ChatGPT

Feasibility of a Blockchain-Integrated Mobile P2P Streaming Platform

This report validates and expands a feasibility study for a blockchain-enabled, mobile-first peer-to-peer (P2P) streaming platform. We cross-reference recent literature and benchmarks (e.g. Livepeer, Theta, WebRTC, Arbitrum) to confirm technical claims, then deepen the analysis of hybrid P2P-CDN architectures, mobile constraints, and edge computing. We expand the technical architecture discussion (fault tolerance, peer orchestration, PWA vs native apps, WebAssembly) and strengthen the blockchain layer (Layer-2/3 scalability, cross-chain media use). Monetization is elaborated via tokenomics modeling, watch-to-earn sustainability, and creator payout scenarios. We detail three innovation concepts (AI-optimized mesh network, gaming-linked economy, micro-CDN) with implementation roadmaps, platform examples, and readiness. Finally, we propose a phased implementation roadmap (with timeline, deliverables, dependencies).

1. Technical Viability & Updated Benchmarks

Recent studies confirm that **hybrid P2P-CDN streaming** can deliver high QoE and low latency at scale. For example, a cloud-based test with 350 DASH clients showed that a hybrid P2P-CDN system achieved *"high users' QoE, low latency, and low edge server energy consumption compared with baseline approaches"* [1]. Similarly, decentralized streaming networks dramatically reduce infrastructure cost: the Livepeer network is designed to lower transcoding costs up to *10*× compared to centralized cloud services [2]. Notably, specialized blockchains like Theta combine an edge network with a blockchain: the Theta Edge Network hosts over 10,000 nodes with *80 PetaFLOPS* of GPU compute (equivalent to ~250 Nvidia A100s) [3], which can power tasks like video transcoding and AI processing. Theta's **Video API** launched in 2021 allows any user to embed decentralized video on websites, where viewers "will… relay video over the Theta Network on a peer-to-peer basis" [4]. This shows proven viability: users watching a stream help distribute it.

- **WebRTC vs HLS Latency:** Modern P2P streaming will likely use WebRTC (real-time protocol) to minimize latency. HLS (Apple's HTTP Live Streaming) typically incurs 10–20 seconds of delay (≈10 s even in low-latency mode) [5]. In contrast, WebRTC enables *ultra-low latency* (~0.5–3 seconds) since it uses UDP and direct peer connections [6]. This means interactive streams (e.g. live mobile events) can appear near real-time if using WebRTC.
- **Mobile Device Constraints:** Mobile peers face battery, CPU, and memory limits. In theory, constant uploading/download can drain battery. For example, one P2P streaming primer notes *"device limitations: P2P streaming can be resource-intensive… [and] energy consumption: mobile devices may drain their battery quickly when involved in constant uploading and downloading"* [7]. However, experiments suggest manageable consumption: in one hybrid CDN-P2P test, an iPhone 11 streaming *and* transcoding a 5-minute video used only ~1.5% of battery [8]. Pure playback had far less draw; transcoding was the heaviest task but still modest (1–1.5% over 5 min) [8]. This indicates that with efficient codecs and short segments, smartphones can participate without crippling battery drain.

- **Throughput & Scalability:** In practice, P2P networks boost aggregate bandwidth. One study shows that peer-assisted streaming "reduces load on any single source and optimizes network usage" (by splitting content among peers) [9]. Platforms like Livepeer have already processed tens of millions of transcoded minutes per quarter (e.g. 35M minutes in Q4 2024 [10]). While direct peer upload speeds on cellular may be limited, peers on Wi-Fi or multi-path (5G + Wi-Fi) can supply chunks to neighbors. Benchmarks of live P2P systems (e.g. M2S, Hycon2.0 in research) show that intelligently combining CDN and peer sources can support thousands of viewers with low startup and rebuffering times.

2. Hybrid P2P-CDN Architecture Analysis

In a **hybrid P2P-CDN** design, streaming segments are delivered from either peers or CDN/edge servers based on network conditions. This leverages both the resilience of CDNs and the scalability of P2P. Key considerations include latency, peer churn, and leveraging edge nodes.

- **Latency Benchmarks:** Hybrid streaming typically uses adaptive bitrate (ABR) and smaller segments (e.g. 2–5 s). With WebRTC delivery, observed end-to-end latencies of 0.5–3 s are feasible [6]. In hybrid systems, an *action tree* can dynamically decide delivery: serve from a stable peer if available, else a local edge node, else CDN origin [11]. This keeps delays low. In practice, careful ABR tuning and prefetch (a few-segment buffer) can keep rebuffering rare, even on mobile networks.
- **Mobile Limitations:** As noted, mobile peers have constrained resources. Device memory limits cache size (some studies cap peer cache to ~5 segments [12]), and mobile NICs may cap uplink. Architecture must account for this by limiting upload loads and fall back to CDN if peers are overloaded. Uplink on cellular is often <5 Mbps; for 1080p video that may not suffice. Thus, a hybrid model ensures that if peer throughput is insufficient, the system "fallbacks to CDN for reliability" [13]. Indeed, best practices recommend exactly this: *"use hybrid models (P2P + server-based) for extra reliability"* [13].
- **Edge Computing Role:** Edge servers or "micro-CDNs" at 5G base stations (or local cloudlets) can cache and transcode video near users. By processing segments at the edge, latency is reduced. Edge computing "aims to reduce latency, improve bandwidth efficiency, and enhance overall performance by processing data closer to the end-user" [14]. For example, edge nodes can quickly serve popular segments or even perform real-time transcoding to suit a device's resolution, minimizing round-trip delays. For live events, placing servers or caches at cell sites can ensure *"minimal delay between the broadcast and viewer's screen"* [15]. Moreover, edge nodes help **bandwidth optimization** by caching repeated content locally [16], reducing core network load.
- **Fault Tolerance:** The system must gracefully handle peer/edge failures. Fault tolerance is achieved by: (1) caching redundancy (multiple peers hold overlapping segments); (2) immediate CDN fallback if peers fail; (3) checksums or hash-based chunk verification to discard corrupted data (mitigating malicious peers) [17]. Frequent peer churn requires robust discovery and replacement: for example, peers can send "heartbeat" signals or trackers monitor availability and switch sources when dropouts occur. Industry practice is to design the P2P layer so any missing chunk is quickly requested from a CDN or another peer, ensuring uninterrupted playback.
- **Dynamic Peer Orchestration:** To optimize delivery, the system should dynamically select peers based on proximity and reliability. Standards like IETF ALTO have been introduced for this purpose: ALTO provides network maps to P2P apps so they can choose nearby peers [18]. In addition, machine-learning approaches have been proposed: e.g. Ma *et al.* introduced an ML-based hybrid CDN-P2P peer-selection system (though without edge support) that improved distribution efficiency

[19]. Practically, an orchestration layer (tracker or edge servers) can rank peers by upload speed, device capabilities, and current cache contents [20]. The cited work describes a *"tree-mesh"* P2P structure where peers share idle bandwidth, storage and even computation [20]. In such schemes, peers report cache occupancy and resources to an edge node, which then uses an "action tree" logic to decide from where (peer, edge, CDN, origin) to fetch a segment [21] [22]. This ensures continuous delivery even as peers join and leave.

3. Expanded Technical Architecture

We extend the original architecture with considerations for fault tolerance, orchestration, deployment, and low-level tech limits:

- **Fault Tolerance & Reliability:** The architecture should include *multiple delivery paths*. For each video chunk request, the system should attempt peer retrieval first (if peers are healthy), but automatically fall back to an edge or CDN server if peers are slow or missing. This dual-path design provides resilience: even if a portion of the peer network fails, the CDN picks up the slack [13]. To further improve reliability, peers can perform **redundant prefetching**: storing not only immediate next segments but also some future segments common to popular streams, allowing for quick switchover if one peer drops out. Additionally, integrating **error correction** (e.g. Forward Error Correction codes) can mitigate packet loss.
- **Dynamic Peer Orchestration:** A key feature is continuous re-evaluation of which peer(s) serve each client. This may use algorithms or even AI/ML. For instance, an edge controller could run light-weight ML models to predict peer longevity or bandwidth availability based on historical data, then assign clients to the "best" peer. Although this is cutting-edge, related research (e.g. Ma *et al.*) shows ML can improve peer selection [19]. The orchestrator might also implement incentives: peers that consistently deliver good service could earn extra tokens or priority in future assignments.
- **PWA vs. Native App Trade-offs:** A mobile-first platform may deploy as a Progressive Web App (PWA) or native apps. **PWAs** have the advantage of "write once, run anywhere" on any modern browser or OS, instant updates, and no app store approval delay. Users can install a PWA from the browser; major streaming services (e.g. cloud gaming) have found PWA versions achieve high retention [23]. However, PWAs currently have **limitations**: on iOS/Safari the PWA cannot receive push notifications; integration APIs for Bluetooth or NFC may be unavailable; and background processes are limited. (For example, Safari lacks full PWA install banners and push support [24].) Native apps, by contrast, have full OS integration (background processing, optimized hardware acceleration for video, richer push capabilities) but suffer from distribution friction (app store reviews) and multiple codebases. In summary:
- *PWA:* easier distribution, web-based updates, cross-platform; but limited by browser restrictions and sometimes lower raw performance.
- *Native:* best performance and OS access, but higher development overhead per platform and longer update cycles.
  Either approach can work; many teams choose a hybrid strategy (e.g. initial launch as a PWA for reach, later native apps for power users).
- **WebAssembly (WASM) Limitations:** To achieve near-native performance in browser streaming (e.g. for video decoding, crypto, or ML inference), WebAssembly is attractive. It allows C/C++ or Rust code to run in-browser faster than JavaScript. However, WASM has known limits [25]: it has *limited DOM access* (requiring JavaScript bridges for UI/DOM interactions), *no built-in garbage collector* (developers must manage memory manually), and can be harder to debug. Browser support is mature, but

certain APIs (like multithreading or SIMD) may not be fully enabled on all devices. Also, WASM cannot directly access GPU codecs (it must rely on WebCodecs/WebGL APIs via JavaScript wrappers). Thus, while WASM can accelerate codec or cryptography, its performance may not match a fully native app, especially on low-end devices. We must account for these constraints: heavy processing tasks might still run more efficiently in a native binary or on an edge server, while WASM is best used for specialized modules (e.g. encryption or custom decoders) where needed.

4. Blockchain Layer & Scalability

The blockchain component handles content rights, tokens, and possibly payments. To avoid the high fees and latency of a base layer (e.g. Ethereum L1), we leverage Layer-2 (L2) and emerging Layer-3 (L3) solutions:

- **Layer-2 (L2) Solutions:** These rollups and sidechains dramatically increase throughput and cut fees. For example, **Arbitrum** and **Optimism** (optimistic rollups on Ethereum) routinely batch thousands of transactions off-chain, posting only compressed proofs to L1. Using Arbitrum for stream micropayments (e.g. per-minute credits) reduces fees to fractions of a cent and achieves ~1000+ TPS at low cost. Other L2s like **Polygon** (zk-rollup or sidechain), **StarkNet** or **zkSync** (zk-rollups) also enable scalable transfers and smart contracts. Even **Bitcoin Lightning Network** or Celer state channels could be used for token micropayments (per-view or per-second), with on-chain settlement only for final balances. The key point: adopting L2 dramatically minimizes transaction fees for creators and viewers. For example, L2s like Arbitrum have block times of 1–5 s and fees orders of magnitude below Ethereum L1.
- **Layer-3 (L3) and Cross-Chain:** Emerging L3s aim to be *application-specific chains* on top of L2s. They offer customization and cross-chain bridges. Notable examples include **XAI Games** (a gaming L3 on Arbitrum enabling thousands of in-game microtransactions cheaply) and **Orbs Network** (decentralized backend services on Ethereum). L3s can natively support high-speed interactions (e.g. NFT minting, governance) with minimal gas. Critically, L3s often include cross-chain messaging: they can interoperate with multiple L1/L2 networks. As one source notes, L3s *"can act as bridges between different L1s or L2s, allowing apps to seamlessly interact across ecosystems"* [26] . For example, a streaming dApp on an L3 could let content and tokens flow between Ethereum and Solana or other chains. Protocols like **Axelar**, **Hyperlane**, or **Wormhole** provide generic cross-chain channels so a token earned by watching on Ethereum L2 could be spent on, say, a Solana-based game or marketplace. This is valuable if different user segments prefer different chains.
- **Fee Minimization:** In addition to L2/3 adoption, we can design *dual-token* economics to manage fees. For instance, separate *governance tokens* and *usage (gas) tokens* can isolate volatility. The Script TV example uses $SCPT (governance) and $SPAY (gas): users earn $SPAY for streaming and spend it on premium content or tipping [27] . Ads and payments are also forced through $SCPT, creating sustained demand [28] . Structuring tokens with in-app sinks (ads, subscriptions, NFTs) and sinks helps prevent inflation. Layer-2 fee relays (like gas abstractions) can allow users to make micropayments in stablecoins or even fiat on-ramp, while the blockchain ledger settles them later.
- **Cross-Chain Streaming Use-Cases:** By using interoperable protocols, a single streaming app could run on multiple blockchains. For example, a user's viewing credits earned on Ethereum's L2 could be bridged via a Polygon chain or Cosmos IBC link to another chain where a content creator resides. This broadens market reach and liquidity. Moreover, if other blockchains specialize in media or gaming, integrating with them means tapping into their user bases and features. For instance, Theta Network itself is EVM-compatible and working on a Metachain (sharded L1/L3) to allow

"permissionless horizontal scaling" [29]. While cross-chain adds complexity, current tech (bridges, oracles) makes it increasingly feasible to operate on multiple chains.

5. Monetization Strategy and Tokenomics

A blockchain P2P streaming platform typically rewards both viewers and creators with tokens. We deepen the strategy by modeling token flows and ensuring sustainability:

- **Watch-to-Earn (W2E) Models:** Decentralized streaming platforms often reward viewers for engagement. As an example, **EarnTV** rewards viewers with its token (ETV) for simply watching promoted content [30]. This "watch-to-earn" concept *"gives viewers a tangible stake in the content they enjoy"* [31]. Similarly, the Script TV case study highlights a token-purchase mechanic: users buy an NFT (a "Script Glass") to enable earning SPAY tokens for content views [27]. Creators can then monetize these tokens by cashing out or using them to purchase on-platform benefits. However, not all analysts are optimistic about W2E sustainability: some argue watch-to-earn is niche and harder to sustain long-term than, say, play-to-earn or learn-to-earn [32]. The risk is that if token rewards are too generous, they may outpace real revenue or quickly deplete token supply. Our strategy must ensure that viewer rewards scale with platform revenue. For instance, allocate a fixed fraction of ad revenue or subscription fees into a "viewer reward pool." As user base grows, token inflation is balanced by these inflows; if growth stalls, rewards slow down.
- **Tokenomics Modeling:** We recommend formal modeling of token supply, demand, and flow. One approach is to create scenarios: for example, assume X tokens minted per week for viewers, Y tokens earned per creator per hour streamed. Link these to projected ad impressions or sponsorship fees. Table 1 (below) illustrates a simplified scenario model under different conditions:

Scenario	Monthly Ad Revenue	Viewer Tokens Pool	Token Value Assumption	Creator Payout per 1k Views
Low Activity	\$10,000	100,000 $TKN	\$0.05	\$0.50 (40 tokens)
Moderate Activity	\$100,000	500,000 $TKN	\$0.10	\$4 (40 tokens)
High Activity	\$1,000,000	2,000,000 $TKN	\$0.20	\$32 (40 tokens)

Table 1: Example token reward model. Tokens are fictional "$TKN". Creator payout assumes a fixed 40-token grant per 1,000 views. In high activity, tokens are worth more due to scarcity and demand.

In this model, sustainability requires that the token value or revenue backing scales with user engagement. Notice how creator payout (in USD) grows with activity. This illustrates that: (a) token issuance rates should be linked to real revenue (e.g. ad spend or subscription fees), (b) there should be token sinks (ads, giftshop, NFTs, premium features) to absorb tokens and support price. In practice, projects like Script TV require advertisers to pay in tokens, directly tying platform usage to token demand [28].
- **Creator Payouts:** We propose paying creators proportionally based on viewer engagement, split between tokens and possibly fiat/crypto. One model is "stream-to-earn": content is tagged for promotion, and for each minute watched, a small token stipend is given to viewers and a larger sum to the creator (similar to EarnTV's model [30]). Alternatively, a subscription or tip system could allow viewers to spend tokens directly on favored creators. The exact split can be adjusted (e.g. 70% of ad revenue to creators, 20% to viewers,

10% to platform). Under low revenue, token payouts shrink (to prevent inflation), which is fair since fewer advertisers would buy tokens. Under high revenue, token value should stabilize or rise, giving creators real gains. Modeling should test extremes: for instance, if revenue collapses, can creators still earn a livable amount? Measures like minimum wage floors (guaranteed minimal reward in tokens) or off-chain payment options (stablecoin micropay) might be needed for short-term viability.

6. Novel Innovation Opportunities

6.1 AI-Powered Mesh Networking

Concept: Integrate AI to optimize the P2P mesh network. For example, machine learning could predict which peers are most likely to have needed chunks and are best-connected, or dynamically compress/encode streams at the edge using AI. An *AI mesh* might use decentralized learning (like federated learning) to train models on network conditions, peer behavior, or content popularity. This could enable smarter caching and routing decisions on-device.

- **Implementation Roadmap:**
- *Phase 1:* Prototype a predictive peer-selection module using historical data (e.g. start with simple heuristics, then train a small model offline).
- *Phase 2:* Incorporate distributed ML inference at edge servers or even on powerful mobile GPUs to optimize real-time chunk requests.
- *Phase 3:* Extend to federated learning, where peers collaboratively improve the model without central coordination, preserving privacy.

- **Comparable Platforms:** Theta's EdgeStore and EdgeNodes project involve GPU-enabled edge computing for AI tasks [3]. Also, Livepeer's AI subnet and Cascade (Q4 2024) show industry interest in real-time video AI processing [33].

- **Technology Readiness:** Moderate (TRL ~4–6). Basic ML routing exists (see [36]), but on-device federated AI is emerging. High-end mobiles (with NPUs) and 5G MEC (multi-access edge computing) make this feasible soon. Over 2–3 years, improvements in mobile AI accelerators will raise practicality.

6.2 Gaming-Integrated Economy

Concept: Combine streaming with gaming-style incentives. For example, viewers could play interactive mini-games or lotteries while watching, earning additional rewards; creators could host "play-along" quests. Conversely, gamers could earn tokens by streaming games (blurring P2P streaming and play-to-earn).

- **Implementation Roadmap:**
- *Phase 1:* Integrate simple gamification (e.g. badges, leaderboards for top viewers) to boost engagement.
- *Phase 2:* Launch interactive features (e.g. spin-the-wheel NFTs, trivia during streams) that reward tokens.

- *Phase 3:* Partner with gaming projects or create an L3 (like XAI Games on Arbitrum) for in-app gaming with streaming. Possibly develop a unified wallet for both streaming tokens and game assets.

- **Comparable Platforms:** XAI Games (Arbitrum L3) demonstrates bridging streaming/gaming economies: it processes thousands of microtransactions per second for games [34] [35]. Projects like Gala Games and DLive show overlap between gamers and streamers. Recently, some services (e.g. BLive's rewards) hint at web3 game integration.

- **Technology Readiness:** Growing (TRL ~5–7). Basic gamification is trivial; fully integrated Web3 games plus streaming is cutting-edge but trending. Blockchain gaming and NFT ecosystems provide a foundation (e.g. Unreal Engine supports Ethereum plugins). Within ~2 years, expect mature frameworks for cross-app token use (see L3 interoperability) enabling this concept.

6.3 Micro-CDN via Edge and Peer Caches

Concept: Create a *micro-CDN* by leveraging peer storage and edge caches as mini-content servers. Instead of a monolithic CDN, this uses many small caches (on users' devices or local servers) to distribute content. For example, each ISP-population center might have a few seeded micro-CDN nodes.

- **Implementation Roadmap:**
- *Phase 1:* Use peer storage and local "edge nodes" (e.g. Wi-Fi access points with USB drives) to cache popular VOD content. Implement content-addressable storage (like IPFS) for deduplication.
- *Phase 2:* Integrate Theta's EdgeStore or similar tech for persistent storage of less-frequently-watched content [36]. Peers can fetch from the nearest micro-node.

- *Phase 3:* Expand to opportunistic caching: e.g. set up mini-CDN nodes in community centers or transit hubs. Use content routing (DNS+smart anycast) to direct viewers to nearest cache.

- **Comparable Platforms:** Theta EdgeStore (alpha since 2022) is essentially a dCDN/key-value store [37]. DWeb projects like IPFS/Filecoin or Arweave also provide decentralized storage networks that can act as CDNs for files. Content delivery networks (e.g. Fastly) are adopting edge computing, but micro-CDN here means using end-host resources.

- **Technology Readiness:** Moderate (TRL ~4–5). Distributed storage tech is proven (IPFS, decentralized storages) but widespread micro-CDN deployment is early. Still, user devices and edge servers with available disk could easily form a robust cache mesh. Experimentation (e.g. test sites in metro areas) could validate performance.

7. Implementation Roadmap

We propose rolling development in **4–5 phases** over roughly 3–4 years, balancing R&D, prototyping, and scaling:

Phase	Timeline	Key Deliverables	Dependencies
1. Design & Proof-of-Concept	0–6 months	- Detailed architecture and whitepaper - Prototype P2P streaming client - Smart contract specifications for token and rewards	Choice of blockchain L1/L2 HD video codec libraries Early dev team assembled
2. Core Development & Testing	6–12 months	- MVP with hybrid P2P-CDN streaming and simple token wallet - Integration with chosen L2 (e.g. Arbitrum) for micropayments - Basic mobile app (PWA or native) - Initial backend (peer tracker/edge server) for orchestration	Completion of PoC Blockchain dev tools (SDKs) Testnet for token issuance
3. Public Beta & Partnerships	12–24 months	- Public beta launch (limited users) - Monetization enabled: ad/staking smart contracts, token faucet - Content acquisition (streamers, studios) and early community build - Analytics dashboard and ABR tuning	Stable network usage metrics Regulatory review (if needed) Ad/sponsor commitments
4. Scaling & Ecosystem Growth	24–36 months	- Scale to thousands of users/streams - Add L2/L3 bridges for cross-chain token flow - Deploy edge caching nodes (micro-CDNs) - Introduce innovation features (gamification, AI optimizations) - Onboard additional chains (e.g. rollup on Optimism, or Cosmos zone)	Successful beta feedback Funding for expansion Edge infrastructure partnerships (5G providers)
5. Maturity & Innovation	36+ months	- Full-feature platform (AI-mesh optimization, gaming modules, full micro-CDN) - Multi-chain interoperability (users on different blockchains) - Long-term content and creator ecosystem (NFTs, premium tiers) - Enterprise offerings (white-label, licensing)	Market traction Regulatory clarity Continued R&D investment

Table 2: Phased roadmap with timeline, deliverables, and dependencies.

Each phase naturally depends on the success of prior work (e.g. a robust P2P engine is needed before adding tokenomics). During phases 2–3, thorough testing (performance, security audits of smart contracts, user UX) is crucial. Dependencies like choosing a blockchain (Ethereum vs others) early will shape development tools. Edge computing expansion (phase 4) may require telecom partnerships or deploying virtual servers near users. Throughout, community feedback and agile pivots should inform feature priorities.

References

Our analysis and expansions draw on recent studies, platform documentation, and case studies [1] [5] [6] [3] [4] [7] [13] [14] [15] [19] [31] [30] [32] [38] [28] [25] [24] [39] [34] [26], ensuring our expanded recommendations are grounded in up-to-date evidence. Each cited source anchors our design choices, from latency expectations to tokenomics examples.

---

[1] [8] [11] [12] [18] [19] [20] [21] [22] Towards Low-Latency and Energy-Efficient Hybrid P2P-CDN Live Video Streaming
https://arxiv.org/html/2403.16985v1

[2] [10] [33] Livepeer Q4 2024 Brief | Messari
https://messari.io/report/livepeer-q4-2024-brief

[3] [4] [29] [36] [37] What is Theta Network
https://docs.thetatoken.org/docs/what-is-theta-network

[5] [6] Optimize latency - Livepeer Docs
https://docs.livepeer.org/developers/guides/optimize-latency-of-a-livestream

[7] [9] [13] [17] What is P2P Streaming? Benefits And How It Works
https://inorain.com/blog/p2p-streaming

[14] [15] [16] The Role of Edge Computing in Video Streaming - Muvi One
https://www.muvi.com/blogs/role-of-edge-computing-in-video-streaming/

[23] [24] Progressive Web Apps | web.dev
https://web.dev/learn/pwa/progressive-web-apps

[25] Webassembly And Progressive Web Apps: The Perfect Pair
https://www.linkedin.com/pulse/webassembly-progressive-web-apps-perfect-pair-fkatc

[26] [34] [35] [39] Introduction to Layer 3 in Blockchain - Chainalysis
https://www.chainalysis.com/blog/introduction-to-layer-3-blockchain/

[27] [28] [38] Script TV Tokenomics Case Study. introduction | by BlockApex | BlockApex | Medium
https://medium.com/blockapex/script-tv-tokenomics-case-study-3b19427fbd51

[30] [31] How blockchain transforms the streaming industry via new business models
https://cointelegraph.com/news/how-blockchain-transforms-the-streaming-industry-via-new-business-models

[32] What is X-to-Earn? Exploring the New Trend in Crypto | IdeaSoft
https://ideasoft.io/blog/what-is-x-to-earn-applications/