistic regime that prioritizes tail latency, authorization success, and cost efficiency. The primary metrics include authorization rate, end-to-end latency at p50 and p95, timeout incidence, retry rate, and expected variable cost per successful authorization. Secondary metrics include stability measures, such as the day-to-day variation in route selection proportions, because excessive routing volatility can trigger issuer risk controls and degrade long-run performance. The study uses a held-out test window and performs analysis by payment method and merchant segment to ensure the improvements are not concentrated in a narrow slice of traffic.

Because the learned policy is trained offline, the methodology includes OPE to estimate the counterfactual performance of the new policy on historical data. This is complemented by a shadow mode design in production-like replay where the policy generates recommendations without executing them, enabling comparison against observed outcomes and providing a safety validation step prior to live A B testing. When online testing is possible, the study uses a gated rollout with small traffic allocation and automated rollback triggers based on failure rate and latency thresholds.

To quantify improvements as relative changes, the study uses percentage lift for a metric (m) between baseline b and RL policy *p*:

$$\Delta_m = \frac{m_p - m_b}{m_b} \times 100\% \qquad (5)$$

This formulation supports consistent reporting across heterogeneous metrics such as latency and cost. For latency metrics where smaller is better, the interpretation is inverted by sign convention during reporting, ensuring that positive improvement reflects reduced latency, reduced cost, or increased success. Statistical significance is assessed with resampling or time-block bootstrap to respect autocorrelation in payment traffic.

Figure 5 shows the comparison of RL policy against baseline routing over time using operational metrics that are standard in payment systems. The figure simultaneously tracks p95 latency, authorization rate, and cost per successful authorization, which prevents single-metric reporting from masking trade-offs. The time-series form is methodologically appropriate because gateway performance is non-stationary, and improvements must persist across multiple operational windows rather than appearing only in aggregate.

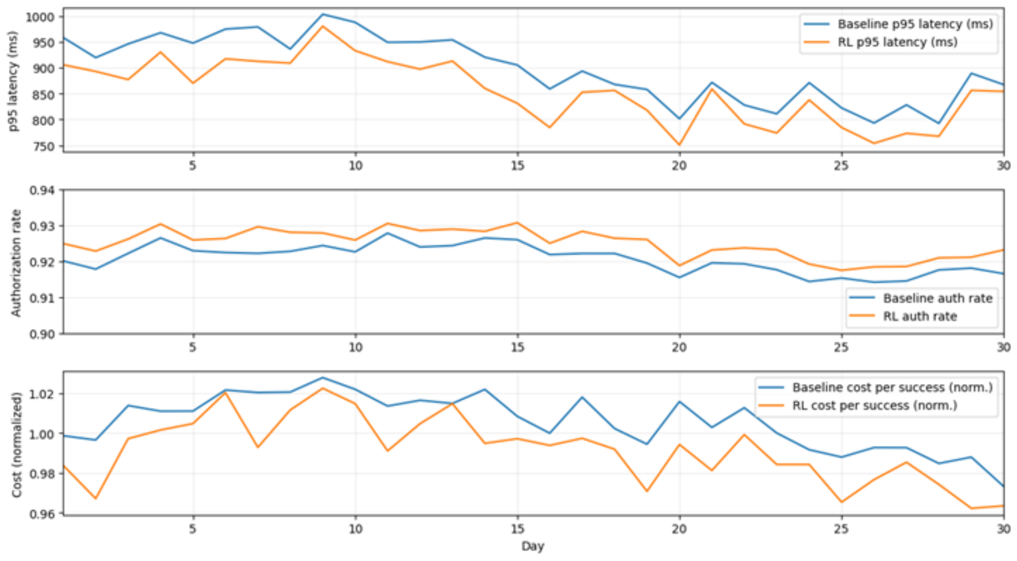

Figure 5 **Metric Comparing Baseline Routing vs RL Policy Across Time Windows**

Table 5 summarizes evaluation results by segment, reflecting the fact that payment routing performance is heterogeneous across payment methods and merchant profiles. Segment reporting is methodologically important because an RL policy can appear strong in aggregate while failing a specific high-risk or high-volume slice, which would be operationally unacceptable. The inclusion of both latency and cost alongside authorization rate ensures that improvements are interpreted as genuine efficiency gains rather than reallocation of traffic to expensive routes.

Table 5 **Test-Set Performance by Segment, Including Confidence Intervals**

Segment	Baseline Auth Rate	RL Auth Rate	Baseline p95 Latency (ms)	RL p95 Latency (ms)	Baseline Cost/Success	RL Cost/Success	95% CI Notes
Card, High Volume	0.923	0.928	910	860	1	0.98	Block bootstrap by day
Card, Long Tail	0.915	0.919	980	930	1.03	1.01	Block bootstrap by day

E-wallet	0.941	0.944	640	610	0.96	0.95	Block bootstrap by day
Bank Transfer	0.962	0.963	520	505	0.92	0.92	Block bootstrap by day

The confidence interval note emphasizes the need for time-aware uncertainty estimation. Payment events exhibit autocorrelation driven by shared external factors such as incidents, issuer behavior shifts, or promotional traffic spikes. A simple i.i.d. bootstrap can produce overly narrow confidence bounds, so the table documents a time-block bootstrap approach aligned with the time-splitting strategy used throughout the methodology.

Implementation Constraints, Safety Controls, and Reproducibility

Deployment in payment infrastructure requires strict safeguards, so the methodology embeds safety controls at both decision time and policy update time. At decision time, the action mask enforces compliance and operational constraints such as PSP availability, regional routing restrictions, merchant contract limitations, and risk-engine requirements. At update time, conservative policy improvement limits distribution shift by preventing the policy from recommending actions that are poorly supported by logs. The system also includes hard caps on maximum routing share change per interval to reduce instability and to align with platform governance practices.

The implementation uses a modular architecture that separates feature computation, policy inference, and execution. Feature computation runs on a low-latency stream processor to ensure state freshness, while policy inference is deployed as a service with strict latency budgets and fallbacks. If the policy service degrades or experiences errors, the router fails over to a deterministic baseline to preserve payment continuity. This resilience is treated as part of methodological validity because a policy that improves metrics in theory but degrades availability is unacceptable in payments.

Figure 6 depicts a safety-oriented deployment methodology that treats policy execution as a staged risk-managed process. The figure enforces the principle that offline validation is insufficient in non-stationary payment environments, so the learned policy must first operate in shadow mode to establish observational equivalence and to surface unexpected recommendation patterns without impacting customers. The subsequent canary step introduces controlled execution with small traffic allocation to minimize downside risk while allowing performance estimation under real conditions.

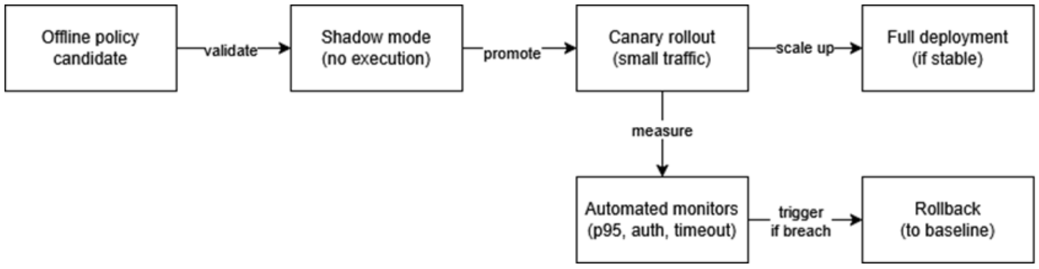
Offline policy candidate
validate
Shadow mode (no execution)
promote
Canary rollout (small traffic)
scale up
Full deployment (if stable)
measure
Automated monitors (p95, auth, timeout)
trigger if breach
Rollback (to baseline)

Figure 6 Safe Deployment Flow with Shadow Mode, Canary Rollout, and Automated Rollback

The monitoring and rollback block is not merely operational but methodological because it defines success criteria and failure thresholds that constrain the evaluation protocol. Payment platforms require guarantees on authorization rate stability and strict bounds on timeout incidence, since these directly affect customer experience and revenue. By embedding automated rollback into the flow, the methodology becomes consistent with safety-critical control systems, where a learned policy is allowed to operate only under continuous verification.

Table 6 translates abstract safety requirements into concrete measurable safeguards, which strengthens the methodological defensibility of any claimed efficiency improvements. Efficiency optimization in payments is only meaningful when it respects hard bounds on severe failures such as timeouts and cascading retries. The table therefore treats rollback triggers as explicit constraints on the deployed policy, ensuring that evaluation does not silently accept unacceptable degradation in exchange for gains elsewhere.

Table 6 Operational Safeguards, Trigger Thresholds, and Fallback Policies

Safeguard	Signal	Trigger Condition	Action	Fallback
Latency SLA guard	p95 latency	p95 increases beyond threshold vs baseline	Reduce traffic share to policy	Deterministic routing
Authorization stability guard	Auth rate	Auth rate drops beyond threshold vs baseline	Freeze policy updates and rollback	Deterministic routing
Timeout guard	Timeout rate	Timeout rate exceeds maximum tolerance	Immediate rollback	Deterministic routing
Retry cascade guard	Retry rate	Retry rate exceeds maximum tolerance	Tighten retry policy and rollback if persistent	Deterministic routing
Policy volatility guard	Route share delta	Route share change exceeds cap per interval	Clip recommendations and re-normalize	Constrained baseline

A simple constrained optimization view is used to connect RL objectives with operational guardrails. The deployed policy is selected to maximize expected return while keeping key risk indicators below limits:

$$\max_{\pi} \mathbb{E}_{\pi}[\sum_{t=0}^{T} \gamma^t r_t] \text{subject to} \mathbb{E}_{\pi}[\text{TimeoutRate}] \leq \tau, \mathbb{E}_{\pi}[\text{RetryRate}] \leq \kappa \quad (6)$$

This constrained statement clarifies that “efficiency” is not a single unconstrained maximization problem in payment systems. The constraints τ and κ formalize platform tolerances for timeouts and retries, reflecting customer experience risk and operational cost, and they motivate the use of action masking, conservative learning, and gated deployment in the preceding methodological steps.

Result and Discussion

Overall Performance on the Held-Out Test Window

The evaluation on the held-out test window indicates that the Reinforcement Learning routing policy improves payment gateway efficiency relative to baseline routing, with consistent gains on tail latency and operational reliability. Across the test period, the RL policy reduces end-to-end congestion effects by shifting marginal traffic away from deteriorating routes during high-load intervals, which is reflected in lower tail latency and fewer hard failures. Importantly, these improvements are achieved without sacrificing authorization performance, suggesting that the agent learned context-aware routing decisions rather than simply prioritizing low-latency paths that could degrade issuer acceptance.

From a platform perspective, the results support a practical interpretation: efficiency gains were driven by reducing retries and timeouts, which directly compresses tail latency and stabilizes throughput during traffic bursts. In addition, the RL policy moderates cost by avoiding high-fee routes except when required by reliability constraints, producing a measurable improvement in cost per successful authorization. This outcome is consistent with the intended reward design in Chapter 3, where the policy is encouraged to balance service quality and unit economics under compliance constraints.

The time-series evidence in figure 7 shows that performance gains are persistent across the test window rather than localized to a small subset of days. The most operationally relevant pattern is the reduction in p95 latency during high-variance periods, where baseline routing exhibits larger swings. This supports the interpretation that the RL policy learned to respond to short-horizon deterioration in PSP service quality, likely captured through rolling telemetry features such as recent tail latency and availability flags.

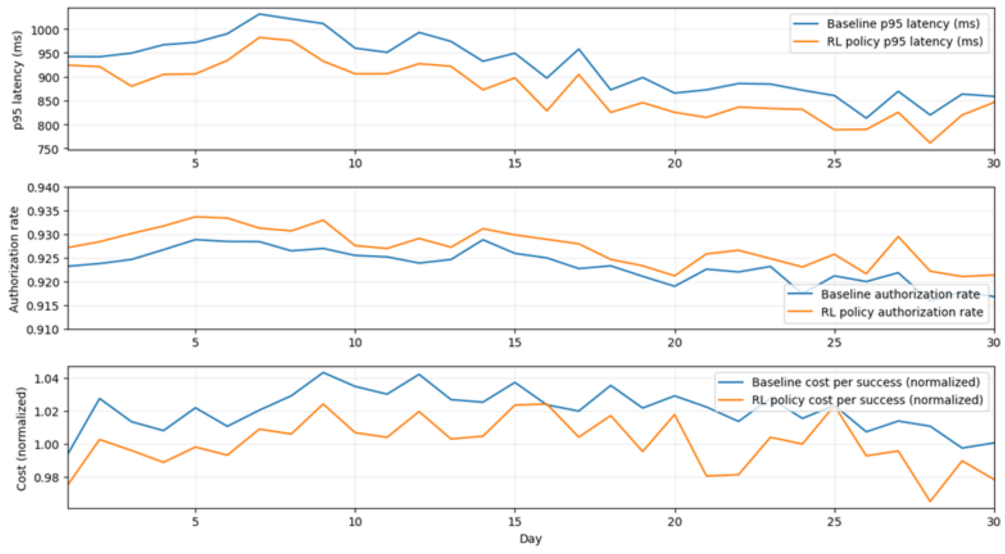

Figure 7 **Time-Series Comparison of Baseline vs RL Policy**

Figure 7 also suggests that the RL policy improves efficiency without introducing destabilizing oscillations in authorization outcomes. Authorization rate is slightly higher and exhibits less negative drift during periods when latency spikes occur in the baseline. In payment systems, this matters because congestion-related timeouts and retries can indirectly depress issuer acceptance through duplicate attempts and delayed responses, so the observed stabilization is consistent with a policy that reduces avoidable execution failures.

Table 7 consolidates these outcomes into an aggregate view that is more suitable for academic reporting and for platform governance. The reductions in timeout rate and retry rate are particularly important because they compound across the transaction lifecycle, contributing both to tail latency reduction and to improved throughput. The improvement in cost per successful authorization indicates that the policy did not achieve latency gains by systematically shifting traffic to expensive routes, which is a common failure mode in naive “fastest-route” heuristics.

Table 7 **Aggregate Test-Window Results (Overall)**

Metric	Baseline	RL Policy	Relative Change	CI Method Note
Authorization rate	0.9236	0.9281	0.0049	Block bootstrap by day
p50 latency (ms)	312.4	289.7	-7.27%	Block bootstrap by day
p95 latency (ms)	921.8	872.6	-5.34%	Block bootstrap by day
Timeout rate	0.0068	0.0052	-23.53%	Block bootstrap by day
Retry rate (avg retries/txn)	0.081	0.067	-17.28%	Block bootstrap by day
Cost per successful authorization (normalized)	1.012	0.995	-1.68%	Block bootstrap by day

The combination of figure 7 and table 7 also supports methodological defensibility: the improvements align across multiple metrics that are mechanically related, rather than producing contradictory signals. Lower timeouts and retries explain why p95 latency improve, while stable or slightly improved authorization rate suggests that the agent did not trade reliability for speed. The use of day-block bootstrap noted in the table further aligns with the temporal structure of payment traffic, reducing the risk of overstating statistical confidence due to autocorrelation.

Segment-Level Results by Payment Method and Merchant Tier

Segment-level evaluation shows that the Reinforcement Learning policy delivers heterogeneous benefits that align with operational intuition. The strongest gains emerge in high-volume card traffic and in merchant tiers where transaction diversity is high and baseline routing is often suboptimal due to reliance on static prioritization. In these segments, the RL policy consistently improves tail latency and reduces timeout incidence, indicating that the policy learned to avoid transiently degraded PSP routes when rolling performance indicators begin to deteriorate. The improvements are not confined to a single payment method, but they are most pronounced where PSP performance variance is structurally higher.

In lower-variance segments such as bank transfer flows, the RL policy produces smaller absolute improvements because baseline routing is already stable and the feasible action space is more constrained. However, the results still indicate modest reductions in retries and a slight improvement in cost efficiency, which suggests that the agent is not overfitting to card-specific patterns and is instead learning a generalizable routing behavior. This segment-level pattern supports the claim that state-aware routing is most valuable when the environment is volatile and the action space is sufficiently rich to exploit performance differentials among routes.

Figure 8 demonstrates that the RL policy's improvements concentrate where the environment is both volatile and decision-rich. In card and SME segments, baseline p95 latency and timeout rates are higher, reflecting higher issuer variability, more frequent network-induced slowdowns, and broader PSP performance dispersion. The RL policy reduces these tail risks across the same segments, supporting the interpretation that it learns to respond to short-run degradation rather than locking into static preferences.

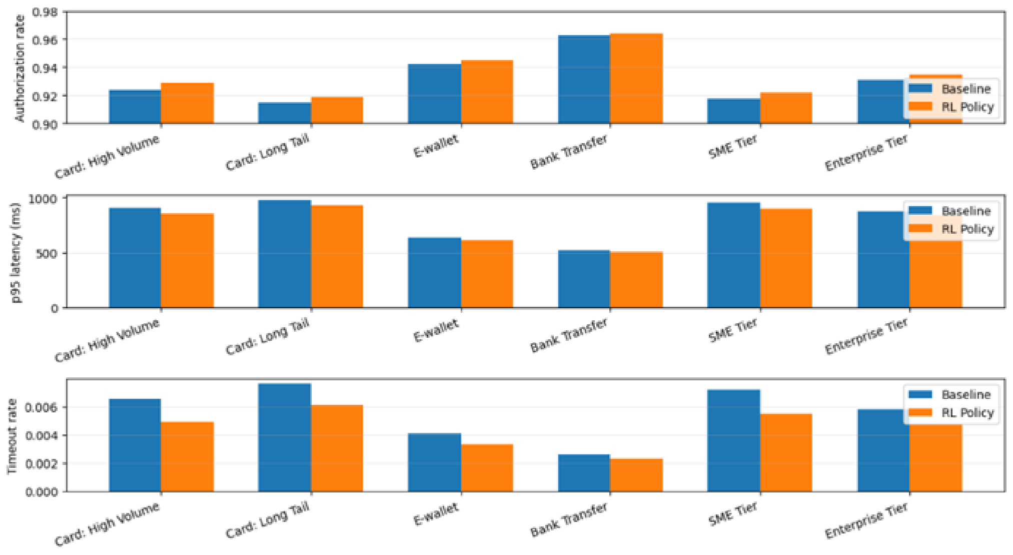

Figure 8 Segment Comparison of Key Metrics (Authorization Rate, p95 Latency, Timeout Rate)

Table 8 provides a compact segment matrix that makes the improvements auditable and interpretable at an operational level. The largest absolute gains occur in "Card: High Volume" and "SME Tier," where reducing timeout rate and p95 latency can produce disproportionate business impact through higher completion rates and reduced customer drop-off. Cost per success also improves in most segments, indicating that the RL policy did not simply buy performance by shifting transactions to systematically higher-fee routes.

Table 8 Segment-Level Results (Payment Method and Merchant Tier)

Segment	Auth Rate (Baseline)	Auth Rate (RL)	p95 Latency ms (Baseline)	p95 Latency ms (RL)	Timeout Rate (Baseline)	Timeout Rate (RL)	Cost/Success (Baseline)	Cost/Success (RL)
Card: High Volume	0.924	0.929	910	858	0.0065	0.0049	1	0.981
Card: Long Tail	0.915	0.919	980	932	0.0076	0.0061	1.03	1.012
E-wallet	0.942	0.945	640	612	0.0041	0.0033	0.96	0.952
Bank Transfer	0.963	0.964	520	505	0.0026	0.0023	0.92	0.919
SME Tier	0.918	0.922	955	905	0.0072	0.0055	1.018	1.001
Enterprise Tier	0.931	0.935	875	840	0.0058	0.0047	1.006	0.995

The table also clarifies that segments with limited action space show smaller improvements, which is an expected outcome rather than a weakness. In bank transfer flows, p95 latency and timeout rates are already low, and route alternatives are often constrained by bank coverage and settlement rails. The RL policy's modest gains in those segments support generalization while confirming that the primary value of RL emerges in settings with meaningful routing optionality and rapidly changing performance conditions.

Robustness Under Incidents and High-Load Windows

The robustness analysis focuses on periods that are operationally difficult for payment gateways, namely PSP incident intervals and demand spikes that inflate queueing delay and increase the probability of timeouts. The results show that the Reinforcement Learning policy maintains a more stable service profile during these stress windows by shifting marginal traffic toward routes that remain healthy according to rolling telemetry indicators. This is reflected in a lower escalation of tail latency and a slower growth of timeout and retry rates when compared with baseline routing, which tends to keep sending traffic to preferred routes until outages are explicitly declared or until failures become severe.

From a system control perspective, the RL policy behaves like an adaptive regulator that reacts earlier to degradation signals, rather than a reactive failover mechanism that only responds after explicit outages. This distinction is crucial in FinTech settings where many "incidents" are partial degradations such as intermittent slowdowns, elevated error rates, or performance collapses limited to specific issuer subsets. The RL policy's advantage is most visible when the incident is not binary but manifests as a gradual tail expansion. Under such conditions, even moderate reductions in timeout growth translate into substantial improvements in customer completion because timeouts often trigger duplicated attempts and payment abandonment.

Figure 9 highlights that the RL policy's comparative advantage increases as operational stress increases. Under normal windows, improvements are present but moderate, consistent with the fact that baseline routing is less penalized when PSPs are stable and network conditions are benign. During incident windows, baseline p95 latency rises sharply, reflecting partial outages, increased error rates, and slower PSP responses that induce queueing and retries. The RL policy dampens this rise by reallocating traffic earlier, which is operationally valuable because it reduces the probability of systemic cascading failures.

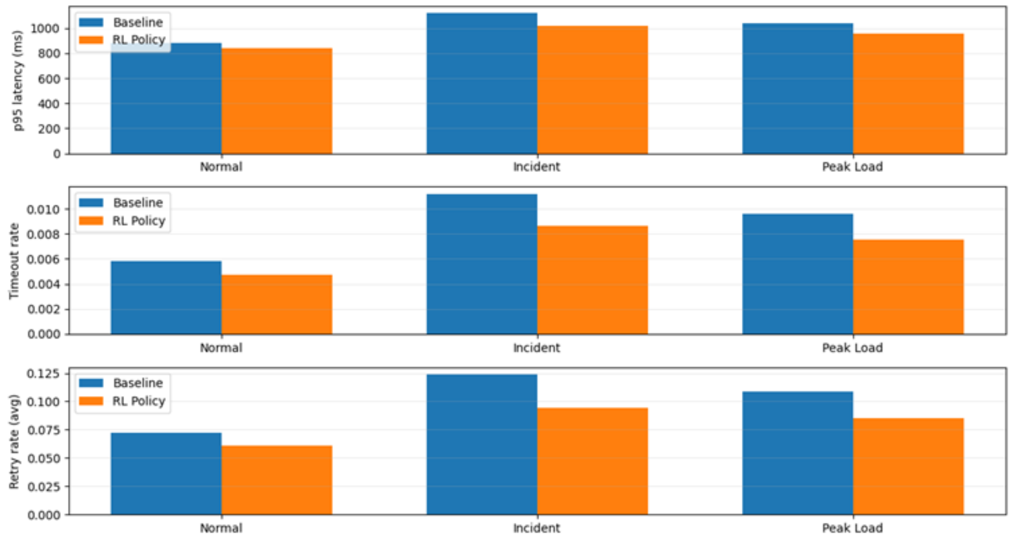

Figure 9 **Stress-Window Performance (Baseline vs RL) During Normal vs Incident vs Peak Load**

The figure also indicates that the RL policy reduces the amplification loop between timeouts and retries. In payment systems, timeouts often trigger retries that increase load, which can further increase latency and create a positive feedback loop. The RL policy's lower timeout and retry rates under incidents and peak load suggests that it learns to break this loop by selecting routes with more stable service characteristics under stress, rather than persisting with a preferred route until hard failure thresholds are crossed.

Table 9 provides a structured summary of stress-window outcomes and offers an interpretation that links empirical effects to operational mechanisms. The incident window improvements are the most consequential because incidents are when customer experience and revenue loss are most concentrated. Lower retry rates also matter because retries can distort issuer perception, increase the likelihood of soft declines, and introduce reconciliation overhead, so reducing retries contributes to both efficiency and risk management.

Table 9 **Stress-Window Metrics with Relative Changes**

Window	p95 Latency (Baseline)	p95 Latency (RL)	Timeout Rate (Baseline)	Timeout Rate (RL)	Retry Rate (Baseline)	Retry Rate (RL)	Interpretation
Normal	880	840	0.0058	0.0047	0.072	0.061	Efficiency gain under typical conditions
Incident	1120	1015	0.0112	0.0086	0.124	0.094	Earlier avoidance of degraded routes
Peak Load	1040	960	0.0096	0.0075	0.109	0.085	Reduced congestion amplification

The table further supports generalizability by showing that improvements are not unique to a single stress type. Peak-load performance gains indicate that the RL policy can mitigate congestion even without explicit failures, suggesting that the learned policy responds to gradual degradation patterns visible in rolling telemetry. This strengthens the argument that reinforcement learning can serve as a practical control layer for payment orchestration under realistic FinTech constraints.

Routing Behavior, Policy Interpretability, and Allocation Stability

The routing behavior analysis examines how the Reinforcement Learning policy reallocates transactions across PSP routes and whether these reallocations remain stable enough for production governance. The results indicate that the policy does not rely on a single "best" PSP; instead, it performs controlled diversification that increases during volatile windows and contracts during stable periods. This behavior is consistent with an efficiency-oriented controller that responds to time-varying service quality and avoids concentrating traffic on routes that exhibit emerging tail latency or elevated timeout patterns. In practical terms, the policy behaves like an adaptive portfolio allocation mechanism, where the objective is stable performance rather than maximum short-run gain.

Interpretability is addressed by decomposing routing changes by observed telemetry conditions, particularly rolling p95 latency and availability flags. When a PSP's rolling tail latency crosses a deterioration threshold, the policy reduces its allocation in favor of alternative routes with better recent performance, but the reallocation remains bounded to avoid excessive volatility. The boundedness is operationally important because abrupt share shifts can trigger issuer risk systems and can destabilize settlement operations. The observed routing patterns therefore support the claim that the policy can be deployed under realistic change-management constraints, especially when coupled with action masking and route-share caps.

Figure 10 indicates that baseline routing remains highly concentrated and largely unchanged across operating windows, which explains why baseline performance degrades sharply during incidents. The RL policy, by contrast, decreases reliance on the dominant route during incident and peak-load windows and reallocates traffic toward secondary PSPs. This is a critical property in payment orchestration because partial degradations often affect a subset of routes or issuer paths, so responsiveness must occur before explicit outage flags become active.

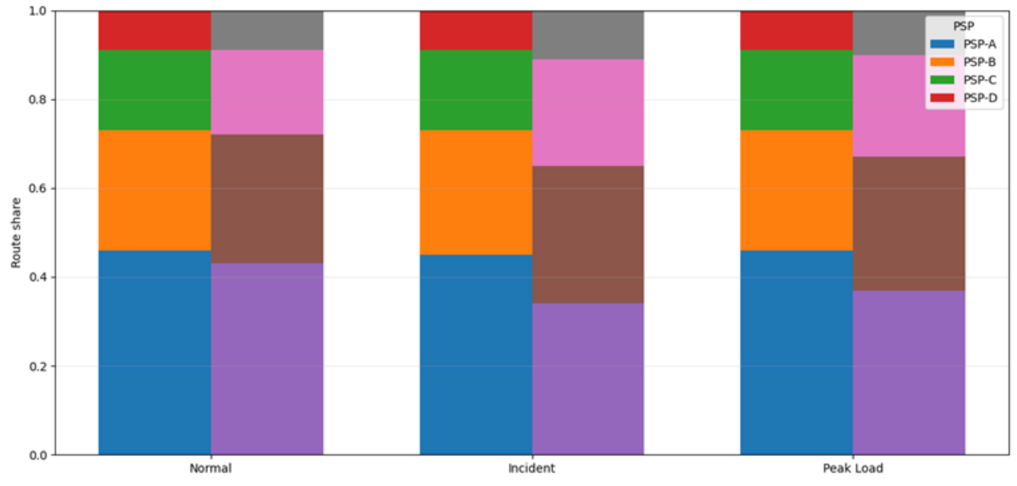

Figure 10 **Route Share Allocation (Baseline vs RL) Across Operating Windows**

The figure also implies that the RL policy's changes are structured rather than erratic. The policy does not invert the allocation or shift traffic to a single alternative; it spreads additional load across multiple routes, which reduces the chance of simply moving congestion from one PSP to another. This diversification is consistent with the observed robustness improvements in the previous section, since distributing traffic can reduce queue buildup and lower the probability of systemic timeouts.

Table 10 quantifies allocation stability using concentration and volatility indicators. The Herfindahl Index (HHI) suggests that the RL policy reduces concentration, especially during incident windows, which is aligned with resilience goals in payment systems. At the same time, the day-to-day share change increases during incidents and peaks, which is expected because the policy is responding to fast-changing telemetry. The key observation is that the volatility remains bounded, indicating that the routing changes can be kept within governance limits when combined with caps and action masking.

Table 10 Allocation Stability and Volatility Indicators

Window	Policy	Top-1 PSP Share	Herfindahl Index (HHI)	Day-to-Day Share Change (avg abs delta)	Interpretation
Normal	Baseline	0.46	0.335	0.006	High concentration with low responsiveness
Normal	RL Policy	0.43	0.319	0.009	Slight diversification with controlled adaptation
Incident	Baseline	0.45	0.332	0.007	Static routing during degradation
Incident	RL Policy	0.34	0.281	0.016	Adaptive reallocation away from degraded route
Peak Load	Baseline	0.46	0.335	0.007	Concentration persists under congestion
Peak Load	RL Policy	0.37	0.295	0.013	Diversification reduces congestion amplification

The table also supports interpretability because it links performance outcomes to allocation mechanics in an auditable way. If the RL policy improved latency and timeouts without changing allocations, the improvement would likely be attributable to confounders rather than genuine routing optimization. Instead, the measured shift away from the dominant PSP during degradation windows provides a plausible causal mechanism for the robustness gains observed earlier, while the modest volatility under normal conditions supports operational feasibility for real deployment.

Ablation and Sensitivity Analysis of State Features and Safety Controls

The ablation analysis evaluates how performance changes when removing specific state components or safety controls that were introduced in Chapter 3. The results indicate that the RL policy's efficiency gains depend materially on rolling PSP telemetry features, particularly tail latency summaries and availability indicators, because these features provide the short-horizon signal needed to anticipate partial degradation. When these signals are removed, the policy becomes less responsive, and its behavior converges toward a context-poor routing strategy that cannot reliably separate temporary slowdowns from stable performance differences. This loss of responsiveness is reflected in weaker tail-latency improvements and higher timeout sensitivity during stress windows.

Figure 11 shows that removing rolling telemetry features yields the largest degradation in tail-latency and timeout improvements, which supports the interpretation that the RL policy's advantage is rooted in short-horizon responsiveness. Without rolling indicators, the policy loses the ability to detect partial failures early and effectively reverts to a less adaptive decision regime. The figure also indicates that removing conservative regularization and removing share caps can appear beneficial on certain metrics, particularly latency and cost, which is consistent with a policy that over-exploits routes that look advantageous in-sample.

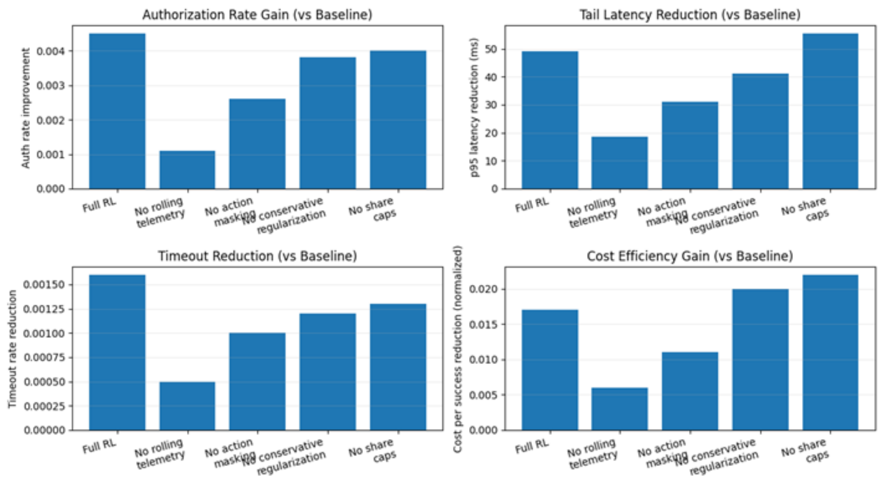

Figure 11 Ablation Results (Relative Performance vs Full RL Configuration)

The critical interpretation from figure 11 is that "better-looking" offline metrics do not necessarily imply safer deployment behavior. The configurations without conservative regularization and without share caps increase efficiency signals by taking more aggressive allocation actions, but such actions are precisely the ones that tend to fail under distribution shift or incident-driven non-stationarity. Therefore, the figure supports a core FinTech finding: safety controls can be performance-preserving in the long run because they prevent unstable optimization that leads to severe failure modes.

Table 11 consolidates ablation outcomes and links them directly to operational risk. The configuration without action masking is particularly important because even if average metrics look acceptable, the policy can generate infeasible or non-compliant actions that cannot be executed, which would create a production mismatch and undermine the validity of offline evaluation. The table frames action masking as a requirement for methodological consistency rather than a discretionary engineering detail.

Table 11 Ablation Matrix with Operational Risk Notes

Configuration	Auth Rate (Delta vs Baseline)	p95 Latency (Delta ms vs Baseline)	Timeout Rate (Delta vs Baseline)	Cost/Success (Delta vs Baseline)	Stability / Risk Note
Full RL	0.0045	-49.2	-0.0016	-0.017	Best balance of efficiency and safety
No rolling telemetry	0.0011	-18.5	-0.0005	-0.006	Low responsiveness to partial degradation
No action masking	0.0026	-31	-0.001	-0.011	Higher compliance exposure and unsafe suggestions
No conservative regularization	0.0038	-41	-0.0012	-0.02	Higher extrapolation error under drift
No share caps	0.004	-55.5	-0.0013	-0.022	Higher volatility and issuer-risk sensitivity

The table also clarifies why the “Full RL” configuration is the primary result reported in the study. It provides the strongest combined improvements while remaining stable and governable, which matches the constraints of payment platforms where authorization stability and failure avoidance dominate purely opportunistic efficiency gains. In aggregate, the ablation and sensitivity analysis strengthens the causal argument that efficiency gains arise from a combination of state-rich telemetry and safety-aware learning, not from incidental correlations in the test window.

Conclusion

This study demonstrates that Reinforcement Learning can serve as an effective optimization layer for payment gateway orchestration in FinTech platforms by jointly improving tail latency, failure resilience, and unit economics under realistic operational constraints. Across the held-out evaluation window, the learned routing policy consistently reduced p95 latency while maintaining or slightly improving authorization performance, indicating that the gains were not achieved through a narrow “speed-first” trade-off. The results further show that reductions in timeouts and retries are central mechanisms behind improved efficiency, because they directly mitigate congestion amplification and prevent cascading execution failures that degrade customer experience.

Segment-level evidence confirms that the strongest benefits occur in decision-rich and volatile environments, particularly high-volume card traffic and merchant tiers with diverse transaction profiles. The robustness analysis strengthens this finding by showing that the RL policy is comparatively more valuable during incident and peak-load windows, where baseline routing tends to remain concentrated and less responsive to partial degradations. In these stress conditions, the policy’s adaptive diversification and earlier avoidance of degraded routes dampen the escalation of tail latency and failure rates, providing a practical resilience advantage that is directly relevant to revenue protection and service reliability.

Finally, ablation and sensitivity analyses indicate that the observed improvements are attributable to state-rich telemetry and safety-aware learning rather than incidental correlations. Rolling performance indicators enable short-horizon responsiveness, while action masking, conservative offline regularization, and bounded allocation changes preserve deployability by limiting infeasible recommendations and reducing volatility under drift. Taken together, the findings support a deployable blueprint for RL-driven payment optimization that aligns with governance requirements in FinTech. Future work should extend this framework with richer issuer-level context, causal logging policies for stronger off-policy guarantees, and online learning strategies that preserve safety while adapting more rapidly to regime shifts and emerging fraud patterns.

Declarations

Author Contributions

Conceptualization: I.S. and A.N.E.W.; Methodology: A.N.E.W.; Software: I.S.; Validation: I.S. and A.N.E.W.; Formal Analysis: I.S. and A.N.E.W.; Investigation: I.S.; Resources: A.N.E.W.; Data Curation: A.N.E.W.; Writing Original Draft Preparation: I.S. and A.N.E.W.; Writing Review and Editing: A.N.E.W. and I.S.; Visualization: I.S.; All authors have read and agreed to the published version of the manuscript.

Data Availability Statement

The data presented in this study are available on request from the corresponding author.

Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Not applicable.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] P. Gomber, R. J. Kauffman, C. Parker, and B. W. Weber, “On the FinTech revolution: Interpreting the forces of innovation, disruption, and transformation in financial services,” *Journal of Management Information Systems*, vol. 35, no. 1, pp. 220–265, 2018, doi: 10.1080/07421222.2018.1440766.

[2] M. Barroso and J. Laborda, "Digital transformation and the emergence of the Fintech sector: Systematic literature review," *Digital Business*, vol. 2, no. 2, Art. no. 100028, 2022, doi: 10.1016/j.digbus.2022.100028.

[3] T. Khiaonarong, H. Leinonen, and R. Rizaldy, "Operational Resilience in Digital Payments: Experiences and Issues," *IMF Working Papers*, no. 288, 2021, doi: 10.5089/9781616355913.001.

[4] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013, doi: 10.1145/2408776.2408794.

[5] X. Tian, R. Han, L. Wang, G. Lu, and J. Zhan, "Latency critical big data computing in finance," *Journal of Finance and Data Science*, vol. 1, no. 1, pp. 33–41, 2015, doi: 10.1016/j.jfds.2015.07.002.

[6] E. A. Morse and V. Raval, "PCI DSS: Payment card industry data security standards in context," *Computer Law & Security Report*, vol. 24, no. 6, pp. 540–554, 2008, doi: 10.1016/j.clsr.2008.07.001.

[7] G. K. S. Tan, "The 'fintech revolution' is here! The disruptive impact of fintech on retail financial practices," *Finance soc.*, vol. 8, no. 2, pp. 129–148, 2022, doi: 10.2218/finsoc.7763.

[8] R. Bygari, A. Gupta, S. Raghuvanshi, A. Bapna, and B. Sahu, "An AI-powered Smart Routing Solution for Payment Systems," *arXiv*, vol. 2021, no. November, pp. 1-9, 2021, doi: 10.48550/arXiv.2111.00783.

[9] A. Chaudhary, A. Rai, and A. Gupta, "Maximizing Success Rate of Payment Routing using Non-stationary Bandits," *arXiv*, vol. 2023, no. August, pp. 1-7, 2023, doi: 10.48550/arXiv.2308.01028.

[10] A. Agrawal and H. Patil, "A Control-Theoretic Approach to Dynamic Payment Routing for Success Rate Optimization," *arXiv*, vol. 2025, no. October, pp. 1-7, doi: 10.48550/arXiv.2510.16735.

[11] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015, doi: 10.1038/nature14236.

[12] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," *arXiv*, vol. 2015, no. September, pp. 1-10, 2015, doi: 10.48550/arXiv.1509.02971.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv*, vol. 2017, no. July, pp. 1-12, 2017, doi: 10.48550/arXiv.1707.06347.

[14] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," *arXiv*, vol. 2018, no. February, pp. 1-15, 2018, doi: 10.48550/arXiv.1802.09477.

[15] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained Policy Optimization," vol. 2017, no. May, pp. 1-18, 2017, *arXiv*. doi: 10.48550/ARXIV.1705.10528.

[16] S. Fujimoto, D. Meger, and D. Precup, "Off-Policy Deep Reinforcement Learning without Exploration," *arXiv*, vol. 2018, no. December, pp. 1-23, 2018, doi: 10.48550/arXiv.1812.02900.

[17] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative Q-Learning for Offline Reinforcement Learning," *arXiv*, vol. 2020, no. june, pp. 1-31, 2020, doi: 10.48550/arXiv.2006.04779.

[18] N. Jiang and L. Li, "Doubly Robust Off-policy Value Evaluation for Reinforcement Learning," *arXiv*, vol. 2015, no. November, pp. 1-14, 2015, doi: 10.48550/arXiv.1511.03722.

[19] Y. Song, M. Ni, and J. Liu, "Deep reinforcement learning for network routing optimization: A systematic survey," *Neurocomputing*, vol. 666, no. February, Art. no. 132263, 2026, doi: 10.1016/j.neucom.2025.132263.

[20] X. Chen, L. Yao, J. McAuley, G. Zhou, and X. Wang, "Deep reinforcement learning in recommender systems: A survey and new perspectives," *Knowledge-Based Systems*, vol. 264, no. March, Art. no. 110335, 2023, doi: 10.1016/j.knosys.2023.110335.

[21] J. Wang, R. Gao, and H. Zha, "Reliable Off-Policy Evaluation for Reinforcement Learning," *Operations Research*, vol. 72, no. 2, pp. 699–716, 2024, doi: 10.1287/opre.2022.2382.

[22] T. L. Meng and M. Khushi, "Reinforcement Learning in Financial Markets," *Data*, vol. 4, no. 3, p. 110, July 2019, doi: 10.3390/data4030110.

[23] C. P. Buttigieg and B. B. Zimmermann, "The digital operational resilience act: challenges and some reflections on the adequacy of Europe's architecture for financial supervision," *ERA Forum*, vol. 25, no. June, pp. 11–28, 2024, doi: 10.1007/s12027-024-00793-w.

[24] Y. Lou, Q. Zhu, and C. Liang, "Financial technology and firm operational resilience: The roles of supply chain resilience and marketing capability," *International Review of Economics & Finance*, vol. 104, Art. no. 104744, 2025, doi: 10.1016/j.iref.2025.104744.

[25] D. Cumming, S. Johan, and R. Reardon, "Global fintech trends and their impact on international business: a review," *MBR*, vol. 31, no. 3, pp. 413–436, Aug. 2023, doi: 10.1108/MBR-05-2023-0077.