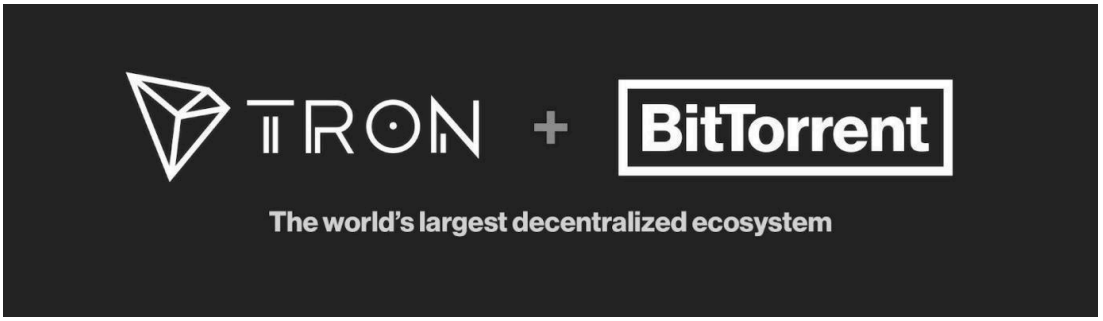


BitTorrent (BTT) White Paper

Oct. 2026

Contents

1. Executive Overview 4
2. Project Background 4
    2.1 High-Volume Data & Content Distribution 4
    2.2 AI Compute Supply and Demand 4
    2.3 On-Chain Settlement & Interoperability 5
    2.4 Ecosystem Design Principles 5
3. Ecosystem Architecture 5
    3.1 Distribution Layer: BitTorrent 5
    3.2 Decentralized Compute Layer: BTTInferGrid 6
    3.3 Network & Settlement Layer: BTTC 6
4. Product I: BitTorrent—Traffic & Delivery Engine 6
    4.1 Design Principles 6
    4.2 P2P Distribution Protocol & Network Primitives 7
        4.2.1 Content Addressing & Integrity Verification 7
        4.2.2 Peer Discovery and Swarm Topology 8
        4.2.3 Piece Exchange and Bandwidth Allocation 8
        4.2.4 Transport and Congestion Control 8
    4.3 Client Node Network 9
    4.4 BitTorrent Speed: A BTT-Based Bandwidth Incentive Layer 10
        4.4.1 Problem Statement: The Structural Absence of Seeding Incentives 10
        4.4.2 Overall Design 11
        4.4.3 Bidding Mechanism 11
        4.4.4 Matching and Auction 11
        4.4.5 Payment Channels and Settlement 12
        4.4.6 From Bandwidth to Universal Resource Exchange 13
    4.5 Distribution Capabilities for Models and Datasets 14
        4.5.1 Workload Characteristics and Adaptability 14
        4.5.2 Artifact Distribution Conventions 15
        4.5.3 Distribution Modes 15
        4.5.4 Long-Tail Availability and Incentives 16
        4.5.5 Positioning and Boundaries 16
5. Product II: BTTInferGrid—the Decentralized Compute Network of the Ecosystem 17
    5.1 Product Positioning: The Ecosystem Compute and Storage Layer 17
    5.2 Participant Roles and Resource Onboarding Mechanism 17
    5.3 AI Inference Workflow and Compute Scheduling 18
    5.4 Verifiable Inference and QoS Mechanisms 18
    5.5 Distributed Storage and Data Service Capabilities 18
    5.6 BTT Settlement, Node Incentives, and Penalties 19
    5.7 Developer Interfaces and Application Integration 19
6. Product III: BTTC—On-Chain Execution and Ecosystem Settlement Layer 19
    6.1 Product Positioning, Target Users, and Use Cases 20
    6.2 Architecture: Three Functional Layers and Key Roles 20
    6.3 Mechanisms: Service Execution and On-Chain Settlement Workflow 21
    6.4 Consensus Mechanism and Security Model 22
7. BTT Economic Model and Token Design 22

7.1 Core Token: BTT 22
7.2 BTT Token Distribution 23
7.3 Staking and Reward Mechanisms 24
7.3.1 Validator Staking Mechanism 24
7.3.2 Validator Partnership Model 24
7.3.3 Delegated Staking Mechanism 25
7.4 Unlock and Release Schedule 25
7.5 Buyback and Burn Mechanism 26
8. Application Scenarios & Ecosystem Development 26
8.1 Decentralized AI Inference and Compute Services 26
8.2 P2P Distribution for Models, Datasets, and Large-Scale Content 27
8.3 Cross-Chain Bridges and Asset Interoperability 27
8.4 DAO Governance and Voting System 27
9. Roadmap and Ecosystem Development Plan 27
9.1 Phased Roadmap 28
9.2 Ecosystem Highlights 28
9.3 Security and Governance 28
10. Conclusion 29

1. Executive Overview

The rapid growth of digital content, generative AI, and Web3 applications is pushing legacy infrastructure past its breaking point. Current architectures face critical bottlenecks, including imbalanced delivery efficiency, fragmented compute supply, steep on-chain fees, and poor settlement scalability.

The BitTorrent ecosystem is architected as an integrated infrastructure system encompassing content delivery, distributed compute, and on-chain scaling and settlement. Three core products work together under a single tokenomic framework:

- **BitTorrent:** A peer-to-peer (P2P) content delivery network that coordinates users, nodes, and bandwidth resources.
- **BTTInferGrid:** An AI inference service that handles the onboarding, job scheduling, execution, and verification of compute resources.
- **BitTorrent Chain (BTTC):** A Layer 2 scaling network designed for heterogeneous public blockchains, providing a high-performance, cost-effective environment for smart contract execution and on-chain settlement.
- **BTT:** The native utility token used across the ecosystem for resource purchases, gas fees, node incentives, staking, and governance.

By converging distributed content delivery, decentralized compute provision, scalable smart contract execution, and unified tokenomics, the BitTorrent ecosystem provides an open, efficient, and sustainable infrastructure layer for Web3 and AI applications.

2. Project Background

2.1 High-Volume Data & Content Distribution

Modern AI models and large-scale datasets often reach tens to hundreds of gigabytes. Distributing files of this scale via centralized servers incurs prohibitive bandwidth costs and egress fees. Centralized endpoints also create single points of failure, causing origin server congestion, service disruptions, and regional downtime during traffic spikes. Traditional delivery pipelines, moreover, are prone to version fragmentation, content tampering, and unverifiable provenance.

BitTorrent leverages its proven peer-to-peer content delivery capacity to coordinate global users, nodes, and bandwidth. Building on this capability, BitTorrent extends its architecture to provide open, high-throughput distributed delivery services optimized for AI models, large datasets, and other heavy digital assets.

2.2 AI Compute Supply and Demand

The exponential rise of generative AI has driven an upsurge in demand for inference compute. However, vast amounts of GPU and compute resources remain fragmented across isolated regions, devices, and service providers. Without an efficient mechanism to match supply with demand, substantial compute capacity sits idle, while developers face high acquisition costs, rigid resource allocation, and unverifiable quality of service.

Purpose-built for AI inference, BTTInferGrid aggregates globally distributed GPU capacity into an open, verifiable, decentralized compute network. Through integrated mechanisms for resource onboarding, job scheduling, inference execution, and result verification, BTTInferGrid provides AI developers and applications with more flexible, cost-effective compute infrastructure.

2.3 On-Chain Settlement & Interoperability

As Web3 applications and decentralized resource protocols mature, the demand for on-chain interactions, asset liquidity, and service settlement keeps growing. However, public blockchains remain siloed, with varying throughput capacities and fee structures that hinder large-scale, cross-chain application interoperability.

BTTC serves as a Layer 2 scaling network connecting heterogeneous public chains. It provides a high-performance, cost-effective execution environment for smart contracts, cross-chain interoperability, and on-chain settlement, enabling seamless value flow for data, compute, and storage services within the BitTorrent ecosystem.

2.4 Ecosystem Design Principles

The BitTorrent ecosystem structures its multi-layered product architecture around content delivery, AI compute, storage, and blockchain infrastructure, guided by four core design principles:

- Layered Synergy: Each product line maintains a clear operational focus while collaborating seamlessly within a unified ecosystem.
- Open Access: Frictionless access for diverse user profiles, node operators, and resource providers to participate in the network.
- Resource Sharing: Maximizing the operational efficiency of globally distributed bandwidth and compute resources.
- Unified Tokenomics: Leveraging BTT as the universal medium of exchange for resource consumption, service payments, and node incentives.

By uniting these product lines under a cohesive technical framework and economic model, the ecosystem delivers an open, efficient, and sustainable infrastructure foundation for Web3 and AI applications.

3. Ecosystem Architecture

As the master brand, BitTorrent anchors the technical identity, user base, and global brand equity of the entire ecosystem. Within this framework, BitTorrent, BTTInferGrid, and BTTC each take on distinct roles across data distribution, decentralized compute, cross-chain messaging, and financial settlement, while BTT serves as the universal medium of exchange for resource utilization, payments, incentives, staking, and governance.

Each product line operates as an independent service tier for users and developers while integrating via standardized interfaces and the BTT tokenomic engine. Together, they establish a closed-loop infrastructure pipeline spanning data distribution, compute execution, outcome verification, and on-chain settlement, providing decentralized infrastructure for Web3 and AI applications.

3.1 Distribution Layer: BitTorrent

BitTorrent acts as the network's primary gateway and data distribution layer. Drawing on its battle-tested P2P protocol, client ecosystem, and distributed node network, BitTorrent aggregates idle edge bandwidth into a shared resource pool to power peer-to-peer file and data transport.

In traditional file-sharing contexts, BitTorrent optimizes transmission efficiency and reduces reliance on centralized servers through mechanisms such as torrent files, magnet links, peer discovery, chunking, and intelligent bandwidth allocation.

For AI workloads, BitTorrent extends these distribution capabilities to serve open-source model weights, training datasets, and other massive data resources. In this role, it serves as the data ingestion engine for BTTInferGrid's inference execution and storage pipelines.

3.2 Decentralized Compute Layer: BTTInferGrid

BTTInferGrid is a decentralized compute network designed for AI inference. It aggregates global GPU nodes to deliver flexible, cost-effective, and verifiable compute capacity for developers and applications.

Developers submit inference workloads through unified APIs, after which the network matches and schedules jobs based on task specifications, node capabilities, and real-time network state. Qualified GPU nodes execute these tasks and earn yield based on their compute contribution.

BTTInferGrid verifies compute output through task challenges, cross-verification, node scoring, and on-chain coordination. The verification results form the basis for fee settlement, reward distribution, and slashing penalties.

3.3 Network & Settlement Layer: BTTC

BTTC functions as the ecosystem's Layer 2 scaling and settlement engine, providing seamless cross-chain interoperability across major networks. It connects heterogeneous blockchains, including TRON, Ethereum, and BNB Chain, and powers cross-chain transfers of assets, data, and state updates with its infrastructure.

BTTC processes cross-chain interactions through a three-layer architecture comprising the Contract Layer, the Delivery Chain, and the BTTC Chain. In particular, the Contract Layer manages asset locking and releasing while logging cross-chain events; the Delivery Chain listens for cross-chain events, generates state checkpoints, and aggregates validator signatures; the BTTC Chain executes smart contracts, updates network state, and powers decentralized applications. Capitalizing on its EVM compatibility, minimal gas fees, and high-throughput transaction processing, BTTC handles service settlements, node staking, reward distributions, and cross-chain payments across the ecosystem.

4. Product I: BitTorrent—Traffic & Delivery Engine

The core value proposition of the BitTorrent ecosystem lies in aggregating distributed upload bandwidth, storage, and content replicas across a vast number of edge endpoints into a coordinated delivery network that operates without a central coordinator. As the world's largest decentralized content delivery protocol with over two decades of production deployment, BitTorrent serves as the ecosystem's traffic and delivery engine. It functions as the data plane for high-volume digital assets—files, software updates, AI model weights, and datasets. It is also the primary gateway for onboarding nodes and wallets at scale, and the protocol-level origin of the "resource contribution—token acquisition—service consumption" economic loop.

4.1 Design Principles

The architecture of the BitTorrent ecosystem is built on five core principles:

- **Openness & Backward Compatibility:** Network upgrades are executed via protocol extension messages to ensure soft-fork compatibility between legacy and updated clients, preventing network fragmentation.

- Resource Control: Resource contribution is fully transparent and user-configurable. The protocol enforces strict resource boundaries (incl. bandwidth and local storage) without interfering with foreground processes, allowing users to opt out at any time.
- **End-to-End Verifiability:** Eliminating reliance on node trust, the protocol guarantees data integrity and transactional security through cryptographic signatures and piece-by-piece verification, effectively mitigating malicious attack vectors.
- **Protocol Neutrality & Market Orientation:** The protocol provides raw metering and settlement primitives without arbitrating content value. The value of delivery and storage is determined solely by market supply and demand.
- **Seamless User Experience:** Native wallet management is embedded directly into the client with automated fee bidding to lower entry barriers, enabling non-technical users to participate in resource sharing without friction.

Guided by these principles, the BitTorrent delivery engine integrates four distinct layers: the Protocol Layer (addressing & exchange), the Client Layer (cross-platform endpoints), the Incentive Layer (Speed bandwidth matching), and the Data Capabilities Layer (AI model and dataset support).

4.2 P2P Distribution Protocol & Network Primitives

4.2.1 Content Addressing & Integrity Verification

Upon publication, digital assets are chunked into fixed-size pieces, and a metainfo descriptor (torrent file) is generated to encode the piece layout and cryptographic hashes. The protocol uses the cryptographic digest of this metainfo, known as the infohash, as the globally unique identifier for the asset. Content addressing decoupled from physical location ensures that as long as the infohash matches, data validity remains invariant regardless of origin node or transit route.

This architecture establishes two fundamental guarantees:

First, **integrity is verifiable end-to-end at piece-level granularity**. Every piece received by a node is independently validated against the piece hash recorded in the metainfo. Corrupted or malicious pieces are detected and discarded instantly at the piece level, requiring zero trust in intermediary relay nodes. Boundary condition: Piece hashes prove byte-level integrity (i.e., confirming that received bytes match the publisher's metainfo digest). Publisher identity and content authenticity are established via higher-layer cryptographic signatures (see Section 4.5.2 for the dual-verification model).

Second, **data transport is source-agnostic**. Data corresponding to a given infohash can be ingested concurrently from P2P peers and HTTP origin endpoints (Web Seeds, BEP-19), with both sources governed by identical integrity constraints. P2P delivery thus integrates seamlessly with legacy centralized infrastructure (e.g., object storage, CDNs, and mirror sites)—where HTTP sources guarantee baseline availability while the P2P layer offloads peak traffic, allowing content providers to adopt decentralized distribution without re-architecting their existing storage stacks.

Under BitTorrent v2 (BEP-52), the protocol upgrades its cryptographic hash function to SHA-256 and builds a dedicated Merkle tree for each file within a torrent. This enables file-level content addressing, cross-torrent deduplication, fine-grained verification anchored to the Merkle root, and selective file download capabilities.

4.2.2 Peer Discovery and Swarm Topology

Nodes actively exchanging pieces corresponding to the same infohash form a swarm. They discover one another through three complementary mechanisms:

- **Trackers** (BEP-3/BEP-23): Centralized rendezvous services run by publishers or third parties where nodes announce their state and receive a list of active peers within the swarm.
- **DHT** (BEP-5): A Kademlia-based distributed hash table that decentralizes the "infohash-to-peer list" mapping across all participating nodes in the network—enabling peer discovery without centralized infrastructure.
- **PEX** (BEP-11): Direct gossip between connected peers exchanging their known peer lists to accelerate topology convergence within the swarm.

DHT and PEX enable swarms to self-organize, establish connectivity, and stay resilient without central coordination; trackers are relegated to optional accelerators and telemetry services. Because the discovery layer has no single point of failure, if any one mechanism fails or is deliberately disabled, the swarm falls back to the remaining pathways and keeps operating without interruption.

4.2.3 Piece Exchange and Bandwidth Allocation

Resource allocation within a swarm is governed by two algorithms:

Rarest-First. When selecting its next download request, a node prioritizes the rarest pieces—those with the lowest copy count across the swarm. This strategy ensures a uniform distribution of pieces, mitigating the risk that critical data becomes unavailable when the few holding nodes go offline. Swarm health is quantified by an availability metric, defined as the number of distributed replicas across the swarm: as long as every piece is retained by at least one active node, the file remains fully downloadable—even in the temporary absence of a complete seeder.

Choking and Tit-for-Tat Reciprocity. Each node limits its active upload capacity to a restricted subset of peers, categorizing connections as either unchoked (authorized to receive data) or choked (temporarily unserviced). A standard choking algorithm ranks peers based on the upload volume received from them since the previous round. The top N peers—where N represents the client's configured unchoke slots—are unchoked, while all remaining peers are choked. Choke states are recalculated periodically, typically on a 15-second cycle. Because seeders require no incoming data, they rank peers strictly by outbound throughput, optimizing purely for overall network efficiency.

In addition to regular slots, clients reserve a small number of **optimistic unchoke** slots (typically one), unchoking peers via random selection or rotation. This mechanism allows new nodes without established reciprocal upload histories to acquire initial pieces and bootstrap tit-for-tat exchanges with other peers.

Choking serves as the primary mechanism for intra-swarm bandwidth allocation: nodes contributing greater upload bandwidth receive prioritized download throughput, while underperforming or free-riding connections are throttled, disconnected, or blacklisted. This mechanism also serves as the integration hook for incorporating BTT bidding logic into resource scheduling (Section 4.4).

4.2.4 Transport and Congestion Control

Data transport runs over uTP (BEP-29) by default, which utilizes LEDBAT congestion control. LEDBAT uses one-way delay variations as a congestion signal, proactively throttling transfer

rates upon detecting queue buildup along the network path. This yields priority to interactive traffic sharing the same egress path—such as web browsing or video conferencing—including traffic generated by other devices on the same local network. For a distribution network, this property provides structural advantages: background distribution tasks can utilize idle upstream bandwidth over extended periods without degrading foreground user experience, enabling consumer devices to remain continuously online with ongoing contribution.

4.3 Client Node Network

The execution layer of the distribution engine operates on a dual-brand product suite—BitTorrent and µTorrent—alongside an open network of third-party clients implementing the same protocol family. Open protocols foster client diversity and market competition, while the official client matrix covers primary user endpoints (see Table 4-1), representing over 2 billion cumulative downloads.

Table 4-1 Official BitTorrent Client Matrix

Client	Positioning	Role in Distribution Ecosystem
BitTorrent Web	Consumer-facing sequential downloading	Lowers the entry barrier by integrating streaming playback directly with P2P transport
BitTorrent Classic	Desktop batch downloading & advanced settings	Provides granular control over batch distribution, seeding, task prioritization, and bandwidth scheduling
BitTorrent Android	Mobile torrent/magnet downloading & playback	Extends the delivery network to mobile devices, supporting Wi-Fi-only execution and remote desktop task dispatch
BitTorrent Neo	Modern, lightweight macOS client	Delivers a streamlined download experience tailored for recent macOS releases
BitTorrent Remote	Remote management interface	Serves as a cross-device control portal for managing home client instances

The client network fulfills three structural roles within the ecosystem:

Node Density and Geographic Distribution. Swarm throughput and resilience scale directly with node count and geographic dispersion. A massive install base ensures that high-demand assets achieve localized, high-throughput distribution even in regions lacking dedicated CDN infrastructure. Clients act as both consumers and distribution nodes: within user-configured limits, every client continuously seeds existing pieces to the swarm.

Native Vector for Wallets and Identity. BTT wallets are embedded directly into supported clients, distributing frictionlessly to general users through standard application updates. The identity model adheres to established protocol conventions: since its inception, the BitTorrent protocol has identified software instances (defined by IP address and port combinations) rather than physical individuals. Binding wallets to client instances follows this exact paradigm, mirroring the cryptographic principle that "whoever owns the private key owns the asset." Beyond wallet passphrases, no auxiliary identity management layer is introduced. Users can earn and spend BTT without needing to manage complex key infrastructures (Principle 5).

Supply-Side Resource Pool. Client nodes featuring high uptime, idle upstream bandwidth, and local storage form the supply-side pool for the bandwidth incentive layer (Section 4.4). They also represent a natural candidate pool for expanding node capacity across other decentralized resource networks within the broader ecosystem.

4.4 BitTorrent Speed: A BTT-Based Bandwidth Incentive Layer

4.4.1 Problem Statement: The Structural Absence of Seeding Incentives

The native protocol's tit-for-tat reciprocity holds only while a download is in progress. Because upload and download speeds are typically asymmetric, a downloader usually finishes the entire file before it uploads an equivalent amount. Once the download completes, continuing to upload—seeding—earns nothing within the protocol, and closing the client costs the user nothing. Seeding is purely altruistic. The systemic result is that users generally leave a swarm having uploaded less than they downloaded, and swarms die out before demand for the content does.

Rarest-first piece selection can slow the decay of a seedless swarm: as long as every piece is still held by at least one peer, downloads can complete. But large-scale measurements of seedless swarms show that in roughly 86% of cases, the remaining peers do not collectively hold a complete copy; only about 14% of seedless swarms can still finish the file from what their peers already have (Kaune et al., 2010).

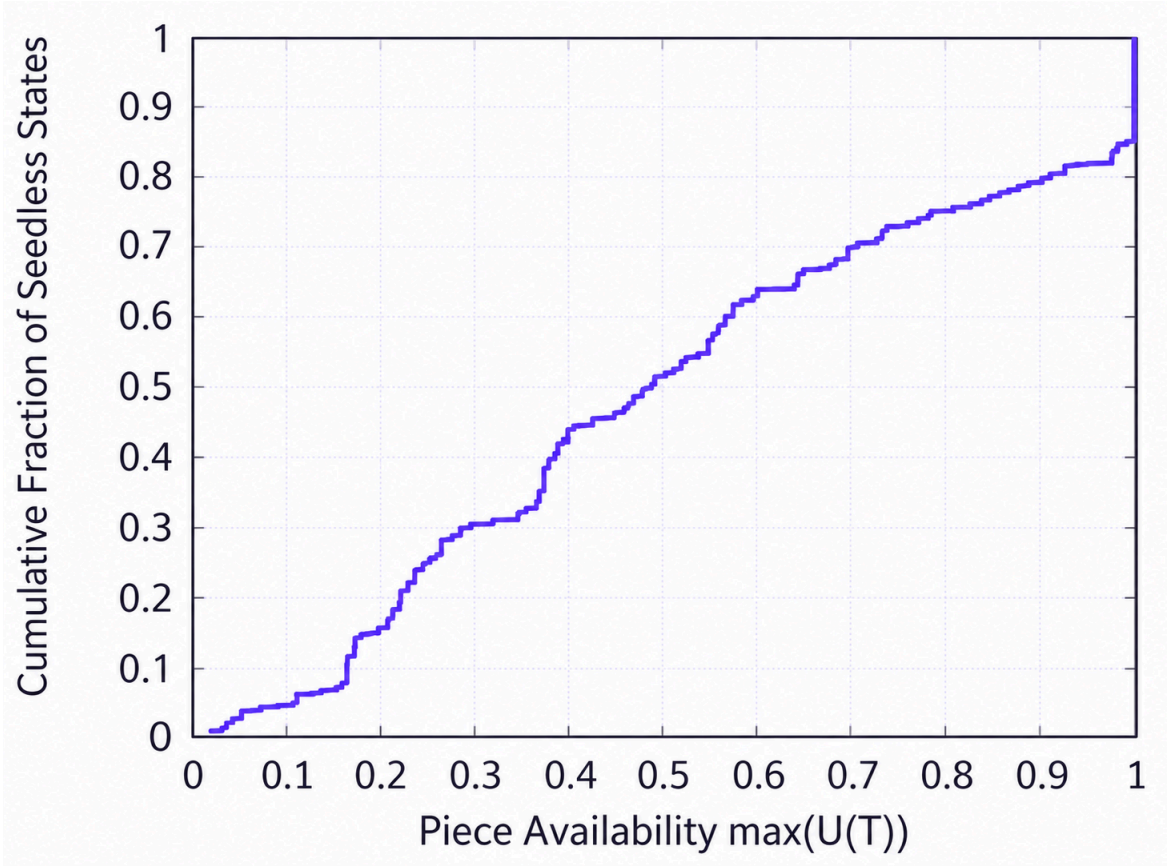


Figure 4-1 Piece Availability Distribution in Seedless Swarms

The absence of seeding incentives is a structural flaw at the protocol layer. We gave serious consideration to building long-term incentives into the protocol core and rejected it, for three reasons. First, "what content deserves long-term seeding" cannot be defined cleanly at the protocol layer without being gamed. Second, the sheer size of the BitTorrent ecosystem makes any hard fork almost certain to fail: a protocol that penalizes non-seeding can be sidestepped simply by switching to an older client that doesn't. Third, value judgments of this kind are better left to the market than baked into the protocol. Our answer, therefore, is to leave the protocol

core untouched and build tokenized bandwidth exchange on top as an extension—BitTorrent Speed. This follows directly from Principles 1 and 4.

4.4.2 Overall Design

BitTorrent Speed is an incentive overlay on the existing protocol, integrated into the BitTorrent and µTorrent clients since 2019. It changes none of the protocol's existing behavior: users who opt out of BTT retain full torrent functionality. In swarms that mix legacy and Speed-enabled clients, token exchange occurs only between Speed-enabled clients; legacy clients behave exactly as before, maximizing upload under native reciprocity.

In Speed, a peer acts as a **service requester** (paying BTT for another peer's bandwidth), a **service provider** (supplying bandwidth in exchange for BTT), or both at once. A Speed session begins with ordinary peer discovery—Speed introduces no new discovery infrastructure, and swarm formation is unchanged from Section 4.2.2. The BTT wallet, automatic bidding, and earnings dashboard are built into the client. This is not mining: earnings come from bandwidth actually delivered, not from proof of work.

Speed does not create physical bandwidth and does not guarantee that any given download will be faster. Actual results depend on the number of seeders in the swarm, peers' available bandwidth, connection quality, file demand, and the proportion of peers running Speed-enabled clients. What Speed does is let peers **bid BTT for priority service, which improves their odds of a faster download**: market forces scarce upload capacity toward peers that value it most, and seeders gain an economic reason to stay in the swarm, keeping files available for longer.

4.4.3 Bidding Mechanism

Bid broadcast. A service requester sends a bid, via a protocol extension message, to every peer holding pieces it needs. The message states how much BTT the requester is willing to pay per piece.

Automatic bidding. By default, the client bids on the user's behalf. The initial release uses a simplified auto-bidding algorithm:

bid = (spending rate × remaining BTT balance in wallet) / remaining download (KB)

where spending rate is a bid-aggressiveness parameter in the range 0.0–1.0, set to 1.0 in the initial release; bidding stops automatically once the remaining download reaches zero. Because peers often serve data for free under native reciprocity, the remaining download shrinks faster than the wallet balance, so the bid tends to rise as the download progresses.

Bid revision and rate limiting. A client may use any heuristic to decide when to rebroadcast a new bid, but to limit protocol message overhead, it does not rebroadcast when the bid changes by less than 10% from the previous bid, and it rebroadcasts no more than once per minute. Manual changes by the user are broadcast immediately. Planned refinements include estimating the market price of bandwidth from bid-message traffic within the swarm and current transfer conditions, and adjusting bids based on piece rarity.

User control. Users retain full control over bidding: a global on/off switch, per-torrent switches, spending-rate adjustment, a price ceiling, and fine-grained manual overrides (Principle 2).

4.4.4 Matching and Auction

Service providers fold BTT bids into the choking decision described in Section 4.2.3. At each choking recalculation interval, the provider first aggregates all currently valid bids and modifies

the existing ranking algorithm to allocate unchoke slots based on a combination of the highest valid bid and historical data reciprocity. When a winning bidder is unchoked, the provider first sends a bid-response extension message indicating the rate to be charged (BTT/byte), followed by the standard unchoke message—ensuring strict layering and isolation between extension and native messages.

Clients may implement any auction format. We recommend a variant of the Vickrey–Clarke–Groves mechanism, which in this setting reduces to a **multi-unit uniform price auction**: each requester competes for exactly one unchoke slot per provider, and all winners pay a uniform price equal to the **highest losing bid**. The combination of uniform pricing and the second-price rule discourages strategic bidding—bidding truthfully is an approximately dominant strategy, eliminating the need for requesters to strategize against other bidders.

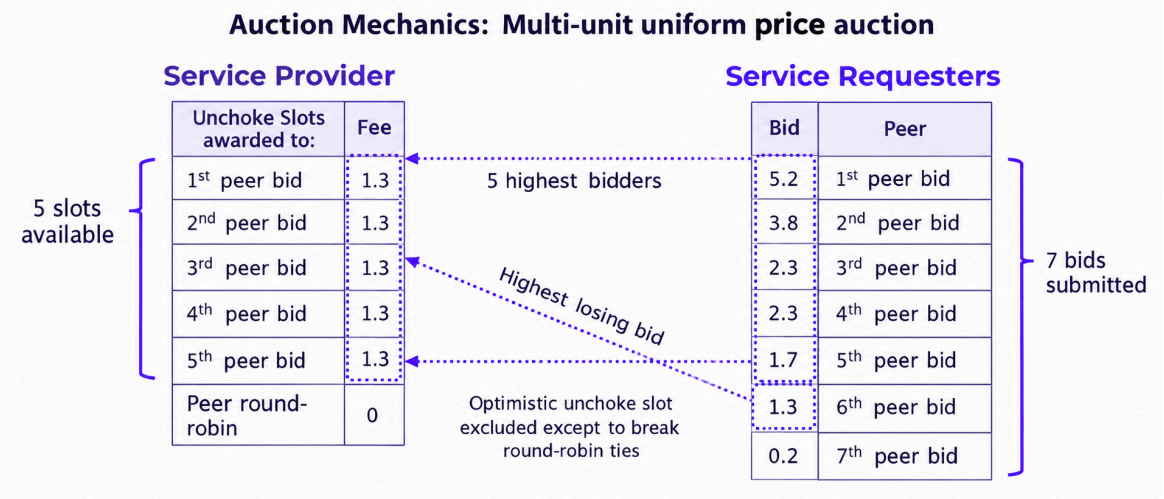


Figure 4-2 Multi-Unit Auction with Uniform Prices

Protecting the optimistic unchoke. The optimistic unchoke slot is the bootstrap path for new peers entering a swarm and must be handled with care. It does not take part in the regular auction; an auction is permitted only as a tie-breaker, when multiple never-before-unchoked peers are exactly tied in rotation order. This guarantees that newcomers without BTT always have a free path into the network—the incentive mechanism must never undermine the network's openness (where Principles 4 and 5 intersect).

4.4.5 Payment Channels and Settlement

Payments are executed over a peer-to-peer payment channel between the requester (K1) and provider (K2). The channel runs on a high-throughput settlement ledger bidirectionally bridged to the BTT main ledger. The protocol operates as follows, with K1 and K2 representing the public keys of the requester and provider, respectively:

1. The requester generates a public key, K1, and requests the provider's public key, K2;
2. The requester opens a payment channel on the settlement ledger designating K2 as the payee, and deposits BTT from the K1 account as escrow;
3. The requester constructs an in-channel transaction allocating the initial payment to K2 and returning the remaining balance to K1;
4. The requester signs the transaction and transmits the signed transaction payload to the provider;

5. Upon discovering the channel for the first time, the provider queries the settlement ledger and validates that the channel parameters match the requester's transaction payload before acknowledging payment.
6. For each delivered data piece, the requester increments K_2's allocation, re-signs the updated transaction, and transmits it to the provider, yielding a piece-by-piece micropayment stream.
7. If the escrow balance is fully drawn before delivery completes, the requester opens a new channel and repeats Steps 2–4.

Channel Settlement & Timeout: Upon transfer completion, the provider signs the last transaction and submits it to the settlement ledger for final balance allocation, permanently closing the channel. If the provider abandons execution mid-transfer, a built-in timelock mechanism prevents capital lockup: upon expiration, the settlement ledger automatically closes the channel and returns all escrowed BTT to the requester.

Failures and Penalties: Requesters with unfulfilled payment commitments upon timelock expiration are immediately choked and denied further data transmission; persistent default triggers temporary IP/key blacklisting by the provider. Symmetrically, providers transmitting corrupted or invalid data blocks face identical blacklisting.

Risk Exposure Analysis: By operating in an interleaved lockstep model—alternating between data piece delivery and signed state incrementation—the protocol tightly bounds counterparty risk. The provider’s maximum exposure at any point is strictly bounded to the bandwidth cost of a single piece, as data delivery precedes payment allocation; conversely, the requester incurs zero financial risk, incrementing payments only after passing cryptographic integrity checks. This design bounds operational exposure while asymmetrically safeguarding resource contributors, operationalizing end-to-end verifiability at the economic layer.

4.4.6 From Bandwidth to Universal Resource Exchange

BitTorrent Speed's track record confirms a foundational thesis: **token-metered resource exchange among low-value, high-frequency, and trustless peers is technically viable within large-scale, consumer-grade networks**. Low-friction entry via embedded wallets and automated bidding, resource isolation through quotas and congestion control, and incentive compatibility anchored to seeding rewards successfully coexist within a mass-market architecture.

The protocol extensions and transaction processing underlying bandwidth exchange are generic: the same set of bidding, matching, and channel-payment primitives can support broader resource categories. We abstract these protocol extensions into a pluggable service framework, enabling multiple BitTorrent (BT) services to operate concurrently over a unified transaction processing layer. New services are formally defined, reviewed, and integrated into the network organically, with standardization governed through the open BitTorrent Enhancement Proposal (BEP) process.

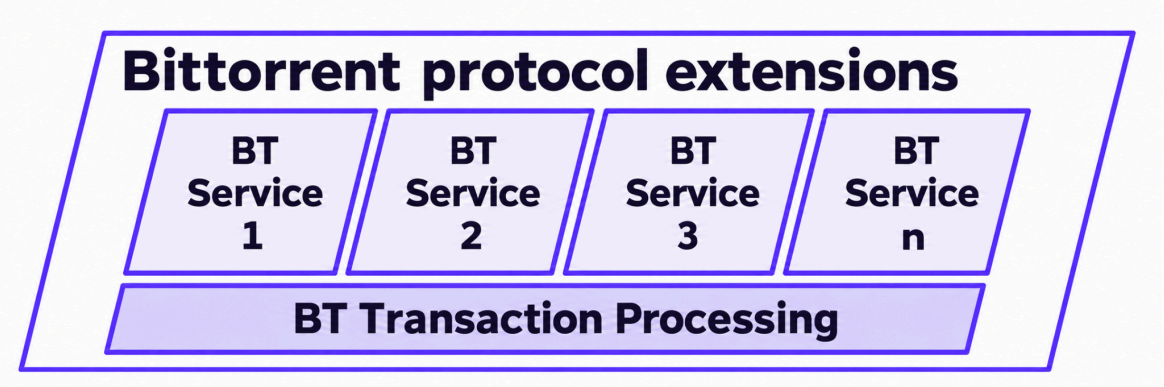


Figure 4-3 BitTorrent Protocol Extensions and Resource Exchange Architecture

4.5 Distribution Capabilities for Models and Datasets

4.5.1 Workload Characteristics and Adaptability

AI workloads introduce a new class of objects to content distribution: model weights, training and evaluation datasets, training checkpoints, container images, and associated resource packages. The BitTorrent protocol is content-agnostic, allowing its general-purpose large-file distribution capabilities to map seamlessly to these assets. Moreover, the workload profiles of these assets fall precisely within the sweet spot of the protocol mechanics outlined in Section 4.2:

- **Large individual object size**: Mainstream open-source model weights range from a few to hundreds of gigabytes, while datasets are often measured in terabytes—making them a prime use case for piece-based, multi-source parallel downloads;
- **Global concurrent pulls at release**: New model version releases exhibit a thundering herd pattern, triggering massive global concurrent downloads within hours—precisely where swarm-scale positive feedback takes effect, directly converting demand peaks into supply peaks.
- **Read-only immutability with concentrated hotspots**: Weights and datasets are not modified in place once published, aligning naturally with the immutable semantics of content addressing.
- **High integrity requirement**: Tampering with weights constitutes a model supply chain attack, where piece-by-piece hash verification provides the first line of defense at the byte level.
- **Long-tail version availability**: Scientific reproducibility and production rollbacks demand continuous access to historical versions—a recurrence of the seeding challenge in AI workloads and a prime use case for BitTorrent Speed incentive primitives.

In contrast, centralized object storage and CDNs under such workloads face two simultaneous bottlenecks: egress bandwidth costs that scale linearly with download volume, and severe network congestion during release peaks. Table 4-2 outlines the mapping from underlying capabilities to the value generated for AI asset distribution, while explicitly indicating the areas that upper-layer systems must supplement above each capability.

Table 4-2 Distribution Layer Capabilities—Mapping and Boundaries for AI Asset Distribution

Underlying Capabilities	Value for Model & Data Distribution	Upper-Layer System Capabilities
Large-file partitioning and multi-source parallel downloads	Distributing large-scale weights, datasets, checkpoints, container images, and resource packages	Asset catalogs, versioning, dependency management, and delta updates

Underlying Capabilities	Value for Model & Data Distribution	Upper-Layer System Capabilities
Piece-level hash verification	Detecting data corruption and invalid pieces during transmission	Publisher signatures, chains of trust, SBOMs/model cards, and provenance proofs
Intra-swarm replica reuse	Downloaders of popular assets simultaneously act as providers, offloading origin server egress bandwidth	Enterprise network policies, observability, scheduling, and rate limiting
BTT bandwidth incentives	Establishing economic incentives for long-term seeding and prioritized provisioning	Pricing, budgeting, anti-fraud mechanisms, settlement, and compliance strategies
BTFS content-addressable storage	Providing the foundation for asset storage, sharing, and CID-based access	Private data encryption, access control, deletion/revocation, data sovereignty, and compliance

4.5.2 Artifact Distribution Conventions

Building on BitTorrent v2's metadata features to operationalize these capabilities, we introduce the **Artifact Torrent** as the standard distribution unit for models and datasets. This convention comprises three core specifications:

Immutable Addressing: Each model or dataset version maps to a unique infohash, defined entirely by its content, preventing in-place overwrites and replacing redirectable URLs with content digests. BitTorrent v2's per-file Merkle roots and publisher signatures combine to form complete supply chain verification credentials: the Merkle root guarantees byte-level integrity, while the signature validates publisher identity. Consumers can execute these verifications offline regardless of download source—whether peers, Web Seeds, or mirrors.

Structured Metadata: Model cards, dataset cards, license identifiers, and version lineage (such as parent infohashes) are bundled as structured metadata within the Artifact Torrent. Because licensing details are inseparably bound to the content, consumers natively acquire both terms of use and provenance lineage upon asset retrieval.

On-Demand Retrieval at File Granularity: Utilizing v2 per-file addressing, consumers can selectively pull specific components—such as weight shards of a target quantization precision or specific dataset subsets—without downloading the entire artifact.

4.5.3 Distribution Modes

Artifact distribution supports three composable deployment configurations:

Public Swarms: Public swarms facilitate the release and distribution of open-source models and public datasets. Publishers need only host an initial seed alongside a Web Seed source; post-release traffic peaks are absorbed directly by the aggregated swarm. Because higher download demand proportionally increases aggregate supply, swarm dynamics naturally alleviate egress bandwidth bottlenecks at the publisher's origin server.

Private Swarms: Private swarms address enterprise and consortium deployments where access is restricted to nodes holding valid cryptographic credentials. Designed to operate seamlessly across air-gapped intranets and dedicated leased lines, private swarms streamline large-scale internal synchronizations—such as rolling weight updates across inference clusters or bulk container/image deployment to edge nodes.

Hybrid-Source Distribution: Hybrid-source distribution enables frictionless, progressive adoption. By integrating Web Seeds, existing model registries and object storage buckets can participate directly as swarm data sources without architectural modifications, overlaying the P2P network as a bandwidth-offloading and acceleration tier. Publishers can incrementally offload egress traffic to the peer-to-peer layer while preserving their legacy storage infrastructure.

4.5.4 Long-Tail Availability and Incentives

Long-tail versions of models and datasets present an availability challenge—characterized by low retrieval frequency coupled with strict accessibility requirements. Scientific reproducibility requires referencing the exact versions utilized in research papers, while production systems require historical checkpoints for rollbacks. Under purely altruistic seeding, these low-popularity artifacts are the first to vanish from the network—a direct recurrence of the seeding problem outlined in Section 4.4.1 within AI workloads.

To address this, we reuse the incentive primitives of BitTorrent Speed without introducing any centralized value judgments. Any party requiring the long-term availability of a specific version—whether the publisher, a dependent project, or any third party—places continuous bids in BTT. This gives seeding nodes a clear economic rationale to retain the artifact and maintain its online availability. Which artifacts are worth retaining is determined entirely by market clearing rather than dictated by protocol operators, embodying the principles of protocol neutrality and market-driven discovery.

4.5.5 Positioning and Boundaries

We define the role of the BitTorrent ecosystem within AI infrastructure explicitly as the **distribution data plane** for models and datasets—rather than a model registry, a data governance platform, or an inference scheduling network.

This delineation is grounded in a candid assessment of its capabilities. Piece-level hash verification ensures byte-accurate transmission, but it does not independently prove publisher identity, data safety, or legal usage rights; furthermore, the distribution layer does not automatically resolve issues of version control, license enforcement, data privacy, or access revocation. For enterprise or regulated AI scenarios, upper-layer mechanisms—such as publisher signatures, chains of trust, model and dataset cards, version catalogs, access controls, encryption, and compliance audits as outlined in the right column of Table 4-2—must be overlaid atop BitTorrent and BTFS. These upper-layer capabilities are implemented by specific applications and third-party systems, allowing the distribution layer to remain neutral and generic. A universal, verifiable distribution layer deployed across hundreds of millions of devices can be confidently reused by higher-level governance frameworks precisely because it does not attempt to replace them.

For decentralized compute networks, the distribution layer serves as the final delivery pipeline for model-to-node deployment. Before executing a task, a compute node retrieves the model image from the nearest swarm via the Artifact Torrent, loading it into memory only after completing Merkle root and signature verifications. Cold-start bandwidth costs are absorbed by the P2P layer, and regional task demand for a specific model automatically translates into higher local distribution throughput. This layered architecture ensures that the BitTorrent distribution layer transcends being a standalone protocol—acting as the primary conduit facilitating AI data flows throughout the broader BTT ecosystem and enabling the seamless convergence of physical compute and storage resources with tokenized incentives.

5. Product II: BTTInferGrid—the Decentralized Compute Network of the Ecosystem

BTTInferGrid is a distributed compute network within the BitTorrent ecosystem tailored specifically for AI inference. By aggregating participant GPUs, it integrates off-chain task execution and result verification with on-chain coordination and settlement. The network drives resource provisioning based on real-world inference demand, aiming to lower the barrier for developers to acquire compute power. Simultaneously, it transforms idle GPUs into sustainable service capacities, provided they meet rigorous performance, reliability, and verification requirements.

5.1 Product Positioning: The Ecosystem Compute and Storage Layer

BTTInferGrid focuses on high-frequency, parallelizable AI inference workloads, providing developers and applications with elastically accessible compute resources. In contrast to centralized cloud infrastructure operated by a single provider, the network broadens the resource pool through distributed GPU provisioning. It ensures service quality through a combination of task scheduling, challenge-based verification, reputation scoring, and economic constraints. The architecture comprises four distinct components: hardware resources, off-chain execution, verification, and on-chain coordination. Within this framework, GPU nodes execute tasks, verification nodes evaluate outcomes, and on-chain smart contracts record performance scores, staking deposits, slashing events, and token rewards.

Storage capabilities are employed to manage model weights, input data, result digests, and task proofs, operating in seamless synergy with BitTorrent's large-file distribution capabilities. The core architectural boundary of BTTInferGrid lies in orchestrating compute resources and verifying service outcomes, rather than guaranteeing that all inference results can be directly proven on-chain. Consequently, different models and tasks must employ verification methods tailored to their specific cost, determinism, and security requirements.

5.2 Participant Roles and Resource Onboarding Mechanism

The network comprises requesters, compute nodes, validator nodes, and data service nodes. Requesters submit inference requests via APIs or applications and pay associated fees. Compute nodes contribute GPUs, executing tasks locally and returning results. Validator nodes handle node selection, challenge tasks, result scoring, and consensus aggregation. Once operationalized, data service nodes provide storage, availability, and retrieval services for models and task data.

Upon onboarding, compute nodes must register their hardware and pass checks covering hardware performance, network connectivity, availability, and benchmark tasks. The scheduling system continuously updates node ratings based on latency, task completion rate, challenge accuracy, and historical reliability. In its early stages, the network may regulate supply quality through performance thresholds and whitelisting, gradually decentralizing as verification and governance mechanisms mature. Validator nodes must meet accuracy and availability standards and stake BTT. Exiting, unbinding, and stake release are subject to a waiting period; validators may exit only after all incomplete tasks, pending disputes, and outstanding penalties are fully cleared.

5.3 AI Inference Workflow and Compute Scheduling

The inference service operates as a closed loop from request submission to settlement:

1. Task Submission: Users submit models, inputs, performance criteria, fee caps, and timeout parameters via APIs or app interfaces.
2. Resource Matching: Validator nodes select compute nodes based on GPU capacity, node scores, latency, availability, and task pricing.
3. Off-Chain Execution: Compute nodes fetch the required models and inputs, perform inference locally, and return outputs, execution metadata, and result summaries.
4. Result Scoring: Validator nodes evaluate results via sample checks, challenge tasks, or cross-node comparison, aggregating scores while filtering out anomalies.
5. State Recording & Settlement: Final scores and task statuses are committed on-chain, where smart contracts distribute fees and rewards based on verified workload, Quality of Service (QoS), and billing rules.

If a node times out, rejects a task, or returns invalid results, the scheduling module can reassign the task to alternative nodes and log the failure in the node's score. Retry behavior—including triggers, retry counts, and cost allocation—is governed by the task policy. Applications requiring high availability may opt for redundant execution or higher SLA tiers, incurring corresponding additional costs.

5.4 Verifiable Inference and QoS Mechanisms

Off-chain inference carries risks such as execution errors, computation skipping, cached output reuse, validator bias, and collusion. BTTInferGrid addresses these challenges through a framework centered on redundant evaluation, incentive alignment, and progressive verification: validator nodes randomly inject challenge tasks—virtually indistinguishable from genuine requests—into the workload, evaluating responses against expected statistical or semantic properties. A single compute node is scored independently by multiple validators; these scores are then aggregated via Yuma Consensus to filter out extreme outliers, mitigating single-point manipulation and score inflation via collusion.

Node scoring factors in challenge accuracy, completed task volume, latency, availability, and historical reliability. Compute node rewards are tied to consensus scores and verified workload. If a validator node's scores consistently deviate from consensus, its reputation weight and reward allocation are downgraded; severe anomalies trigger stake slashing. The verification system can progressively evolve from statistical consensus and challenge audits to Trusted Execution Environments (TEEs), deterministic re-execution, and Zero-Knowledge (ZK) proofs, with clearly defined scopes and cost trade-offs for each approach. For non-deterministic generative models, verification results denote compliance with protocol-defined confidence thresholds rather than absolute mathematical exactness.

5.5 Distributed Storage and Data Service Capabilities

Functioning as an extensible capability alongside inference, distributed storage stores and distributes model weights, input data, execution logs, result summaries, and verification credentials—minimizing redundant transfers and maximizing task data availability. Large files can be distributed across the BitTorrent network, with BTTInferGrid binding verifiable file identifiers, versioning info, and access conditions to specific tasks.

The storage service defines specifications for client-side encryption, file chunking, redundancy policies, integrity verification, proofs of retrieval, gateway permissions, and data deletion rules.

When handling sensitive data, the system strictly adheres to the principle of minimal disclosure: raw inputs and private outputs are never exposed directly on-chain. The blockchain records only task statuses, hash commitments, and data essential for settlement. These capabilities will be rolled out progressively following the completion of technical specifications, performance benchmarking, and security audits.

5.6 BTT Settlement, Node Incentives, and Penalties

BTT serves as the medium of payment for inference and downstream data service fees, while also acting as the asset for validator staking and penalties. User fees can be metered based on request counts, compute duration, throughput, model specifications, and service tiers. Pricing dynamically adjusts according to available GPU supply, inference demand, and node quality, with optional cost floors to prevent nodes from operating at a loss—which leads to degraded service—when revenues fail to cover hardware, power, and bandwidth costs.

Compute node earnings are jointly determined by verified workload, consensus score, task completion rate, and historical reliability; idle capacity or unverified work earns no meaningful rewards. Validator nodes receive rewards for executing challenges, scoring, and identifying anomalies. When a validator deviates from consensus, manipulates scores, or participates in collusion, its rewards are reduced, and its stake may be subject to partial or total slashing. Protocol fees may be allocated to network operations, reward pools, or processed according to defined rules. Long-term incentives should progressively transition from subsidies and token emissions toward organic user fees, linking supply expansion directly to network utilization to mitigate the risks of unbacked expansion and reward dilution.

5.7 Developer Interfaces and Application Integration

Developers submit tasks, query statuses, retrieve results, and inspect fees via a unified API, specifying service requirements by model, latency, budget, verification level, and redundancy strategy. The interface returns task IDs, node selection statuses, itemized billing, verification results, and error codes, while supporting timeouts, cancellations, retries, and asynchronous callbacks. SDKs encapsulate authentication, model parameters, file identifiers, payment handling, and output parsing, minimizing integration overhead for existing AI applications.

Production environments further require testnets, quota management, price discovery, node/task monitoring, SLA metrics, and developer support channels. This whitepaper outlines conceptual directions—such as API/application interfaces, enterprise-dedicated compute pools, and SLA tiers—whereas specific SDKs, testnets, and monitoring utilities remain subject to formal product specifications and are not presented as live features.

6. Product III: BTTC—On-Chain Execution and Ecosystem Settlement Layer

BTTC serves as the on-chain execution and settlement infrastructure for the BitTorrent ecosystem, providing unified contract execution, state recording, and value settlement capabilities for compute, storage, distribution, and other ecosystem services. Leveraging an EVM-compatible execution environment, a validator network, and the BTT economic mechanism, BTTC structures service requests, operational statuses, and settlement outcomes into transparent, verifiable, and auditable on-chain workflows.

6.1 Product Positioning, Target Users, and Use Cases

BTTC is built around four primary ecosystem participants: everyday users can access on-chain applications cost-effectively and settle payments in BTT; content publishers, along with data and model providers, can register service offerings, licensing authorizations, and settlement statuses; developers can deploy DeFi, GameFi, payment, and ecosystem service contracts within an EVM-compatible environment; while compute and storage service providers can execute task acknowledgments, service metering, staking constraints, and revenue settlements via BTTC.

The value of BTTC lies in establishing a composable, trustless coordination layer for ecosystem services. BTTInferGrid, the BitTorrent distribution network, or third-party applications can submit service requests, node states, and settlement proofs to BTTC, where smart contracts execute state verification, fee debits, reward distributions, and dispute logging according to predefined rules. For high-frequency applications, BTTC provides a low-cost contract execution environment; for resource-heavy services, it supplies unified interfaces for task registration, proof submission, and on-chain settlement.

6.2 Architecture: Three Functional Layers and Key Roles

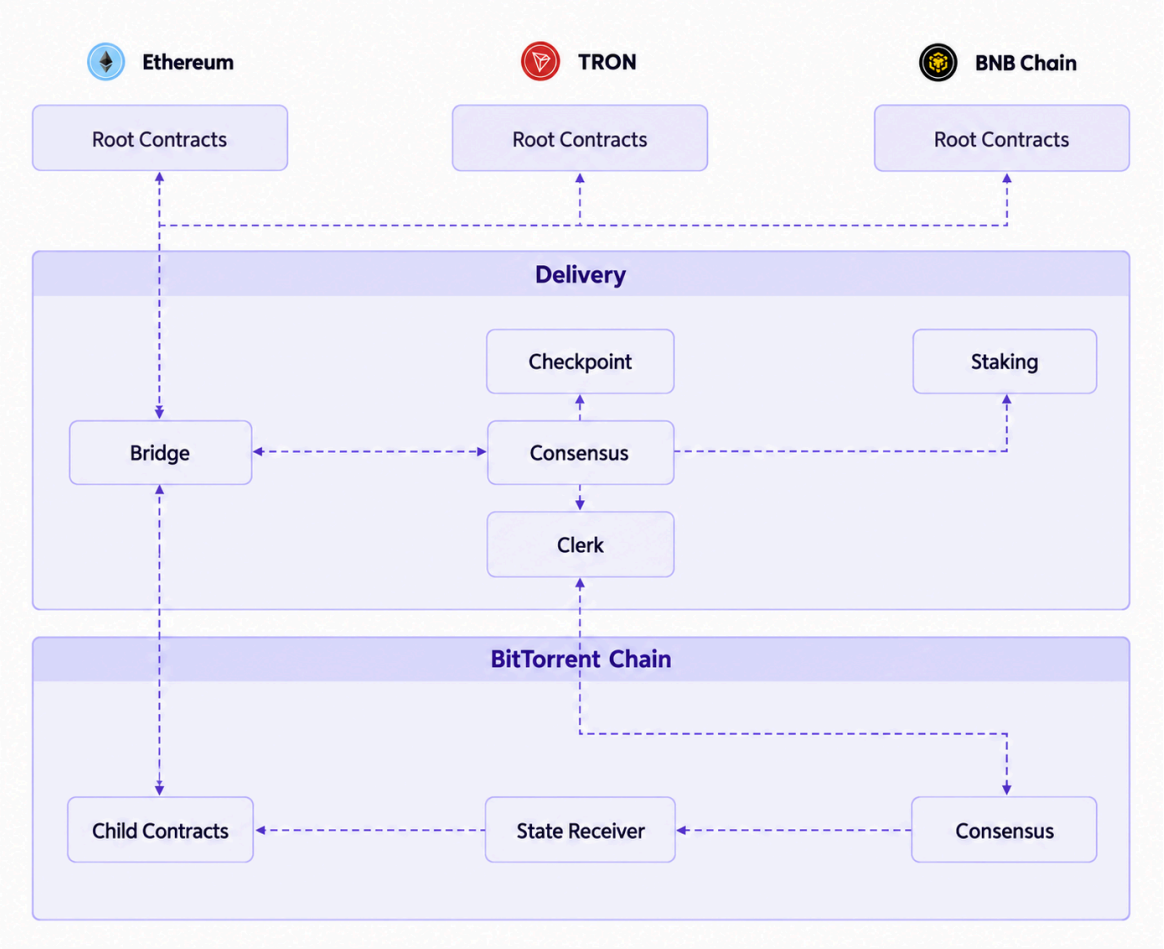


Figure 6-1 BTTC's Three-Layer Architecture

BTTC employs a three-layer modular architecture where each layer operates with distinct responsibilities, collaborating through on-chain events and consensus mechanisms:

- Contract Layer (Root Contracts): Serves as the entry point for network governance and asset management, responsible for validator registration, staking and delegation, parameter configuration, fee management, and state recording for ecosystem service contracts.
- Delivery Chain: Handles validator coordination, consensus confirmation, and network governance state maintenance. Validators verify network events and state changes, forming a consensus result through PoS/BFT consensus to provide a stable verification and governance foundation for the operation of the BTTC Chain.
- BTTC Chain: Handles smart contract execution and operational state hosting, compatible with EVM and Solidity, responsible for processing transactions, executing application logic, recording service states, and settling fees and revenues.

Key roles in the network include validators, block producers, delegators, and developers. Validators participate in Delivery Chain consensus and network governance; block producers are rotated from validators to generate BTTC Chain blocks, handling transaction batching and state execution; delegators participate in the network and share incentives through staking or delegation; while developers deploy applications, service protocols, and settlement contracts. The permissions, staking requirements, and operational records of each role are strictly governed by on-chain rules.

6.3 Mechanisms: Service Execution and On-Chain Settlement Workflow

BTTC structures service requests, execution confirmations, and fee settlements into a standardized on-chain workflow across five sequential stages:

1. Request Registration: Users or applications submit task, resource utilization, or payment requests to service contracts, which log request parameters, fee caps, and unique request identifiers.
2. Service Confirmation: Compute, storage, or distribution nodes accept tasks, and the system logs the service provider, execution conditions, and expected delivery status.
3. Result Submission: Upon service completion, nodes or verification modules submit result summaries, completion proofs, and required performance indicators.
4. Rule Verification: Smart contracts inspect submitted results against protocol-defined states, signatures, timeouts, and billing rules, rejecting duplicate, expired, or non-compliant submissions.
5. Settlement Execution: Upon successful verification, smart contracts deduct fees, distribute node earnings, process protocol fees, and archive states using BTT. In the event of anomalies, the system triggers retry, slashing, or dispute resolution workflows.

This workflow bridges service delivery with financial settlement, ensuring end-to-end traceability for requests, executors, result summaries, and payment records. Diverse ecosystem services can reuse unified account, gas fees, staking, and settlement capabilities, while defining custom task formats, QoS standards, confirmation conditions, and dispute rules tailored to specific business needs. BTTC handles exclusively on-chain coordination and settlement, operating alongside off-chain execution services such as compute execution, file transfer, and storage retrieval.

6.4 Consensus Mechanism and Security Model

The security of BTTC relies on a clear division of labor between Delivery Chain consensus and BTTC Chain block production. The Delivery Chain employs a Tendermint-based PoS/BFT consensus mechanism: in each round, a designated proposer submits a block or state proposal, after which validators sequentially perform pre-vote and pre-commit steps. State transitions achieve finality only upon securing signatures from over two-thirds of active validators; if faulty or malicious nodes exceed one-third, consensus halts rather than committing corrupted state, thereby preserving network state consistency.

The BTTC Chain utilizes a rotating validator mechanism for block production. Block producers batch transactions, execute smart contracts, and update execution state; this deterministic rotation enhances throughput and block production predictability. Producers that miss consecutive block windows can be replaced, mitigating the impact of single-node failures on network performance. To bolster security, validator signing keys are strictly decoupled from control permissions; critical state updates must meet threshold signature requirements and undergo on-chain contract verification in accordance with protocol rules.

The security model covers the entire lifecycle of service registration, state progression, and settlement execution: every request carries a unique identifier to prevent duplicate processing; block height, timestamps, and state-machine constraints restrict replay attacks and out-of-order execution; critical states must be confirmed via consensus and cannot be modified independently by any single validator; and smart contracts accept only inputs that strictly conform to format, permission, and signature specifications. Validator staking, signing attendance, block production performance, and delegation relationships are recorded on-chain, establishing an objective foundation for slashing penalties, node replacement, and community oversight.

7. BTT Economic Model and Token Design

BTTC's economic model is designed to foster a self-sustaining, robust, and resilient incentive system resistant to centralized abuse. By leveraging staking mechanisms, transaction fees, delegation, and governance tools built around the native token BTT, BTTC ensures sustained network security, active validator engagement, and dynamic ecosystem growth, delivering stable, long-term support for decentralized cross-chain architecture.

7.1 Core Token: BTT

The BitTorrent Token (BTT) serves as the foundational asset within the BTTC network, enabling essential functions including staking, incentives, and on-chain payments. Validator nodes must stake BTT to participate in consensus mechanisms. Cross-chain activities generate transaction fees and rewards that are paid in BTT, which also serves as the medium of exchange for on-chain interactions, smart contract deployments, and data storage. BTT maintains a fixed supply managed strictly by on-chain smart contracts, ensuring full transparency in all economic activities.

The original total issuance of BTT was 990,000,000,000 tokens. After the BTTC mainnet launched and BTT was redenominated, legacy tokens were designated as BTTOLD, with new BTT distributed at a 1:1,000 swap ratio to adjust the total supply to 990,000,000,000,000. This redenomination does not alter BTT's underlying market value or original allocation proportions.

7.2 BTT Token Distribution

The initial BTT token distribution is allocated across public sale, private sale, seed round, airdrops, team and foundation, ecosystem building, and partnerships. The table below shows each category's share based on the original issuance metrics, along with the corresponding amount under the new BTT denomination.

Table 7-1 Initial BTT Allocation Breakdown

Category	Allocation	Amount (New BTT)	Description
Public Sale	6.0%	59,400,000,000,000	Public Sale
Private Sale	2.0%	19,800,000,000,000	Private Sale
Seed Sale	9.0%	89,100,000,000,000	Seed Round
TRON Airdrop	10.1%	99,990,000,000,000	TRON User Airdrop
BitTorrent Protocol Airdrop	10.0%	99,000,000,000,000	BitTorrent Protocol User Airdrop
Team	19.0%	188,100,000,000,000	Team and BitTorrent Foundation
TRON Foundation	20.0%	198,000,000,000,000	TRON Foundation
BitTorrent Ecosystem	19.9%	197,010,000,000,000	BitTorrent Ecosystem Building
Partnerships	4.0%	39,600,000,000,000	Partnerships

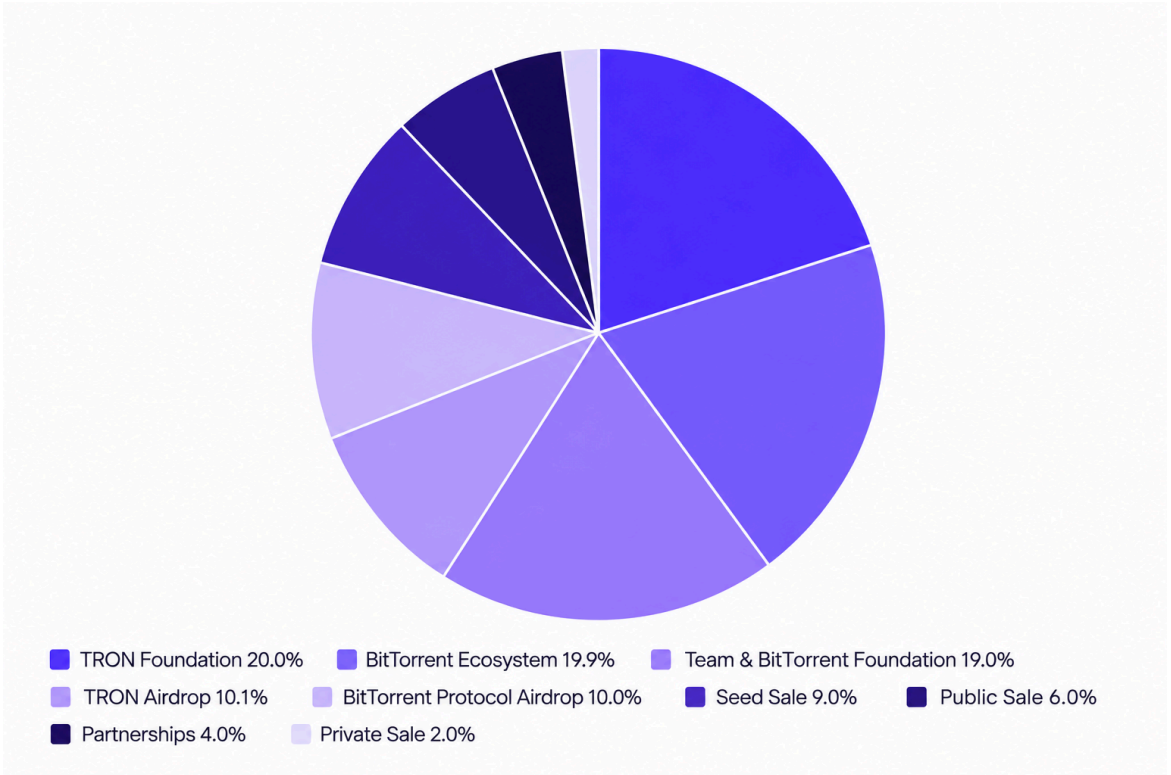


Figure 7-1 Initial BTT Token Distribution

7.3 Staking and Reward Mechanisms

BTTC uses a BTT-based staking framework for decentralized consensus and governance. Anyone looking to engage with network operations and earn rewards may choose to stake BTT and run their own validator node, partner with others (see validator partnership model), or delegate tokens to participate in consensus. Smart contracts handle all staking and rewards allocation on-chain, ensuring transparency, fairness, and credibility. Staking rewards are capped and released annually at a preset rate. The BTTC 2.0 upgrade introduces dynamic controls that adjust release timing and allocation ratios based on network conditions and community governance. Smart contracts manage the entire incentive system without centralized oversight.

7.3.1 Validator Staking Mechanism

Validators are the key actors in the BTTC consensus layer. They sign cross-chain checkpoints on the Delivery Chain and take turns producing blocks on the BTTC Chain. To become a validator, participants must stake at least one trillion BTT tokens in the root contract on TRON MainNet and bind a unique NFT identity. Design highlights:

- Equal Voting Weight: Each validator gets one vote per node, regardless of stake size, preventing voting power concentration.
- Lock-up & Exit Periods: Lock-up and exit periods apply to BTT staking, preventing short-term arbitrage and frequent hops among validators.
- On-Chain Transparency: All staking and reward data is publicly available on-chain, providing full transparency for audits and community oversight.
- Incentives:
  - Signature Rewards: Validators earn a share of cross-chain transaction fees for signing checkpoints on the Delivery Chain.
  - Block Rewards: Validators earn a share of transaction fees when producing blocks on the BTTC Chain.
  - Checkpoint Bonuses: Once a checkpoint created on the Delivery Chain is successfully verified by the destination chain, the system grants extra rewards to the validators who signed and submitted the proof for that round.

Rewards are distributed via smart contracts, ensuring fairness and on-chain auditability.

7.3.2 Validator Partnership Model

To accommodate real-world scenarios where node operation and token capital are managed by different parties, BTTC natively supports a validator partnership model. This allows multiple addresses to jointly operate as a single validator node. Design highlights:

- Multi-Address Collaboration: Multiple addresses can stake collectively and be treated as one validator by the system.
- Automatic Reward Distribution: Rewards from signing and block production are automatically distributed based on each address's staking share.
- On-Chain Execution: Smart contracts manage reward payouts entirely on-chain, removing the need for external agreements and their associated risks.
- Broad Applicability: Ideal for collaborative setups involving DAOs, communities, node service providers, and stakeholders.

The validator partnership model only expands how validators organize; it neither changes the governance framework nor introduces new roles, yet it substantially boosts both flexibility and network decentralization.

7.3.3 Delegated Staking Mechanism

Through delegated staking, BTT holders can participate in network consensus and earn rewards, without the need to run a node themselves. Design highlights:

- Flexible Delegation: BTT holders can freely choose and delegate to validators via the TRON MainNet.
- Reward Sharing: Validators share their signing and block-production rewards with delegators in proportion to each delegator's stake.
- Lock-up & Exit Periods: Delegated tokens are locked up on-chain and remain locked until the exit period ends.
- Value Drivers: By reducing user entry hurdles, the mechanism strengthens validator network resilience and invigorates governance across the ecosystem.

While delegators do not run nodes themselves, they can earn on-chain rewards through a staking model designed to be fair and transparent.

7.4 Unlock and Release Schedule

BTT has completed its redenomination. Historical BTTOLD amounts referenced below can be converted to the new BTT at a 1:1,000 ratio. Allocation proportions and release logic remain unchanged.

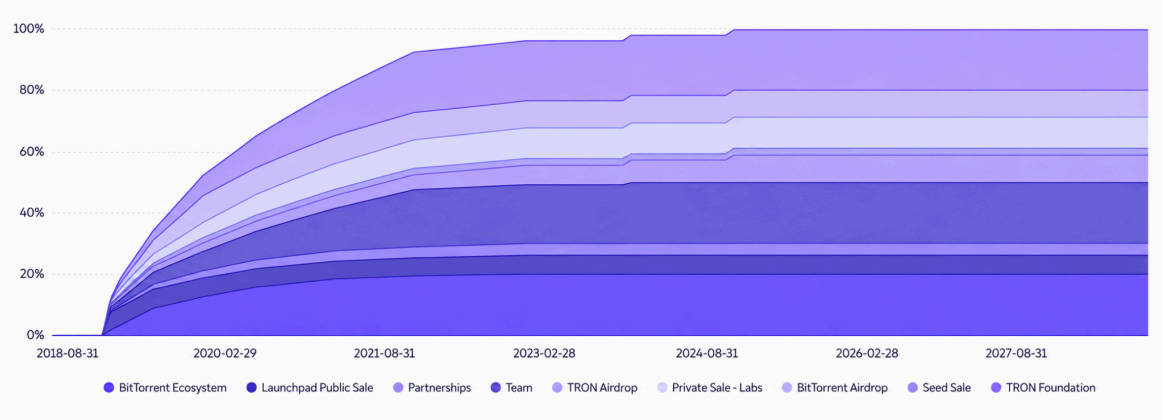


Figure 7-2 BTT Token Release Schedule

Table 7-2 BTT Release Schedule and Rules

Category	Allocation	Release Start	Release Completion	Release Rule
Launchpad Public Sale	6.0%	2019-01-31	2019-01-31	6.0% released in full at TGE/completion of the public sale
Private Sale	2.0%	2019-01-31	2019-12-31	Linear release over 12 months
Seed Sale	9.0%	2019-01-31	2019-12-31	Linear release over 12 months
Team	19.0%	2019-01-31	2021-12-31	Linear release over 36 months

Category	Allocation	Release Start	Release Completion	Release Rule
TRON Foundation	20.0%	2019-01-31	2021-12-31	Linear release over 36 months
TRON Airdrop	10.1%	2019-01-31	2025-02-11	The first airdrop took place on January 31, 2019, with 2,970,000,000 BTTOLD (0.3% of total supply); the second took place on February 11, 2019, with 10,890,000,000 BTTOLD (1.1% of total supply). From March 11, 2019, to February 11, 2025, monthly distributions were made to TRX holders, increasing year by year: 990,000,000; 1,072,500,000; 1,155,000,000; 1,237,500,000; 1,320,000,000; and 1,402,500,000 BTTOLD, respectively.
BitTorrent Airdrop	10.0%	2019-01-31	2029-01-31	Front-loaded release tapering over time, with 10.0% completed in January 2029
BitTorrent Ecosystem	19.9%	2019-01-31	2029-01-31	Front-loaded release tapering over time, with 19.9% completed in January 2029
Partnerships	4.0%	2019-01-31	2029-01-31	Front-loaded release tapering over time, with 4.0% completed in January 2029

7.5 Buyback and Burn Mechanism

To enhance BTT's value capture and deflationary framework, BitTorrent will launch a long-term buyback and burn program starting in Q3 2026. All revenue generated across BitTorrent's decentralized operations will be allocated to repurchasing BTT on the open market at prevailing prices.

On October 16, 2026, BitTorrent executed its first BTT buyback and burn, permanently removing 21,621,611,510,729.6 BTT from circulation by sending them to a burn address. This represents approximately 2.1840% of the initial total supply under the new BTT denomination. Transaction hash: 4d9596617780e329bbd4e644ee0dffd780ec4cf3749f324d03eb98157eb04ae5.

As decentralized services such as BTTInferGrid scale, top-line revenue growth will expand the capital pool for subsequent repurchases. This links token supply contraction directly to real ecosystem cash flows.

8. Application Scenarios & Ecosystem Development

By synergizing the BitTorrent file-sharing protocol, BTTInferGrid, and BTTC, the ecosystem delivers core infrastructure for decentralized AI inference, model and dataset distribution, cross-chain liquidity, and multi-chain governance.

8.1 Decentralized AI Inference and Compute Services

Developers and applications can submit inference workloads to BTTInferGrid, where the network schedules and matches tasks to qualifying GPU nodes based on job specifications, hardware profiles, and node availability. Once computation is complete, nodes return inference outputs, which are evaluated by verification mechanisms to ensure task validity and output integrity.

Workload states, service fees, and node incentives are recorded and settled on-chain via BTTC. This forms an end-to-end operational pipeline: Job Submission → Compute Orchestration → Inference Execution → Output Verification → On-Chain Settlement. The architecture is optimized for production-grade API consumption, large-model inference, and high-concurrency batch processing.

8.2 P2P Distribution for Models, Datasets, and Large-Scale Content

Large AI model weights, training corpora, and associated binary artifacts typically involve massive file sizes, high latency, and prohibitive centralized bandwidth costs. Leveraging its established peer-to-peer network, BitTorrent fragments these large assets into discrete chunks, distributing propagation across globally decentralized nodes so clients can pull data concurrently from multiple peers.

Model creators, researchers, and content publishers can distribute models, training sets, and data bundles across the BitTorrent network. Cryptographic content hashes guarantee data integrity, while official channels and cryptographic signatures verify provenance. Supported by BitTorrent Speed and localized seeding incentives, the network achieves high throughput, low latency, and sustained data availability for large-scale AI assets.

8.3 Cross-Chain Bridges and Asset Interoperability

BTTC leverages the Delivery Chain to establish a cross-chain data and state relay mechanism, providing a unified channel for asset transfers and arbitrary message passing across heterogeneous blockchain networks. The protocol natively supports major token standards—including ERC-20, TRC-20, and BEP-20—and relies on a checkpoint mechanism to validate and broadcast cross-chain events.

When a user initiates an asset lock on the source chain, the Delivery Chain detects and verifies the event, compiles a cryptographic checkpoint, and submits it to the destination chain. Upon verification, the destination chain releases or mints the corresponding asset, completing the transfer. Developers can also leverage BTTC's cross-chain primitives to build multi-chain DEXs, cross-chain lending markets, liquidity aggregators, and other composable multi-chain applications.

8.4 DAO Governance and Voting System

As DAOs increasingly deploy across multiple blockchain networks, member identities, governance tokens, and voting records become fragmented across disparate chains. BTTC's state synchronization and cross-chain verification mechanisms bridge governance data across these networks, establishing a unified foundation for multi-chain state relay and execution.

Governance proposals can be initiated on a designated network, with token holders participating according to protocol rules. Once voting concludes, verified results synchronize to relevant chains or governance contracts for on-chain execution. Developers can route cross-chain voting, treasury disbursements, and protocol parameter updates through BTTC, creating an end-to-end governance lifecycle from proposal submission to execution.

9. Roadmap and Ecosystem Development Plan

Centered on AI and large-scale model applications, the BitTorrent ecosystem will drive the convergence of model distribution, decentralized inference, and on-chain settlement,

systematically establishing an open service network that connects model developers, compute providers, and application teams.

9.1 Phased Roadmap

Phase 1: Foundation and Core Infrastructure

Streamlining open-source model distribution, version integrity checks, and GPU node onboarding, the team will complete the end-to-end integration of BTTInferGrid inference, output verification, and BTT fee settlement, establishing baseline service infrastructure for large-model applications.

Phase 2: Capability Expansion and Enterprise Readiness

The team will progressively extend support to text, image, and speech models, enable workloads such as knowledge-base question answering and Retrieval-Augmented Generation (RAG), and refine elastic scheduling and data access management to meet the deployment needs of developers and enterprises.

Phase 3: Open Ecosystem and Agentic Workflows

The roadmap will explore multi-model invocation, tool collaboration, and continuous task execution. Leveraging BTTC to refine cross-service metering and settlement, it will support the growth of AI agent applications and drive open collaboration across models, compute, and application services.

Each phase will advance progressively based on technical validation, service quality benchmarks, and market demand, with specific milestones tracked and updated alongside product releases.

9.2 Ecosystem Highlights

- **Model and Compute Partnerships**: Connect open-source model communities with GPU resource providers to advance model adaptation, benchmark performance, and expand reliable node capacity.
- **Developer and Application Support:** Refine APIs, SDKs, sandboxes, and starter templates while deploying technical partnerships and phased grant programs to accelerate production deployments.
- **Sustainable Incentives:** Align BTT payment flows and incentive structures with verified service requests and active resource contributions, ensuring network expansion matches real utility demand.

9.3 Security and Governance

Continuously advance smart contract and codebase audits while upgrading model provenance verification, inference validation, and data protection mechanisms. Major protocol adjustments will be introduced through public proposals, community review, and staged rollouts, with ecosystem growth evaluated against uptime, verified compute throughput, and developer retention.

10. Conclusion

As Web3 and AI continue to converge, model and dataset distribution, compute provisioning, cross-chain communication, and value settlement are emerging as essential pillars of next-generation digital infrastructure. Building on its established P2P network and cross-chain technology, the BitTorrent ecosystem is further expanding into decentralized AI compute capabilities, committed to seamlessly connecting users, developers, node resources, and blockchain networks.

Within this architecture, BitTorrent delivers peer-to-peer distribution for models, datasets, and large-scale content. BTTInferGrid networks globally distributed GPU nodes to supply developers and applications with decentralized inference compute. Leveraging a three-layer architecture, PoS consensus, and cross-chain relays, BTTC provides cross-chain interoperability and on-chain settlement for digital assets, data, and service states. Anchoring these layers, BTT serves as the underlying economic engine, integrating operational utility across every product line with on-chain incentives.

These three product lines unite through a continuous pipeline—Model & Data Distribution → Decentralized Inference → Cross-Chain Interoperability & On-Chain Settlement—transforming fragmented bandwidth, compute, and blockchain resources into unified infrastructure for Web3 and AI applications. Moving forward, the BitTorrent ecosystem will continue advancing open access, cryptographic verification, cost-efficient execution, and sustainable coordination, refining network capabilities and developer support frameworks as the ecosystem scales.

We invite users, compute providers, validators, engineering teams, AI application developers, and ecosystem partners worldwide to build with BitTorrent and shape an open, efficient, and verifiable decentralized infrastructure.