Trust-Based Adaptive LQR Control for Autonomous Multi-Tenant Network Management

Ahmed Ben Ali
Université de Toulouse
LAAS-CNRS
Toulouse, France
Email: abenali@laas.fr

Yann Labit
Université de Toulouse
LAAS-CNRS
Toulouse, France
Email: ylabit@laas.fr

Benoit Nougnanke
Telecom SudParis
Palaiseau, France
Email: knougnanke@telecom-sudparis.eu

***Abstract*—In multi-tenant network environments, deploying Linear Quadratic Regulators at scale is impeded not by theory but by a practical bottleneck: cost matrices require per-instance manual tuning, yet optimal parameters vary across tenants and evolve as traffic dynamics change, making per-instance configuration infeasible at large scale. Prior adaptive and gain-scheduling approaches either require offline enumeration of operating regimes [1] or designer-specified tuning parameters [2], and prior work has not addressed autonomous Q-matrix discovery from online statistical observation of plant behavior. We present a trust-based adaptation mechanism that continuously monitors traffic predictability via Coefficient of Variation analysis and autonomously maps observed statistics to LQR cost matrix parameters, selecting among provably stable controllers without any prior knowledge of tenant traffic profiles. Using NS-3 simulations with multi-phase dynamic traffic and scalability experiments across varying tenant populations, we show the system autonomously discovers the full control spectrum from uniform initialization, significantly reduces queue occupancy and latency compared to manually-tuned fixed LQR, and maintains equivalent fairness at scale. This work demonstrates that statistical plant characterization can drive zero-touch controller synthesis with formal stability guarantees, offering a practical path to autonomous LQR deployment where per-instance expert tuning is infeasible.**

***Index Terms*—Adaptive Control, LQR Control, Autonomous Systems, Q-Matrix Adaptation, Zero-Touch Deployment, Congestion Control**

I. Introduction

Autonomous parameter adaptation in optimal control systems is a longstanding challenge: while Linear Quadratic Regulator (LQR) design provides optimal closed-loop performance guarantees, its cost matrices must be tuned to the plant's operating regime. When plant dynamics vary across instances—as in networked control systems serving thousands of heterogeneous flows—manual per-instance configuration becomes the primary deployment bottleneck. Existing protocols and resource managers address related objectives but use fixed parameters tuned offline, lacking mechanisms for per-instance autonomous adaptation as operating conditions evolve [3], [4]. Unlike prior adaptive controllers, our system does not learn a new policy but autonomously selects among provably stable LQR controllers based on online traffic predictability—achieving online statistical adaptation while preserving the formal stability guarantees of LQR theory. Unlike gain scheduling, which requires offline identification of all operating regimes, the proposed mechanism autonomously discovers scheduling signals directly from online traffic statistics, requiring no prior knowledge of plant behavior.

This paper presents a trust-based LQR adaptation system that removes per-instance manual cost matrix configuration by continuously monitoring traffic predictability via Coefficient of Variation (CV) analysis, automatically mapping observed plant characteristics to cost matrix parameters, and enabling all controllers to initialize identically before autonomously converging to traffic-appropriate control levels. Simulation results demonstrate 36% queue and latency reduction compared to fixed-parameter LQR while maintaining equivalent fairness, with extended evaluation showing continuous adaptation across the complete control spectrum. The approach transforms LQR from a specialist tool requiring expert tuning into a zero-touch solution for large-scale heterogeneous deployments.

We make three contributions: **(1)** a trust-based Q-matrix adaptation mechanism that autonomously selects among provably stable LQR controllers by mapping online CV measurements to cost matrix parameters, with formal stability guarantees across the adaptation spectrum via LMI analysis; **(2)** a zero-touch deployment framework where all controllers initialize identically and autonomously converge to traffic-appropriate control levels without operator intervention; and **(3)** NS-3 simulation validation achieving 36% performance improvement and favorable computational scaling, with resource analysis suggesting deployment viability across software, P4-programmable switch, and FPGA platforms. Section II surveys related work. Section III presents the networked control system model. Section IV describes the trust-based adaptation mechanism. Section V provides evaluation. Section VI concludes.

II. Related Work

Adaptive and Self-Tuning Control. Classical self-tuning regulators adapt controller parameters online by estimating plant characteristics [5], [6], but assume a fixed model structure and do not address multi-instance heterogeneity. Gain scheduling [7] pre-computes controllers for a finite set of operating points and switches between them based on a scheduling variable — the closest prior art to our approach. However, gain scheduling requires offline characterization of all operating regimes, whereas our trust mechanism discovers operating regimes online without prior knowledge. A more recent advance, Campos et al. [1] develop an adaptive gain-scheduling scheme for polytopic uncertain systems that adaptively provides a tuning parameter for gain-scheduling

implementation using LMI-based conditions, proving asymptotic stability under parameter adaptation. While conceptually related, their approach still requires the controller gains to be pre-determined offline by solving a set of LMIs over the polytope vertices — our system removes this offline phase entirely by using CV as an online scheduling signal derived from observed traffic. For LQR specifically, Elkhatem and Engin [2] propose adaptive Q-matrix selection for UAV stabilization under disturbances, adjusting weighting matrices via state variable feedback and a designer-specified preference factor. This demonstrates the broader applicability of online Q-matrix adaptation but relies on a fixed preference factor set offline — our trust mechanism derives the equivalent parameter autonomously from statistical traffic observation. Robust $\mathcal{H}_\infty$ control [8] guarantees stability under bounded plant uncertainty but trades performance for conservatism. Model Predictive Control [9] handles constraints explicitly but is computationally intensive for high-instance-count deployments. Our approach occupies a distinct position: it adapts LQR cost matrices online using a scalar statistical signal (CV), maintaining formal stability guarantees via LMI analysis while requiring no offline regime enumeration, preference factor specification, or predictive model.

Control-Theoretic Congestion Control. Hollot et al. [5], [6] established fluid-flow TCP models and PI/PID controller design for Active Queue Management (AQM) with formal stability guarantees — the foundational framework this work extends. Misra et al. [10] developed fluid-based AQM analysis for network topology changes. These works assume fixed controller parameters tuned for specific operating conditions and do not address the multi-tenant heterogeneity or online parameter adaptation problem. Our previous TAQM work [11] built on this foundation with per-tenant LQR controllers using CUBIC TCP fluid-flow dynamics, but required manual per-tenant cost matrix configuration — the gap this paper closes.

Data-Driven and Application-Layer Approaches. Data-driven methods [12] and end-to-end datacenter protocols [3], [4] address related but orthogonal objectives and lack formal stability guarantees during adaptation—the specific gap this paper closes.

III. Networked Control System Model

This work builds on the TAQM (Tenant-Aware Active Queue Management) architecture previously established in [11], which instantiates a networked control system where each tenant (plant instance) has its own LQR controller regulating queue length via TCP drop probability. We summarize the control model only to highlight the tuning challenge; full derivations and proofs appear in the TAQM paper. TAQM's architecture comprises three components: **(1)** per-tenant packet classification and queueing preventing cross-tenant interference, **(2)** weighted fair scheduling ensuring each tenant i receives bandwidth $BW_i = \frac{w_i}{\sum_j w_j} C$ proportional to configured weights w_i, and **(3)** per-tenant LQR controllers computing drop probabilities $p_i(t)$ to minimize queue occupancy while maintaining stability. Figure 1 illustrates this established architecture augmented with our trust-based adaptation mechanism (yellow/orange components).

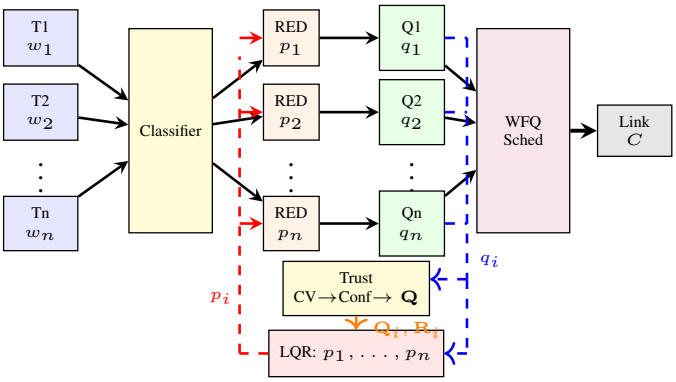

Fig. 1. Trust-based LQR architecture built on TAQM foundation [11]. Queue observations (blue) feed both Trust and LQR modules. **Trust module (our contribution)** computes confidence from traffic patterns and modulates LQR **Q**-matrix (orange). LQR computes drop probabilities p_i (red) for AQM modules.

CUBIC TCP Model. TAQM uses CUBIC TCP fluid-flow dynamics [11] to model congestion window evolution:

$$\frac{dW_i}{dt} = \frac{\alpha_i}{\varrho_0} - \beta_i \cdot \frac{W_i(t) \cdot W_i(t-\varrho_0) \cdot p_i(t-\varrho_0)}{\varrho_0} \tag{1}$$

and queue dynamics:

$$\frac{dq_i}{dt} = \frac{N_i(t) \cdot W_i(t)}{\varrho_i(t)} - C_i \tag{2}$$

where N_i is flow count, $\varrho_i(t) = \varrho_0 + q_i(t)/C_i$ is RTT, $C_i = \frac{w_i}{\sum_j w_j} C$ is allocated bandwidth from weighted fair scheduling, and α_i, β_i are TCP parameters. All model function hypotheses including stability conditions and equilibrium existence were verified in the prior work [11]. Linearization around equilibrium (W_0, q_0, p_0) yields state-space form:

$$\frac{d}{dt}\begin{bmatrix}\delta W_i \\ \delta q_i\end{bmatrix} = \mathbf{A}_i \begin{bmatrix}\delta W_i \\ \delta q_i\end{bmatrix} + \mathbf{B}_i \delta p_i \tag{3}$$

where matrices $\mathbf{A}_i$ and $\mathbf{b}_i$ depend on operating point and system parameters. The architecture ensures per-tenant isolation—tenant i's dynamics depend only on (W_i, q_i, p_i), not other tenants' states—while weighted fair scheduling guarantees B_i independently of queue lengths, decoupling fairness from queue optimization.

LQR Control Framework: For each tenant i, TAQM's LQR controller minimizes the cost function:

$$J_i = \int_0^\infty \left(\mathbf{x}_i^T \mathbf{Q}_i \mathbf{x}_i + u_i^T \mathbf{R}_i u_i\right) dt \tag{4}$$

where $\mathbf{x}_i = [\delta W_i, \delta q_i]^T$ is state deviation, $u_i = \delta p_i$ is control input, $\mathbf{Q}_i = \text{diag}(q_w, q_e)$ penalizes window and queue deviations, and $\mathbf{R}_i$ penalizes control effort.throughout this work we will consider $\mathbf{R}_i = 1$. The optimal control law is:

$$u_i(t) = -K_i \mathbf{x}_i(t), \quad \mathbf{K}_i = \mathbf{R}_i^{-1} \mathbf{B}_i^T \mathbf{P}_i \tag{5}$$

where P_i solves the Continuous-time Algebraic Riccati Equation (CARE):

$$\mathbf{A}_i^T \mathbf{P}_i + \mathbf{P}_i \mathbf{A}_i - \mathbf{P}_i \mathbf{B}_i \mathbf{R}_i^{-1} \mathbf{B}_i^T \mathbf{P}_i + \mathbf{Q}_i = 0 \tag{6}$$

The established design sets $q_w, i = 1.0$ and varies $q_e, i \in [q_{\min}, q_{\max}]$ to control aggressiveness. Small q_e, i (e.g., 0.0002) yields aggressive control minimizing queues but becomes unstable under traffic bursts—rapid rate fluctuations cause large corrections that overshoot, creating oscillations. Large q_e (e.g., 0.04) yields conservative control maintaining stability during bursts but tolerating unnecessarily large queues for steady traffic. The fundamental challenge is that optimal q_e must match traffic burstiness: steady flows require small q_e; bursty flows require large q_e to avoid destabilization.

The Gap Our Work Addresses. The original TAQM system requires manual selection of q_e based on expected traffic patterns—operators must configure predictable flows with small q_e and variable flows with large q_e. This manual tuning demands: **(1)** prior knowledge of traffic characteristics, **(2)** expert understanding of LQR sensitivity, and **(3)** static configuration unable to adapt to workload changes. With thousands of tenants exhibiting diverse, time-varying patterns, manual per-tenant tuning becomes the deployment bottleneck preventing practical large-scale operation. **Our contribution addresses this bottleneck through trust-based adaptation that automatically discovers appropriate q_e values by continuously observing traffic patterns, enabling zero-touch deployment where all tenants initialize identically and the system autonomously converges to traffic-appropriate control levels without operator intervention.**

IV. Trust-Based Adaptation Mechanism

The adaptation pipeline executes three stages every $\Delta_{\text{trust}} = 5$s per tenant (Algorithm 1): **(S1)** compute traffic burstiness via CV over a 30s sliding window; **(S2)** update scalar confidence $c_i \in [0, 100]$ by mapping CV to a signed increment; **(S3)** map c_i to $q_{e,i}$ via exponential modulation and re-solve the CARE if the change is significant. The core insight is that traffic *predictability*—rate variability relative to the mean—directly determines safe control aggressiveness: steady traffic tolerates small q_e (tight regulation without overshoot); bursty traffic requires large q_e to avoid destabilization.

Algorithm 1 Trust-Based Q-Matrix Adaptation (per tenant i, every Δ_{trust})

// S1: Traffic predictability
Compute μ_i, σ_i over sliding window ($SW = 30$s)
$\text{CV}_i \leftarrow \sigma_i / \mu_i$
// S2: Confidence update
$\delta_i \leftarrow \text{CVTOINCREMENT}(\text{CV}_i)$ *(Eq. 10)*
$c_i \leftarrow \text{sat}(c_i + \delta_i,\ 0,\ 100)$
// S3: Q-matrix modulation and gain recomputation
$q_{e,i} \leftarrow q_{\max} \cdot (q_{\min}/q_{\max})^{c_i/100}$ *(Eq. 11)*
if $|\Delta c_i| \geq 1.0$ **then**
  Solve CARE $\Rightarrow \mathbf{P}_i$; compute $\mathbf{K}_i \leftarrow \mathbf{R}_i^{-1}\mathbf{B}_i^T\mathbf{P}_i$
end if

A. Traffic Predictability via Coefficient of Variation

For tenant i, we monitor packet arrivals over observation period $\Delta_{\text{obs}} = 0.1$ seconds, computing instantaneous sending rate $r_i(k) = \frac{\text{packets}_{i,k} - \text{packets}_{i,k-1}}{\Delta_{\text{obs}}}$ at each interval k. Over a sliding window of $SW = 30$ seconds (300 samples), we compute:

$$\mu_i = \frac{1}{300} \sum_{j=k-299}^{k} r_i(j) \tag{7}$$

$$\sigma_i = \sqrt{\frac{1}{300} \sum_{j=k-299}^{k} (r_i(j) - \mu_i)^2} \tag{8}$$

$$\text{CV}_i = \frac{\sigma_i}{\mu_i} \tag{9}$$

The Coefficient of Variation quantifies traffic burstiness: CV ≈ 0 indicates constant rate (highly predictable); CV > 2 indicates highly variable patterns (unpredictable). This metric directly maps to control strategy: low CV traffic can sustain aggressive control without oscillation risk, while high CV traffic requires conservative control to maintain stability. The 30-second window balances responsiveness to pattern changes with stability against transient fluctuations.

Rationale for CV metric: Standard deviation alone lacks scale-invariance. Hurst parameter requires long traces unsuitable for real-time adaptive control. Spectral analysis has high computational cost. CV provides a normalized burstiness measure with O(1) online computation via running statistics, clear interpretation, and direct mapping to control parameters—properties essential for autonomous operation at scale.

B. Confidence Updates and Policy Discovery

We maintain per-tenant confidence $c_i(t) \in [0, 100]$ initialized at 50 (neutral). Every $\Delta_{\text{trust}} = 5.0$ seconds, the system updates confidence based on observed CV:

$$\delta_{\text{CV}_i}(\text{CV}_i) = \begin{cases} +2.0 & \text{if CV}_i < 0.5 \text{ (very steady)} \\ +0.5 & \text{if } 0.5 \leq \text{CV}_i < 1.0 \text{ (steady)} \\ 0.0 & \text{if } 1.0 \leq \text{CV}_i < 1.5 \text{ (moderate)} \\ -0.5 & \text{if } 1.5 \leq \text{CV}_i < 2.0 \text{ (bursty)} \\ -2.0 & \text{if CV}_i \geq 2.0 \text{ (very bursty)} \end{cases} \tag{10}$$

The CV thresholds derive from empirical analysis of datacenter traffic traces [13]: constant-rate flows exhibit CV $<$ 0.5; moderate interactive traffic shows CV $\in [0.5, 1.5]$; highly variable patterns exceed CV = 2.0. The zero increment for CV $\in [1.0, 1.5]$ provides a deliberate dead zone around the moderate regime, preventing unnecessary CARE re-solves and avoiding confidence oscillation for tenants near the stability boundary. The asymmetric magnitude design (± 2.0 for extremes, ± 0.5 for moderate) is deliberate: at the maximum rate of ± 2.0 per $\Delta_{\text{trust}} = 5$s, worst-case convergence from neutral ($c = 50$) to any extreme ($c = 0$ or $c = 100$) requires 125 seconds. This intentionally exceeds the 30-second CV observation window, ensuring that the controller cannot react faster than the statistical estimator can resolve traffic patterns—a design choice that prevents oscillatory confidence updates in the presence of transient traffic bursts.

C. Automatic Policy Enforcement via Q-Matrix Modulation

Confidence maps to $\mathbf{Q}$-matrix error weight via exponential modulation:

$$q_{e,i} = q_{\max} \times \left(\frac{q_{\min}}{q_{\max}} \right)^{c_i/100} \tag{11}$$

with boundaries $q_{\min} = 0.0002$ (aggressive) and $q_{\max} = 0.04$ (conservative). This provides:

$$c_i = 0 \Rightarrow q_{e,i} = q_{\max} = 0.04 \text{ (conservative policy)} \quad (12)$$

$$c_i = 50 \Rightarrow q_{e,i} = \sqrt{q_{\min} \cdot q_{\max}} \approx 0.00283 \text{ (neutral)} \quad (13)$$

$$c_i = 100 \Rightarrow q_{e,i} = q_{\min} = 0.0002 \text{ (aggressive policy)} \quad (14)$$

The exponential form ensures: **(1)** smooth variation—small confidence changes yield proportionally small q_e changes, preventing abrupt policy shifts, and **(2)** wide range—confidence span [0,100] maps to 200× q_e variation, enabling the complete control spectrum from conservative to aggressive operation. This automatic policy enforcement requires no operator intervention.

D. Controller Recomputation and Stability Guarantees

When confidence changes trigger q_e update, the system recomputes LQR gains by updating $\mathbf{Q}_i = \text{diag}(1.0, q_{e,i})$, solving CARE for new $\mathbf{P}_i$, and computing new gain $\mathbf{K}_i = \mathbf{R}_i^{-1}\mathbf{B}_i^T\mathbf{P}_i$. Recomputation occurs only when $|c_i(t) - c_i(t - \Delta_{\text{trust}})| \geq 1.0$, preventing excessive computation for minor variations. CARE solution takes <1ms per tenant on modern hardware, negligible overhead for 5-second update intervals.

Since q_e varies within the bounded interval $[q_{\min}, q_{\max}]$, we can certify closed-loop stability for all admissible $\mathbf{Q}_i$ values simultaneously. Specifically, the adaptation defines a polytopic parameter variation: $\mathbf{Q}_i(\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & \lambda q_{\min} + (1-\lambda) q_{\max} \end{pmatrix}$, $\lambda \in [0,1]$, $\lambda \in [0,1]$. A common quadratic Lyapunov function $V(x) = x^T\mathbf{P}x > 0$ certifying $\dot{V} < 0$ for all λ can be found by solving the LMI:

$$(\mathbf{A}_i - \mathbf{B}_i\mathbf{K}_i(\lambda))^T\mathbf{P} + \mathbf{P}(\mathbf{A}_i - \mathbf{B}_i\mathbf{K}_i(\lambda)) \prec 0, \quad \forall\lambda \in [0,1] \quad (15)$$

Since the closed-loop matrix is affine in λ, feasibility at the two vertices $\lambda \in \{0,1\}$ implies feasibility for all λ, so only two LMI checks are required;both were verified at the simulated operating point (C_i=8.57 Mbps, N_i=5 flows, ϱ_0=0.1 s, q_0=10 pkts): the CARE solution at $q_{\max}$ yields $\mathbf{P} = \left[\begin{smallmatrix} 0.0167 & 0.0012 \\ 0.0012 & 0.0020 \end{smallmatrix}\right] \succ 0$ (eigenvalues 0.0019, 0.0167), which also satisfies $\max\lambda(A_{\text{cl}}^T\mathbf{P} + \mathbf{P}A_{\text{cl}}) = -0.035$ at the aggressive vertex ($q_{\min}$, poles -62.5, -8.8) and -0.042 at the conservative vertex ($q_{\max}$, poles -61.8, -12.6), confirming $\dot{V} < 0$ across the full adaptation range.

E. Zero-Touch Deployment and Continuous Adaptation

The mechanism provides zero-touch deployment: all tenants initialize at $c_i(0) = 50$ (neutral $q_e \approx 0.003$), then autonomously converge to traffic-appropriate levels without manual intervention. For a constant-rate tenant with CV = 0.3, confidence gradually increases ($\delta = +2$ per 5s) toward aggressive control; for a bursty tenant with CV = 3.5, confidence decreases ($\delta = -2$ per 5s) toward conservative control. Operators deploy with uniform initialization—no per-tenant configuration, traffic profiling, or expert tuning required.

Unlike fixed LQR requiring manual retuning when workloads change, trust-based adaptation continuously adjusts throughout system lifetime. When tenant traffic patterns shift, CV tracking detects the change, confidence updates follow, and control aggressiveness adapts automatically—handling tenant onboarding/offboarding, seasonal pattern changes, and application evolution without operator intervention.

V. Experimental Evaluation

A. Experimental Setup

We implement trust-based adaptation in NS-3.44 using the TAQM architecture [11]. Topology: 27 tenants (9 per class) connect via 100 Mbps access links (1ms delay) to a router with 30 Mbps bottleneck (10ms delay). Tenant classes have weights $w = \{0.5, 1.0, 2.0\}$ yielding bandwidth allocations $\{4.3, 8.6, 17.1\}$ Mbps. All flows use CUBIC TCP with 1448-byte packets; The base RTT of ≈100ms is intentionally adversarial relative to sub-millisecond datacenter RTTs: fluid-flow stability margins improve as R_0 decreases [5] (closed-loop poles scale with $1/R_0$), so realistic intra-rack delays would tighten poles and accelerate confidence convergence. The tested results therefore constitute a conservative lower bound on adaptation performance. Trust parameters: observation $\Delta_{\text{obs}} = 0.1$s, window $SW = 30$s, update $\Delta_{\text{trust}} = 5$s. LQR parameters: control $\Delta t = 0.5$s, $\mathbf{Q} = \text{diag}(1.0, q_e(c))$.

Metrics: Weighted Jain's Index for fairness [11], queue occupancy (packets), end-to-end latency (ms), confidence evolution (%), CV tracking. We evaluate three aspects: **(1)** performance improvement over baselines, **(2)** autonomous adaptation across dynamic traffic patterns, and **(3)** deployment feasibility at scale.

B. Baseline Comparison and Performance Gains

We compare three approaches under 120-second simulation with mixed-burst traffic: **(1) No Control**—per-tenant architecture with static RED ($p = 0.02$), **(2) Fixed LQR**—conservative fixed $\mathbf{Q} = \text{diag}(1.0, 0.001)$, and **(3) Trust-Based LQR**—autonomous adaptation with zero-touch deployment. All use identical weighted fair scheduling. The Fixed LQR setting $q_e = 0.001$ reflects best-practice manual tuning for heterogeneous deployments: without prior knowledge of each tenant's traffic pattern, operators must configure conservatively to guarantee stability against worst-case bursty tenants, sacrificing performance on steady-traffic tenants. Trust-based adaptation removes this trade-off by discovering per-tenant optimal settings autonomously.

TABLE I
Performance Comparison: Autonomous Adaptation vs Fixed Approaches

Approach	Fairness	Queue (pkt)	Latency (ms)
No Control	0.934	46.9	118.3
Fixed LQR	0.934	17.7	53.6
Trust-Based (Ours)	0.930	**11.3**	**34.2**

36% improvement vs manually-tuned Fixed LQR.

Table I and Figure 2 demonstrate the performance gains of autonomous adaptation. All approaches achieve equivalent fairness (0.930–0.934, ≈0.93), validating architectural decoupling—weighted fair scheduling guarantees bandwidth allocation regardless of queue control policy. Trust-Based LQR reduces mean queue to 11.3 packets versus 17.7 for manually-tuned Fixed LQR (**36% reduction**), with corresponding latency improvement from 53.6ms to 34.2ms (**36% reduction**).

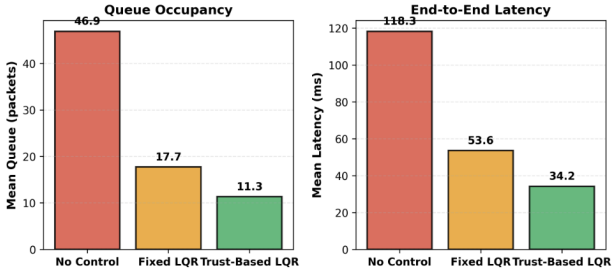

Fig. 2. Performance comparison across queue occupancy and end-to-end latency metrics. Trust-Based LQR (green) achieves 36% reduction in both metrics versus manually-tuned Fixed LQR (orange): queue reduces from 17.7 to 11.3 packets, latency from 53.6ms to 34.2ms. No Control baseline (red) demonstrates unmanaged performance.

The autonomous system discovers more aggressive control ($q_e \approx 0.002$) than conservative manual configuration ($q_e = 0.001$), demonstrating that automatic policy discovery outperforms a representative conservative manual configuration for heterogeneous workloads where per-tenant optimal settings are unknown a priori.

C. Autonomous Adaptation Across Dynamic Traffic

We demonstrate continuous adaptation over 600 seconds with four traffic phases validating zero-touch operation. Using 9 tenants (3 per class) enables detailed per-tenant visualization. Traffic phases: **(1)** All moderate burst (0-150s), **(2)** Differentiated patterns—Light→constant, Normal→heavy, Premium→moderate (150-300s), **(3)** All heavy burst (300-450s), **(4)** Selective recovery (450-600s).

Figure 3 shows autonomous operation: all tenants begin at confidence=50% (red dashed "Initial" line), representing zero-touch deployment with uniform initialization. During initial 150 seconds with uniform moderate-burst traffic, tenants remain near initialization (45–55%), demonstrating stable baseline. At t=150s, traffic patterns change substantially—the system autonomously discovers appropriate policies achieving **0-100% confidence span across all tenant classes**: Light tenants with constant traffic reach 0–20% (aggressive control), Normal tenants with heavy bursts decrease to 30% (conservative control), Premium tenants increase toward 80–100% (aggressive control). We note explicitly that the exponential mapping of Eq. (11) is an engineering design choice rather than an optimality derivation: LMI analysis certifies stability at the boundary points $q_{\min}$ and $q_{\max}$; the specific trajectory between them is selected for its smoothness and dynamic range properties, consistent with standard gain-scheduling practice [7]. Deriving an optimal mapping from CV to q_e analytically remains an open problem. This automatic policy enforcement requires no operator intervention.

At t=300s, all tenants experience uniform heavy bursts creating collective stress. Confidence decreases collectively toward 0–30%, demonstrating CV-driven adaptation responding to traffic characteristics rather than time-based schedules. At t=450s, differentiated patterns return, yielding selective recovery with final states reflecting traffic characteristics: Light 37%, Normal 93%, Premium 100%. This intelligent pattern-dependent recovery—where each tenant class autonomously converges to appropriate control levels based on observed traffic rather than returning to uniform settings—demonstrates traffic-aware operation. Traditional fixed LQR would require manual retuning at each phase transition; autonomous adaptation handles all transitions without operator intervention.

Measured adaptation: Light 0–50% confidence (14.1× Q-variation), Normal 30–93% (**26.7**×—widest due to most diverse patterns), Premium 50–100% (14.1×). These values reflect the confidence ranges actually traversed (e.g., 63 percentage points for Normal), within the full 200× design capacity across $[0, 100\%]$. Queue remains bounded 10–30 packets despite large control variation, consistent with LMI-certified stability. Mean queue: 13.0 packets across all phases. **Key finding:** Full-range autonomous adaptation (0–100% confidence) demonstrates complete control spectrum utilization from uniform zero-touch initialization, automatically discovering 26.7× parameter variation without manual tuning or traffic profiling.

D. Scalability and Deployment Feasibility

We validate scalability across 3–243 tenants per bottleneck link. The trust mechanism exhibits favorable computational scaling: per-tenant processing decreases from 3100ns per tenant (3 tenants) to 40ns per tenant (243 tenants) due to amortization effects. Total per-packet processing remains O(1) at 6–9μs regardless of tenant count, demonstrating aggregate overhead dominated by fixed-cost operations (WFQ scheduling, packet classification) rather than per-tenant adaptation computation. Memory footprint grows sub-linearly at $O(N^{0.89})$: 3KB for 3 tenants to 130KB for 243 tenants per link.

Distributed architecture: The system is distributed per-link—each bottleneck independently manages only traversing tenants. A datacenter with 1000 total tenants and 50 bottleneck links has 20–100 tenants per link, with each link running independent adaptation. This distributed approach ensures per-link overhead remains bounded as total tenant count scales to thousands, enabling practical large-scale deployment.

Parameter robustness: Performance variation remains <5% across $SW \in [10, 60]$s and $\Delta_{\text{trust}} \in [2, 10]$s, except pathological combinations (very short window <10s with rapid updates <2s causing oscillations). Current defaults ($SW = 30$s, $\Delta_{\text{trust}} = 5$s) work effectively across tested tenant counts (3–243) and traffic patterns. CV thresholds tolerate ±50% variations with <8% performance impact. $Q_{\min}$ and $Q_{\max}$ boundaries tolerate ±30% with <10% performance change. This robustness enables zero-touch deployment with default parameters across different bottleneck locations (core, aggregation, Top of Rack) and traffic mixes without per-site tuning—operators deploy identical configurations at all bottlenecks.

Implementation platforms: CV computation is O(1) via running statistics, confidence updates every 5s, and CARE solving <1ms per tenant, yielding <5% CPU for typical bottlenecks (50–100 tenants). For P4-programmable switches, the tested configuration (243 tenants) would require 608KB—well within typical switch SRAM capacity (10–32MB) [14]. These resource projections suggest deployment viability across software, P4-programmable switch, and FPGA platforms at strategic bottlenecks, with each point independently managing local traffic and no cross-link coordination required.

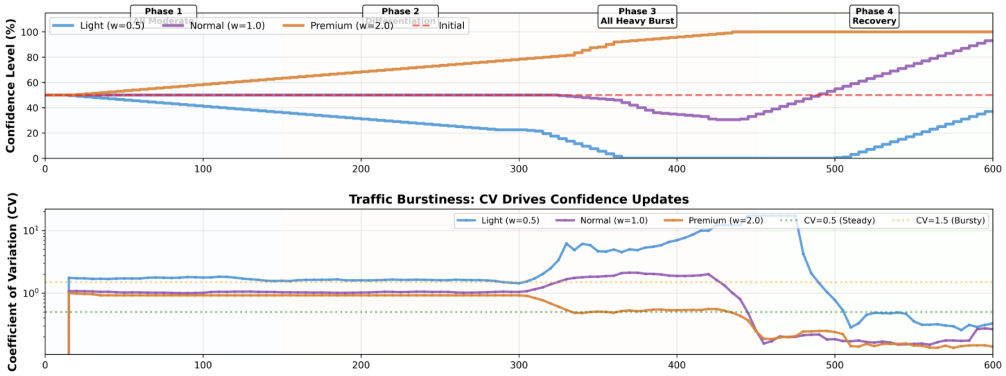

Fig. 3. Autonomous adaptation over 600s demonstrating zero-touch operation. **Phase 1 (0-150s):** Self-calibration from uniform initialization at confidence=50% (red dashed line). Phase 2 (150-300s): Automatic policy discovery achieving 0-100% confidence range. Phase 3 (300-450s): Collective stress response. Phase 4 (450-600s): Selective traffic-aware recovery. System autonomously discovers Q-matrix variation up to 26.7× without manual tuning. Queue remains bounded between 10 and 30 packets throughout this experiment.

VI. Conclusion

We presented a trust-based adaptive LQR system that autonomously tunes cost matrix parameters from online traffic observations, removing per-tenant manual configuration for multi-tenant network deployments. The system continuously monitors traffic predictability via CV analysis and automatically discovers per-tenant control policies without operator intervention. All tenants initialize identically at neutral settings and autonomously converge to traffic-appropriate control levels, achieving 26.7× parameter variation across observed workload diversity. NS-3 evaluation under the tested conditions demonstrates: **(1)** 36% queue and latency reduction versus manually-tuned Fixed LQR with equivalent fairness (≈0.93), showing automatic policy discovery outperforms a representative conservative manual configuration for heterogeneous workloads, **(2)** continuous autonomous adaptation over 600 seconds achieving 0–100% confidence range across dynamic traffic phases without operator intervention, and **(3)** scalability to 243 tenants per link with O(1) processing and distributed per-link architecture. LMI analysis certifies closed-loop stability across the full adaptation spectrum.

The trust-based approach transforms LQR from a manually-configured specialist tool into a self-optimizing controller suitable for large-scale heterogeneous deployments. Operators initialize the system uniformly—no per-instance configuration, plant profiling, or expert tuning required. The system continuously adapts as plant characteristics evolve throughout deployment lifetime, handling instance onboarding, pattern shifts, and operating point changes without manual intervention. Implementation feasibility across software, P4-programmable switch, and FPGA platforms suggests practical deployment readiness.

Future work includes exploring multi-dimensional traffic features beyond CV for enhanced policy discovery, asymmetric confidence update rates for improved transient response, and coordinated adaptation considering aggregate bottleneck state for system-wide optimization. Several limitations warrant future investigation: the CV metric, while computationally efficient, may not capture heavy-tailed distributions or correlated bursts; evaluation uses conservative 100ms RTTs and up to 243 tenants per link, and real testbed validation at datacenter scale remains a natural next step. The trust-based adaptation principle—using statistical characterization to guide automatic controller synthesis—extends naturally to other autonomous control problems where optimal cost matrix parameters vary with observed plant behavior.

References

[1] V. C. da Silva Campos, A.-T. Nguyen, and R. M. Palhares, "Adaptive gain-scheduling control for continuous-time systems with polytopic uncertainties: An LMI-based approach," *Automatica*, vol. 133, p. 109856, 2021.

[2] A. S. Elkhatem and S. N. Engin, "Robust LQR and LQR-PI control strategies based on adaptive weighting matrix selection for a UAV position and attitude tracking control," *Alexandria Engineering Journal*, vol. 61, no. 8, pp. 6275–6292, 2021.

[3] M. Alizadeh, A. Greenberg, D. A. Maltz *et al.*, "DCTCP: Efficient packet transport for the commoditized data center," in *Proc. ACM SIGCOMM*, 2010.

[4] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types," in *Proc. USENIX NSDI*, 2011.

[5] C. V. Hollot, V. Misra, D. Towsley, and W. Gong, "A control theoretic analysis of RED," *IEEE/ACM Trans. Networking*, vol. 9, no. 3, pp. 266–277, 2001.

[6] ——, "On designing improved controllers for AQM routers supporting TCP flows," in *Proc. IEEE INFOCOM*, 2002.

[7] P. F. Quet, B. Ying, and H. Ozbay, "Delay scheduling for improved delay robustness in AQM design," in *Proc. American Control Conference*, 2003.

[8] K. Zhou and J. C. Doyle, *Essentials of Robust Control*. Prentice Hall, 1996.

[9] A. Bemporad, M. Mannelli, and M. Morari, "Model predictive control for networks of linear systems," in *Proc. IEEE CDC*, 2002.

[10] V. Misra, W. Gong, and D. Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows," *IEEE/ACM Trans. Networking*, vol. 8, no. 3, pp. 277–290, 2000.

[11] A. Ben Ali, Y. Labit, and B. Nougnanke, "TAQM: Guaranteeing fairness and minimizing SLA violations for self-managing datacenters," *IEEE ICIN*, 2026.

[12] N. Jay, N. Rotman, B. Godfrey *et al.*, "A deep reinforcement learning perspective on internet congestion control," in *Proc. ACM SIGCOMM*, 2019.

[13] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," *ACM SIGCOMM CCR*, vol. 40, no. 4, 2010.

[14] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda, "Less is more: Trading a little bandwidth for ultra-low latency in the data center," in *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, 2012, pp. 253–266.