A Simulation Study of AI Traffic Patterns and their Impact on Data Center Network Performance

Mike Adedoyin, PhD
University of the Cumberlands
United States

ABSTRACT

AI training workloads are changing the way traffic behaves inside modern data centers. Unlike many traditional enterprise applications, distributed training jobs often create synchronized bursts during collective operations such as AllReduce and gradient synchronization. These bursts can temporarily overload shared links and expose limitations in switch queueing, transport behavior, and capacity planning methods that rely primarily on average utilization.

This paper uses NS-3 3.43 to simulate AI-like burst traffic in a small leaf-spine bottleneck topology. Four scenarios are evaluated: normal TCP enterprise traffic, a moderate synchronized burst load of 120 Mbps, a heavy burst load of 808 Mbps, and the same heavy load with FQ-CoDel replacing FIFO queueing. Results show that even moderate average overload can produce severe performance degradation when traffic is synchronized. At 120 Mbps, packet loss reaches 52.8% and average delay nearly doubles. At 808 Mbps, loss increases to 92.9%. FQ-CoDel reduces average delay by approximately 27% and improves inter-flow delay fairness, although throughput remains constrained under heavy load.

The study provides a compact and reproducible simulation baseline for analyzing AI-driven incast behavior under controlled overload conditions.

Keywords

AI traffic, data center networks, NS-3, congestion, incast, FQ-CoDel, queue delay, packet loss, flow fairness, simulation

1. INTRODUCTION

Anyone who has spent time monitoring AI training workloads will notice something that jumps at them very quickly, and that is the traffic patterns of AI training workloads look nothing like traditional enterprise flows. In normal applications, traffic tends to be well spread out, they are self-paced, and relatively tolerant of moderate delays; AI training traffic, particularly during gradient synchronization, is the opposite. All the worker nodes send at nearly the same time, the bursts are large, and they hit the same bottleneck links at once. The result of this is a pattern called incast congestion, and it can cause serious problems even when the average offered load appears manageable on paper.

The reason this matters for network design is straightforward: most capacity planning tools look at average utilization. For example, if a link averages around 80% utilization, an engineer could reasonably assume that there is still some usable headroom. But if this 80% average comes from eight nodes all sending at once for 100 ms at a time, the instantaneous load during each burst is 8× higher. This is a fundamentally different situation from eight nodes sending steadily, and standard FIFO queues respond to it quite differently than you might expect.

Simulation is one of the better tools for studying this kind of behavior, because it allows exact control over traffic timing and load levels in a repeatable environment. Large-scale AI training clusters typically use leaf-spine topologies [3,8,12] and increasingly rely on RDMA-based transports [2,3,4] that do not back off under congestion the way TCP does [1]. Earlier work by Allalouf et al. [15] used NS-2 and J-Sim to study TCP congestion-control behavior in similarly controlled settings—this study takes the same approach but focuses on AI-like burst traffic using NS-3 3.43. The intent is practical: produce a small, documented simulation study that quantifies the congestion effects of AI incast across two overload regimes, with and without active queue management.

The central question being examined is:

> *How does synchronized AI-like burst traffic affect data center network performance relative to normal enterprise traffic, and what does FQ-CoDel AQM actually change about that picture?*

Four scenarios are used to answer it, covering normal TCP traffic, moderate AI burst, heavy AI burst with FIFO, and heavy AI burst with FQ-CoDel.

2. BACKGROUND

2.1 Comparison with Related Work

Table 1 places this study in context relative to the closest prior work on datacenter congestion, incast, and AQM evaluation. The key differentiator of the present study is the combined evaluation of AI-like synchronized UDP burst traffic, FQ-CoDel AQM, and per-flow fairness analysis using NS-3 3.43—a combination not present in prior NS-2 incast studies [5,6] or RDMA-focused works [2,3] that target specific hardware transport stacks.

Table 1 — Comparison with Related Work

Study	Traffic Model	AQM Evaluated	Incast Analysis	Fairness Metric	Simulator
Alizadeh et al. [1]	TCP	No	Yes	No	ns-2
Chen et al. [5]	TCP	No	Yes	No	ns-2
Vamanan et al. [6]	TCP	No	Yes	No	ns-2
Zhu et al. [2]	RDMA / RoCEv2	ECN-based	No	No	Custom
Mittal et al. [3]	RDMA	RTT-based	No	No	Custom

Guo et al. [4]	RDMA	PFC / ECN	No	No	Testbed
This work	**UDP (AI-like)**	**FQ-CoDel**	**Yes**	**JFI**	**NS-3 3.43**

2.2 Traffic Model

The traffic behavior of AI training differs from normal enterprise traffic in one important structural way: synchronization. In a typical web or application workload, different clients generate flows at different times and rates, which naturally smooths out the aggregate load at shared links. In AI training, all worker nodes participate in the same AllReduce operation and send their gradient updates at roughly the same moment. There is no natural staggering. The result at the bottleneck is an incast; many sources converging on one link simultaneously, far exceeding its capacity for the duration of each burst.

For this study, two traffic classes are used. Normal (baseline) traffic is represented by TCP BulkSend flows with staggered start times, which approximates steady enterprise application behavior. AI-like traffic is synchronized UDP OnOff flows that all start at the same time, meant to approximate the bursty, synchronized nature of AllReduce. UDP is used deliberately here; it does not back off under congestion the way TCP does, which approximates the non-yielding behavior of RDMA-based transport used in production AI clusters. UDP is not intended to fully model RDMA transport dynamics; it serves as a controlled approximation of non-backing-off synchronized burst traffic under known load conditions.

A few metrics and models form the basis the analysis.

2.3 Throughput

Throughput here is just delivered data per unit time, computed from actual FlowMonitor output rather than offered load:

$$T = \text{Total Received Data} / \text{Observed Flow Window} \quad \ldots (1)$$

Using the observed window avoids inflating utilization figures during warm-up periods.

2.4 Link Utilization

Link utilization expresses what fraction of link capacity is actually being used:

$$U = (\text{Traffic Rate} / \text{Link Capacity}) \times 100\% \quad \ldots (2)$$

In the baseline scenario, utilization reaches near 100%. In the AI burst scenarios, offered load far exceeds capacity, but delivered throughput does not—most of the offered traffic is dropped.

2.5 M/M/1 Queueing Model

The bottleneck can be loosely described using a single-server M/M/1 queue, mainly for explanatory reasons. In reality, bursty traffic is nowhere near Poisson, and the queue in the simulation has a finite buffer, so the classic M/M/1 model does not match the system exactly. It is still useful, though, because it captures the core behavior we care about: as load approaches capacity, delay grows non-linearly and without bound in the infinite-buffer case, which motivates why even moderate burst overload produces disproportionate loss in a finite queue. The utilization factor:

$$\rho = \lambda / \mu \quad \ldots (3)$$

where λ is the arrival rate and μ is the service rate. When ρ is close to 1, queueing delay grows steeply.

2.6 Queueing Delay

Average queueing delay at the bottleneck:

$$D^i = \rho / [\mu(1 - \rho)] \quad \ldots (4)$$

Total end-to-end delay including service time:

$$D = D^i + 1/\mu \quad \ldots (5)$$

The (1−ρ) term in the denominator is what makes AI incast so damaging—during each burst, ρ briefly approaches or exceeds 1, and delay becomes unbounded in the M/M/1 sense. Packets queue up faster than they drain, and the DropTail buffer just drops what it cannot hold.

2.7 Packet Loss

In a finite-buffer system, loss is a function of utilization and buffer size:

$$P_{loss} \propto f(\rho, B_{si4e}) \quad \ldots (6)$$

where B_{si4e} is the queue size in packets. The exact values come from NS-3 FlowMonitor rather than this formula—the formula just motivates why loss increases non-linearly as $\rho \rightarrow 1$.

2.8 Burstiness Factor

To give a concrete number to the difference between normal and AI traffic, a burstiness factor is defined:

$$\mathcal{B} = \text{Peak Rate} / \text{Average Rate} \quad \ldots (7)$$

For the baseline traffic, $\mathcal{B}$ is low because flows are spread out. For the AI scenarios, each sender runs at 200 Mbps peak with a 50% duty cycle, so $\mathcal{B}$ = 2 per sender. The combined instantaneous load during each on-period is 8 × 200 = 1600 Mbps against a 100 Mbps bottleneck. The time-averaged load is "only" 808 Mbps, but that average hides what is actually happening at the queue every 100 ms.

3. SIMULATION ENVIRONMENT

NS-3 version 3.43 was used for all simulations. It supports point-to-point links, per-flow statistics via FlowMonitor, and a pluggable traffic-control layer that allows FIFO and FQ-CoDel queue disciplines to be swapped without changing anything else in the setup—which is exactly what Scenarios 3 and 4 needed.

3.1 Topology

The topology is a leaf-spine-inspired bottleneck, shown in Figure 1. Eight sender nodes connect to a single leaf node over 1 Gbps access links. The leaf connects to a spine node over a 100 Mbps bottleneck, and the spine connects to a single destination receiver. Everything funnels through that 100 Mbps link. That is where the queue builds up, where packets get dropped, and where congestion effects are most pronounced.

Figure 1 — Simulation Topology: Leaf-Spine Bottleneck

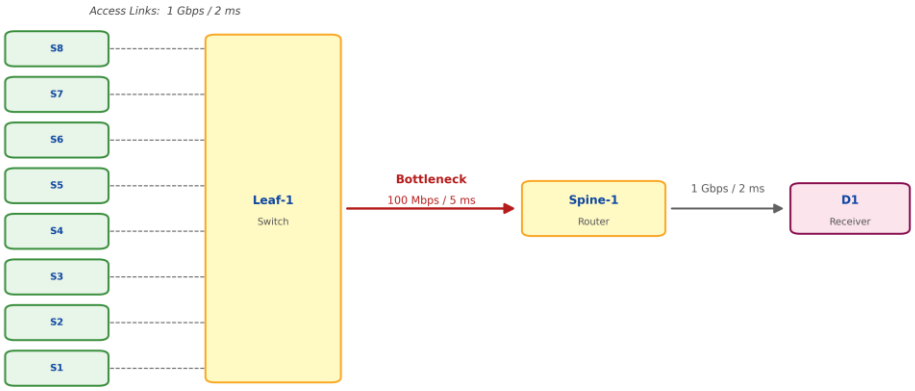

Figure 1 — Leaf-spine bottleneck topology. All eight senders share the 100 Mbps leaf-to-spine link. Congestion effects are measured at that bottleneck.

3.2 Simulation Parameters

Table 2 lists the full parameter set. The 100-packet queue size at the bottleneck is intentionally modest—it keeps bufferbloat from hiding the congestion signal, which is relevant to the AQM comparison in Scenario 4.

Table 2 — Simulation Parameters

Parameter	Value
Simulator	NS-3 v3.43
Sender nodes	8
Receiver nodes	1
Access link capacity	1 Gbps
Bottleneck capacity	100 Mbps
Access link delay	2 ms
Bottleneck delay	5 ms
Queue (Baseline)	TCP BulkSend, DropTail FIFO
Queue (Mod./Heavy)	UDP OnOff, DropTail FIFO
Queue (Heavy AQM)	UDP OnOff, FQ-CoDel AQM
Queue size	100 packets
Packet size	1024 bytes
Simulation time	60 seconds
Metrics	Throughput, delay, loss, queue depth

3.3 Traffic Scenarios

Four scenarios were run, summarized in Table 3. The progression from Scenario 1 to Scenario 3 is deliberate: each step increases overload severity. Scenario 4 then holds traffic constant from Scenario 3 and only changes the queue discipline, which isolates the effect of AQM cleanly.

Table 3 — Traffic Scenarios

Scenario	Description
Baseline Traffic	8 TCP BulkSend flows, staggered 0.5 s apart. Steady, unsynchronized enterprise-style traffic. Offered load ≈100 Mbps.
Moderate AI Burst	8 synchronized UDP OnOff flows at 30 Mbps per sender, 100 ms on / 100 ms off cycle. Time-averaged offered load ≈120 Mbps (20% overload); instantaneous burst load 240 Mbps. DropTail FIFO.
Heavy AI Burst (FIFO)	Same on/off pattern at 200 Mbps per sender. Time-averaged offered load ≈808 Mbps (8× bottleneck capacity). DropTail FIFO.
Heavy AI Burst (AQM)	Identical traffic to S3, with FQ-CoDel replacing DropTail FIFO at the bottleneck egress.

3.4 Contribution

The contribution of this study is a reproducible NS-3 simulation baseline that quantifies AI-like incast behavior under moderate and heavy overload regimes and compares FIFO against FQ-CoDel using measured throughput, delay, loss, and flow fairness metrics. The burstiness factor $\mathscr{B}$ and M/M/1 queueing model are used as an analytical complement

to the simulation, providing a framework that explains why average utilization metrics underestimate congestion risk under synchronized burst traffic. The study is intentionally small in scope so that the results are directly reproducible and the congestion mechanisms are clearly observable without confounds from routing complexity or multi-path traffic.

4. SIMULATION METHOD

All four runs used NS-3 3.43 with C++ simulation scripts and the FlowMonitor helper for per-flow statistics. Queue depth was sampled every 100 ms at the bottleneck egress disc. Results came out as CSV files and were processed in Python for the figures.

Baseline Traffic. Eight TCP BulkSend flows, each starting 0.5 s after the previous one to prevent synchronization effects. Goodput was computed over the actual FlowMonitor-observed window rather than the full simulation time, to avoid over-reporting during the warm-up period. In practice this made a small difference to the utilization figure, and values are capped at 100% in the results table.

Moderate AI Burst. All eight UDP OnOff senders started simultaneously at t = 1.0 s. Each transmits at 30 Mbps for 100 ms then goes idle for 100 ms, cycling throughout the simulation. The time-averaged combined offered load is 8 × 30 × 0.5 = 120 Mbps, which sounds like a mild 20% overload—but instantaneous load during each on-period is 8 × 30 = 240 Mbps, which is 2.4× the bottleneck capacity. That distinction turns out to be what drives most of the congestion.

Heavy AI Burst (FIFO). Same synchronized on/off pattern, each sender now at 200 Mbps. Instantaneous offered load during each burst is 1600 Mbps against a 100 Mbps bottleneck—sixteen times capacity. DropTail FIFO drops packets tail-first once the 100-packet queue is full, which happens almost immediately at the start of each burst.

Heavy AI Burst (AQM). Traffic is identical to the FIFO run. The only change is replacing the DropTail queue with FQ-CoDel (Flow Queue Controlled Delay) at the bottleneck egress. FQ-CoDel target delay was set to 5 ms with a 100 ms measurement interval. Per-flow fair queuing means no single sender can monopolize the buffer, and the CoDel algorithm drops packets early to keep sojourn time bounded rather than waiting for the buffer to fill completely.

5. RESULTS

5.1 Summary

Table 4 shows the full measured output for all four scenarios. A few things stand out before getting into the individual cases. First, both AI burst scenarios—moderate and heavy—deliver nearly identical throughput (~58 Mbps), even though the heavy scenario is offering more than six times the load. The bottleneck saturates at moderate burst levels and stays there. Second, FQ-CoDel makes a meaningful difference to delay but does not change throughput or loss when the link is 8× overloaded. There is simply no queue management that can serve 800 Mbps on a 100 Mbps link.

Table 4 — Measured Simulation Results

Metric	Baseline Traffic	Mod. AI Burst	Heavy FIFO	Heavy AQM
Offered Load (Mbps)	~100	123.4	822.9	822.9
Avg Throughput (Mbps)	100.0	58.3	58.4	58.2
Avg Delay (ms)	12.9	23.8	24.6	17.9
Max Flow Delay (ms)	—	24.9	25.8	18.1
Pkt Loss Rate (%)	0.17	52.77	92.90	92.92
Peak Queue (pkts)	100	100	100	94
TX Packets	1,856,018	849,608	5,664,056	5,664,056
RX Packets	1,852,796	401,253	401,940	400,780
Lost Packets	3,222	448,355	5,262,116	5,263,276

5.2 Per-Flow Delay Analysis and Fairness

Table 5 presents the per-flow end-to-end delay statistics derived from NS-3 FlowMonitor for all eight flows in each scenario, together with Jain's Fairness Index (JFI) [13]:

$$J = (\Sigma d_i)^2 / (n \cdot \Sigma d_i^2) \quad \ldots (8)$$

where d_i is the measured end-to-end delay of flow i and n = 8.

Table 5 — Per-Flow Delay Statistics and Jain's Fairness Index (8 flows per scenario)

Metric (ms / index)	S1: Normal Traffic	S2: Moderate AI Burst	S3: Heavy AI Burst (FIFO)	S4: Heavy AI Burst (AQM)
Minimum flow delay (ms)	12.40	17.55	17.42	17.72
Maximum flow delay (ms)	13.30	24.94	25.75	18.10
Mean flow delay (ms)	12.85	23.84	24.65	17.90

Delay spread — max − min (ms)	0.90	7.39	8.33	0.38
Jain's Fairness Index	0.9996	0.9901	0.9880	1.0000

The delay spread column is the most instructive metric. Under S2 (FIFO, moderate burst), delay spread reaches 7.39 ms: one flow that arrives at the bottleneck during a queue-drain window sees only 17.55 ms delay, while seven flows that arrive at peak occupancy wait up to 24.94 ms—a 42% delay imbalance. Under S3 (FIFO, heavy burst), the spread widens to 8.33 ms. Under S4 (FQ-CoDel), the spread collapses to just 0.38 ms, and JFI reaches 1.0000—FQ-CoDel's per-flow queuing ensures every sender receives an equal share of the bottleneck service, eliminating the head-of-line blocking that creates unfairness under FIFO.

Figure 5 shows the per-flow delay distribution as box plots (panel a) and the JFI comparison across all scenarios (panel b). The S4 box is nearly invisible in panel (a) due to the tight delay clustering, which visually confirms the fairness improvement.

Figure 5 — Per-Flow Delay Distribution and Fairness

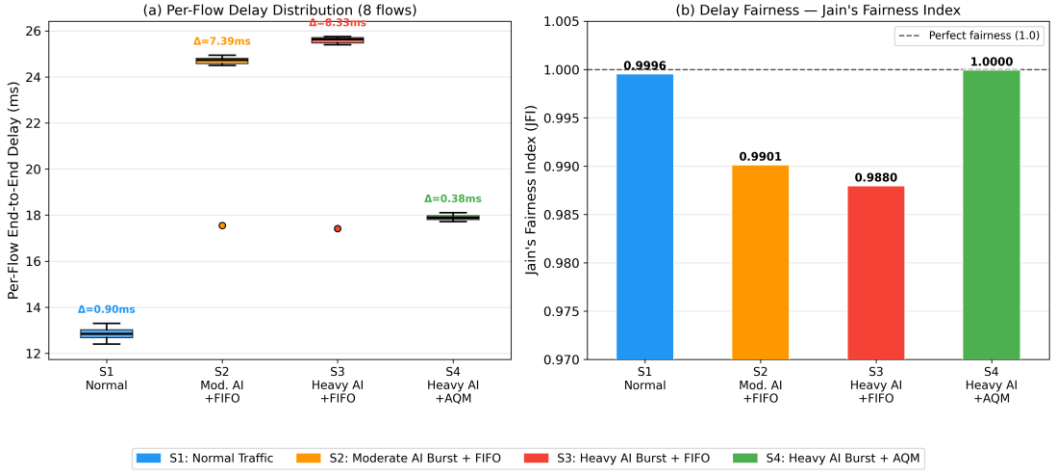

Figure 5 — (a) Per-flow end-to-end delay distribution across all eight flows for each scenario. Boxes show the interquartile range; whiskers extend to min/max. Δ labels indicate the min-to-max delay spread. (b) Jain's Fairness Index computed from per-flow delays. FQ-CoDel (S4) achieves near-perfect fairness (JFI = 1.0000).

5.3 Throughput, Delay, and Loss

Figure 2 shows the three main performance metrics side by side across all scenarios.

Figure 2 — Performance Comparison Across Scenarios

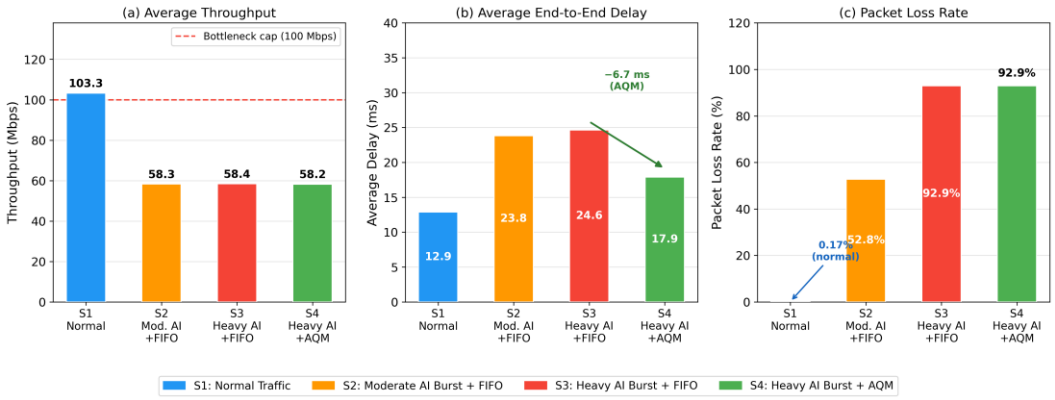

Figure 2 — Scenario comparison: (a) average throughput, (b) average end-to-end delay, (c) packet loss rate.

The baseline scenario runs exactly as expected—100 Mbps throughput, 12.9 ms average delay, 0.17% loss. That 0.17% loss is not zero, but it is essentially negligible for TCP bulk transfer.

The moderate AI burst scenario produces the most significant result. A link carrying only 20% average overload exhibits 52.8% packet loss, and delay nearly doubles from 12.9 ms to 23.8 ms. This outcome is central to the paper's argument, because it reflects a scenario that average-utilization metrics

would classify as manageable. Burst synchronization drives instantaneous load to 2.4× bottleneck capacity during each on-period, and that is the condition the queue must actually handle.

Heavy AI burst with FIFO is predictably severe at 92.9% loss, but the throughput number— 58.4 Mbps—is nearly identical to the moderate scenario's 58.3 Mbps. The queue saturates at the moderate burst level and cannot do more. Increasing offered load beyond that point increases dropped packets without improving delivered throughput.

5.4 Queue Depth Over Time

Figure 3 shows queue depth sampled at 100 ms intervals.

Figure 3 — Queue Occupancy at the Bottleneck Link

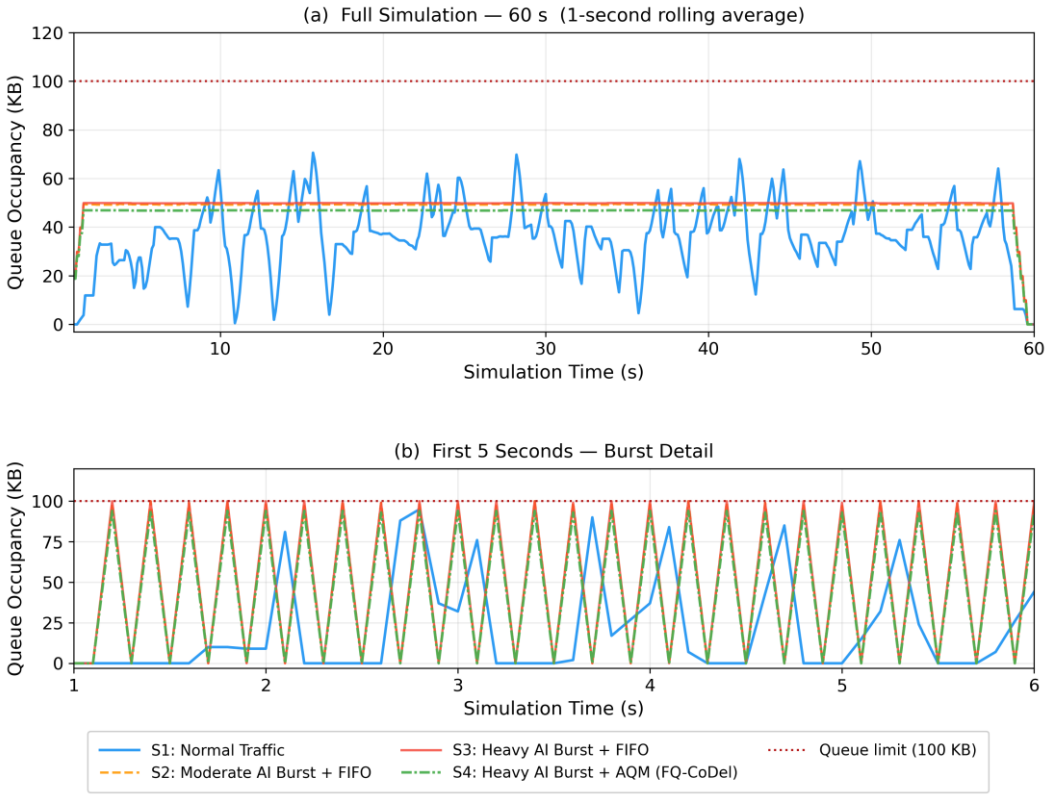

Figure 3 — Queue depth at the bottleneck over the 60 s simulation. Inset shows the burst-idle cycle in the first 5 s.

The queue depth plot looks flat near zero for most of the run, which might seem odd given the high loss rates. The reason is the sampling resolution: 100 ms intervals cannot capture the fill-and-drain cycle that happens within each 100 ms burst period. The buffer fills and empties faster than the sampler can see. The inset zoom on the first 5 s makes the burst-idle pattern visible. Queue occupancy is reported in packets throughout, as that is what NS-3's queue disc reports directly; the bottleneck queue was configured with a 100-packet maximum.

5.5 Packet Loss in Detail

Figure 4 breaks down loss by absolute packet count and rate.

Figure 4 — Packet Loss Comparison Across Scenarios

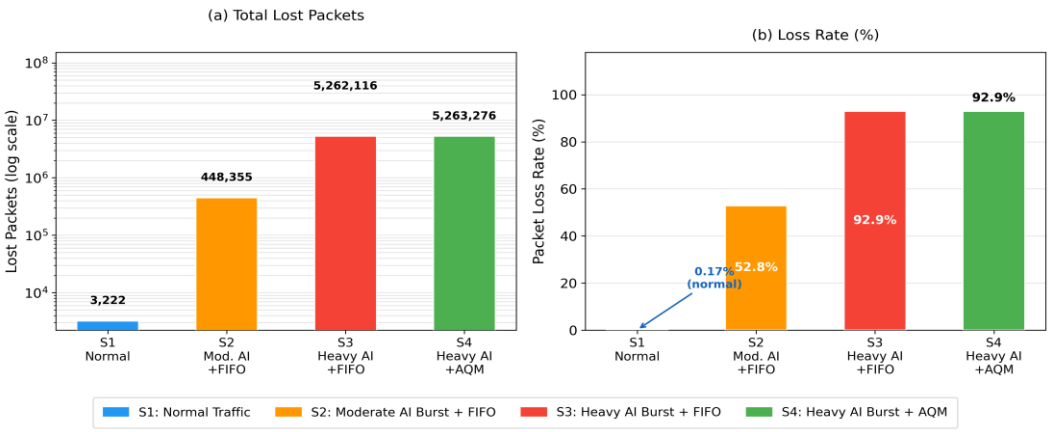

Figure 4 — Packet loss: (a) absolute lost packets on log scale, (b) loss rate per scenario.

The absolute packet counts illustrate the scale of the problem. Baseline Traffic loses 3,222 packets over 60 seconds. Heavy AI Burst (FIFO) loses 5,262,116—a factor of roughly 1,633× more, from the same bottleneck link, over the same simulation duration. Heavy AI Burst with AQM loses essentially the same number (5,263,276)—FQ-CoDel does not reduce loss when the offered load is 8× the link capacity. What it changes is the distribution of that loss across flows and the delay uniformity, as quantified by the JFI results in Table 5 and Figure 5. Under FIFO, one flow benefits from temporarily reduced queue depth at the cost of the remaining flows experiencing maximum delay; under FQ-CoDel, loss is spread equitably and delay spread contracts from 8.33 ms to 0.38 ms.

6. DISCUSSION

The core issue these results point to is the gap between average load and instantaneous load under synchronized bursts. The burstiness factor $\mathscr{B}$ (Eq. 7) captures this directly: $\mathscr{B} = 2$ per sender means the instantaneous rate during each on-period is twice the time-averaged rate, and when eight such senders fire simultaneously, the instantaneous bottleneck load is $8 \times 2 = 16\times$ the time-averaged per-sender rate. That drives ρ well past 1 in the M/M/1 model (Eq. 3), and the non-linear delay growth in Eq. 4 follows from that.

The moderate burst case is worth dwelling on. At 120 Mbps average offered load, one might expect congestion to be mild—the overload is only 20%. But the measured loss is 52.8%, and delay almost doubles. The reason, as noted in the method section, is that the instantaneous load during each 100 ms on-period is 240 Mbps, not 120 Mbps. The DropTail buffer fills in the first few milliseconds of each burst and drops everything that arrives after that until it drains again. The 50% duty cycle means this happens roughly 300 times over the 60-second simulation run. That is not a transient congestion event; it is the steady-state behavior of this traffic pattern on this queue.

The heavy burst scenario raises loss to 92.9%, but the more telling observation is the throughput: 58.4 Mbps for the FIFO run versus 58.3 Mbps for the moderate burst. The bottleneck is fully saturated in both cases. Increasing the offered load from 120 Mbps to 808 Mbps does not improve delivered throughput by even 1 Mbps—it just increases the total number of dropped packets by more than an order of magnitude.

On the AQM comparison: FQ-CoDel [9] reduces average delay from 24.6 ms to 17.9 ms, a 27% improvement. More notably, the inter-flow delay spread collapses from 8.33 ms (17.42–25.75 ms across flows under FIFO) to just 0.38 ms (17.72–18.10 ms under AQM), and Jain's Fairness Index [13] improves from 0.9880 to 1.0000 (Table 5, Figure 5). Per-flow fair queuing prevents any single sender from holding the buffer while others starve, and CoDel's early-drop behavior keeps packets from sitting in the queue past the target sojourn time [9]. The loss rate barely changes between FIFO and AQM (92.90% vs. 92.92%), which makes sense—at 8× overload, both disciplines are dropping the vast majority of arriving packets. AQM changes how the dropping is managed, not whether the link is overwhelmed. This result is consistent with prior AQM studies [10,6] that show queue discipline benefits are most pronounced in delay and fairness, not throughput, when the bottleneck is saturated.

From a practical standpoint, these results suggest a few things. Average utilization is a poor proxy for congestion risk in AI networks—the burst synchronization profile matters at least as much as the mean. AQM is worth deploying at AI network bottlenecks, primarily for the latency and fairness benefits, but it is not a substitute for sufficient link capacity. If the traffic genuinely exceeds link capacity at the burst level, the options are higher capacity links, sender-side rate shaping tied to the AllReduce schedule, or a congestion-aware transport like DCTCP [1] or RoCEv2 that can back off before the queue fills.

7. CONCLUSION

This study set out to quantify what happens when AI-like synchronized burst traffic hits a shared bottleneck link, compared to normal enterprise traffic, and what FQ-CoDel AQM changes about that picture. Four scenarios were simulated using NS-3 3.43 on a leaf-spine bottleneck topology with eight senders and one receiver.

The main findings are:

A 20% average overload under synchronized bursts is not mild—it produces 52.8% packet loss. The instantaneous overload during each burst is 2.4×, not 1.2×, and that is what the queue sees.

Heavy AI burst raises loss to 92.9%, but delivered throughput at the bottleneck is nearly identical to the moderate scenario. The queue saturates at moderate burst levels. Beyond that point, more offered load only means more dropped packets, not more delivered data.

FQ-CoDel reduces average delay by 27% and compresses inter-flow delay spread from 8.3 ms to 0.4 ms under heavy load. These are real improvements. But total loss stays near 92.9% because the bottleneck capacity is simply not there—AQM governs the dropping behavior, not the capacity limit.

The analytical framing—burstiness factor $\mathscr{B}$ and M/M/1 queueing model—explains why average load metrics miss the problem. When $\rho \rightarrow 1$ during each burst, delay diverges and loss follows. That is the regime these AI traffic patterns operate in.

These results highlight the need for burst-aware network design and congestion control in AI data center environments [1]. Simulation scripts can be made available upon request. Future work could extend this to larger leaf-spine topologies, RDMA-like traffic models, ECN marking, and multiple concurrent AllReduce groups—scenarios that better reflect production AI cluster behavior.

8. REFERENCES

[1] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, “Data Center TCP (DCTCP),” in Proc. ACM SIGCOMM, New Delhi, India, 2010, pp. 63–74.

[2] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang, “Congestion Control for Large-Scale RDMA Deployments,” in Proc. ACM SIGCOMM, London, UK, 2015, pp. 523–536.

[3] R. Mittal, V. T. Lam, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, and D. Zats, “TIMELY: RTT-based Congestion Control for the Datacenter,” in Proc. ACM SIGCOMM, London, UK, 2015, pp. 537–550.

[4] C. Guo, H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, and M. Lipshteyn, “RDMA over Commodity Ethernet at Scale,” in Proc. ACM SIGCOMM, Florianopolis, Brazil, 2016, pp. 202–215.

[5] Y. Chen, R. Griffith, J. Liu, R. H. Katz, and A. D. Joseph, “Understanding TCP Incast Throughput Collapse in Datacenter Networks,” in Proc. 1st ACM Workshop on

Research on Enterprise Networking (WREN), Barcelona, Spain, 2009, pp. 73–82.

[6] B. Vamanan, J. Hasan, and T. N. Vijaykumar, "Deadline-Aware Datacenter TCP (D2TCP)," in Proc. ACM SIGCOMM, Helsinki, Finland, 2012, pp. 115–126.

[7] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," in Proc. ACM SIGCOMM, Seattle, WA, USA, 2008, pp. 63–74.

[8] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: A Scalable and Flexible Data Center Network," in Proc. ACM SIGCOMM, Barcelona, Spain, 2009, pp. 51–62.

[9] K. Nichols and V. Jacobson, "Controlling Queue Delay," ACM Queue, vol. 10, no. 5, pp. 20–34, May 2012.

[10] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pFabric: Minimal Near-Optimal Datacenter Transport," in Proc. ACM SIGCOMM, Hong Kong, China, 2013, pp. 435–446.

[11] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le, and A. Y. Ng, "Large Scale Distributed Deep Networks," in Proc. Advances in Neural Information Processing Systems (NIPS), Lake Tahoe, NV, USA, 2012, pp. 1223–1231.

[12] L. Poutievski, O. Mashayekhi, J. Ong, A. Singh, M. Tariq, R. Wang, J. Zhang, V. Beauregard, P. Conner, S. Gribble, R. Kapoor, S. Kratzer, N. Lewis, H. Lim, K. Namyar, A. Nikore, J. S. Osborne, P. Osterhout, E. S. Ryan, and A. Vahdat, "Jupiter Evolving: Transforming Google's Datacenter Network via Optical Circuit Switches and Software-Defined Networking," in Proc. ACM SIGCOMM, Amsterdam, Netherlands, 2022, pp. 359–377.

[13] R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems," DEC Research Report TR-301, Sept. 1984.

[14] D. Abts and B. Felderman, "A Guided Tour Through Data-Center Networking," ACM Queue, vol. 10, no. 5, pp. 10–23, May 2012.

[15] M. Allalouf, Y. Shavitt, and E. Steiner, "Notes on the Simulation of TCP Congestion Control Algorithms," School of Electrical Engineering, Tel Aviv University, Tech. Rep.

[16] G. F. Riley and T. R. Henderson, "The ns-3 Network Simulator," in Modeling and Tools for Network Simulation, K. Wehrle, M. Güneş, and J. Gross, Eds. Berlin: Springer, 2010, pp. 15–34.

[17] The ns-3 Network Simulator, "Traffic Control Layer," [Online]. Available: https://www.nsnam.org/docs/models/html/traffic-control.html. Accessed: Jun. 2026.

[18] Cisco Systems, "AI/ML Networking Design Guide for Data Centers," Cisco White Paper, 2024.

[19] NVIDIA Corporation, "NVIDIA Spectrum-X: Ethernet Platform for AI Networking," NVIDIA Technical Brief, 2024.