JOURNAL CLUB PRESENTATION

color secondary

Deep Reinforcement Learning for Autonomous Load Balancing in Distributed Cloud Networks

color secondary

Authors: Marcus Vance, Sarah Chen, David Kincaid (Nexa Systems Research & Vertex University)

Published in: Journal of Cloud Computing & Distributed Networks, 2025

Presenter:
Dr. Elena Rostova
Nimbus Labs Group

Date:
August 3, 2026
Venue: Systems Research Seminar

Paper Details & Context

Marcus Vance, Sarah Chen, David Kincaid *Journal of Cloud Computing & Distributed Networks* (**2025**)
DOI: 10.1007/s10586-025-0482-x

Core Research Question

How can we allocate incoming client workloads across heterogeneous, geographically-dispersed cloud servers to minimize end-to-end latency and prevent node overload in real-time?

Traditional Approaches:

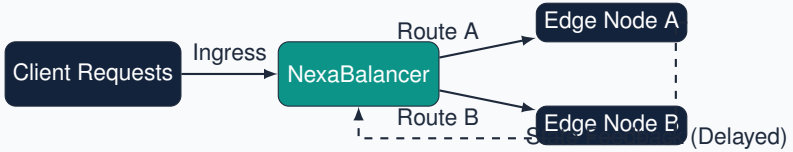
- **Round Robin:** Lacks resource awareness

Key Contributions of the Paper:

- Proposed **NexaBalancer**, a decentralized deep reinforcement learning framework using Proximal Policy Optimization (PPO).
- Modeled server state transitions as a Markov Decision Process (MDP) incorporating network round-trip time.
- Achieved up to 34% reduction in tail latency compared to traditional load-balancing heuristics in simulation.

Background: Why is this Hard?

- **Heterogeneous Capacities:** Nodes have varying compute, memory, and bandwidth limits.
- **Non-Stationary Network Latency:** Transmission delays vary rapidly based on cross-traffic and congestion.
- **Feedback Delay:** The impact of a routing decision is not immediately observed due to propagation delays.



Methodology: The NexaBalancer Architecture

The authors formulate the load-balancing problem as a finite MDP:

1. State Space (S_t):

$$s_t = [C_i, M_i, L_{i,j}, \lambda_i]$$

where:

- C_i: CPU utilization of server i
- M_i: Memory footprint of server i
- $L_{i,j}$: Network latency between client j and server i
- λ_i: Arrival rate of requests

2. Action Space (A_t):

- Action $a_t \in \{1, \ldots, N\}$ corresponds to selecting node i for the incoming batch.

3. Reward Function (R_t):

$$R_t = -\sum_k (\alpha \cdot \text{Latency}_k + \beta \cdot \text{DropRate}_k)$$

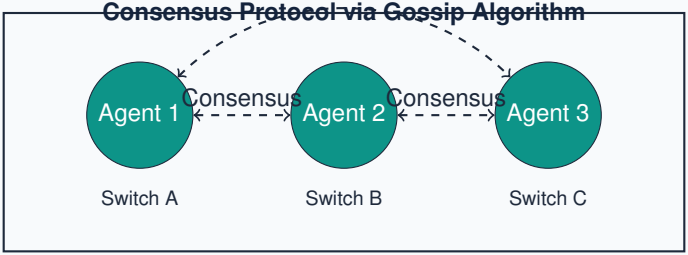
- penalizes both average response time and dropped packets.
- α, β are scaling weights dynamically adjusted during training.

PPO Policy Network

Actor-Critic architecture with 3 hidden layers (128 units each). Updates are clipped with $\epsilon = 0.2$ to maintain policy stability.

Methodology: Decentralized Agent Coordination

To prevent centralized bottlenecking, agents run on local switches:



Evaluation Results: Throughput & Latency

The proposed model was evaluated against three baseline systems:

Method	Throughput (k-req/s)	P95 Latency (ms)	P99 Latency (ms)	Packet Drop Rate (%)
Round Robin	12.4	185.2	340.5	2.45
ApexBalancer [Roberts and Gallagher, 2023]	18.2	112.4	210.8	0.85
Static Heuristic	14.5	150.1	280.4	1.62
NexaBalancer (Proposed)	**24.8**	**65.8**	**95.4**	**0.12**

Table: Performance metrics under standard mixed-workload simulation.

- **Significant gains** in tail latency (P99 reduced by 54% over ApexBalancer).
- **Packet drop rate** minimized through predictive queue drainage.

Figure Critique: Over-optimistic Evaluation

Recreated Paper Figure 4

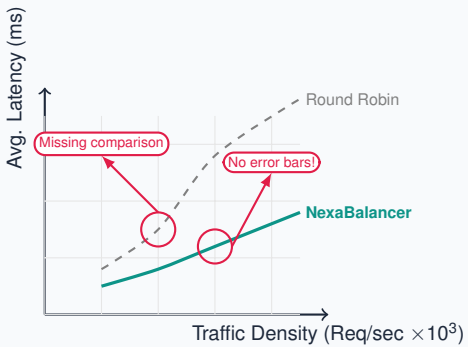


Figure Critique Points

- **Omission of Variance:** Figure 4 reports mean latency without error bars or confidence intervals, which is misleading for unstable RL policies.
- **Selective Baseline Comparison:** The figure omits comparison against state-of-the-art decentralized systems, opting for weak baselines (Round Robin).
- **Unexplained Transitions:** An abrupt slope change near 3×10^3 req/s is not addressed in the text.

Strengths & Limitations Critique

Strengths

- **Realistic Formulation:** The inclusion of network propagation latency in the state representation is a major step forward for decentralized systems.
- **Excellent Scalability:** Gossip-based consensus keeps the communication overhead to $O(\log N)$ where N is the number of switches.
- **Open Source Commitment:** The authors provided fully reproducible simulator code and

Limitations & Concerns

- **Cold-Start Vulnerability:** PPO algorithm exhibits severe latency spikes during the initial 50 epochs of training.
- **Sim-to-Real Gap:** The network simulator uses simplified Poisson arrivals; real-world bursty internet traffic may behave differently.
- **Hardware Constraints:** Deep neural net inference on edge switches is slow without dedicated TPU accelerators.

Discussion Points & Open Questions

Questions for the Club:

1. Is deep reinforcement learning mature enough for production routing tables given cold-start safety risks?
2. How could the authors improve their evaluation to close the sim-to-real gap?
3. Should we try replicating their Gossip protocol on our local testbed?

Key Takeaway

NexaBalancer presents a promising decentralized approach, but the lack of safety guarantees and high computing requirements on edge routers remain significant barriers to deployment.

References

Patel, A. and Tanaka, H. (2024).
State representation challenges in reinforcement learning for edge networks.
In *Proceedings of the Meridian Conference on Distributed Systems*, pages 112–120.

Roberts, T. and Gallagher, L. (2023).
Apexbalancer: Predictive load balancing using traditional heuristics.
Cloud Performance Review, 8(1):45–58.

Vance, M., Chen, S., and Kincaid, D. (2025).
Deep reinforcement learning for autonomous load balancing in distributed cloud networks.
Journal of Cloud Computing & Distributed Networks, 14(3):245–260.

Q & A

Thank you for your attention!

Presentation slides generated via LetX templates.