Online Learning Based Server Selection in Peer-to-Peer Content Delivery Networks

Hadi Hosseini[1], Sam Shippey[1], Alireza Keshavarz-Haddad[2], and Ehsan Aryafar[1]
[1]Portland State University, USA; [2]Shiraz University, Iran

***Abstract*—Content Delivery Networks (CDNs) are experiencing rapidly growing demand due to the proliferation of large-scale video streaming and interactive multimedia applications. To improve scalability and reduce infrastructure costs, recent systems have begun adopting Peer-to-Peer Content Delivery Networks (PCDNs) to complement or replace traditional centralized CDNs. PCDNs utilize excess compute and storage capacity of edge IoT devices (e.g., routers, smart home appliances) to deliver content to end users. In this architecture, a client connects to multiple edge servers simultaneously, receiving data streams in parallel to improve throughput and reliability. While prior work has focused on transport-layer scheduling—i.e., how to distribute packets across multiple paths—little attention has been given to the problem of distributed server selection at each client. To address this gap, we formulate decentralized server selection as an online learning problem, where each client must repeatedly choose a subset of servers under uncertain and time-varying performance. We model this as a combinatorial multi-armed bandit (CMAB) problem and develop a simulation framework to evaluate representative algorithms across stochastic and adversarial settings, including EXP3, Thompson Sampling, and BROAD. We show through extensive simulations that (i) in stationary settings, distributed learning algorithms converge to allocations with total throughput close to the Nash equilibrium outcome, with EXP3 achieving the best trade-off between convergence speed, efficiency, and fairness, (ii) Under network dynamics, EXP3 maintains the highest fairness and stability with near-capacity throughput, and (iii) At the application level, video rebuffering is driven more by throughput stability than average throughput, highlighting a trade-off between adaptability and smooth playback across all algorithms.**

***Index Terms*—Edge Computing, Peer-to-Peer CDNs, Multi-Armed Bandits, Multipath Parallel Data Download**

I. Introduction

The demand for large-scale content delivery continues to grow rapidly, driven by the proliferation of video streaming platforms, social media applications, and immersive multimedia services. Content Delivery Networks (CDNs, Fig. 1a) are the primary infrastructure supporting this demand by deploying geographically distributed servers to cache and deliver content to end users. However, the rising cost of provisioning and operating large-scale CDN infrastructures [1], [2] has motivated the exploration of alternative, cost-effective architectures that leverage resources at the network edge.

One promising approach is the Peer-to-Peer Content Delivery Network (PCDN) architecture (Fig. 1b), where content is distributed using the excess storage and compute capacity of edge IoT devices (peers), such as home routers and smart appliances. A PCDN can be deployed either independently or alongside a CDN to reduce the load on the underlying CDN infrastructure [3]. In a PCDN, when a client requests content, it first contacts a **tracker entity** that maintains a directory of available peers and their stored content. The client/tracker then selects the set of peers that will serve the client.

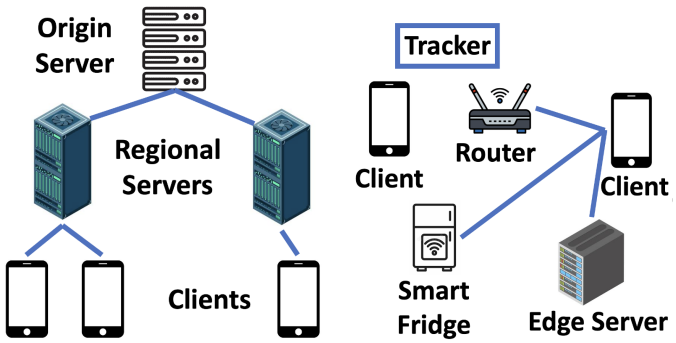


Fig. 1: **(a) CDNs typically use a hierarchical structure with origin servers at the core and regional servers in the middle, (b) PCDNs use spare capacity of edge IoT devices. Clients query a tracker entity to discover peers and download content from multiple devices simultaneously to improve bandwidth and robustness. In the example shown here, three IoT devices serve the client.**

Recent deployments, such as ByteDance's Swarm system [4], demonstrate that PCDNs can significantly reduce infrastructure costs while maintaining acceptable performance by serving each client in parallel through multiple servers (peers). This parallel (i.e., multi-server-to-single-client) data transmission increases aggregate bandwidth, reduces latency, and improves robustness to link or server failures. Over the past few years, several multipath transport-layer scheduling algorithms have been proposed for PCDN architectures to optimally distribute packets from multiple servers to each client [3], [4], [5], [6]. However, little attention has been given to the problem of server selection for each client.

One approach to addressing the server selection problem is to use a centralized algorithm at the tracker entity to assign servers to each client. While this method simplifies resource management, it introduces scalability challenges and creates a single point of failure. Furthermore, maintaining accurate, up-to-date knowledge of peer capacity and load becomes increasingly difficult as the number of participating devices grows and network conditions fluctuate.

979-8-3315-4568-0/26/$31.00 © 2026 IEEE

An alternative approach is to allow each client to autonomously select peers in a fully distributed manner. However, designing such a system presents several challenges. Peer bandwidth is highly variable and often unknown to clients in advance. Multiple clients may simultaneously connect to the same peer, causing congestion and reducing throughput. Additionally, clients typically act selfishly, seeking to maximize their own performance without coordinating with others.

In this work, we investigate whether distributed decision-making combined with online learning can achieve efficient resource allocation in PCDNs without centralized coordination. Specifically, we formulate peer selection as a distributed Multi-Armed Bandit (MAB) problem[1], where each peer corresponds to an arm and the observed download bandwidth serves as the reward signal. Within this framework, clients can gradually identify high-performing peers while adapting to dynamic network conditions and fluctuations in peer capacities.

To evaluate the effectiveness of this approach, we conduct extensive simulations using a custom PCDN simulation framework implemented using SimPy [11]. We study several online learning algorithms, including Thompson Sampling [12], EXP3-based methods [13], and other bandit-based approaches (e.g., BROAD [14]), and analyze their performance in terms of throughput, fairness, convergence behavior, and switching cost. Our results demonstrate that distributed online learning algorithms can achieve performance comparable to centralized scheduling while maintaining scalability and robustness to network dynamics. Specifically, our contributions can be summarized as follows:

- **Simulator Development:** We develop a custom simulator in Python to evaluate distributed learning-based peer selection in PCDNs under varying network conditions and with support for capturing video streaming performance. We plan to publicly release the software upon acceptance of this paper to enable the community to extend it and compare additional AI-based peer selection algorithms.
- **Performance Evaluation:** We conduct extensive simulations to evaluate the performance of different online learning algorithms as a function of server capacities, network dynamics, and traffic type (e.g., short videos and file downloads). We show that **(i) Stationary Environments:** distributed learning algorithms converge to allocations with total throughput close to the Nash equilibrium outcome, with EXP3 achieving the best balance between convergence speed, efficiency, and fairness; **(ii) Dynamic Environments:** under network dynamics, all methods sustain near-capacity throughput, but EXP3 provides superior robustness in terms of fairness and stability; and **(iii) Video Performance:** At the application level, video playback quality—measured by rebuffering ratio—is governed more by throughput stability than average throughput, revealing a trade-off between adaptability and smooth viewing experience across all algorithms.

[1]Multi-armed bandits are a class of online learning problems characterized by partial (bandit) feedback, where the learner observes outcomes only for the selected actions [7], [8], [9], [10].

The rest of this paper is organized as follow. We discuss the related work in Section II. We present our system model and assumptions in Section III. Section IV describes the online learning algorithms studied in this paper. We present the results of our extensive performance evaluation in Section V. Finally, we conclude the paper in Section VI.

II. Related Work

Content Delivery Architectures. Modern content delivery relies on a spectrum of architectures that balance performance, cost, and scalability. Traditional CDNs, such as Akamai Technologies [15] and Cloudflare [16], deploy geographically distributed server infrastructures to serve content from locations close to end users, thereby reducing latency and improving availability. In contrast, PCDNs [4], [5], [6] augment or partially replace dedicated infrastructure by incorporating end-user devices into the delivery process, enabling more scalable and cost-efficient content distribution through resource sharing at the network edge. Hybrid CDN architectures [17], [18], [19] combine these two paradigms, leveraging the reliability and control of centralized servers alongside the elasticity of peer-to-peer overlays, allowing systems to better handle flash crowds and operate effectively in regions with limited infrastructure. Further, PCDNs and hybrid CDNs leverage excess capacity of edge IoT devices and employ a multi-server-to-single-client transport layer to increase communication throughput and reliability.

Combinatorial Multi-Armed Bandits (CMAB). The CMAB framework has been studied under both stochastic and adversarial settings to address the exploration–exploitation trade-off under partial feedback. In stochastic environments, Thompson Sampling (TS) [12], [20], [21] is a Bayesian approach that samples actions according to their posterior optimality and achieves strong empirical performance. In contrast, adversarial algorithms such as EXP3 [13], [22] use exponential weighting to ensure sublinear regret even under worst-case rewards. More recently, mirror-descent-based methods such as BROAD [14] provide adaptive, data-dependent guarantees in adversarial and combinatorial semi-bandit settings. These methods span stochastic, adversarial, and adaptive regimes, making them natural baselines for CMAB problems with non-stationarity and interaction effects across arms.

III. System Model

In this section, we describe the system model used throughout the paper and outline the assumptions underlying our algorithms and simulations. We consider a PCDN architecture in which clients download content from multiple peers simultaneously. Peer selection is modeled as a decentralized decision problem, where clients independently choose peers based solely on locally observed performance. To capture uncertainty in peer capacity and the lack of coordination among clients, we formulate peer selection as a distributed combinatorial Multi-Armed Bandit (CMAB) problem with shared-resource rewards.

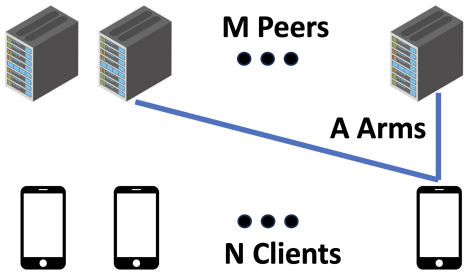


Fig. 2: **There are N clients and M peers in the system. For each chunk, a given client selects A peers and concurrently downloads the chunk from the peers leveraging a multipath transport protocol such as MPRD [3].**

A. PCDN Model

We consider a PCDN *swarm* consisting of N *clients* (players) and M *peers*. Each client attempts to download a file divided into T *chunks*, which are downloaded sequentially. At the beginning of each chunk download, the client receives a set of candidate peers that store the required chunk[2].

Each client maintains simultaneous connections to A peers, and we refer to each connection as an *arm*. This multi-server-to-single-client parallel data download increases communication bandwidth and improves robustness [3], [5].

At the beginning of each chunk download, client i selects a set of A peers, denoted by S_i, where $|S_i| = A$. These connections remain active for the duration of the chunk download. Each peer j has a capacity C_j, representing the maximum total bandwidth it can provide to connected clients. We assume that each peer divides its capacity equally among all connected clients (arms).

B. Strategy Space and Payoff

We model peer selection as a strategic interaction among clients. Let $\mathcal{M} = \{1, 2, \ldots, M\}$ denote the set of available peers. The strategy space of client i is

$$\mathcal{S}_i = \{S' \subseteq \mathcal{M} \mid |S'| = A\} \tag{1}$$

A strategy profile for all clients is denoted by

$$S = (S_1, S_2, \ldots, S_N), \tag{2}$$

where $S_i \in \mathcal{S}_i$ is the peer set selected by client i.

Let $N_j(S)$ denote the total number of client connections (arms) assigned to peer j under strategy profile S. Since peer j shares its capacity equally among all connected clients, each connection to peer j receives a bandwidth equal to $\frac{C_j}{N_j(S)}$. The payoff of client i, defined as its total throughput across all its arms under strategy profile S, is therefore

$$r_i(S) = \sum_{j \in S_i} \frac{C_j}{N_j(S)}. \tag{3}$$

[2]In practice, chunks are not necessarily available at every peer. A lightweight tracker entity or selector service typically maintains metadata indicating which peers store each chunk. Clients query this service to obtain the set of candidate peers containing the desired chunk.

Note that as peer capacity is shared among all connected clients (arms), the payoff of each client depends on the peer selections of other clients.

C. Connection Constraints and Assumptions

We impose the following assumptions throughout the paper.

- **Multiple Connections.** Each client maintains A simultaneous connections to peers.
- **Distinct Peers.** No two arms of the same client may connect to the same peer.
- **Equal Capacity Sharing.** Each peer divides its capacity (bandwidth) equally among all connected clients.
- **Limited Information.** Clients do not know the peer capacities C_j or the number of other clients connected to a given peer. They only observe the bandwidth obtained from a peer after connecting to it.
- **Selfish Behavior.** Clients act independently and attempt to maximize their own throughput.

D. Combinatorial Multi-Armed Bandit Problem Formulation

The peer selection problem can be viewed as a distributed online learning problem, where each client acts as a bandit agent that repeatedly selects peers and observes the resulting throughput from each selected peer. The client then uses this feedback to update its future decisions. The objective for each client is to maximize cumulative reward over time (e.g., throughput), while balancing exploration (trying less-known servers) and exploitation (choosing currently high-performing ones).

Specifically, each chunk download defines a decision round. At the beginning of each round, a client selects A peers and observes the bandwidth obtained from each selected peer (semi-bandit feedback). Importantly, the reward obtained from a peer is not independent: since multiple clients may select the same peer simultaneously, the achieved throughput depends on the (unknown and time-varying) level of contention induced by other clients' actions. This introduces coupling across agents and makes the reward process non-stationary from the perspective of any single client.

This problem naturally falls under the framework of online learning because decisions are made sequentially, feedback is revealed only after actions are taken, and the system dynamics are initially unknown. More specifically, it is a combinatorial multi-armed bandit (CMAB) problem. Here, instead of selecting a single arm per round, the client selects a subset of A arms simultaneously. The action space is therefore combinatorial, consisting of all $\binom{M}{A}$ possible subsets. The reward depends on the joint selection (e.g., additive under ideal conditions or sub-additive due to shared bottlenecks), and the learner must infer the quality of individual peers and their interactions from partial feedback. This combinatorial structure, together with bandit (or semi-bandit) feedback and inter-agent coupling, distinguishes the problem from classical single-play bandits and motivates algorithms tailored to CMAB settings.

IV. Online Learning Algorithms

In this section, we describe the algorithms studied in this paper. We focus on three prominent online learning algorithms, adapted to match our problem setting: EXP3 [13], BROAD [14], and Thompson Sampling [12][3]. We also include two baseline methods: Random and Centralized Nash, the latter representing a centralized load balancer with full knowledge of peer capacities. Note that the online learning algorithms operate without knowledge of peer capacities or the number of clients.

EXP3. EXP3 [13] is an adversarial multi-armed bandit algorithm that maintains a weight $w_j(t)$ for each peer j. The selection probability is computed as

$$p_j(t) = (1-\gamma)\frac{w_j(t)}{\sum_{k=1}^{M} w_k(t)} + \frac{\gamma}{M} \tag{4}$$

where $\gamma \in (0,1)$ controls exploration. Each client i samples A distinct peers according to $p_j(t)$.

Let $S_i(t)$ denote the set of selected peers by client i. After observing the reward $r_j(t)$ for each $j \in S_i(t)$, we form an importance-weighted estimate:

$$\hat{r}_j(t) = \frac{r_j(t)}{p_j(t)}, \quad \forall j \in S_i(t) \tag{5}$$

and update the weights as

$$w_j(t+1) = w_j(t)\exp\left(\frac{\gamma \hat{r}_j(t)}{M}\right), \quad \forall j \in S_i(t) \tag{6}$$

while weights for $j \notin S_i(t)$ remain unchanged.

In our setting, EXP3 enables each client to adaptively favor peers that yield higher throughput while maintaining exploration, making it robust to contention and non-stationarity.

BROAD. BROAD [14] is an optimistic adversarial bandit algorithm based on log-barrier mirror descent. It maintains a probability distribution over peers, along with a prediction vector $\mathbf{m}_t \in \mathbb{R}^M$ that estimates the next round's losses, a reference distribution $\mathbf{w}'_t \in \Delta_M$ where Δ_M is the M-dimensional probability simplex[4], and a learning rate vector $\boldsymbol{\eta}_t \in \mathbb{R}^M$ with an element for each peer.

At each round, the client computes the selection distribution $\mathbf{w}_t$ by solving the following log-barrier regularized optimization problem:

$$\mathbf{w}_t = \arg\min_{\mathbf{w}\in\Delta_M} \langle \mathbf{w}, \mathbf{m}_t\rangle + \sum_{j=1}^{M} \frac{1}{\eta_{t,j}}\left(\frac{w_j}{w'_{t,j}} - 1 - \log\frac{w_j}{w'_{t,j}}\right) \tag{7}$$

The resulting distribution is then mixed with a small uniform exploration component to obtain $\bar{\mathbf{w}}_t$, from which each client samples A distinct peers without replacement.

[3]Upper Confidence Bound (UCB) [23] was also evaluated, but its performance was inferior to the other algorithms and is therefore omitted from the results due to the page limitations.

[4]The M-dimensional probability simplex describes the space of all probability distributions over M discrete options. More specifically it is the set of points Δ_M where for every point $x \in \Delta_M$, $\sum_{i=1}^{M} x_i = 1$ and $x_i \geq 0$ $\forall i \in \{0, 1, 2, ..., M\}$.

In our setting, the feedback corresponds to observed throughput. Since BROAD is formulated in terms of losses, we define the loss for peer j as the negative reward:

$$\ell_j(t) = -\frac{C_j}{N_j(t)}. \tag{8}$$

After observing the selected peers, the client constructs an importance-weighted loss estimate $\hat{\ell}_j(t)$ based on the sampling distribution. BROAD then computes a correction term $\mathbf{a}_t$ that captures the discrepancy between the predicted losses $\mathbf{m}_t$ and the observed estimates. The learning rates are also updated. In the PCDN setting, BROAD produces smooth updates to peer selection probabilities, leading to lower switching behavior and more stable allocations. However, due to its gradual updates, it may adapt more slowly to rapid changes in peer load and network conditions.

Thompson Sampling. Thompson Sampling [12] models each peer as an uncertain reward source and represents that uncertainty with a posterior distribution. At each round, the client samples one value from each peer's posterior and selects the A peers with the largest sampled values. This implements probability matching: peers with higher expected performance are selected more often, while uncertain peers are explored.

In the classical formulation, Thompson Sampling assumes bounded rewards and models each arm using a Beta distribution. Specifically, with normalized rewards $\tilde{r}_j \in [0,1]$, each peer is associated with a posterior $\mathrm{Beta}(\alpha_j, \beta_j)$, updated as

$$\alpha_j \leftarrow \alpha_j + \tilde{r}_j, \quad \beta_j \leftarrow \beta_j + 1 - \tilde{r}_j. \tag{9}$$

However, in our PCDN setting, rewards correspond to throughput values, which are continuous, unbounded, and exhibit variability due to contention and network dynamics. To better capture this behavior, we adopt a Gaussian model and maintain a posterior of the form

$$\mu_j \sim \mathcal{N}(\hat{\mu}_j(t), \sigma_j^2(t)). \tag{10}$$

At each round, we sample

$$\tilde{\mu}_j(t) \sim \mathcal{N}(\hat{\mu}_j(t), \sigma_j^2(t)), \tag{11}$$

and select the A peers with the highest sampled values.

After observing throughput, posterior parameters are updated using standard Gaussian [12]. This formulation is better suited for our setting as it models continuous rewards and captures uncertainty arising from fluctuating peer loads.

Random. In the Random policy, at each round, each client selects A peers from a total of M available peers uniformly at random. This baseline does not use feedback.

Centralized Nash. As a centralized benchmark, we consider a global allocation computed with full system knowledge of peer capacities and client assignments. The objective is to reach a stable allocation in which no client can improve its throughput through unilateral deviation. The resulting allocation corresponds to a (pure-strategy) Nash equilibrium, where the maximum deviation gain is zero. This solution requires centralized coordination and full knowledge of the system at each round, making it impractical for large-scale dynamic PCDN architectures.

V. Performance Evaluation

In this section, we evaluate the performance of distributed online learning algorithms for peer selection in PCDNs under a variety of system conditions. We consider both stationary and dynamic environments, as well as application-level performance in video streaming scenarios.

A. Simulation Framework

Simulator Development. We develop a custom simulation framework in Python to evaluate distributed learning-based peer selection algorithms in a peer-to-peer content delivery network (PCDN). The simulator follows a time-driven (round-based) design, where the system evolves in discrete time steps. At each round, each client independently selects a set of A peers and observes the resulting throughput based on current network conditions, and update their internal learning parameters. This round-based abstraction aligns naturally with bandit algorithms, where actions and rewards are observed sequentially over time.

The simulator supports both stationary and dynamic operating conditions. In the stationary setting, key system parameters such as peer capacities and the number of active clients remain fixed throughout the simulation, allowing us to study convergence properties and equilibrium behavior in stable environments. In contrast, the dynamic setting captures realistic network variability by allowing either peer capacities or the client population to evolve over time. These dynamics are modeled through stochastic processes (e.g., Poisson processes to model client arrivals and departures), creating a non-stationary environment that requires continuous adaptation from the online learning algorithms.

At each round, the interaction between clients and peers is modeled at an aggregate flow level. Each peer has a finite capacity that is shared equally among its connected clients, and the throughput of each client is computed as the sum of rates obtained from its selected peers. The simulator updates system states including allocations, loads, and capacities at every round, enabling the analysis of both efficiency and stability under varying conditions. To ensure statistical reliability, we perform Monte Carlo simulations over 30 independent runs. In each run, environmental dynamics are kept consistent across algorithms, while their internal stochastic decisions evolve independently. Performance metrics such as total throughput, Jain's fairness index, switching cost and maximum deviation gain are recorded over time and averaged across runs, providing a comprehensive evaluation of algorithm behavior under both stationary and dynamic scenarios.

Network Settings. We consider a PCDN system with $M = 6$ peers and $N = 20$ clients. **Unless otherwise specified, each client is equipped with $A = 2$ arms (i.e., representing a common dual-connectivity architecture).** This setting satisfies $N > M$ and $M > A$, reflecting practical scenarios where many clients compete over limited heterogeneous resources with a non-trivial combinatorial action space.

We evaluate two representative capacity configurations to capture different levels of system heterogeneity. In the first setting, peers exhibit highly heterogeneous capacities spanning several orders of magnitude, given by

$$C = [10000, 1000, 100, 30, 10, 1] \text{ Mbps},$$

with an aggregate system capacity of 11141 Mbps. This configuration models realistic PCDN environments where peers have widely varying bandwidth capabilities. In the second setting, we consider a near-homogeneous configuration where peer capacities are within the same range,

$$C = [500, 450, 480, 420, 510, 460] \text{ Mbps},$$

resulting in an aggregate capacity of 2820 Mbps. This setting isolates the effects of contention and learning dynamics under reduced resource disparity. Together, these configurations enable evaluation across both highly skewed and balanced network conditions.

We compare five algorithms in this framework. The first three algorithms, namely EXP3, Thompson Sampling, and BROAD, are online learning methods that enable clients to adaptively select peers based on observed rewards without coordination. In addition, we include two baselines: a Random policy, where clients select peers uniformly at random, and a Centralized Nash solution, which assumes full system knowledge and computes a load-balanced allocation corresponding to a Nash equilibrium. These baselines serve as reference points for comparison, representing uncoordinated behavior (Random) and an ideal centralized allocation (Centralized Nash), thereby enabling a clear evaluation of the efficiency and fairness of distributed learning approaches.

Impact of the Number of Arms. In addition to the above configurations, we investigate the impact of the number of arms by varying $A \in \{1, 2, 3, 4\}$ under the heterogeneous capacity (first) setting. Increasing A expands the combinatorial action space and introduces additional coupling between client decisions due to shared resources. This experiment allows us to evaluate how action space complexity affects learning dynamics, convergence behavior, and the throughput–fairness trade-off. We focus on the EXP3 algorithm in this particular analysis, as it demonstrates strong performance relative to other online learning algorithms in the heterogeneous setting (as shown later).

B. Temporal Behavior in Stationary Environments

In this section, we investigate the temporal behavior of the considered algorithms in terms of stability, switching cost, throughput and fairness. For the learning algorithms, we use the following parameters: EXP3 is configured with an exploration parameter $\gamma = 0.07$, BROAD uses a learning rate $\eta = 0.05$, and Thompson Sampling assumes a Gaussian prior with initial mean 0.5, initial precision 1, and observation noise variance $\sigma^2 = 0.05$ [12]. These parameter choices provide a balanced trade-off between exploration and exploitation, allowing the algorithms to adapt efficiently to the underlying system dynamics while maintaining stable learning behavior. All time-series results are smoothed using a moving average

with a window size of 200 to reduce short-term fluctuations and highlight long-term trends.

Maximum Deviation Gain. To evaluate the convergence behavior of the system, we adopt the *maximum deviation gain* as a metric to quantify how close the system is to a stable operating point. Intuitively, this metric captures the *maximum improvement any client can achieve by unilaterally changing any of its selected peers while all other clients keep their strategies fixed*. Formally, let $\epsilon(t)$ denote the maximum deviation gain at round t. It is defined as the maximum increase in throughput that any client can obtain by deviating from its current allocation to an alternative feasible action. A system is said to be in an ϵ-equilibrium if no client can improve its throughput by more than ϵ through unilateral deviation. As $\epsilon(t)$ approaches zero, the system converges to a stable state close to a Nash equilibrium. We use this metric to characterize the convergence time of each algorithm. Specifically, convergence is declared when $\epsilon(t)$ stabilizes below a small threshold and exhibits negligible variation over a sustained time window. This approach captures both the magnitude and stability of deviation gains, ensuring that the system has reached a steady operating regime rather than a transient fluctuation.

Fig. 3a illustrates the evolution of the maximum deviation gain $\epsilon(t)$ in the heterogeneous capacity setting. The vertical dashed lines represent the convergence time for each algorithm, where variation in $\epsilon(t)$ falls below a fixed threshold. EXP3 exhibits a clear and consistent convergence trend, with $\epsilon(t)$ decreasing rapidly over time and stabilizing at a relatively low value. Thompson Sampling also converges, but at a slower rate and to a higher steady-state deviation compared to EXP3. In contrast, BROAD shows little to no convergence, with $\epsilon(t)$ remaining relatively high and fluctuating over time. The Random policy exhibits the worst behavior, maintaining a consistently large deviation gain, indicating a lack of stability. As expected, the Centralized Nash solution achieves zero deviation gain throughout, confirming that it corresponds to a Nash equilibrium allocation.

Fig. 3b shows the convergence behavior in the near-homogeneous capacity setting. Notably, BROAD achieves a lower ϵ compared to the heterogeneous setting. This is because, under near-homogeneous capacities, the reward differences between peers are small, reducing the penalty of suboptimal selections and enabling BROAD's smoother, gradient-based updates to maintain more balanced allocations across peers. As a result, the incentives for unilateral deviation are reduced, leading to a lower deviation gain. Moreover, the performance gap between algorithms is significantly reduced compared to the heterogeneous setting, highlighting the critical role of capacity diversity in enabling efficient learning. Fig. 3c presents the impact of the number of arms on convergence for EXP3. Increasing A significantly accelerates convergence and reduces the steady-state deviation gain. In particular, larger values of A allow clients to exploit multiple peers simultaneously, improving load balancing and reducing incentives for unilateral deviation. As a result, the system approaches equilibrium more rapidly as the action space expands.

Findings: Convergence to stable and near-equilibrium states strongly depends on both network heterogeneity and action space size. EXP3 consistently achieves faster convergence and lower deviation gains compared to other distributed algorithms, particularly in heterogeneous environments. Moreover, increasing the number of arms improves both convergence speed and stability by enabling more efficient resource utilization and reducing contention-induced imbalance.

Switching Cost. To evaluate the stability of client decisions over time, we consider the *switching cost*, which captures how frequently clients change their selected peers across consecutive rounds. Specifically, the switching cost at each round is defined as the average number of changes in selected peers per client, normalized by the total number of arms ($N \times A$). This normalization ensures that the switching cost is independent of the system size and directly comparable across different settings. A lower switching cost indicates more stable and consistent peer selections, while a higher value reflects frequent reconfiguration and increased system overhead.

Fig. 3d shows the evolution of the average switch rate in the heterogeneous capacity setting. EXP3 exhibits a clear decreasing trend over time, indicating that clients progressively stabilize their peer selections as learning converges. Thompson Sampling also reduces its switching rate, but more gradually and to a higher steady-state value compared to EXP3. In contrast, BROAD maintains a relatively constant switching rate after a brief initial drop, suggesting limited stabilization in its decisions. The Random policy exhibits persistently high switching, reflecting the absence of learning, while the Centralized Nash solution achieves zero switching as it corresponds to a fixed allocation.

Fig. 3e presents the switching behavior in the near-homogeneous capacity setting. In this case, EXP3 and Thompson Sampling maintain relatively high and nearly constant switching rates over time, indicating that clients continue to explore due to the lack of strong reward differentiation among peers. BROAD, however, achieves a lower and more stable switching rate, as its smoother updates promote more consistent selections in environments with similar peer capacities. The Random and Centralized Nash policies exhibit behavior consistent with the heterogeneous case.

Fig. 3f illustrates the impact of the number of arms on switching cost for EXP3 under the heterogeneous capacity setting. Increasing A significantly reduces the switching rate and accelerates stabilization. With more arms, clients can distribute their traffic across multiple peers, reducing the need for frequent reconfiguration and enabling more stable allocations. As a result, both the convergence speed and steady-state switching cost improve as A increases.

Findings: Switching cost is strongly influenced by both network heterogeneity and action space size. EXP3 achieves the lowest switching cost in heterogeneous environments due to effective learning and stabilization, while BROAD performs better in near-homogeneous settings due to its smoother updates. Increasing the number of arms reduces switching by enabling more flexible and stable resource allocation.

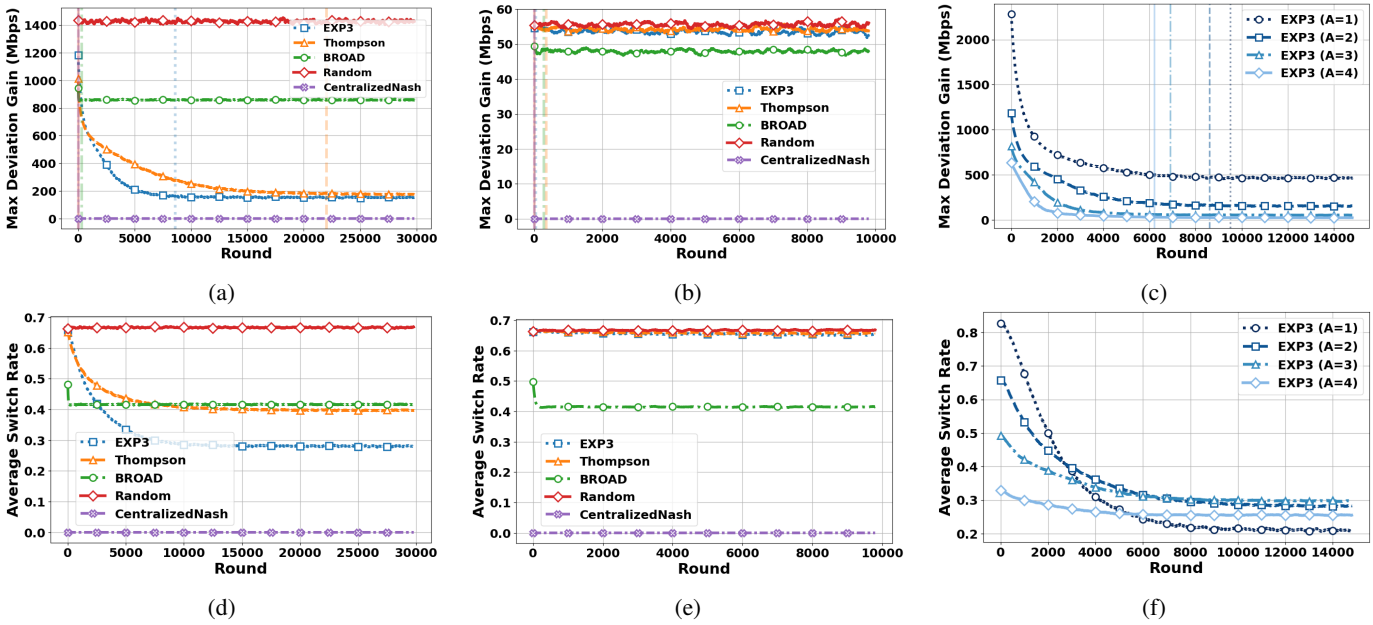


Fig. 3: **(a) Evolution of the maximum deviation gain $\epsilon(t)$ in the heterogeneous capacity setting. The vertical dashed lines represent the convergence time for each algorithm, where variation in $\epsilon(t)$ falls below a fixed threshold, (b) Evolution of the maximum deviation gain $\epsilon(t)$ in the near-homogeneous capacity setting, (c) Impact of the number of arms on the convergence of EXP3 under the heterogeneous capacity setting. Increasing A (number of arms) accelerates convergence and reduces the steady-state deviation gain, (d) Average switching cost over time in the heterogeneous capacity setting. EXP3 exhibits a clear reduction in switching rate as learning progresses, (e) Average switching cost over time in the near-homogeneous capacity setting. EXP3 and Thompson Sampling maintain relatively high and stable switching rates, BROAD achieves a lower and more stable switching rate, (f) Impact of the number of arms on switching cost for EXP3 under the heterogeneous capacity setting.**

Jain Fairness Index. We evaluate the fairness performance of all algorithms using the Jain fairness index over time under the same settings. In the heterogeneous setting (Fig. 4a), EXP3 and Thompson Sampling achieve significantly higher fairness compared to BROAD and Random. EXP3 converges rapidly to near-optimal fairness (close to 1), indicating its strong ability to balance load across highly uneven peer capacities. Thompson Sampling follows a similar trend but converges slightly slower and to a marginally lower fairness level. In contrast, BROAD stabilizes at a substantially lower fairness value (around 0.6), suggesting that its update mechanism struggles to properly distribute load in highly skewed environments. The Random policy yields the lowest fairness due to its lack of adaptation, while the Centralized Nash solution provides the optimal fairness benchmark.

In the near-homogeneous setting (Fig. 4b), all learning-based algorithms achieve relatively similar fairness levels, with only minor differences between them. Since peer capacities are close in value, the system naturally tends toward balanced allocations even without sophisticated learning. As a result, the fairness gap between algorithms diminishes, and all approaches operate near a high fairness regime. BROAD performs slightly better in this setting compared to the heterogeneous case, reflecting its stability in environments with reduced resource disparity. Finally, when varying the number of arms (Fig. 4c), increasing A significantly improves fairness for EXP3. With more arms, each client can distribute its traffic across multiple peers, reducing congestion imbalance and accelerating convergence toward fair allocations. As A increases from 1 to 4, fairness improves both in terms of convergence speed and final steady-state value, approaching near-optimal fairness for larger A.

Findings: EXP3 consistently achieves high fairness, especially in heterogeneous environments, and increasing the number of arms significantly enhances fairness by enabling better load distribution across peers.

Throughput–Fairness Trade-off at Convergence. We evaluate the trade-off between total system throughput and fairness by comparing the *steady-state performance* of all algorithms after convergence. Specifically, the reported values correspond to time-averaged throughput and Jain fairness index computed after the convergence time T_c, indicated by the vertical colored dashed line in Figs. 3a, 3b, 3c, ensuring that the system behavior reflects stable operating conditions.

In the heterogeneous capacity setting (Fig. 4d), a clear trade-off between throughput and fairness is observed in the steady state. EXP3 achieves a near-optimal balance, with high throughput and close to the optimal fairness. Thompson Sampling achieves slightly higher throughput than EXP3 but at the cost of reduced fairness. In contrast, BROAD achieves high throughput but suffers from significantly lower fairness, indicating an imbalanced allocation of resources even after convergence. The Random policy performs poorly in terms of fairness. The Centralized Nash solution achieves optimal fairness; however, since it selects the first Nash equilibrium it identifies and remains fixed at that point over time, the

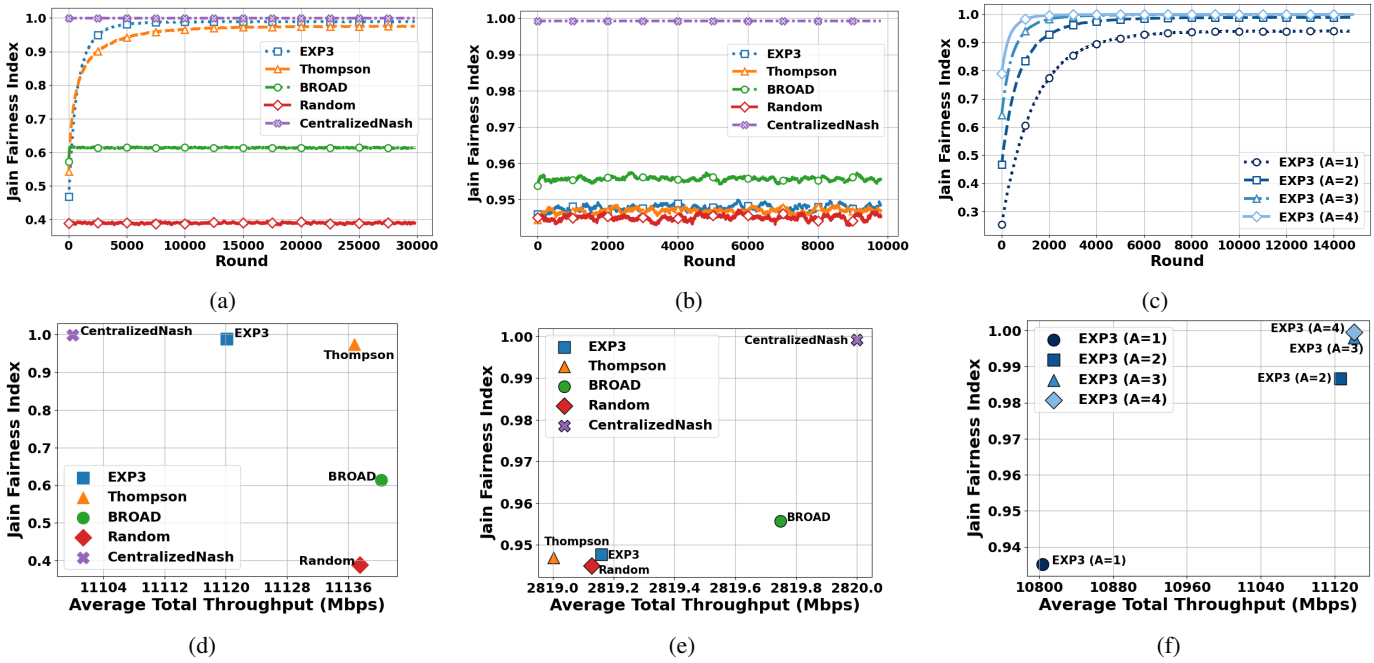


Fig. 4: **(a) Jain fairness index over time in the heterogeneous capacity setting. EXP3 and Thompson Sampling achieve high fairness. BROAD stabilizes at a lower fairness level, (b) Jain fairness index over time in the near-homogeneous capacity setting. All algorithms achieve similar fairness levels. BROAD performs comparatively better than in the heterogeneous case, (c) Impact of the number of arms on fairness for EXP3 under the heterogeneous capacity setting. Increasing A significantly improves both convergence speed and steady-state fairness, (d) Throughput–fairness trade-off at convergence in the heterogeneous capacity setting. EXP3 achieves the best balance between throughput and fairness, (e) Throughput–fairness trade-off at convergence in the near-homogeneous capacity setting, (f) Impact of the number of arms on the throughput–fairness trade-off for EXP3 under the heterogeneous capacity setting.**

resulting allocation does not necessarily have the highest throughput. These results demonstrate that EXP3 offers the best balance between efficiency and fairness under stable conditions in highly heterogeneous environments.

In the near-homogeneous setting (Fig. 4e), the trade-off becomes less pronounced at convergence. Due to the similarity in peer capacities, all algorithms achieve comparable throughput and fairness levels in the steady state. The performance gap between algorithms diminishes, and even the Random policy performs reasonably well. BROAD slightly improves fairness compared to other distributed algorithms, while EXP3 and Thompson Sampling remain competitive. Overall, the system naturally operates near a fair and efficient regime once convergence is reached. Finally, in the *multi-arm setting* (Fig. 4f), increasing the number of arms A improves both throughput and fairness in the steady state for EXP3. As A increases, clients can simultaneously utilize multiple peers, leading to better load distribution and higher aggregate throughput. The results show a consistent shift toward the upper-right region of the plot after convergence, indicating simultaneous gains in both metrics. This highlights that increasing decision flexibility reduces the inherent trade-off between throughput and fairness in stable network conditions.

Findings: At convergence, EXP3 achieves the best throughput–fairness balance in heterogeneous environments, while increasing the number of arms improves both metrics and effectively mitigates the trade-off between them.

C. Impact on Video Performance

PCDN architectures are increasingly adopted in short-video applications [3], [4], [5], [6], as they leverage edge IoT resources to improve scalability and reduce latency. Thus, in addition to transport statistics such as throughput and fairness, we also evaluate the impact of peer selection algorithm choice on video streaming performance. We find that video streaming favors stability over higher, less stable throughput.

Video Simulation Setting. To evaluate the algorithms studied in this paper, we design an MPEG-DASH simulator which uses an Adaptive Bitrate (ABR) Strategy to control the quality of the video as it is played back. We provide a short description of both MPEG-DASH streaming and the ABR strategy which we evaluate here below.

In MPEG-DASH streaming, a video is provided to a client in segments, short portions of that video a few seconds long. Each segment is downloaded sequentially and placed into the client's buffer for playback. Segments are made available at multiple encoding qualities (and thus multiple file sizes) so that the client can balance high quality video playback and low rebuffering ratio, i.e. the time spent not playing the video while waiting for the next segment to download. By choosing a lower quality segment, the client can speed up download time at the expense of quality and vice-versa.

The client policy used to select the quality of the next segment is known as an ABR strategy. In this evaluation, we focus primarily on BOLA [24] as our ABR strategy as it was

capable of achieving the lowest rebuffering ratios out of others evaluated while still keeping quality high.

We simulate video playback by assuming that each client chooses which peers to download from at the start of each segment, meaning that each throughput trace value represents the throughput available to the client for the next segment. The client does not know these values ahead of time.

We evaluate a standard reference video known as Big Buck Bunny [25], with a bitrate ladder matching that of the same video encoded at 60fps as hosted on YouTube. The ladder includes segments at 3, 5, 14, and 28 Mbps and the segments are 5 seconds in length each.

Video Performance. In this section we present the results of our video evaluation. We focus primarily on *rebuffering ratio*, i.e. the portion of time spent stalled downloading the next segment while not playing back the previous segment. Rebuffering ratio is computed by

$$\frac{\text{rebuffering\_time}}{\text{total\_playback\_time + rebuffering\_time}}.$$

We focus on throughput traces from the first setting, where peer bandwidths are highly heterogeneous. In the second setting, all algorithms allow BOLA to achieve both zero rebuffering and consistently high quality, picking the highest quality 97% of the time.

In the first setting, we find that rebuffering ratio requires significantly larger quality sacrifices in order to improve: Using only the lowest quality segments allows for near-zero (with the worst 5% of stalls making up 0.8%) rebuffering ratio, while using only the 14 Mbps segments resulted in a mean rebuffering ratio of 11-21%, and in some cases as high as 53%. Using BOLA, which allowed for the lowest rebuffering ratios out of all ABR strategies evaluated, mean rebuffering ratio sits between 27% (using BROAD) and 41% (using Random), while 95th percentile stall times vary between 48% (using Thompson Sampling) and 55% (using BROAD). The results are shown in detail for BOLA in Fig. 5a.

That BROAD achieves both the best average case performance and the worst worst case performance can be explained by its tendency to stick with the same peers for long periods of time. BROAD has one of the lowest average switch rates and thus provides a stable, if relatively low, bandwidth for the ABR strategy to work with. However, if a client associates with low quality peers, it is effectively stuck with them, resulting in higher worst-case rebuffering ratio.

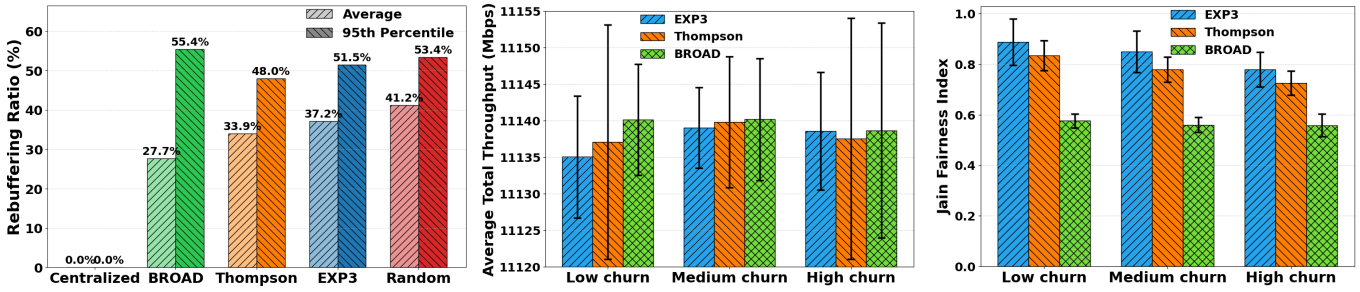
Findings: High rebuffering ratio indicates that while different algorithms can result in different performance, the core issue is buffer depletion downstream of changing client bandwidths. This instability is one of the largest hurdles to decentralizing peer selection in PCDNs, however, this effect is most pronounced when peer bandwidths differ significantly.

D. Impact of Network Dynamics

In practical PCDN systems, network conditions are inherently dynamic due to user arrivals and departures. Such dynamics introduce non-stationarity into the environment, which directly affects the robustness of online learning algorithms. In particular, client churn continuously alters the load distribution across peers, thereby changing the reward structure observed by clients over time. As a result, algorithms must not only learn an efficient allocation but also adapt to evolving system conditions. This creates a fundamental challenge: maintaining both high throughput and fairness while reacting to stochastic changes in network topology.

To capture this behavior, we model network dynamics through client churn using stochastic processes. Specifically, client arrivals are modeled as a Poisson process with rate λ, while client session durations follow an exponential distribution with mean T. This modeling choice reflects well-established assumptions in network traffic analysis, where arrivals are memoryless and session durations exhibit variability. We consider three representative churn regimes: low churn ($\lambda = 0.0025, T = 8000$), medium churn ($\lambda = 0.004, T = 5000$), and high churn ($\lambda = 0.01, T = 2000$). The choice of these parameters ensures that the expected number of active clients remains constant across all scenarios, given by $\mathbb{E}[N] = \lambda T = 20$, which is consistent with the client population considered in the stationary setting. While the average number of clients remains the same, different (λ, T) pairs result in distinct churn characteristics: low churn corresponds to infrequent arrivals with long session durations, leading to a relatively stable population, whereas high churn corresponds to frequent arrivals and short-lived sessions, resulting in rapid turnover of clients. This allows us to isolate the impact of network dynamics on algorithm performance while controlling for the average system load.

Fig. 5b illustrates the average total throughput of different algorithms under these churn conditions. From the throughput perspective, all algorithms achieve comparable average throughput across different churn levels, indicating that the system operates close to its capacity despite network dynamics. However, notable differences arise in terms of variability. BROAD and Thompson Sampling achieve slightly higher average throughput in some scenarios, but this comes at the cost of significantly larger variance—particularly for Thompson Sampling—as reflected by the error bars. This higher variability indicates less stable performance over time, likely due to their more aggressive adaptation to instantaneous reward fluctuations, which makes them sensitive to the continuous changes in client population caused by churn. In contrast, EXP3 demonstrates more consistent throughput with lower variability, highlighting its robustness to stochastic network dynamics. This stability stems from its balanced exploration–exploitation mechanism, which prevents overreaction to transient changes in peer loads. The advantage of EXP3 becomes even more pronounced when considering fairness (Fig. 5c). EXP3 consistently achieves the highest fairness across all churn scenarios, followed by Thompson Sampling. In comparison, BROAD exhibits significantly lower fairness–only slightly above 0.5 in terms of Jain fairness index–despite achieving comparable throughput. This indicates that BROAD tends to favor certain clients over others, leading to imbalanced



(a) (b) (c)

Fig. 5: **(a) Average and 95th percentile rebuffering ratio under BOLA [24]. BROAD has the lowest average but worst tail performance, reflecting its low switching behavior, while Thompson Sampling and EXP3 are more balanced. Random performs worst on average, highlighting the impact of bandwidth instability, (b) Average total throughput under different churn regimes. All methods achieve similar throughput, but EXP3 shows lower variability, while Thompson Sampling and BROAD exhibit higher variance due to sensitivity to network dynamics, (c) Jain fairness index under different churn levels. EXP3 consistently achieves the highest fairness, followed by Thompson Sampling, while BROAD shows significantly lower fairness, indicating imbalance under dynamic conditions.**

resource allocation under dynamic conditions.

Findings: Network dynamics primarily impact fairness and stability rather than average throughput. EXP3 offers the best trade-off between fairness and robustness, consistently maintaining high fairness with relatively low variability. Thompson Sampling provides competitive but slightly inferior performance, while BROAD, despite achieving high throughput, suffers from poor fairness and high variability, making it less suitable for dynamic PCDN environments.

VI. Conclusion

In this paper, we studied distributed server selection in PCDNs as a combinatorial multi-armed bandit problem. Simulations show that distributed online learning achieves performance comparable to centralized solutions while remaining scalable and robust to network dynamics. EXP3 offers the best balance of convergence, efficiency, fairness, and robustness. At the application level, video performance depends more on throughput stability than average throughput, highlighting a trade-off between adaptability and playback consistency.

VII. Acknowledgements

This work was supported in part by NSF CNS 1942305.

References

[1] U. Krishnaswamy, R. Singh, N. Bjorner, and H. Raj. Decentralized cloud wide-area network traffic engineering with blastshield. In *Proceedings of USENIX NSDI*, 2022.

[2] R. Xie, Q. Tang, S. Qiao, H. Zhu, F. Yu, and T. Huang. When serverless computing meets edge computing: Architecture, challenges, and open issues. In *IEEE Wireless Communications*, 2021.

[3] S. Srinivasan and E. Aryafar. Multipath parallel reverse segment download (mprd) for peer-to-peer content delivery. In *Proceedings of IEEE ICDCS*, 2025.

[4] D. Wei, J. Zhang, H. Li, Z. Xue, Y. Peng, X. Pang, R. Han, Y. Ma, and J. Li. Swarm: Cost-efficient video content distribution with a peer-to-peer system. In *arXiv:2401.15839*, 2024.

[5] D. Wei, J. Zhang, H. Li, Z. Xue, Y. Peng, X. Pang, Y. Liu, R. Han, and J. Li. Pscheduler: Qoe-enhanced multipath scheduler for video services in large-scale peer-to-peer cdns. In *Proceedings of IEEE INFOCOM*, 2024.

[6] R. Zhang, H. Wang, S. Shi, X. Pang, Y. Peng, Z. Xue, and J. Liu. Enhancing resource management of the world's largest pcdn system for on-demand video streaming. In *Proceedings of USENIX ATC*, 2024.

[7] B. Atalar and C. Joe-Wong. Neural combinatorial clustered bandits for recommendation systems. In *Proceedings of AAAI*, 2025.

[8] T. Kim, J. Zuo, X. Zhang, and C. Joe-Wong. Edge-msl: Split learning on the mobile edge via multi-armed bandits. In *Proceedings of IEEE INFOCOM*, 2024.

[9] X. Liu, J. Zuo, H. Xie, C. Joe-Wong, and J. C. S. Lui. Variance-adaptive algorithm for probabilistic maximum coverage bandits with general feedback. In *Proceedings of IEEE INFOCOM*, 2023.

[10] X. Liu, J. Zuo, S. Wang, C. Joe-Wong, J. C. S. Lui, and W. Chen. Batch-size independent regret bounds for combinatorial semi-bandits with probabilistically triggered arms or independent arms. In *Proceedings of NeurIPS*, 2022.

[11] O. Lünsdorf. Simpy: Discrete-event simulation for python, November 2023. PyPI release 4.1.1.

[12] D. Russo, B. V. Roy, A. Kazerouni, I. Osband, and Z. Wen. A tutorial on thompson sampling. In *Foundations and Trends in Machine Learning*, 2018.

[13] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The non-stochastic multi-armed bandit problem. In *SIAM Journal on Computing*, 2002.

[14] C. Wei and H. Luo. More adaptive algorithms for adversarial bandits. In *Proceedings of Machine Learning Research*, 2018.

[15] Akamai technologies. https://en.wikipedia.org/wiki/Akamai_Technologies.

[16] Cloudflare inc. https://en.wikipedia.org/wiki/Cloudflare.

[17] B. Zolfaghari, G. Srivastava, S. Roy, H. R. Nemati, F. Afghah, T. Koshiba, A. Razi, K. Bibak, P. Mitra, and B. Rai. Content delivery networks: State of the art, trends, and future roadmap. In *ACM Computing Surveys*, 2020.

[18] R. Farahani, H. Amirpour, F. Tashtarian, A. Bentaleb, C. Timmerer, H. Hellwagner, and R. Zimmermann. Richter: hybrid p2p-cdn architecture for low latency live video streaming. In *Proceedings of ACM MHV*, 2022.

[19] R. Farahani, A. Bentaleb, E. Centinkaya, C. Timmerer, R. Zimmermann, and H. Hellwagner. Hybrid p2p-cdn architecture for live video streaming: An online learning approach. In *Proceedings of IEEE GLOBECOM*, 2022.

[20] S. Agrawal and N. Goyal. Thompson sampling for contextual bandits with linear payoffs. In *Proceedings of Machine Learning Research*, 2013.

[21] W. Chen, Y. Wang, and Y. Yuan. Combinatorial multi-armed bandit: General framework and applications. In *Proceedings of Machine Learning Research*, 2013.

[22] N. Cesa-Bianchi and G. Lugosi. Combinatorial bandits. In *Journal of Computer and System Sciences*, 2012.

[23] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of machine learning research*, 2002.

[24] K. Spiteri, R. Urgaonkar, and R.K. Sitaraman. Bola: Near-optimal bitrate adaptation for online videos. *IEEE/ACM Transactions on Networking*, 2020.

[25] Big buck bunny, 2009. https://peach.blender.org/.