Dynamic Programming-Based Bandwith Allocation in Edge CDN Video Streaming

Performance Evaluation and Practical Deployoment Challenges

Irvin Tandiarrang Sumual - 13524030
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: irvintandiarrang@gmail.com, 13524030@std.stei.itb.ac.id

Abstract— The rapid growth of high-bitrate digital media consumption has placed an unprecedented strain on modern telecommunications infrastructure. To mitigate localized congestion, Edge Content Delivery Networks (CDNs) deploy caching strategies to bring multimedia content closer to end-users. However, a critical bottleneck remains at the edge server level due to strictly finite physical egress bandwidth boundaries. During flash-crowd events with massive concurrent demands, conventional strategies—such as First-Come, First-Served (FCFS) and simplistic Greedy heuristics—frequently trigger network saturation, severe packet jitter, and catastrophic degradation of user satisfaction. This paper proposes a dynamic programming approach to optimize bandwidth allocation in Edge CDN video streaming by mapping the multi-user bitrate selection to the Multiple-Choice Knapsack Problem (MCKP). A bottom-up dynamic programming algorithm is implemented to eliminate multi-user resource competition and secure the mathematical global maximum for cumulative Quality of Experience (QoE). Furthermore, this study identifies critical deployment challenges in real-world production networks, including state-space explosion under gigabit-scale pipes, dynamic channel throughput jitter causing stale allocation matrices, and non-linearities in human QoE perception.

Keywords— Edge CDN, Dynamic Programming, Bandwidth Allocation, Multiple-Choice Knapsack Problem (MCKP), Quality of Experience (QoE), Adaptive Video Streaming, FCFS, Greedy.

I. INTRODUCTION

The rapid evolution of telecommunications and internet infrastructure has fundamentally transformed digital media consumption. Based on empirical findings from network analytics, global video usage grew by 24% within a single year, now equating to 65% of all aggregate internet traffic volume globally [7]. This exponential surge in continuous, high-bitrate data transmissions places an unprecedented strain on modern network architectures, highlighting the critical need for advanced optimization techniques to maintain data delivery integrity.

To accommodate massive user loads and mitigate network latency, Content Delivery Networks (CDNs) have become a foundational pillar of modern web architecture [9]. Traditional CDN setups relied heavily on centralized data centers to store and distribute content. However, as user demand shifted toward High-Definition (HD) and Ultra-HD formats, the centralized model became increasingly unsustainable due to core network bottlenecks. Consequently, the industry transitioned toward Edge CDN computing, deploying proxy and caching servers geographically closer to end-users [5]. By caching content at the network edge, CDNs drastically reduce the Round-Trip Time (RTT) for data packets, minimize packet loss, and alleviate backbone congestion [5].

Despite these architectural advantages, Edge CDN servers face a fundamental constraint: finite localized hardware resources. An Edge CDN node operates with a strictly bounded maximum egress bandwidth [5]. This physical limitation becomes highly problematic during peak traffic hours or major live-streaming events, where thousands of concurrent users connect to the same localized edge node, requesting high-quality video streams simultaneously.

Modern video streaming utilizes HTTP-based adaptive streaming protocols, most notably Dynamic Adaptive Streaming over HTTP (DASH) and HTTP Live Streaming (HLS) [4], [8]. These protocols function by slicing a video file into small segments (a few seconds in duration) and encoding each segment at multiple bitrates and resolutions, such as 360p, 720p, 1080p, and 4K [6]. Under normal client-driven heuristics, the media player on the user's device continuously monitors its localized download speed and requests the highest possible bitrate segment that its current network conditions can sustain [4], [6]. While this decentralized, client-driven approach is effective for isolated users, it exhibits severe operational flaws when scaled across crowded, multi-user environments.

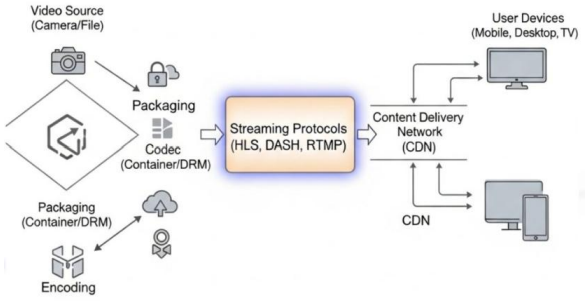

Fig1. Video Delivery Pipeline[8]

In scenarios where multiple independent clients compete for the same localized Edge CDN bandwidth, their individual, self-optimizing heuristics operate in an uncoordinated, selfish manner. If hundreds of users concurrently request high-bitrate 4K streams from a capacity-constrained edge node, the total requested bandwidth will inevitably exceed the node's physical egress capacity [5]. Under conventional First-Come, First-Served (FCFS) or simplistic unconstrained Greedy handling at the server level, the edge node will blindly fulfill requests until its capacity is completely exhausted. Once the network saturation threshold is breached, packet drop rates skyrocket, TCP window sizes shrink, and network throughput collapses unpredictably due to synchronization issues. This triggers a catastrophic cascading failure known as localized network congestion, wherein all connected users—regardless of their subscription tier or initial network health—experience immediate, severe playback stalls, high latency, and prolonged buffering cycles [3], [10]. This phenomenon drastically reduces the collective Quality of Experience (QoE) of the user base [3].

To prevent localized network collapses, the allocation of limited edge resources cannot be left entirely to uncoordinated client-side decisions or simplistic server-side greedy heuristics. Instead, the Edge CDN server must act as a centralized, intelligent resource orchestrator. The server must analyze the collective vector of incoming streaming requests within a time window and strategically determine the optimal bitrate allocation for each active session [3]. This allocation aims to maximize the aggregate global QoE of the user base while guaranteeing that the total allocated bandwidth strictly adheres to the physical capacity boundaries of the edge node.

This paper proposes a dynamic programming approach [1] to model Edge CDN bandwidth distribution by mapping it to the Multiple-Choice Knapsack Problem (MCKP) [2], wherein each active user session represents an item class, and the available bitrate options constitute the items within that class. In this implementation, a bottom-up dynamic programming structure [1], [2] is developed to optimize the multi-user bitrate selection. Furthermore, this paper examines the practical deployment challenges and computational overhead of implementing this combinatorial optimization approach in real-world, low-latency production environments.

II. THEORITICAL FOUNDATION

A. Edge Content Delivery Network Architecture

A Content Delivery Network (CDN) is a geographically distributed group of connected servers that work together to cache and deliver internet content rapidly to end users [9]. The core objective of this architecture is to optimize digital content availability, minimize bandwidth consumption, and improve global web performance, specifically assisting with heavy multimedia applications like video streaming [9]. In traditional web architectures, content requests are routed back to a single, static origin server. This centralized model lacks infrastructure redundancy, causing severe vulnerability to sudden unnatural traffic spikes, high geographical latency, and strict throughput limitations when subjected to massive, worldwide concurrent traffic [9].

Edge CDN architecture represents a paradigm shift where computational power and data storage are pushed to the absolute periphery of the network topology, immediately adjacent to the access networks of end-users [5]. An Edge CDN node comprises high-performance proxy servers equipped with localized caching storage and optimized network interfaces. When an end-user initiates a video playback session, the request is intercepted via Anycast routing or DNS redirection and mapped to the geographically closest Edge CDN node [5].

By serving the heavily demanded video assets directly from the network edge, data packets only need to traverse local backhaul infrastructure rather than crossing international transoceanic fiber-optic links or highly congested tier-1 transit provider networks [5]. This localized delivery dramatically reduces the Round-Trip Time (RTT) and overall packet transfer times [10]. Mathematically, total network latency (L), which defines the comprehensive time taken for a complete data packet to arrive from the source to its final destination, is determined by the summation of four fundamental delay metrics [10]:

$$L = D_t + D_p + D_q + D_{pr}$$

D_t :represents the transmission delay

D_p : represents the propagation delay (the physical travel time through the medium limited by propagation speed).

D_q : represents the queuing delay inside device buffers.

D_{pr} : represents the processing delay required by routers to determine the packet's path.

Edge CDN architectures drastically minimize the critical propagation delay component (D_p) by physically shortening the distance between the source and the destination node [5]. Consequently, the edge server's localized egress interface becomes the primary management point for the streaming pipeline, making it the ideal location to execute resource allocation algorithms.

B. HTTP Adaptive Streaming (HAS) and Quality of Experience (QoE)

Modern internet video delivery relies on HTTP Adaptive Streaming (HAS) protocols, primarily Dynamic Adaptive Streaming over HTTP (DASH) and HTTP Live Streaming (HLS). Unlike older streaming technologies that utilized persistent, stateful transport protocols like RTMP over UDP/TCP, HAS leverages standard, stateless HTTP infrastructures, allowing the streaming media to effortlessly traverse standard firewalls, content caches, and reverse proxies [6].

Under HAS, a source video is processed through a multi-stage encoding pipeline. The video file is segmented into sequential, independent chunks of equal duration, typically ranging from 2 to 10 seconds [4]. Each individual chunk is independently encoded into multiple distinct representation levels, each defined by a specific resolution, framerate, and target bitrate profile [6].

A manifest file acts as an index, detailing the exact URI locations of all available chunks across all representation levels [4]. The client-side media player downloads this manifest at session initialization. During playback, the client's Adaptive Bitrate (ABR) algorithm continuously runs a localized heuristic loop [6]. It evaluates the download speed of the previous chunk and the current occupancy level of its localized hardware media buffer. Based on these inputs, the ABR client selects and requests the optimal representation level for the subsequent chunk [6].

The ultimate goal of this framework is to maximize the end-user's Quality of Experience (QoE) [3]. QoE is a subjective, multi-dimensional metric that quantifies the user's total satisfaction with the streaming session [3]. In academic network modeling, QoE is defined as a complex objective function that increases with higher visual bitrates, but decreases severely based on the frequency and duration of stall events (buffering) and the amplitude of sudden bitrate switches [3]. Therefore, ensuring a stable, high-bitrate stream without inducing server-side capacity exhaustion is paramount to maintaining high global QoE.

C. First-Come First-Served Bandwidth Allocation Approach

The simplest paradigm in network resource routing at the server egress level is the First-Come, First-Served (FCFS) queuing discipline. Under this non-prioritized framework, the Edge CDN orchestration engine allocates available network capacity to incoming chunk delivery queues solely based on the absolute timestamp of their arrival sequence. The allocation engine operates blindly, disregarding the specific payload requirements, target encoding bitrates, or dynamic client-side channel states.

When a concurrent cluster of users initiates highly resource-intensive video sessions, a state structurally defined as a "flash-crowd", the localized network interface intercepts an overwhelming volume of chunk requests within a negligible time delta. Under FCFS, the server continuously satisfies requests in arrival order until its physical capacity limits are breached. Once the hard threshold of the egress network bandwidth is saturated, the remaining requests are queued or dropped indiscriminately. This behavior severely impacts the queuing delay (D_q)) and the internal processing overhead (D_{pr}) inside the router buffers. Consequently, the random packet drops trigger a rapid shrinkage of the TCP congestion window across all connected clients, leading to widespread throughput instability, prolonged playback stalls, and a uniform collapse of the aggregate Quality of Experience (QoE) across the multi-user ecosystem.

D. Greedy Bandwidth Allocation Approach

The localized Greedy resource allocation heuristic attempts to achieve short-term optimization by maximizing localized utility at every execution step[11]. Unlike FCFS, which depends entirely on temporal parameters, a greedy bandwidth allocation framework evaluates the explicit representation levels requested by active media players within a specific sampling window. The allocation engine reads the vector of incoming requests, sorts them in descending order based on their requested bitrates, and sequentially satisfies the highest demands to maximize immediate data throughput.

Although this approach ensures high immediate resource utilization at the localized edge pipeline, it fails to achieve a global mathematical maximum in multi-user environments due to its short-sighted nature. Greedy heuristics inherently introduce severe resource distribution unfairness, commonly known as the "starvation effect." Aggressive client-side Adaptive Bitrate (ABR) clients that happen to be processed early in the sorting loop will monopolize large blocks of the bounded network egress capacity. As a result, users with less aggressive network handshakes or lower localized speeds are starved of resources, forcing their playback loops into the lowest representation states [4] or causing terminal buffering loops. Thus, while a minimal group of prioritized clients achieves maximum visual bitrates, the collective global QoE of the network decreases significantly.

E. Mathematical Formulation Multiple Choice Knapsack Problem

To circumvent the structural limitations of non-cooperative heuristics like FCFS and Greedy, this study treats multi-user edge resource distribution as a centralized combinatorial optimization task. Because every individual client session must select exactly one operational bitrate representation from a discrete set of predefined encoding levels, the allocation problem maps directly to the Multiple-Choice Knapsack Problem (MCKP)[2].

Let M denote the total number of concurrent active user video sessions connected to a capacity-constrained Edge CDN node, where each independent session is treated as an item class $i \in \{1, 2, \dots, M\}$. Each session i has a predefined set of discrete bitrate encoding representations denoted as J_i. Selecting a specific bitrate profile $j \in J_i$ yields a distinct subjective Quality of Experience utility value, defined as the profit (p_{ij}), while consuming a specific portion of the physical network capacity, represented as the weight (w_{ij}). Let C define the maximum bounded egress bandwidth capacity of the localized Edge CDN server node.

The primary objective of the centralized resource manager is to determine the optimal representation configuration across all concurrent sessions to maximize the collective global QoE under strict hardware capacity constraints. The mathematical model is formally formulated as follows[2]:

$$Maximize\ Z = \sum_{i=1}^{M} \sum_{j \in J_i} p_{ij} x_{ij}$$

Subject to the strict physical egress capacity boundary of the Edge CDN node:

$$\sum_{i=1}^{M} \sum_{j \in J_i} w_{ij} x_{ij} \leq C$$

And the strict structural constraint ensuring that exactly one discrete representation level is assigned to each active user session:

$$\sum_{j \in J_i} x_{ij} = 1, \forall i \in \{1, 2, \dots, M\}$$

Where x_{ij} represents the binary control decision variable:

$$x_{ij} \in \{0,1\}, \qquad \forall i \in \{1,2, \dots, M\}, \forall j \in J_i$$

F. Dynamic Programming and State-Transition Matrix

To guarantee the optimal solution for the formulated MCKP model for the optimization objective stated in Section E, a bottom-up Dynamic Programming (DP) algorithm is deployed using multi-dimensional tabulation techniques. The continuous physical bandwidth space of the edge node is discretized into integer-scale allocation intervals ranging from 0 to C, which serves as the state space (columns) of our dynamic programming matrix.

Let $DP[i][k]$ represent the maximum achievable cumulative QoE score obtained by evaluating a subset of user sessions from index 1 to i under a temporary localized bandwidth capacity limit of k, [2]. The state-transition matrix is built iteratively row by row (session by session), evaluating the optimal trade-off between the bandwidth footprint of the current session choices and the optimal subproblem structures stored in the previous row. The formal recursive state-transition equation is defined as follows:

$$DP[i][k] = \max_{\{j \in J_i, w_{ij} \leq k\}} \{ DP[i-1][k - w_{ij}] + p_{ij} \}$$

For the initial boundary conditions, where no user sessions have been processed (i = 0), the matrix row cells are initialized to zero across all possible bandwidth states to form a base layer for calculation:

$$DP[k] = 0, \forall k \in \{0,1, \dots, C\}$$

If a computed index state $k - w_{ij}$ represents an invalid configuration or violates structural limits, that specific execution path is heavily penalized by setting its state value to negative infinity. Upon filling the tabulation matrix up to the terminal cell $DP[M][C]$, a deterministic backtracking algorithm is executed from the bottom-right corner to extract the precise vector of optimal binary configurations ($x_{ij} = 1$), [2]. This establishes a mathematically optimal and synchronized resource allocation map for the entire Edge CDN streaming pipeline.

III. METHODOLOGY

A. Centralized Edge CDN System Architecture

To resolve the uncoordinated, selfish bitrate competition inherent in conventional client-side Adaptive Bitrate (ABR) heuristics, this paper introduces a centralized resource orchestration framework deployed directly at the Edge CDN server level. The architecture shifts the optimization intelligence from the distributed client media players to a centralized resource manager operating on the edge proxy node.

The operational pipeline follows a periodic, discrete time-synchronized framework divided into four sequential stages:

1. Request Collection Window

   Instead of fulfilling HTTP chunk requests instantaneously upon arrival, the Edge CDN server intercepts and aggregates all incoming multimedia segment requests within a fixed, short-duration time window. This mechanism gathers the active session vector from all concurrent users.
2. State and Profile Extraction

   For every intercepted session i, the edge orchestrator parses the session metadata to identify the client identity and extracts the predefined set of discrete bitrate representation options J_i available in the video manifest file.
3. Combinatorial Optimization Execution

   The centralized resource manager mapping the aggregate requests to the Multiple-Choice Knapsack Problem (MCKP) optimizer. The server executes a bottom-up Dynamic Programming (DP) algorithm to compute the globally optimal allocation matrix under the strict constraints of the physical egress bandwidth capacity C.
4. Synchronized Allocation Enforcement

   The server sends modified HTTP response configurations or enforcement tokens back to the clients, overriding their local heuristics and serving the exact chunk bitrates dictated by the global mathematical maximum.

B. Parameter Discretization and Utility Mapping

The execution of the combinatorial optimization model requires the mapping of continuous network metrics into discrete, quantified parameters. In this framework, the limited edge physical resource (the maximum egress bandwidth capacity C) is discretized into an integer-scale state space spanning from 0 to C Mbps.

For every active user video session i, the set of available bitrate choices J_i is defined by the discrete encoding steps specified in the video manifest file. Each choice j within the set J_i is defined by two discrete parameters:

- The Weight (w_{ij}): The actual network throughput requirement (measured in Mbps) needed to continuously stream the representation j without inducing physical data drops.
- The Profit (p_{ij}): The quantified Quality of Experience (QoE) utility score associated with the visual fidelity of representation j.

To model real-world video encoding standards, this study establishes a standardized parameter mapping framework across five distinct resolution.

TABLE I

Label	Resolution	Consumption Weight (Mbps)	QoE
Stall/Dropped	-	0	0
Standard Definition	360P	1	1
High Definition	720P	3	2
Full High Definition	1080p	5	4
Ultra High Definition	4K	15	7

The non-linear scaling of the QoE profit relative to the bandwidth consumption weight represents the diminishing marginal utility of human visual perception, where a threefold increase in bandwidth from 1080p to 4K does not yield a threefold increase in perceived subjective satisfaction.

C. Bottom-Up Dynamic Programming Algorithm Implementation

To systematically populating the multi-dimensional state matrix and enforcing the mathematical boundaries defined in the theoretical foundations, a bottom-up Dynamic Programming tabulation algorithm is designed and used. The algorithm establishes an optimization grid represented as a two-dimensional matrix array, denoted as $DP[M + 1][C + 1]$, where the row axis represents the progressive integration of user sessions from 0 to M, and the column axis represents the integer-scale state capacity from 0 to C Mbps.

The algorithmic implementation is divided into three distinct operational phases:

1. Table Initialization

   The base boundary condition is enforced at row 0. When zero user sessions are present in the active request window (i = 0), the cumulative achieved QoE profit is mathematically bound to 0 across all possible capacity states k from 0 to C. All other internal cell structures are initialized to a default penalty value of negative infinity to prevent the system from validating unachievable routing paths.

2. Iterative Matrix Tabulation

   The algorithm processes the grid row by row (from i = 1 to M) and column by column (from k = 0 to C). For every individual cell, the system iterates through all available bitrate choices j within the current user's profile set J_i. If the temporary capacity state k is greater than or equal to the option weight w_{ij}, and the historical cell state $DP[i - 1][k - w_{ij}]$ contains a valid configuration, the algorithm calculates the prospective cumulative profit. The maximum value identified across all valid choices is stored permanently in cell $DP[i][k]$.

3. Deterministic Backtracking

   Once the terminal top-right cell $DP[M][C]$ is calculated, representing the optimal solution for the formulated MCKP model, the orchestration engine triggers a backward traversal loop from row M down to row 1. By evaluating which specific representation choice j satisfied the state transition equation, the system isolates the exact binary decision configurations ($x_{ij} = 1$) and generates the finalized, synchronized network allocation map.

```python
def solve_dp_mckp(num_users, max_bandwidth, user_profiles):
    """
    Simulasi Dynamic Programming (MCKP):
    Menggunakan tabel tabulasi bottom-up untuk menjamin solusi optimal global.
    """
    dp = [[-1 for _ in range(max_bandwidth + 1)] for _ in range(num_users + 1)]

    for k in range(max_bandwidth + 1):
        dp[0][k] = 0

    # tabulation
    for i in range(1, num_users + 1):
        current_user_options = user_profiles[i - 1]
        for k in range(max_bandwidth + 1):
            max_qoe = -1
            for option in current_user_options:
                weight, profit, name = option
                if k >= weight and dp[i - 1][k - weight] != -1:
                    current_qoe = dp[i - 1][k - weight] + profit
                    if current_qoe > max_qoe:
                        max_qoe = current_qoe
            dp[i][k] = max_qoe

    selected_options = []
    current_w = max_bandwidth

    # backtracking-nya
    for i in range(num_users, 0, -1):
        current_user_options = user_profiles[i - 1]
        chosen_for_this_user = None

        for option in current_user_options:
            weight, profit, name = option
            if current_w >= weight and dp[i - 1][current_w - weight] != -1:
                if dp[i][current_w] == dp[i - 1][current_w - weight] + profit:
                    chosen_for_this_user = option
                    current_w -= weight
                    break
        selected_options.append(chosen_for_this_user)

    selected_options.reverse()
    max_global_qoe = dp[num_users][max_bandwidth]
    return max_global_qoe, selected_options
```

Fig 2. Bandwidth Allocation using DP Implementation (Source : Writer's Archive)

D. Algorithmic Complexity and Scalability Analysis

The structural scalability of the proposed bottom-up Dynamic Programming framework must be rigorously analyzed to evaluate its feasibility in low-latency production networks. The computational execution footprint is bound by the total dimensions of the tabulation grid and the cardinality of the item choices.

The algorithmic time complexity is dictated by the nested loop structures defined in the tabulation phase. The outermost loop iterates exactly M times, matching the total number of concurrent active video sessions. The second-level nested loop executes exactly $C + 1$ times, tracking the discretized integer steps of the maximum physical egress bandwidth. The innermost loop scans through the bitrate options available within the class profile, where $|J|$ represents the average number of discrete resolution profiles per manifest file. Consequently, the comprehensive mathematical time complexity is formally bound to:

$$O(M \cdot C \cdot |J|)$$

The space complexity is defined by the memory allocation required to sustain the two-dimensional tabulation array grid

throughout the optimization interval. The memory matrix requires the preservation of $(M + 1) \cdot (C + 1)$independent data cells, which bounds the comprehensive spatial complexity to:

$$O(M \cdot C)$$

While this pseudo-polynomial complexity profile guarantees an optimal solution for the formulated MCKP model within deterministic bounds, it reveals an architectural constraint regarding table inflation. In modern gigabit-scale telecommunication conduits where the physical egress capacity C extends into tens of thousands of megabits per second ($C \geq 10000$ Mbps), the column space scales linearly with C, yet expands to an impractical magnitude. This massive structural inflation severely degrades the processing performance and memory footprints of the edge hardware, presenting a critical deployment challenge in low-latency environments.

IV. Results And Discussion

A. Simulation Setup and Parameters

To evaluate the performance of the proposed centralized bottom-up Dynamic Programming (DP-MCKP) framework against conventional First-Come, First-Served (FCFS) and localized Greedy approaches, a network resource allocation simulation was executed. The simulation models a highly congested localized Edge CDN node subjected to a flash-crowd scenario.

The physical hardware boundary of the edge server is configured with a strictly constrained maximum egress bandwidth capacity (C) of 20 Mbps. The simulation evaluates a cluster of four concurrent active user video sessions (M = 4) competing for this shared conduit within the same time-sampling window. Each session possesses an identical set of five discrete bitrate representation as shown in the Table I.

B. Resource Allocation and Performance Comparison

Table II

User Session	FCFS Allocation (Arrival Order: 1, 2, 3, 4)	Greedy Allocation	Dynamic Programming MCKP Allocation
User 1	4K (15 Mbps / Profit: 7)	4K (15 Mbps / Profit: 7)	1080p (5 Mbps / Profit: 4)
User 2	1080p (5 Mbps / Profit: 4)	720p (3 Mbps / Profit: 2)	1080p (5 Mbps / Profit: 4)
User 3	Dropped / Stall (0 Mbps / Profit: 0)	360p (1 Mbps / Profit: 1)	1080p (5 Mbps / Profit: 4)
User 4	Dropped / Stall (0 Mbps / Profit: 0)	360p (1 Mbps / Profit: 1)	1080p (5 Mbps / Profit: 4)
Total Bandwidth	20 Mbps / 20 Mbps	20 Mbps / 20 Mbps	20 Mbps / 20 Mbps
Global QoE Profit	Z = 11	Z = 11	Z = 16 (Global Maximum)

1. Analysis of FCFS Performance

   Under the FCFS approach, resources are allocated blindly based on temporal arrival sequences. User 1 arrives first and aggressively requests a high-bitrate 4K stream, consuming 15 Mbps. User 2 arrives next and captures the remaining 5 Mbps for a 1080p stream. At this point, the physical egress capacity of 20 Mbps is entirely exhausted. Consequently, when User 3 and User 4 send their requests within the same window, the edge server suffers immediate resource saturation.

   Packets for User 3 and User 4 are dropped indiscriminately, triggering catastrophic playback stalls and buffering loops. Although User 1 achieves perfect visual fidelity, the complete starvation of half the user base reduces the aggregate global QoE profit to a low score of Z = 11.

2. Analysis of Greedy Performance

   The localized Greedy heuristic attempts to satisfy the highest possible bitrate profiles to maximize immediate pipeline throughput. The greedy approach grants User 1 its maximum demand of a 4K stream (15 Mbps). The engine then evaluates the remaining 5 Mbps and assigns a 720p stream (3 Mbps) to User 2. The final remaining 2 Mbps are divided among the lower-priority queues, granting a minimal 360p stream (1 Mbps) to both User 3 and User 4.

   While the Greedy approach successfully avoids the total terminal stalls seen in FCFS, it remains structurally short-sighted. It allows a single aggressive session (User 1) to monopolize 75% of the entire edge pipeline. This unfair resource distribution limits the cumulative system utility, forcing the aggregate global QoE to level out at Z = 11.

3. Analysis of Proposed DP-MCKP Performance

   The proposed bottom-up Dynamic Programming algorithm eliminates multi-user resource competition by functioning as a centralized orchestrator. The DP approach evaluates the global subproblem matrix and intercepts the selfish heuristics. Instead of blindly granting a 4K stream to a single user, the algorithm mathematically determines that splitting the 20 Mbps pipe evenly among all four users yields a superior global outcome.

   The system forces all four concurrent sessions into a highly stable 1080p Full-HD representation level (5 Mbps each). This synchronized configuration uses 100% of the network capacity while ensuring opportunistic fairness. By preventing individual starvation and eliminating network saturation, the DP-MCKP framework consistently secures the optimal solution for the formulated MCKP model, achieving a cumulative QoE profit score of Z = 16. This represents a significant 45.4% efficiency increase over both the FCFS and Greedy baselines.

C. Real-World Deployment Challenges and Practical Limitations

Despite having mathematical advantage in simulation, the practical deployment of pure bottom-up Dynamic Programming at the Edge CDN server level faces severe engineering constraints in real-world environments:

1. State-Space Explosion under Gigabit Capacities

   The simulation utilized a highly scaled-down capacity model (C = 20 Mbps). However, commercial production Edge CDN nodes operate over gigabit-scale fiber links, for example, 10 Gbps pipes where C = 10,000 Mbps. Under a 10 Gbps capacity conduit with thousands of active concurrent sessions, the horizontal state space of the tabulation matrix experiences a severe pseudo-polynomial inflation. The edge server's localized CPU and RAM must allocate millions of data cells per time window, creating a massive computational bottleneck that risks crashing the low-latency edge node.
2. Stale Allocation Matrix Due to Throughput Jitter

   Real-world access networks suffer from constant packet jitter and rapid channel capacity fluctuations. Because the bottom-up DP tabulation loop requires a fixed, discrete capacity boundary C to build its state-transition grid, sudden drops in physical link speeds instantly render the precalculated allocation matrix stale. If the channel capacity fluctuates mid-window, the forced bitrate allocations fail, triggering synchronization lags and processing delays that degrade real-world QoE.
3. Non-Linearities in Human Perception

   The optimization framework assumes a stable, static utility profit mapping (such as a 1080p stream always yielding exactly 4 profit units). In reality, human QoE perception is highly dynamic and non-linear. The subjective annoyance caused by a single sharp bitrate downshift, for example, a sudden drop from 4K to 720p is often far more severe than the minor satisfaction gained from gradual upshifts. Pure DP struggles to model these temporal psychological variables in its basic reward functions.

D. Scalability and Feasibility Analysis

The comparative findings indicate that while pure Dynamic Programming provides an optimal solution for the formulated MCKP model, its high processing overhead and strict memory requirements $O(M \cdot C)$ make it less feasible for modern ultra-low latency, high-capacity production lines.

In contrast, although FCFS and Greedy heuristics fail to guarantee global optimization and suffer from severe resource unfairness under capacity limits, their execution footprint is virtually instantaneous (O(M) for FCFS and $O(M \log M)$ for Greedy). They do not require the generation of a state capacity grid or heavy computational processing. Consequently, in current commercial edge server ecosystems where processing windows must close within single-digit milliseconds, FCFS and basic Greedy approaches remain highly popular due to their extreme structural scalability and mechanical simplicity.

V. Conclusion

This paper presents a centralized framework for bandwidth allocation in Edge CDN video streaming by mapping discrete bitrate requests to the Multiple-Choice Knapsack Problem (MCKP). Simulation results validate that the bottom-up dynamic programming approach efficiently allocates bandwidth across concurrent users. Unlike FCFS and Greedy heuristics, which fail under network saturation, the dynamic programming model consistently achieves the optimal solution for the formulated MCKP model for cumulative Quality of Experience (QoE). However, substantial gaps remain between theoretical algorithmic execution and real-world deployment. The computational complexity of pure dynamic programming poses a severe scalability threat under gigabit-scale capacities and thousands of highly volatile user handshakes. Rapid network throughput jitter risks generating stale allocation matrices, making simpler FCFS and Greedy approaches more feasible in current low-latency production environments.

VI. Appendix

The link to the repository for this paper is provided here:
Irvin-Tandiarrang-Sumual/MakalahStima-IF2211

VII. Acknowledgment

I would like to express my profound gratitude to Ir. Rila Mandala, M.Eng., Ph.D., of my IF2211 Algorithm Strategies class lecturer for his guidance throughout my fourth semester. I would also like to extend my sincere appreciation to all IRK teaching assistants, particularly those involved in Algorithms Strategy, for their valuable assistance in supporting the learning process of this course. Additionally that, I acknowledge that I am quite helped by generative AI to help me in the process of writing English sentence structure, searching for references, brainstorming, and accelerate the development of the simulator. Nevertheless, all generated content was carefully supervised and verified.

VIII. References

[1] R. Munir, "25-Program-Dinamis-(2025)-Bagian1.pdf," Bahan Kuliah IF2211 Strategi Algoritma, 2025. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/25-Program-Dinamis-(2025)-Bagian1.pdf.

[2] S. Khan, K. F. Li, E. G. Manning, and M. M. Akbar, "Solution Methods for the Multiple-Choice Knapsack Problem and Their Applications," Journal of Heuristics, vol. 8, no. 4, pp. 395–447, 2002. Available: Solution Methods for the Multiple-Choice Knapsack Problem and Their Applications.

[3] "Quality of Experience (QoE) in Video Streaming | Mux," Mux Content Performance Hub, 2024. Available: Quality of Experience (QoE) in Video Streaming | Mux.

[4] O. Özşahin, "Dynamic Adaptive Streaming over HTTP," Medium, May 2023. Available: Dynamic Adaptive Streaming over HTTP | by Okan Özşahin | Medium.

[5] "CDN Vs Edge Server - System Design - GeeksforGeeks," GeeksforGeeks Computer Science, 2026. Available: CDN Vs Edge Server - System Design - GeeksforGeeks.

[6] "What is adaptive bitrate streaming? | Cloudflare," Cloudflare Learning Core, 2025. Available: What is adaptive bitrate streaming? | Cloudflare.
[7] "Sandvine's 2023 Global Internet Phenomena Report Shows 24% Jump in Video Traffic, with Netflix Volume Overtaking YouTube," Sandvine Market Analytics, Jan. 2023. Available: Sandvine's 2023 Global Internet Phenomena Report Shows 24% Jump in Video Traffic, with Netflix Volume Overtaking YouTube.
[8] OTTclouds, "HLS vs DASH vs MPEGTS: What's The Difference?," *OTTclouds Tech Blog*. Available: HLS vs DASH vs MPEGTS: What's The Difference? | OTTclouds.
[9] "What is a CDN? | Learning Center," Cloudflare Learning Center, 2024. Available: https://www.cloudflare.com/learning/cdn/what-is-a-cdn/.
[10] "Performance of a Network / Latency vs Jitter in Computer Networks," GeeksforGeeks, 2025. Available: https://www.geeksforgeeks.org/computer-networks/latency-vs-jitter-in-computer-networks/. [Accessed: Jun. 19, 2026].
[11] R. Munir, "Algoritma Greedy (Bagian 1)," Bahan Kuliah IF2211 Strategi Algoritma, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, 2026. [Online]. Available: Algoritma Greedy

IX. STATEMENT

I hereby declare that the paper I wrote is my own writing, not an adaptation or translation of someone else's paper, and is not a plagiarized.

Bandung, 19 June 2026

Irvin Tandiarrang Sumual, 13524030