Gaming per sec : Peer-to-Peer Cloud Gaming System

Abstract

Abstract In a world dominated by expensive GPUs and growing demand for high-end computing power, access to modern games remains exclusive to those who can afford elite hardware. This paper proposes a decentralized, peer-to-peer (P2P) cloud gaming system a community-driven platform where individuals can contribute their CPU power in exchange for tokens, and redeem those tokens to play games when needed. We aim to democratize gaming access through trust, transparency, and community computation.

1. Introduction The democratization of gaming has always been constrained by the asymmetry of computational power. While cloud gaming services attempt to bridge this gap, they remain limited by centralized infrastructure, high subscription costs, and regional restrictions. Millions of users across the globe, especially those with low-end devices, are unable to access the immersive experience of modern games simply due to hardware limitations. This paper introduces a decentralized, peer-to-peer (P2P) cloud gaming architecturea trust-based protocol where users share their idle CPU and GPU resources in exchange for tokens. This system empowers contributors with a meaningful incentive and allows players with weaker hardware to run high-performance games through distributed computing. The platform operates as a self-regulating, secure network of contributors and consumers, creating a circular ecosystem where gaming becomes a community-driven experience, rather than a privilege of the few. Our goal is to spark a revolution where users not only play, but also power each others gamesrestoring the spirit of cooperation, decentralization, and accessibility. This paper outlines the core architecture, economic model, technical challenges, and proposed cryptographic solutions required to make this vision a reality.

2. CPU Sharing

At the core of the system lies a decentralized exchange of computing poweran agreement between two parties: the contributor (who offers processing power) and the player (who consumes that power to run a game). Instead of transferring digital coins as in a blockchain, we transfer compute cycles.
Each interaction in the network is recorded as a CPU Sharing Unit (CSU), which includes:
Contributor ID A unique identifier of the device or user offering compute resources.
Player ID The recipient of the compute power.   Session Time Duration of resource usage (e.g., 1 hour).   Token Rate The token cost agreed for the session.   Resource Hash A cryptographic hash that validates the resource offered.   Proof of Contribution A record

signed by the player, verifying successful gameplay. These transactions are broadcasted to the network, where they're verified and stored in a decentralized ledger, ensuring transparency and fairness. Just as Bitcoin transactions are irreversible and cryptographically verified, our system ensures that every compute-sharing session is trustless and verifiablerequiring no central server to match users or validate usage. Contributors are rewarded in P2P Gaming Tokens (PGTs), which they can later use to play games themselves or trade on token markets. This transforms CPU power into a digital asset, allowing gamers worldwide to earn playtime by simply helping others.

3. Delay Management and Task Timestamping

In a decentralized peer-to-peer gaming environment, latency is one of the most critical challenges.
To ensure fair gameplay and a responsive experience, the system must not only allocate CPU power but also coordinate time-sensitive execution. To address this, we introduce the concept of the Timestamped Task Scheduler (TTS) a distributed mechanism that logs when a compute task begins, how long it runs, and whether it meets minimum performance guarantees. Every time a player initiates a session, the network performs the following: 1. Task Creation: A task package is created, including the game's configuration, system requirements, session start time, and expected duration.
2. Timestamping: This package is hashed and timestamped by a randomly selected set of trusted nodes, ensuring that the declared start and end times are publicly verifiable.   3. Ping Monitoring:
The network monitors the connection between contributor and player to ensure latency remains within acceptable limits.   4. Delay Compensation: If the contributing node cannot meet timing requirements due to network lag or local performance issues, the session may auto-switch to backup contributors or be paused with zero token cost until a stable connection resumes.   5.
Immutable Record: The timestamped record is logged into the distributed ledger as a verifiable block, ensuring trust even in asynchronous or delayed environments. By synchronizing time and enforcing deadlines using cryptographic timestamps and network consensus, we reduce the risk of resource hogging, latency fraud, or unresponsive gameplay. The Timestamped Task Scheduler is what keeps the peer-to-peer network synchronized, fair, and scalableenabling a seamless experience even across millions of nodes with varying network qualities.

4. Proof of Contribution (PoC)

In our peer-to-peer cloud gaming system, instead of mining blocks, users contribute CPU and GPU cycles to help others play. But how do we ensure someone actually contributed fairly and didnt just claim rewards without doing the work? Thats where Proof of Contribution (PoC) comes in. Each participating contributor must solve a real-time task: running parts of the game engine or rendering frames for the remote player. Unlike traditional proof-of-work, which is purely mathematical, our proof is functionala user proves they helped someone else play a game in real time. The Flow of Contribution: 1. Task Assignment The network assigns small, verifiable tasks (e.g., render 10 frames of gameplay, simulate physics, etc.) to nodes offering their power. 2. Live Hash Proofs As contributors compute, they generate real-time performance logs (hashes of completed outputs) that are signed and shared back to the network. These logs are lightweight, tamper-proof, and cryptographically verifiable. 3. End-to-End Validation The player (who is receiving the game feed) can sign off on the experience (e.g., lag-free, valid rendering), and the network cross-verifies this with expected results. 4. Token Reward Once validated, the contributor receives N tokens per hour, based on the amount of power and quality of service provided. Why Its Trustworthy: The system uses hashes of rendered frames or simulation states as proof that the game actually ran on a real machine. Cheating the system is nearly impossiblebecause you must deliver real-time, verifiable output under strict timing windows. It replaces meaningless computation (like Bitcoin mining) with meaningful community supporthelping others experience high-end gaming.

5. Network

To enable a seamless and decentralized gaming experience, our network must be fast, reliable, and secure. The network is responsible for distributing tasks, verifying contributions, and facilitating gameplay between contributors and players. Here's how it works: 5.1 Node Roles There are two types of nodes in the network: Contributor Nodes: Offer their CPU/GPU power to render game frames, compute physics, or stream content. Player Nodes: Request computing power to run a game they otherwise couldnt handle locally. A user can be both at different times: contribute when idle, and play when in need. 5.2 Task Distribution When a player requests to play a game, the system breaks the load into micro-tasks (e.g., rendering frames, executing logic). These micro-tasks are distributed to contributors based on latency, availability, and reputation. 5.3 Real-Time Sync Each frame or state change must be synchronized in milliseconds. The system employs low-latency WebRTC or UDP channels for fast delivery. To ensure stability, a redundancy protocol runs in parallelif one node fails to deliver in time, another instantly takes over. 5.4 Data Verification Every task response is signed with the contributors private key and hashed. Responses are verified by: The system (hash matches expected outcome) The player (confirmation of good performance) Random auditing by the network 5.5 Reputation System Each node builds a trust score based on past contributions. Nodes with higher scores get prioritized for gameplay and earn slightly more tokens per hour. Misbehaving or offline nodes are penalized or temporarily removed. 5.6 Fault Tolerance & Resilience The network is self-healing. If a node drops mid-game, the task is reassigned in real time. To minimize disruption, the system buffers a few seconds of game logic ahead on multiple contributors.

6. Incentive

To build a thriving and self-sustaining peer-to-peer cloud gaming ecosystem, we introduce a token-based reward system that incentivizes contributors while ensuring fair access to computing power. 6.1 Token System Contributors receive N Tokens for every hour of verified computing contribution (CPU, GPU, or network bandwidth). Players must spend tokens to request computing resources and play games through the system. This mechanism ensures a circular economy where everyone is motivated to contribute and can later redeem those efforts for playtime. 6.2 Earning Model The reward rate (tokens/hour) adjusts dynamically based on: Demand and supply of contributors Task complexity Contributors reputation Contributors with higher trust scores or low latency connections may earn bonuses. 6.3 Burning and Scarcity A small percentage of tokens are burned (permanently removed) from the system after every transaction, ensuring long-term scarcity and value preservation. 6.4 Optional Monetization Users who consistently contribute may optionally sell excess tokens on a decentralized marketplace. This creates an earning opportunity for users in low-resource regions or with idle computing infrastructure. 6.5 Gamification and Loyalty Badges, leaderboards, and staking mechanisms encourage continuous participation. Long-term contributors gain tiered access to premium features or early access to games. 6.6 Community Incentives A portion of the token supply is reserved for: Bug bounty programs Developer grants Community-driven upgrades This ensures that growth is guided by the very users and contributors powering the network.

7. Reclaiming Disk Space

In traditional cloud gaming systems, massive server farms store and process full game installations.

In our decentralized model, contributors temporarily host data. To ensure system scalability and efficiency, we propose a smart caching and cleanup mechanism. 7.1 Temporary Game Data When a contributor's system is selected for hosting a game session, only the essential game files are downloaded. These files are cached in a secure, sandboxed environment. After the session ends or times out, the system automatically purges unused or outdated data. 7.2 Adaptive Caching The system tracks game popularity and user demand. Frequently played games may have longer cache durations to reduce future download overhead. Rarely played or large titles are cleared quickly to free up space for new sessions. 7.3 Disk Contribution Preferences Contributors can choose how much of their disk space they're willing to share: e.g., 10 GB for short games, 50 GB for high-end titles. The system respects these limits and allocates tasks accordingly. 7.4 Compression and Deduplication Game files are compressed before distribution. Duplicate files across multiple games are deduplicated to save space. Only delta updates (changes from the last version) are sent to contributors. 7.5 Offline Preloading To support areas with unstable internet: Contributors can preload popular game files overnight or during low-traffic hours. These preloaded chunks are used when requested by players nearby, reducing delay and dependency on central servers. 7.6 Security and Isolation All temporary game data is stored in isolated containers to ensure: No access to user files No risk of malware persistence Auto-cleanup upon completion

8. Simplified Token Verification

To facilitate fast, secure, and lightweight verification of user contributions and consumption within the network, we propose a Simplified Token Verification (STV) mechanism. This system ensures contributors and players can verify their tokens without downloading the entire ledger or relying on centralized authorities. 8.1 Token as Proof of Contribution Each unit of compute time (e.g., 1 hour of CPU usage) is rewarded with a token. This token: Is cryptographically signed by the system. Includes metadata (time, task type, contributing device ID). Can be validated independently by any node using public system keys. 8.2 Lightweight Clients Instead of requiring each node to hold the full transaction history: Devices run a lightweight verification client. It queries network nodes for Merkle proofs of past token records. The user only needs block headers or compact proof chains, not the entire blockchain. This makes it feasible for even low-end devices to participate and verify their balance with minimal storage or computational overhead. 8.3 Fraud Resistance Token issuance and spending are linked to nonces and timestamps, making double-claims or forged activity detectable. Contributors cannot reuse the same compute cycles to claim multiple tokens. 8.4 Public Auditing All token issuance and redemption logs are publicly visible and append-only. A contributor can prove their work was valid. A gamer can prove they had enough tokens to consume compute time. Third-party auditors can scan the network for anomalies and abuse. 8.5 Self-Custody and Portability Tokens can be: Held in a users local wallet. Transferred to another device or account. Even potentially traded or exchanged on future decentralized gaming marketplaces. All this is done using deterministic key pairs, offering users full control without intermediaries.

9. Work Combining and Splitting

The core technical challenge in replicating GPU-like performance using distributed CPU resources lies in the effective combination and splitting of computational work. Just as Bitcoin allows splitting and merging of value, our system allows users to fragment workloads and aggregate results across multiple peers. 9.1 GPU-Like Simulation Using Multiple CPUs Parallel Execution: GPU rendering tasks are broken into micro-tasks (e.g., shaders, texture rendering, physics simulations). Task Sharding: These micro-tasks are distributed to contributors based on availability, latency, and computational power. Result Aggregation: Each contributor returns partial results, which are reassembled in real time by the systems orchestrator. This mimics GPU pipelines by leveraging hundreds or thousands of CPU threads across the network. 9.2 Micro-Task Ledgering Each micro-task is: Identified with a unique task ID and hash. Bound to a token commitment (so the contributor can be rewarded). Logged in a verifiable, timestamped structure. This allows tracking, load balancing, and rewarding granular contributions. 9.3 Redundancy and Verification To ensure correctness and prevent malicious actors: Redundant computation is introduced the same task is sent to multiple peers. Consensus logic ensures the correct output is verified by majority or hashed reference. Incorrect results are flagged, and reputation systems lower trust scores for faulty nodes. 9.4 Modular Architecture Low Latency Modules (e.g., physics, hit detection) are handled by trusted, high-speed contributors with low ping. High Throughput Modules (e.g., rendering frames) are split into tile-based jobs and parallelized. This enables real-time multiplayer gameplay by optimizing based on task type. 9.5 Dynamic Load Balancing As nodes join and leave: Tasks are dynamically reassigned. Dead contributors tasks are reassigned or fall back to local fallback CPU render. Load is intelligently spread based on bandwidth, CPU, and past performance.

10. Privacy

Privacy is a critical concern in any distributed system especially one where players and compute contributors interact in real-time. In our Peer-to-Peer Cloud Gaming System, multiple layers of privacy are embedded to protect both players and contributors from tracking, data leakage, and impersonation. 10.1 Anonymous Task Distribution Each compute task is: Sent without revealing the identity or location of the player receiving the results.   Assigned through an anonymized routing layer (similar to Tor relays or onion routing).   Encrypted end-to-end between the orchestrator and compute node. This ensures contributors do not know who they are rendering for, and vice versa. 10.2 Token Transaction Privacy To prevent exposure of contribution patterns or usage history: Token transactions are abstracted using zk-SNARK-style zero-knowledge commitments or ring signatures.   Contributors and players only reveal proof of computation or consumption, not the history of what was played or rendered. This keeps contribution rewards unlinkable to real-world identities. 10.3 Session Isolation Each gameplay session operates in a sandboxed environment: No personal device information is shared with contributors. Contributors environments are reset and hardened after each session to prevent data residue or reverse probing. No permanent link is ever formed between contributors and gamers. 10.4 IP and Metadata Obfuscation To guard against IP sniffing and timing analysis: Relays, proxies, or fog nodes are deployed.   Metadata like frame rates, timings, and control inputs are obfuscated or normalized to prevent fingerprinting. This makes it difficult for anyone to correlate activity patterns with real users. 10.5 Contributor Reputation, Not Identity The system tracks contributor behavior using reputation scores based on reliability, speed, and correctness not identity: Contributors earn trust without revealing who they are.   Bad actors are excluded automatically through behavior, not manual moderation. Privacy becomes a foundation, not an afterthought ensuring a truly decentralized, censorship-resistant experience for all.

11. Conclusion

We have proposed a decentralized, peer-to-peer cloud gaming system designed to break the barriers imposed by hardware limitations, centralized infrastructure, and economic inequality. By allowing users to contribute unused CPU power in exchange for tokens and later redeem those tokens to access high-performance gaming experiences we introduce a circular, self-sustaining ecosystem of

cooperation and fair access. This system addresses not only performance scalability and cost-efficiency, but also touches on the deeper values of the web: freedom, accessibility, and privacy.
At its core, this project is still in its prototype phase a vision in motion. Your feedback, suggestions, and technical contributions today could shape the foundation of a system that scales tomorrow. Like how Satoshi Nakamoto quietly lit a spark that grew into a global financial revolution, this paper aims to ignite curiosity, discussion, and perhaps a movement. We invite developers, gamers, researchers, and dreamers to explore, question, build, and improve upon this idea. Every voice, every node, every cycle of compute you share brings us closer to a world where gaming belongs to everyone. Who knows? Maybe, just maybe, this idea becomes the Bitcoin of cloud gaming.