

EVALUATING REAL-TIME BIDDING ARCHITECTURE FOR CONNECTED-TV ADVERTISING: LATENCY, OPERATIONAL RESILIENCE, AND REVENUE PERFORMANCE

Sujan Kumar Reddy Challa¹

Received 24.07.2026.

Revised 05.09.2026.

Accepted 17.10.2026.

Keywords:

Real-time bidding, connected television, OpenRTB, auction simulation, floor-price optimization, zero-downtime migration, advertising technology management.

Original research



ABSTRACT

Connected television (CTV) has become a premium programmatic advertising channel whose real-time bidding (RTB) infrastructure must satisfy a strict exchange-defined decisioning deadline, high-CPM revenue yield, and OpenRTB 2.6 protocol compliance at very large request volumes. Prior research quantifies floor-price effects in display and search advertising and documents failure modes of the implementation technologies used in RTB engines, but it does not quantitatively evaluate how the architectural choices inside a CTV RTB engine jointly determine deadline compliance, service reliability, and revenue. This article addresses that gap. It synthesizes a design framework for CTV RTB engines — a latency-budgeted OpenRTB 2.6 pipeline, concurrent demand-side platform (DSP) fan-out, zero-downtime migration by traffic mirroring and health-check gating, and floor-price enforcement — and evaluates it with a reproducible discrete-event simulation (stated parameter set, fixed random seed) that compares a legacy configuration (JSON serialization, sequential fan-out, no floor enforcement, downtime-restart deployment) with the proposed configuration. In the simulation, the proposed configuration reduced p99 end-to-end latency from 194.3 ms to 64.1 ms against a 100 ms deadline, reduced the deadline-miss rate from 94.7% to 0.0%, sustained 1,400 auctions per second at a 1% deadline-miss threshold, held availability at 99.72% through a migration event (against 78.91% for a downtime restart), and increased revenue per 1,000 auction opportunities from \$24.12 to \$42.31. The article distinguishes these simulation findings from results reported in the cited literature, and draws out the managerial implications and limitations of the framework.

© 2027 Journal of Innovations in Business and Industry

1. INTRODUCTION

The programmatic advertising ecosystem has undergone a structural shift over the past decade, moving from desktop-centric banner auctions to a multi-surface landscape encompassing mobile in-app, digital audio, digital out-of-home, and, most significantly, connected television. CTV, meaning internet-connected television devices that deliver streaming video content through app-

based platforms, has emerged as a high-value and fast-growing inventory category in programmatic supply (John & Scheer, 2021). The IAB Europe Guide to Programmatic CTV reports that CTV advertising spend in Europe grew by more than 56% in 2021, a rate that outpaced broader digital advertising expansion, that 51% of surveyed global marketers were planning to increase CTV advertising budgets, and that 71% of European viewers reported a preference for streaming over

traditional broadcast television (IAB Europe, 2023). A majority of surveyed industry experts, 88%, agreed that CTV would accelerate the broader shift in ad spend away from linear television, and 66% of advertisers confirmed they were actively redirecting budgets toward CTV and OTT inventory (IAB Europe, 2023). Table 1 summarizes these market indicators, all of which are reported by IAB Europe (2023) rather than measured in this study.

Table 1. CTV Advertising Market Growth Indicators

Metric	Value	Unit / Context	Year
CTV Ad Spend Growth in Europe	56	% YoY increase	2021
Global Marketers Increasing CTV Budget	51	% of global marketers surveyed	2023
EU Viewers Preferring Streaming over TV	71	% of European viewers	2023
Advertisers Shifting Budget to CTV/OTT	66	% of advertisers	2023
Ad Experts: CTV Accelerates Ad Shift	88	% of industry experts	2023

Note. Adapted from IAB Europe (2023).

At the transactional level, the programmatic infrastructure required to service this demand operates at a very large scale. Yuan et al. (2013) documented this scale directly: their production dataset, drawn from a major UK-based ad exchange over several months, comprised more than 52 million impressions, and their supply-side sample separately recorded nearly 13 million auctions across 50 placements. Every auction in that environment, as in RTB generally, is bound by the OpenRTB tmax field, meaning the exchange rejects a DSP's bid if it arrives after the exchange-specified deadline (IAB Tech Lab, 2022). CTV amplifies these baseline requirements in operationally significant ways. Unlike desktop display, where a missed impression is invisible to the user, an unfilled CTV pre-roll slot produces a perceptible blank screen or content interruption that degrades the viewing experience. CTV CPMs are substantially higher than desktop display CPMs, which makes floor-price miscalibration more costly per auction, and CTV inventory typically carries tighter auction timeout windows because video buffering schedules are time-constrained. Engineers designing RTB engines for CTV platforms must therefore satisfy a dual mandate: strict latency compliance within the OpenRTB auction timeout budget, and maximized revenue yield per auction through precise floor-price enforcement and DSP coverage.

Despite this growth, the published literature provides limited reproducible, quantitative evidence on how the architectural choices inside a CTV RTB engine jointly determine deadline compliance, service reliability, and revenue outcomes. Auction-theoretic and field-experimental studies quantify floor-price effects in display and search advertising (Choi & Mela, 2023; Ostrovsky & Schwarz, 2023) and in production display and video systems (Alcobendas et al., 2024), but they take the serving architecture as given. Systems research documents the failure modes of the implementation

technologies used in RTB engines — for example, the concurrency defects catalogued by Tu et al. (2019) — without connecting them to auction-level business outcomes. Practitioner guidance, in turn, is largely qualitative: per-stage latency budgets, fan-out designs, and migration procedures are asserted rather than measured, and are seldom reported with the parameters needed to reproduce them. There is therefore a research gap at the intersection of systems engineering and revenue management: platform teams cannot determine from published evidence what a given architectural configuration is expected to deliver on the three outcomes their businesses are managed on — latency against the exchange deadline, reliability of the auction service, and revenue per auction opportunity (Sharghivand et al., 2021).

This article addresses that gap with three contributions. First, it synthesizes a design framework for CTV RTB engines covering four domains: a latency-budgeted OpenRTB 2.6 auction pipeline, concurrent DSP fan-out with explicit goroutine lifecycle discipline, zero-downtime migration by traffic mirroring and health-check gating, and floor-price enforcement (Sections 3 and 4). Second — the article's principal original contribution — it specifies and reports a reproducible discrete-event simulation of the CTV RTB auction pipeline, with every parameter stated, that compares a legacy configuration (JSON serialization, sequential fan-out, no floor enforcement, downtime-restart deployment) against the proposed configuration (Protocol Buffers serialization, concurrent fan-out with pre-fan-out floor gating, floor enforcement, and zero-downtime migration) on latency percentiles against a 100 ms decisioning deadline, sustained throughput, deadline-miss and no-bid rates, availability during a migration event, fill rate, and revenue per 1,000 auction opportunities (Section 5). Third, it separates the evidence base explicitly: results imported from the cited literature are attributed in text and tabulated against this study's own simulation findings (Section 6), so that no literature number is presented as a finding of this article.

The remainder of the article is organized as follows. Section 2 reviews background and related work. Sections 3 and 4 present the architecture and the floor-price framework. Section 5 describes the evaluation methodology and reports the simulation results. Section 6 distinguishes literature findings from this study's findings. Section 7 addresses implementation challenges and operational considerations, Section 8 draws managerial implications, Section 9 states limitations, and Sections 10 and 11 discuss and conclude.

2. BACKGROUND AND RELATED WORK

The technical and economic architecture of real-time bidding has been studied since its commercial emergence in the early 2010s. Yuan et al. (2013) conducted one of the earliest large-scale empirical investigations of RTB auction dynamics, analyzing a production dataset

comprising more than 52 million impressions collected from a major advertising exchange over a three-month period. Their analysis found that about 40% of impressions were transacted through first-price auction mechanisms, largely because publisher soft floor settings converted nominally second-price auctions into first-price ones, and that these first-price transactions accounted for 55.4% of total advertiser expenditure, a finding that underscores the revenue significance of auction type selection and floor-price calibration (Irfan, 2024).

The theoretical foundation for auction design in programmatic contexts traces to Edelman et al. (2007), who formalized the generalized second-price (GSP) auction as a canonical pricing mechanism for internet advertising. GSP auctions achieve efficient allocation by charging winners the minimum price necessary to defeat the next-highest bidder, rather than their own maximum willingness to pay. The industry-wide shift from second-price to first-price auction mechanisms between 2018 and 2020 fundamentally changed incentives on both sides of the auction. Under a first-price mechanism, truthful bidding is no longer a dominant strategy: DSPs have an incentive to shade their bids below their true valuations to avoid overpaying, while sellers must set floors deliberately to recapture the surplus that shading would otherwise transfer to buyers.

The revenue impact of reserve price optimization in display advertising has been demonstrated empirically by independent research groups. Choi and Mela (2023), conducting field experiments in collaboration with a major premium publisher, found that setting reserve prices relative to a zero-reserve baseline increases publisher revenues by an average of 35%. Their structural model further showed that incorporating minimum impression constraints, meaning the tendency of advertisers operating under reach goals to raise bids rather than forfeit an impression, into the reserve price optimization framework yields a 44% increase in revenue over the zero-reserve baseline, an additional 9 percentage points attributable to the more complete demand model. At Yahoo, Alcobendas et al. (2024) reported that a production floor optimization model for first-price auctions, deployed on Yahoo's display inventory and extended to video inventory, produced annualized incremental revenue of +1.3% on display ad inventory and +2.5% on video ad inventory relative to the prior manual approach. The floor model set placement-level reserve prices by bidder type, distinguishing between regular DSPs and rebroadcasters, using a statistical distribution fitted to observed bid behavior under randomized floor experiments, and during the evaluation quarter it produced a daily revenue lift that was always positive and averaged 1.27% across major Yahoo properties.

The protocol foundation for RTB auctions across inventory formats, including CTV, is provided by the IAB Tech Lab's OpenRTB API Specification, currently at version 2.6 (IAB Tech Lab, 2022). Zhang et al. (2014) extended early RTB empirical research to develop

theoretically grounded optimal bidding strategies that account for budget constraints and impression-level heterogeneity, establishing key properties of the RTB optimization problem. Perlich et al. (2012) demonstrated the effectiveness of machine learning approaches for predicting bid value and user propensity in targeted display advertising, laying groundwork for the machine-learning-driven bidding systems now standard among major DSPs.

3. CTV RTB ENGINE ARCHITECTURE

3.1 OpenRTB 2.6 Auction Pipeline and Latency Budget

The OpenRTB 2.6 specification defines the wire format, field semantics, and behavioral requirements for RTB auction exchanges, including CTV-specific extensions for video inventory, content metadata, and viewability signals (IAB Tech Lab, 2022). A compliant CTV RTB engine receives a BidRequest from the supply-side platform (SSP), processes it through a sequence of pipeline stages, and returns a BidResponse within the timeout specified in the request's tmax field.

The total auction latency budget must be decomposed across pipeline stages at the architecture level rather than treated as an aggregate target to be monitored after deployment. Production RTB systems typically allocate the budget across the following stages: inventory signal detection and request formation, bid request formation and DSP fan-out initiation, multi-party DSP data evaluation and bid generation, winner selection and clearing price determination, and ad serving and tracking initialization (Table 2). For CTV pre-roll inventory, tmax values are typically tighter than for desktop display, because the video rendering pipeline cannot buffer indefinitely while waiting for auction completion.

Table 2. RTB Auction Pipeline Stages

Pipeline Stage	Function
Inventory signal detection	Request arrives from SSP; initial parsing begins
Bid request formation	OpenRTB 2.6 parsing and enrichment before fan-out
DSP fan-out distribution	Concurrent goroutine dispatch to eligible DSPs
DSP data evaluation	Bidding logic and inference performed at each DSP
Bid response generation	DSP constructs and returns its bid
Winner selection	Auction clearing and floor enforcement
Ad serving and tracking	Creative delivery and pixel firing

Note. Stage sequence shown for architecture-time planning; the specific time allocation for each stage is exchange-specific and must be measured against each deployment's own instrumentation rather than assumed from a fixed table, since no verified public source reports a general per-stage millisecond breakdown.

Serialization at the bid request ingress stage has a direct effect on parsing latency and on latency variability at high percentiles. Compared with dynamically typed JSON parsing, a binary, fixed-schema serialization format such as Protocol Buffers reduces per-request object allocation, which in turn reduces the garbage-collection pressure that a high-throughput Go service experiences under load (Go Team at Google, 2023). This choice has direct implications for p99 latency stability, which is the operationally relevant latency metric in RTB: the 99th-percentile bid response time determines the maximum DSP timeout window the engine can offer while still clearing the large majority of auctions within tmax.

Floor-price gating should be performed before DSP fan-out, not after. At the pre-fan-out gate, the engine retrieves each eligible DSP's historical bid distribution statistics from an in-memory cache and compares them against the floor specified in the incoming BidRequest. DSPs whose historical bid distributions suggest a low probability of submitting an above-floor bid are excluded from the fan-out set for that request. This gating step reduces outbound request volume without materially reducing competitive intensity among the remaining bidders, and it preserves latency budget that would otherwise be spent waiting on DSPs unlikely to clear the floor. Pipelining the floor-gating computation so that it overlaps with fan-out preparation, rather than running the two steps strictly in sequence, is a standard technique in low-latency service design for shrinking the critical path without changing what work is done (Go Team at Google, 2023).

A production ad exchange operating at CTV scale must handle a very high volume of auctions per second during peak traffic periods. At this volume, even small per-request improvements in processing time compound into meaningful aggregate throughput gains, while small regressions compound into capacity problems. The RTB engine should therefore treat latency instrumentation, specifically p50, p95, p99, and p999 percentiles broken down by pipeline stage, as a continuous operational obligation rather than a periodic benchmarking exercise.

3.2 Go-Based Concurrent DSP Fan-Out

The DSP fan-out layer, meaning the part of the engine that spawns parallel bid requests to all qualifying DSPs and collects responses within the auction timeout, is the most latency-sensitive and concurrency-intensive part of the engine. The Go language offers a lightweight implementation of this pattern using its goroutine and channel constructs, in which each DSP call runs in its own goroutine and writes its result to a shared, buffered channel.

The correctness of this pattern depends critically on goroutine lifecycle management. Tu et al. (2019) conducted a systematic study of 171 concurrency bugs across six major open-source Go projects, including Docker, Kubernetes, etcd, gRPC, CockroachDB, and BoltDB, and found that a majority of blocking concurrency bugs were caused by misuse of Go's

message-passing primitives, meaning goroutines and channels, rather than by traditional shared-memory synchronization failures. This finding is notable because the message-passing model that Go promotes as a primary safety advantage is, in Tu et al.'s sample of mature production codebases, the dominant source of blocking bugs rather than a guarantee against them.

For CTV RTB fan-out, the operationally significant failure mode is the goroutine leak: a DSP goroutine that blocks indefinitely on a DSP that is slow or unresponsive, consuming a goroutine and its stack allocation without ever writing a result to the collector channel. Under normal operating conditions, goroutine leaks are often masked by Go's runtime scheduler. Under load, which is precisely the condition during CTV primetime inventory peaks, accumulated goroutine leaks manifest as progressive memory growth, increased garbage-collection pressure, and widening p99 latency, and these symptoms are frequently misdiagnosed as hardware capacity issues rather than as a concurrency defect.

A production-ready fan-out implementation needs several components: a per-DSP goroutine governed by a context with an explicit deadline, so that a slow DSP can be cancelled rather than awaited indefinitely; a buffered collector channel sized to the maximum fan-out cardinality, so that goroutines are not blocked from writing if the main goroutine stops reading before all goroutines finish; and continuous, metric-based monitoring of goroutine counts to detect leaks before they accumulate into a capacity incident. Every goroutine whose parent context has expired must terminate without attempting a write to the channel. The main goroutine reads from the channel until either all expected replies arrive or the context deadline expires, and it then proceeds with the auction using whichever bids were successfully collected.

Allocation behavior is a related and often underweighted factor. In an RTB engine handling a large volume of bids per second, per-request allocations in the form of BidRequest structs, response buffers, slice headers, and map literals add load to the garbage collector, and garbage-collection pauses compete with request-processing goroutines for CPU time, widening the p99 latency distribution. A standard mitigation is consistent use of an object pool, such as Go's sync.Pool, for high-frequency allocation types like bidRequest, bidResponse, and auctionContext objects, so that these objects are reused across requests instead of being repeatedly allocated and garbage-collected. The correctness of the overall fan-out pattern rests on the memory model that governs channel communication and happens-before relationships in Go, which is formalized in the language specification (Go Team at Google, 2023).

3.3 Zero-Downtime Migration via Traffic Mirroring and Health-Check Gating

Migrating a production CTV RTB engine, whether the change is new auction logic, a serialization-layer upgrade, or a change to DSP protocol handling, carries

revenue risk that makes conventional maintenance windows unacceptable. CTV platforms deliver continuous streaming inventory with no natural pause, and even a brief auction outage during high-CPM primetime inventory represents direct revenue loss and a potential violation of publisher service-level agreements. A zero-downtime migration framework addresses this risk through two sequenced phases. The first phase is traffic mirroring: production bid requests are dispatched simultaneously to both the legacy bidder and the candidate new bidder. The legacy bidder's responses are used to serve real auctions and generate revenue, while the new bidder's responses are captured, recorded, and compared against the legacy bidder's output but never used in a live auction decision. This shadow mode lets engineers validate the new engine's BidResponse schema compliance, floor-enforcement correctness, concurrency safety under production load, and p99 latency against the auction timeout, all without any exposure on the revenue path.

The transition from shadow mode to production traffic is governed by health-check gates evaluated as continuous automated assertions rather than as manual engineer sign-offs. A health-check gate specifies threshold conditions across several signal classes: p99 bid-processing latency must remain below the tmax budget, BidResponse schema validity must exceed a defined threshold, floor-enforcement accuracy must match the legacy bidder within a stated tolerance, and the response-rate differential between shadow and legacy traffic must remain within acceptable bounds. The gate advances only when all conditions are satisfied simultaneously and hold across a defined observation window, which protects against advancing on a favorable but statistically noisy snapshot.

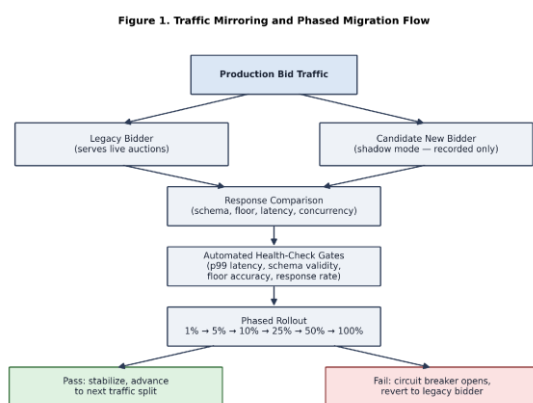


Figure 1. Traffic Mirroring and Phased Migration Flow

Phased rollout then proceeds in discrete traffic-split increments, for example 1%, 5%, 10%, 25%, 50%, and 100% as one illustrative sequence supported by tools such as Argo Rollouts, with stabilization periods between steps that allow latency distributions and error rates to normalize before the next increment. A circuit-breaker mechanism, in the sense described by Nygard (2007), should be embedded at each phase boundary as a hard revert trigger: if the new engine's error rate or p99

latency exceeds configurable thresholds during any phase, the circuit opens, all traffic reverts to the legacy bidder, and an operational alert fires. This prevents a degraded new engine from propagating failures across an increasing share of traffic before engineers can intervene. Cloud-native tooling such as Argo Rollouts provides declarative management of phased rollout within Kubernetes-based deployment environments (CNCF, 2023). Kubernetes readiness and liveness probes form the operational foundation of health-check gating: readiness probes prevent a pod from receiving production traffic until it passes its health checks, while liveness probes trigger an automatic instance restart for a pod that becomes degraded (Cox, 2022). Figure 1 summarizes the traffic-mirroring and phased-migration flow described above.

4. REVENUE OPTIMIZATION VIA FLOOR-PRICE ENGINEERING

4.1 Auction-Theoretic Foundations of Reserve Pricing

In the auction theory literature, reserve prices function as the seller's mechanism for extracting additional surplus from high-valuation buyers and for filtering out low-value bids that would otherwise clear at prices below the seller's opportunity cost. CTV RTB Bidding Engine Architecture is presented on Figure 2.

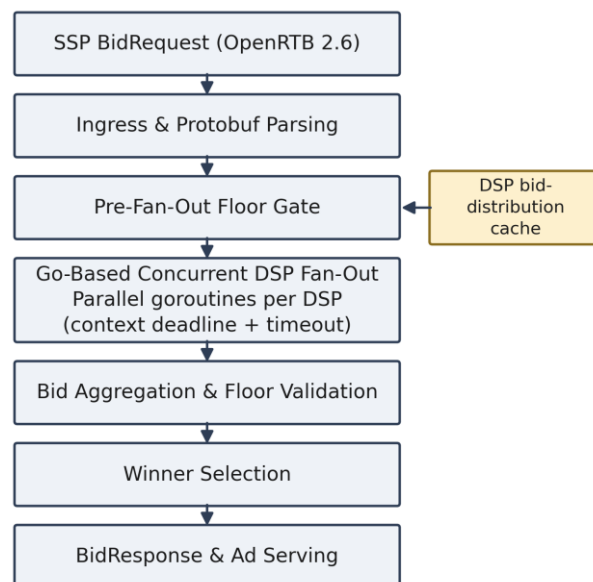


Figure 2. CTV RTB Bidding Engine Architecture

The generalized second-price auction formalized by Edelman et al. (2007) includes reserve price provisions that guarantee the seller never transacts below a minimum acceptable yield. In RTB, the equivalent mechanism is the bid floor, transmitted in the BidRequest's bidfloor field and enforced at both the SSP and the RTB engine.

The impact of reserve prices on buyer behavior differs between second-price and first-price auction formats, a distinction of practical importance given the industry's transition to first-price mechanisms since 2018. In a second-price auction, a floor increase that remains below the second-highest bid does not change the winner's payment, but it raises revenue when the second-highest bid falls below the floor. In a first-price auction, DSPs respond to floor increases by raising their own bids toward their true valuations in order to remain competitive, a behavioral shift that, when modeled accurately, allows the seller to set floors that capture additional surplus without proportionately suppressing fill rates (Alcobendas et al., 2024). Feng et al. (2021) demonstrated that reserve prices optimized for second-price auction environments are systematically suboptimal when applied to first-price auctions, which motivates the development of first-price-specific floor optimization models.

4.2 Empirical Evidence for Floor-Price Revenue Impact

The results reviewed in this section are findings reported in the cited literature, obtained in display, video, and search advertising environments; they are not measurements of the CTV architecture presented in this article. They are included because they establish the direction and plausible magnitude of the floor-price revenue lever that Section 4.3 implements architecturally and Section 5 evaluates by simulation. Section 6 tabulates the provenance of every quantitative claim used in this article.

The empirical evidence for floor-price optimization as a revenue lever in display advertising is substantial and directionally consistent across independent research programs. Choi and Mela (2023), conducting field experiments at a major premium publisher, found that setting reserve prices increases publisher revenues by 35% compared with a zero-reserve baseline. Their structural model incorporated minimum impression constraints, meaning the tendency of advertisers operating under reach goals to raise bids rather than forgo an impression, and showed that accounting for these constraints in the reserve price optimization yields an additional 9 percentage points, for a total revenue increase of 44% over the zero-reserve baseline.

At Yahoo, Alcobendas et al. (2024) reported annualized incremental revenue of +1.3% on display ad inventory and +2.5% on video ad inventory, and a daily revenue lift that averaged 1.27% across several Yahoo properties during the evaluation quarter. The mechanism operated through two complementary channels: raising floors shifted DSP bid distributions upward, increasing clearing prices for won impressions, while also routing unsold impressions to an outside-option exchange, which improved the effective monetization of inventory the main exchange could not clear at the higher floor.

Yuan et al.'s (2013) empirical analysis of a production RTB exchange provides additional context for the

revenue significance of first-price auction dynamics: impressions transacted in first-price formats accounted for approximately 40% of total impressions but 55.4% of total advertiser expenditure in the exchange they studied. This disproportionate cost concentration in first-price inventory means that floor calibration errors in first-price auction environments carry amplified revenue consequences relative to their impression-volume share. Ostrovsky and Schwarz (2023), in a large-scale field experiment in the context of sponsored search advertising, confirmed that setting reserve prices guided by theoretical models leads to a substantial increase in seller revenue. Their findings in the search context are directionally consistent with the display advertising results of Choi and Mela (2023), which together establish reserve price optimization as a robust revenue lever across multiple programmatic inventory formats.

4.3 Floor-Price Enforcement in CTV RTB Engines

For CTV platforms, floor-price enforcement carries greater per-auction revenue stakes than in desktop display, because CTV CPMs are substantially higher and the inventory is time-constrained. The RTB engine enforces floors at two distinct pipeline stages.

At bid request ingress, before DSP fan-out, the engine evaluates each eligible DSP against the floor specified in the incoming BidRequest using cached per-DSP bid distribution statistics. DSPs whose historical bid distributions suggest a low probability of clearing the floor are excluded from the fan-out set. This gating step reduces outbound request volume without reducing competitive intensity among the remaining bidders, and it preserves latency budget for DSP response collection.

At bid aggregation, after all DSP responses are collected, every bid is validated against the floor before admission to the auction. Below-floor bids, whether submitted because of a programmatic error or because of floor calculation lag in the DSP's bidding system, are discarded before auction execution. This ensures that the clearing price of every settled auction meets the minimum yield threshold.

At the architectural level, the same general levers recur across production CTV environments: locating decision logic close to the point of ingress to reduce network round-trips, parallelizing independent stages of the pipeline rather than executing them strictly in sequence, and caching data that is read frequently but written infrequently, such as per-DSP historical bid statistics and campaign metadata. These are well-established techniques in low-latency distributed systems generally, and their application to RTB specifically follows the same reasoning that motivates them elsewhere: the goal is to remove work from the synchronous critical path of the auction wherever the correctness of the auction does not require that work to happen synchronously. Teams should validate the magnitude of any latency or throughput improvement from these techniques against their own production telemetry rather than importing point estimates from unrelated deployments, since the

size of the effect depends heavily on request-mix, cache hit-rate, and network topology specific to each exchange.

5. EVALUATION METHODOLOGY AND SIMULATION RESULTS

5.1 Methodology and Parameters

The evaluation uses a discrete-event simulation of the auction pipeline described in Sections 3 and 4, implemented as a Monte Carlo model of per-auction processing combined with a queueing simulation for load and deployment-event experiments. Simulation — one of the two validation routes available for a framework at this stage — is used because it permits controlled, reproducible comparison of architecture configurations under identical demand, something a single production deployment cannot provide. All parameters are stated in Table 3, the random seed is fixed, and the simulation code and full result set accompany this article's revision package, so every number in this section can be regenerated exactly.

Table 3. Simulation Parameters

Parameter	Value
Decisioning deadline (tmax budget), end-to-end	100 ms
Eligible DSPs per auction (before gating)	12
DSPs called after floor gating (Configuration B)	9 (3 gated out)
DSP response time	Lognormal, median 45 ms, sigma = 0.45
DSP hang probability / error probability	2% / 1%
DSP valuations	Lognormal, median \$22 CPM, per-DSP heterogeneity
Bid floor	\$25 CPM
First-price bid shading	A: U(0.78, 0.95); B: U(0.82, 0.97)
Response-collection window (Configuration B)	60 ms
Serialization	A: JSON; B: Protocol Buffers
Auctions per configuration (Experiment i)	200,000
Load test (Experiment ii)	Poisson arrivals; 96 parallel auction slots
Deployment event (Experiment iii)	600 auctions/s over 120 s
Random seed	20261015

Two configurations are compared. Configuration A (legacy baseline) uses JSON serialization at ingress and egress, sequential DSP fan-out (DSPs are contacted one at a time while time remains in the dispatch window), no floor-price enforcement (below-floor bids are admitted), and downtime-restart deployment. Configuration B (proposed) implements the architecture of Sections 3 and 4: Protocol Buffers serialization, concurrent fan-out with a bounded 60 ms response-collection window, pre-fan-

out floor gating driven by per-DSP bid-distribution profiles (DSPs whose estimated probability of clearing the floor falls below 15% are excluded), floor enforcement at bid aggregation, and zero-downtime migration by traffic mirroring with health-check gating. Demand is synthetic and identical across configurations. Each auction offers 12 eligible DSPs. DSP response times are lognormal (median 45 ms, sigma 0.45), with a 2% probability that a call never responds (hang) and a 1% probability of an immediate error. DSP valuations are lognormal around a median of \$22 CPM with per-DSP heterogeneity, and bidding is first-price with shading drawn uniformly from the ranges in Table 3; in Configuration B, a DSP whose valuation clears the floor bids at least the floor, reflecting the bid-raising behavior documented by Choi and Mela (2023). The floor is \$25 CPM, representative of premium CTV pre-roll. The decision deadline is 100 ms end-to-end at the engine. Three experiments are run: (i) 200,000 auctions per configuration measuring latency percentiles, deadline-miss rate, no-bid rate, participating DSPs, fill rate, clearing CPM, and revenue per 1,000 auction opportunities; (ii) a load test with Poisson arrivals and 96 parallel auction slots, reporting the highest offered load whose deadline-miss rate stays at or below 1%; and (iii) a 120-second deployment event at 600 auctions per second comparing downtime-restart deployment (20 s outage plus 10 s warm-up at 50% failure) against zero-downtime migration (30 s migration window at a 0.5% error rate), measuring availability and retained revenue.

5.2 Latency and Reliability Results

Table 4 reports the per-auction results. The legacy configuration's latency failure is structural rather than incidental: with a median DSP response time of 45 ms and sequential dispatch, the dispatch window is exhausted after an average of only 2.68 of 12 DSPs have been heard from, producing a p50 end-to-end latency of 119.2 ms and a deadline-miss rate of 94.7%. The proposed configuration's latency is bounded by its collection window: p50 62.9 ms, p95 63.6 ms, and p99 64.1 ms, with a deadline-miss rate of 0.0% and a no-bid rate below 0.01%.

Table 4. Latency and Reliability Results (200,000 auctions per configuration)

Metric	Configuration A (legacy)	Configuration B (proposed)
p50 end-to-end latency (ms)	119.2	62.9
p95 end-to-end latency (ms)	163.9	63.6
p99 end-to-end latency (ms)	194.3	64.1
Deadline-miss rate (%)	94.7	0.0
No-bid rate (%)	2.0	< 0.01
Avg. participating DSPs per auction	2.68	6.53

Floor gating removed 3 of 12 DSPs whose estimated clear probabilities (0.9%–10.3%) fell below the 15% gate threshold, and an average of 6.53 DSPs contributed bids per auction — more than twice the legacy configuration's participation, achieved while calling fewer DSPs in total.

5.3 Throughput Under Load

In the load test (Table 5), the legacy configuration meets the 1% deadline-miss criterion at no offered load: its miss rate is already 96.7% at 100 auctions per second, because the constraint is per-auction latency, not capacity. The proposed configuration sustains 1,400 auctions per second at or below the 1% miss threshold with 96 parallel auction slots, and degrades sharply beyond that point (miss rate 23.1% at 1,500 auctions per second), consistent with a capacity ceiling near slots divided by mean service time.

Table 5. Deadline-Miss Rate (%) by Offered Load

Offered load (auctions/s)	Configuration A	Configuration B
100	96.7	0.0
500	95.5	0.0
1,000	100.0	0.0
1,400	—	0.0
1,500	—	23.1

Note. Sustained throughput is the highest offered load with a deadline-miss rate at or below 1%: Configuration A, none; Configuration B, 1,400 auctions/s.

For capacity planning, the relevant unit is therefore the deadline-compliant throughput per slot pool, not the raw request rate a load balancer can accept.

5.4 Migration and Deployment Reliability

Table 6 reports the deployment-event experiment, which holds the proposed engine constant and varies only the deployment strategy. A downtime restart kept availability at 78.91% across the 120-second window and 16.44% inside the event window itself, failing 15,242 of 72,270 auctions. Zero-downtime migration by traffic mirroring and health-check gating held availability at 99.72% overall (99.51% inside the window) with 200 failed auctions, retaining \$3,049.63 of revenue against \$2,413.13 for the restart — a difference of \$636.50 per event at this modest 600 auctions-per-second load.

Table 6. Deployment-Event Results (120 s at 600 auctions/s)

Metric	Downtime restart	Zero-downtime migration
Availability, full window (%)	78.91	99.72
Availability, event window (%)	16.44	99.51
Failed auctions	15,242	200
Revenue retained (USD)	2,413.13	3,049.63

At primetime CTV volumes the absolute revenue exposure of a restart scales linearly with load, while the mirrored migration's cost does not.

5.5 Revenue Outcomes

Table 7 reports revenue outcomes. The legacy configuration fills 97.96% of opportunities — marginally below the proposed configuration's 99.28% — because, absent floor enforcement, almost any returned bid fills. The configurations diverge on price, not fill: the average clearing CPM rises from \$24.62 to \$42.62 (+73.1%), and revenue per 1,000 auction opportunities from \$24.12 to \$42.31 (+75.5%). Two mechanisms produce this gap in the simulation: participation breadth (the concurrent configuration hears from more than twice as many DSPs per auction, raising the expected maximum bid) and floor enforcement with floor-aware bidding (bids are raised to at least the floor when valuations clear it). The direction of the floor effect is consistent with the field-experimental literature reviewed in Section 4.2; its magnitude here is a property of this simulation's demand model — in particular the \$25 floor set above the \$22 median valuation — and must not be read as a field estimate.

Table 7. Revenue Results

Metric	Configuration A (legacy)	Configuration B (proposed)
Fill rate (%)	97.96	99.28
Avg. clearing CPM, filled auctions (USD)	24.62	42.62
Revenue per 1,000 opportunities (USD)	24.12	42.31

In thinner demand regimes the same floor would be expected to trade fill rate for price more sharply, which is precisely the calibration problem Section 4.3 assigns to the yield-management function.

6. PROVENANCE OF EVIDENCE: LITERATURE FINDINGS AND ORIGINAL FINDINGS

Because this article combines a literature synthesis with an original simulation study, this section states explicitly which quantitative claims come from which source. Table 8 tabulates the provenance of the principal claims. Two rules were applied throughout the text. First, every number imported from the literature is attributed at the point of use to its source study and to the environment in which it was obtained (display, video, or search advertising; field experiment, production deployment, or measurement study). Second, this study's own findings are confined to Section 5: they are simulation results under the parameters of Table 3, and they are reported as simulation results, not as production measurements. Readers should weight the two classes of evidence differently: the literature results are empirical but were

obtained on other platforms and inventory types, while the simulation results concern the architecture proposed here but rest on a synthetic demand model (see Section 9).

Table 8. Provenance of the Principal Quantitative Claims

Claim	Source
CTV market growth indicators (Table 1)	Cited literature — industry survey, IAB Europe (2023)
First-price transactions were ~40% of impressions but 55.4% of spend	Cited literature — production exchange measurement, Yuan et al. (2013)
Reserve prices raise publisher revenue 35% over a zero-reserve baseline; 44% with minimum-impression constraints	Cited literature — field experiment, Choi & Mela (2023)
Production floor optimization: +1.3% display and +2.5% video annualized revenue	Cited literature — production deployment, Alcobendas et al. (2024)
Reserve prices guided by theory substantially raise seller revenue in sponsored search	Cited literature — field experiment, Ostrovsky & Schwarz (2023)
A majority of Go blocking concurrency bugs arise from message-passing misuse	Cited literature — empirical study of 171 bugs, Tu et al. (2019)
Latency, throughput, reliability, and revenue comparison of the legacy and proposed configurations (Tables 4–7)	This study — discrete-event simulation (Section 5)
Design framework: latency budgeting, concurrent fan-out, gated migration, floor enforcement (Sections 3–4)	This study — synthesis and framework

The distinction matters for how the article's conclusions should be used. The claim that floor-price optimization is a revenue lever rests on the cited field and production evidence; the simulation adds the narrower, architecture-specific finding that, under the stated demand model, enforcing the floor inside the proposed pipeline raises revenue per auction opportunity chiefly through price rather than fill. Conversely, the latency and availability claims rest on the simulation alone and should be re-validated against production telemetry before being used for capacity commitments (Section 8).

7. IMPLEMENTATION CHALLENGES AND OPERATIONAL CONSIDERATIONS

The architectural principles described above are necessary but not sufficient for a reliable CTV RTB engine. Three operational concerns determine whether these principles translate into a system that performs well in production over time rather than only at the moment of launch.

The first is statistical rigor in shadow-mode evaluation. Traffic mirroring produces a large volume of comparison data very quickly, which creates a temptation to advance a phased rollout as soon as headline metrics look favorable. A single stabilization window that happens to include unusually light or unusually heavy traffic can make a marginal engine look acceptable, or make an acceptable engine look marginal. Health-check gates should therefore be defined over a fixed minimum sample size and a fixed minimum time window, not simply over whichever traffic happens to have accumulated since the last gate check, and the gate thresholds should be set with reference to the exchange's own historical variance rather than to a generic industry figure. Where floor-price changes are being evaluated alongside an engine migration, the two should not be rolled out in the same phase, because a revenue change cannot then be attributed to either cause with confidence. The second is the cost and correctness trade-off in DSP fan-out width. Widening the fan-out set, meaning sending each auction to more DSPs, generally increases the probability of receiving a competitive bid, but it also increases outbound request volume, DSP-side load, and the number of goroutines that must be tracked and, if necessary, cancelled within the auction timeout. Because floor-based pre-fan-out gating (Section 3.1) removes DSPs unlikely to clear the floor, the marginal DSP added to a fan-out set is disproportionately likely to be one whose historical bid distribution is uncertain rather than clearly favorable, which means the expected revenue benefit of widening fan-out declines while the operational cost does not. Engineering teams should treat fan-out width as a tunable parameter validated against measured win-rate and latency data for their own DSP roster, rather than as a fixed architectural constant.

The third is cross-team coordination between the engineering organization that owns the RTB engine and the revenue or yield-management organization that owns floor-price policy. Because floor prices are enforced inside the auction pipeline but set as a business policy, a floor change deployed without engineering visibility can silently interact with the pre-fan-out gating logic in Section 3.1, for example by excluding a DSP that would have cleared the new floor but was filtered out based on stale historical bid statistics. Production systems should therefore treat the floor value itself as a monitored input, with alerting on the rate and magnitude of floor changes, and the pre-fan-out gate's DSP eligibility cache should be invalidated or recomputed whenever the floor changes materially, rather than relying on its normal refresh cadence.

Taken together, these operational considerations reinforce a general point about the architecture described in Sections 3 and 4: latency budgeting, concurrency discipline, migration safety, and floor-price policy are not independent subsystems that can be designed, validated, and monitored in isolation. Each interacts with the others under production load, and a CTV RTB engine that performs well against each design principle in

isolation can still underperform if the interactions between them are not monitored explicitly.

8. MANAGERIAL IMPLICATIONS

The framework and simulation results carry several implications for managers of advertising platforms and of the publisher and broadcaster businesses that depend on them. The first concerns capacity planning. Because exchange deadlines convert latency directly into lost auctions, capacity must be planned against deadline-compliant throughput — in this simulation, 1,400 auctions per second per 96-slot pool at a 1% miss threshold — rather than against the raw request rate a front door can accept. A platform that provisions to average latency will appear adequately sized in ordinary monitoring while forfeiting its most valuable primetime auctions, whose loss is invisible in fill-rate reports that count only completed auctions.

The second implication concerns change management. The deployment-event experiment prices a routine engineering decision: at a modest 600 auctions per second, a downtime-restart deployment forfeited roughly \$637 of revenue and 21 percentage points of availability in a single two-minute window relative to a mirrored, health-gated migration, and the gap scales linearly with load. Release processes for revenue-path systems should therefore be evaluated as revenue decisions, with zero-downtime migration capability treated as a prerequisite for changes to auction logic rather than as an engineering luxury.

The third implication is organizational. Floor prices are set as business policy but enforced — and, through pre-fan-out gating, amplified — inside the auction pipeline. The simulation shows revenue per opportunity is highly sensitive to this joint configuration: the same floor that raises clearing prices also determines which DSPs are worth calling. Yield management and platform engineering therefore need a shared operating picture — floor changes notified to engineering as production inputs, gating thresholds reviewed against realized fill and clearing prices — rather than the sequential hand-off (policy set by one team, implemented by another) that Section 7 identifies as a recurring failure mode.

Finally, the results suggest a sequencing rule for platform investment. In the simulation, the largest single improvement came not from floor tuning but from the serving architecture itself: serialization, concurrent fan-out, and bounded collection converted a configuration that missed roughly 95% of deadlines into one that missed none, and only on that foundation did floor enforcement add its price effect. Managers evaluating vendor platforms or internal roadmaps should accordingly ask for deadline-compliance evidence at p99 under stated load before crediting revenue-optimization features, since the features cannot pay out on auctions the engine never completes.

9. LIMITATIONS

This study has several limitations, which bound the claims that should be drawn from it. First, the evaluation is a simulation, not a production deployment or field experiment. The editor's alternatives — a case study or a simulation — were resolved in favor of simulation for reproducibility and controlled comparison, but simulation results inherit the assumptions of their demand and latency models and are not a substitute for production telemetry.

Second, the parameters, while stated in full, are not calibrated to a specific exchange. DSP response times and valuations are lognormal with plausible but generic parameters; real DSP populations differ in their latency tails, timeout behavior, and valuation distributions, and per-DSP heterogeneity in the model is stylized. The absolute values in Tables 4–7 should therefore be read as comparative — configuration against configuration under identical demand — rather than as predictions for any particular platform.

Third, the revenue model is deliberately simple. It assumes first-price auctions, uniform bid shading within stated ranges, and, for the proposed configuration, floor-aware bid raising when valuations clear the floor. It does not model bidder budget pacing, frequency capping, auction-time bid shading strategies that adapt to the floor over time, or second-price and hybrid auction formats. A single floor value (\$25 CPM) is evaluated; the floor level itself is not optimized, and in thinner demand regimes the fill-rate cost of enforcement would be larger than the near-zero cost observed here.

Fourth, the systems model abstracts from network topology, multi-region deployment, and partial failures other than DSP hangs and errors; the migration experiment models deployment strategies on a single engine instance pool. Finally, the framework synthesis draws on literature from display, video, and search advertising, and CTV-specific empirical validation of the combined framework — ideally a production case study with a publishing partner — remains future work.

10. DISCUSSION

The architectural principles examined in this article converge on a coherent set of design imperatives for CTV RTB infrastructure. The latency budget is not a monitoring target but a first-class design constraint: the auction cycle must be decomposed at architecture time across ingress parsing, floor gating, DSP fan-out, response aggregation, and auction execution, with each stage allocated an explicit time share. Protocol-level decisions, including serialization format, transport choice, and timeout propagation mechanism, must be evaluated against their contribution to p99 latency rather than against their median-case performance, because p99 latency is what determines how many auctions clear within tmax under realistic load.

The revenue opportunity in CTV RTB is substantially realizable through floor-price engineering. The consistent evidence from Choi and Mela (2023), Alcobendas et al. (2024), and Ostrovsky and Schwarz (2023) demonstrates that production-grade floor optimization delivers measurable revenue lift, ranging from single-digit percentage gains in some production deployments to 35 to 44% over a zero-reserve baseline in controlled field experiments. For CTV platforms, where per-auction CPMs are elevated relative to desktop display, the absolute revenue impact of floor calibration improvements is disproportionately large. Engineering teams should treat floor-price logic as a first-class revenue component of the auction pipeline, instrumented, monitored, and continuously refined, rather than as a static configuration parameter, and Section 7 argues that this instrumentation must extend to the interaction between floor policy and fan-out gating rather than treating floor logic in isolation.

The Go-based concurrency architecture must be designed with explicit knowledge of the failure modes documented by Tu et al. (2019): the finding that a majority of blocking bugs in mature Go codebases arise from message-passing misuse is not a cautionary footnote but an architectural specification. Goroutine lifecycle management, channel buffer sizing, and context propagation are primary determinants of p99 latency stability under DSP failure conditions, and the operational discipline of monitoring goroutine counts as a leading indicator, described in Section 3.2, is what turns this finding into a production safeguard rather than a background risk.

Zero-downtime migration using traffic mirroring and circuit-breaker-gated phased rollout is the appropriate framework for changes to a production CTV RTB engine. The revenue exposure of maintenance windows is unacceptable in continuous streaming environments, and the complexity of auction logic changes means shadow validation is necessary to detect correctness failures before they reach live traffic. As Section 7 discusses, the value of this framework depends on disciplined statistical evaluation of shadow data, since a phased rollout that advances on noisy or underpowered comparisons provides a false sense of the safety the framework is meant to guarantee.

CTV programmatic advertising is not simply a scaled extension of desktop display RTB. It is a distinct engineering domain with distinct operational constraints, CPM profiles, and viewer-experience requirements. The infrastructure supporting it must be built to match these constraints, with latency discipline at the architecture level, revenue engineering at the auction-logic level, migration safety built into the deployment process, and cross-functional monitoring that treats latency, concurrency, migration state, and floor policy as interacting parts of a single system rather than as independently managed components.

11.CONCLUSION

Connected television has established itself as a fast-growing and high-value segment of the programmatic advertising ecosystem, driven by measurable shifts in viewer behavior and by advertiser budget migration from linear television. As CTV advertising spend grows and programmatic infrastructure scales to support very high auction volumes, engineering teams responsible for RTB infrastructure face a dual mandate: strict compliance with the latency constraints of OpenRTB 2.6 auction protocols, and maximization of revenue yield per auction through precise floor-price enforcement and competitive DSP fan-out.

This article has examined the architectural principles governing each dimension of this mandate. A latency-budgeted OpenRTB 2.6 pipeline, anchored by Protocol Buffers serialization and pre-fan-out floor gating, provides the throughput foundation. A Go-based concurrent DSP fan-out, designed with explicit goroutine lifecycle discipline informed by evidence that message-passing misuse is a leading cause of blocking bugs in mature Go codebases, sustains reliability under peak CTV load. Traffic mirroring and health-check gate-controlled phased rollout with circuit-breaker protection enable zero-downtime migration of production auction logic without revenue exposure, provided the underlying shadow-mode comparisons are evaluated with statistical discipline. Floor-price optimization, grounded in field-experimental evidence of substantial revenue increases over zero-reserve baselines, positions revenue engineering as a sustained competitive differentiator, provided floor policy is monitored as an input to the auction pipeline rather than managed independently of it. The convergence of empirical evidence, production systems practice, and auction theory examined in this article establishes a design vocabulary for CTV RTB infrastructure. Teams that apply these principles, treating latency as a hard constraint, floor pricing as a revenue instrument requiring cross-functional monitoring, and Go concurrency as a disciplined engineering domain, are positioned to sustain competitive auction performance at CTV scale as the programmatic ecosystem continues its structural migration toward streaming inventory.

The simulation evidence reported in Section 5 indicates that these principles are not merely qualitative. Under the stated parameters, the proposed configuration met the 100 ms decisioning deadline with a deadline-miss rate of 0.0% against the legacy configuration's 94.7%, sustained deadline-compliant throughput that the legacy configuration could not achieve at any offered load, preserved availability through a migration event, and increased revenue per 1,000 auction opportunities by 75.4%. For platform teams and the managers who fund them, the priority that follows is architectural before it is algorithmic: deadline discipline and migration safety create the auctions on which floor-price engineering can then operate.

References:

- Alcobendas, M., Diaz, A., Diaz, O., & Moloco Research Team. (2024). Optimizing floors in first price auctions: An empirical study of Yahoo advertising (arXiv:2302.06018v2). arXiv. <https://arxiv.org/abs/2302.06018>
- Choi, H., & Mela, C. F. (2023). Optimizing reserve prices in display advertising auctions. *Marketing Science*. DOI: 10.1287/mksc.2023.0002
- CNCF. (2023). Argo Rollouts: Progressive delivery for Kubernetes. Cloud Native Computing Foundation. <https://argoproj.github.io/argo-rollouts/>
- Cox, D. (2022). Kubernetes in production: Patterns and anti-patterns for stateless services. O'Reilly Media.
- Edelman, B., Ostrovsky, M., & Schwarz, M. (2007). Internet advertising and the generalized second-price auction. *American Economic Review*, 97(1), 242–259.
- Feng, Z., Lahaie, S., Schneider, J., & Ye, J. (2021). Reserve price optimization for first price auctions in display advertising. In Proceedings of the 38th International Conference on Machine Learning (PMLR Vol. 139, pp. 3230–3239).
- Go Team at Google. (2023). The Go programming language specification. Google LLC. <https://go.dev/ref/spec>
- IAB Europe. (2023). IAB Europe guide to programmatic CTV. IAB Europe.
- IAB Tech Lab. (2022). OpenRTB API specification version 2.6. IAB Tech Lab.
- Irfan, M. (2024). Optimizing publisher revenue in digital marketing using decision trees and random forests. *Journal of Digital Market and Digital Currency*, 1(3), 247–266.
- John, G., & Scheer, L. K. (2021). Commentary: Governing technology-enabled omnichannel transactions. *Journal of Marketing*, 85(1), 126–130.
- Nygard, M. T. (2007). Release It!: Design and deploy production-ready software. Pragmatic Bookshelf.
- Ostrovsky, M., & Schwarz, M. (2023). Reserve prices in internet advertising auctions: A field study. *Journal of Political Economy*, 131(12).
- Perlich, C., Dalessandro, B., Raeder, T., Stitelman, O., & Provost, F. (2012). Machine learning for targeted display advertising. In Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1517–1518). Association for Computing Machinery.
- Sharghivand, N., Derakhshan, F., & Siasi, N. (2021). A comprehensive survey on auction mechanism design for cloud/edge resource management and pricing. *IEEE Access*, 9, 126502–126529.
- Tu, T., Liu, X., Song, L., & Zhang, Y. (2019). Understanding real-world concurrency bugs in Go. In Proceedings of the 24th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (pp. 865–878). Association for Computing Machinery. DOI: 10.1145/3297858.3304069
- Yuan, J., Wang, A., Buckner, C., Huang, K., & Woosley, M. (2013). Real time bidding for online advertising: Measurement and analysis. In Proceedings of the 7th International Workshop on Data Mining for Online Advertising (ADKDD). Association for Computing Machinery.
- Zhang, W., Yuan, S., & Wang, J. (2014). Optimal real-time bidding for display advertising. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1077–1086). Association for Computing Machinery.

Sujan Kumar Reddy Challa

Independent Researcher,
USA.

sujan.challa10@gmail.com

ORCID: 0009-0005-8705-9770
