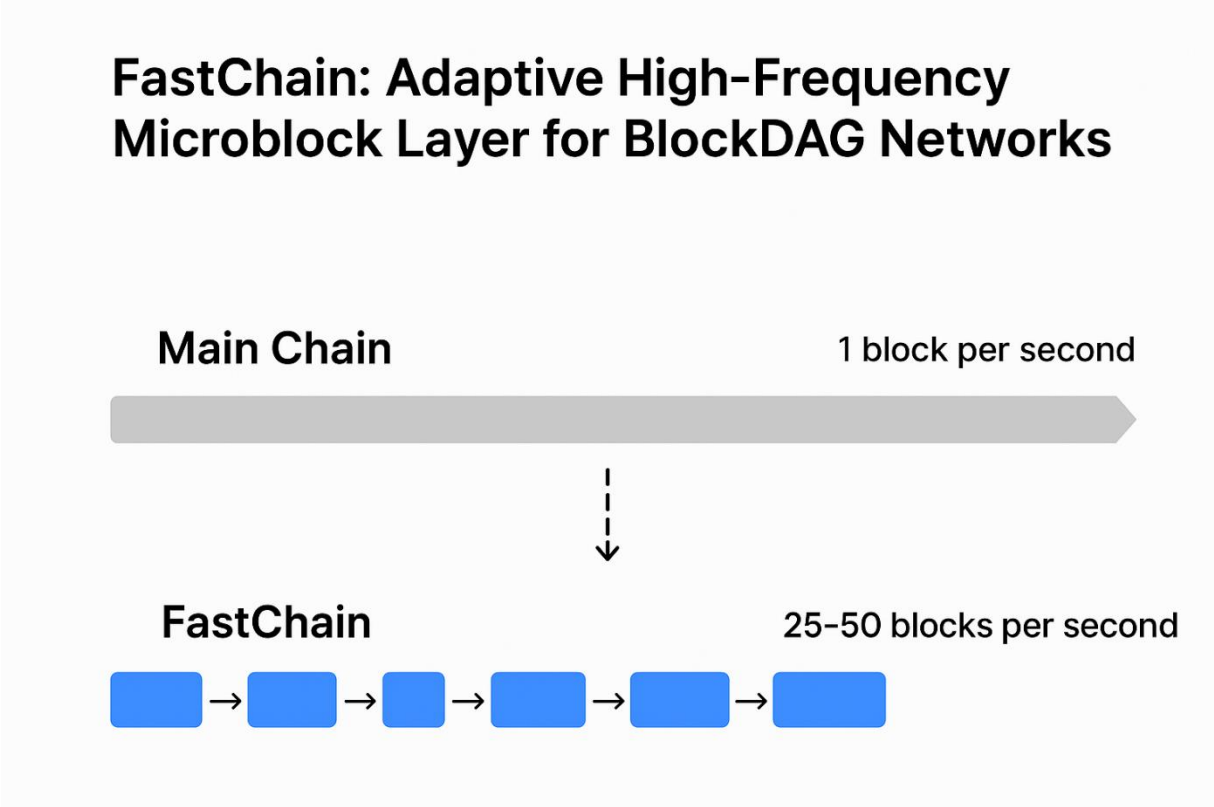
FastChain: Adaptive High-Frequency Microblock Layer for BlockDAG Networks

@Cryptis – Cryptix-Network.org

Abstract



FastChain is a high-frequency microblock layer that operates in parallel with a DAG-based main chain network. The system is designed to achieve stable transaction throughput of around 25 blocks per second (BPS) on standard hardware. Under optimal, locally controlled conditions, throughput of up to 50 BPS is theoretically possible.

FastChain combines four core mechanisms:

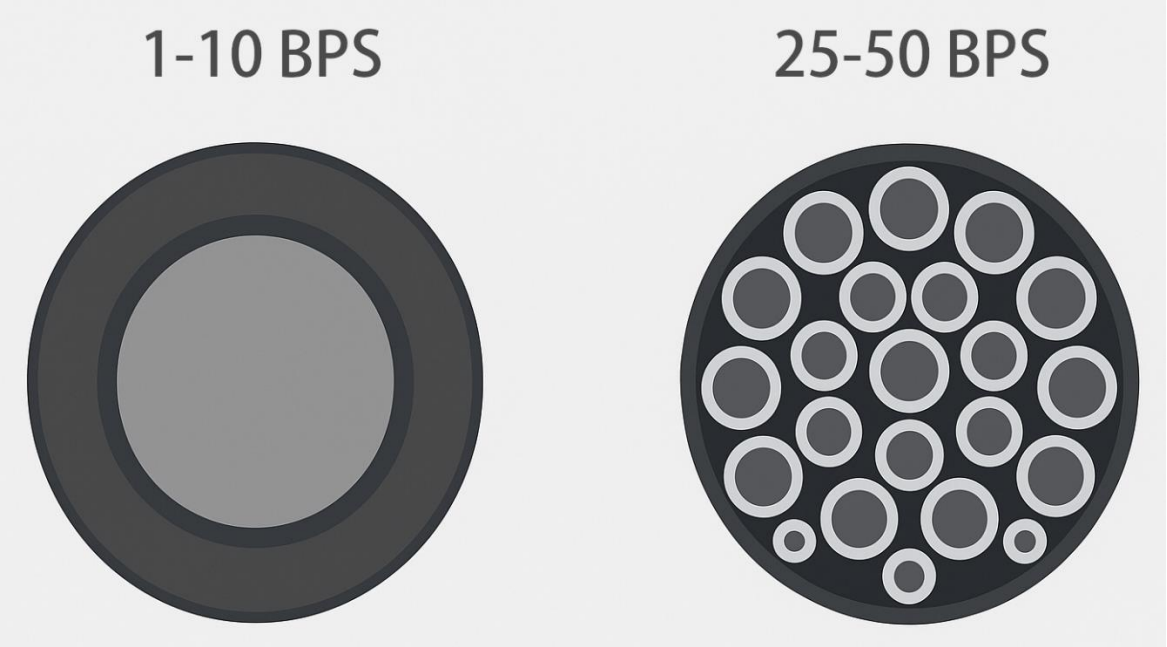
1. Small, parallel microblocks enabling rapid transaction processing.
2. Adaptive control of block rate and network fanout to minimize latency and forks.
3. Lazy-loaded transaction data, reducing bandwidth consumption and node load.
4. Dynamic reward mechanisms that incentivize participation and maintain network consistency.

The architecture is based on three foundational principles:

- **High-frequency microblock production within a DAG:** Allows parallel processing and rapid propagation of transaction IDs.
- **Layered finality:** Separates fast transaction capture from definitive finality.
- **Adaptive control mechanisms:** Automatically tune block rate and network parameters based on real-world physical limits such as latency. Rather than imposing hard protocol limits, the system optimizes for globally stable performance.

---

1. Motivation



Even modern BlockDAG systems such as Kaspa currently achieve only ~10 BPS globally. Effective scaling is limited less by the nominal block rate and more by global network factors such as latency, bandwidth, peer topology, and the inherently stochastic (non-deterministic) block generation in PoW-based DAGs.

Common issues include:

- High fork rates when multiple blocks are produced nearly simultaneously.
- Network congestion when header propagation and validation lag behind block production.
- Delayed finality due to increasing reorganization depth as block rate increases.

While DAG-based approaches fundamentally improve scalability compared to linear blockchains, publicly tested systems rarely exceed ~10 BPS in practice.

FastChain addresses these limitations through:

- **Parallel microblock production:** Multiple microblocks can be created and disseminated concurrently.
- **Adaptive block rate and fanout control:** The system dynamically adjusts block rate in response to latency, fork rate, and mempool growth.

- **Layered finality:** A fast capture layer for rapid transaction intake combined with a main chain for irreversible finality.

Together, these techniques enable stable throughput around 25 BPS and up to 50 BPS under optimal conditions. Achieving global 50 BPS is unlikely without additional technologies (compression, preloading, predictive deterministic mechanisms). Nonetheless, FastChain significantly reduces network instability.

---

2. Architecture

2.1 Microblock Layer

FastChain operates an independent microblock layer parallel to the main chain. Microblocks are extremely small (50–100 bytes) and contain only essential metadata:

- Transaction IDs (1–3 per block)
- Parent IDs (2–3), forming the DAG structure
- Optional signature
- Timestamp

Full transaction bodies are retrieved on demand (lazy-load). This saves bandwidth and allows nodes to operate within their performance limits.

2.2 Layered Finality

FastChain separates rapid transaction processing from irreversible finality:

- **Capture Layer (FastChain):** Produces microblocks at high frequency (25–50 BPS) and propagates transaction IDs quickly. Underperforming nodes may be temporarily excluded.
- **Finality Layer (Main Chain):** Aggregates microblocks, validates full transactions, and provides irreversible finality. Nodes remain active here even when excluded from the fast layer.

This separation allows fast processing without compromising the security of the main chain.

2.3 Adaptive Dual-Loop Control

FastChain uses two complementary feedback loops:

1. **Fast Gossip Loop**
    - Interval: 5–20 ms (depending on target BPS)
    - Dynamically adjusts fanout for efficient microblock header propagation
    - Objective: minimize latency and forks

2. **Slow Blockrate Loop**
    - Interval: 10–60 seconds
    - Adjusts microblock interval based on p90 header propagation, fork rate, and mempool growth
    - Objective: stabilize block rate around ~25 BPS; up to 50 BPS locally

Together, the loops adapt block production to real-world network conditions and node capabilities. The network automatically converges to its physical performance limits.

---

3. Adaptive Node Rewards

FastChain implements a dynamic incentive system to reward participation while maintaining stability:

- Rewards are linked to microblock production but adjusted to avoid network congestion.
- Underperforming nodes may be temporarily excluded from the fast layer while remaining active on the main chain.
- Goals: economic equilibrium, low fork rate, and long-term participation.

---

4. Lazy-Load Transaction Data

- Microblocks contain only transaction IDs.
- Full transaction data is fetched only when needed, depending on node performance.

Benefits:

- Faster header propagation
- Reduced bandwidth load
- Stable microblock frequency

This allows nodes to operate efficiently without pushing the network beyond its global limits.

---

5. Network Economics and Incentives

- Nodes participating in the fast layer receive additional rewards.
- Underperforming nodes are temporarily excluded to maintain stability.
- Adaptive rewards prevent overproduction, ensure consistency, and promote balanced participation.

---

6. Bandwidth and Feasibility Analysis

Block interval:

- 25 BPS → 1 block every 40 ms
- 50 BPS → 1 block every 20 ms

Fast Gossip Loop:
Must be faster than the block interval: 5–20 ms.

Slow Blockrate Loop:
10–60 seconds.

Example throughput calculation:
Maximum_BPS = (U × e) / (B_s × k)

Where:

- B_s = 100 bytes (block size)
- k = 6 (average parents)
- e = 0.8 (efficiency factor)
- U = 100 KB/s (upload bandwidth)

Therefore:
Maximum_BPS ≈ 133 (theoretical, idealized)

Storage requirement:
100 bytes × 50 BPS × 86,400 seconds/day ≈ 0.432 GB/day

CPU:
Less than 5000 signature verifications per second → standard hardware is sufficient.

The network automatically adjusts the block rate according to real latency and bandwidth conditions.

---

7. Summary

FastChain enables robust, scalable, and adaptive high-frequency transaction processing:

- Parallel microblocks and layered finality provide fast processing without sacrificing irreversible finality.
- Adaptive control maintains stable block rates and considers physical network limits.
- Lazy-loading minimizes bandwidth and supports heterogeneous node performance.
- Economic incentives encourage long-term participation without endangering the main chain.

Key advantage:
No hard protocol limits. Instead, adaptive optimization enables maximum stable performance under real-world global conditions.

FastChain provides a practical solution for decentralized high-frequency micropayments and scalable DAG-based blockchain networks.

Personal Note:

It's important to understand that forcing a fixed BPS (Blocks Per Second) is one of the biggest mistakes developers can make. The most crucial factor—the network's health and stability—is uncontrollable and determined by the users. To maximize network speed, it must be adaptive, dynamically adjusted based on predefined factors. This approach eliminates the need to select a lower fixed value, even if temporarily a 2–3x increase in speed is possible but not sustainable. Testing and enforcing a fixed BPS therefore becomes unnecessary.

Another key point is that not every block requires the network's full throughput. Even if, theoretically, each block could be 100 KB, this will never happen in practice—not even for an hour. This allows for the concept of a "Fastchain," which increases effective throughput. Even if a single blockchain were used by 60% of all crypto users worldwide, the network would not need its maximum capacity. Developers often strive for extremely high TPS (transactions per second), but these high values are rarely needed now or in the foreseeable future. While high theoretical speeds may look impressive on paper, they are rarely practical. Prioritizing **high real-world speeds, efficient data transfer, and energy-conscious operation** is likely a better approach. It also allows broader accessibility, including for users with older hardware.

Block rewards can also be dynamically adjusted based on the block rate. Similarly, high-performance and fast nodes should be rewarded. Currently, there is little incentive to run a node in the crypto ecosystem, which has led to mining pools handling roughly 80% of activity.

Python-like pseudocode:

```
#########################
# Parameters
base_fanout = 7
min_fanout = 5
max_fanout = 10
target_hdr_p90 = 10.0    # ms

min_block_interval = 20    # ms (50 BPS max)
max_block_interval = 1000  # ms (1 BPS min)
current_interval = 20      # ms start

fork_target = 0.02         # 2% fork rate ideal
mempool_target = 10        # transactions per second growth target

# EMA helpers
def ema(prev, value, alpha=0.2):
    return prev*(1-alpha) + value*alpha
```

```
# State
hdr_p90_ema = target_hdr_p90
mempool_growth_ema = 0
fork_rate_ema = 0
fanout = base_fanout

#########################
# Fast gossip loop (run every 5-20ms)
def gossip_control_loop():
    global hdr_p90_ema, fanout
    observed_p90 = measure_hdr_p90()  # ms over last tick
    hdr_p90_ema = ema(hdr_p90_ema, observed_p90, alpha=0.3)

    # Compute suggested fanout based on latency
    k = 3.0
    suggested = round(base_fanout + k*(hdr_p90_ema/target_hdr_p90 - 1))
    suggested = max(min_fanout, min(max_fanout, suggested))

    # Hysteresis: only change by +-1 per tick
    if suggested > fanout:
        fanout = min(fanout + 1, suggested)
    elif suggested < fanout:
        fanout = max(fanout - 1, suggested)

    # Ensure high-quality peers are prioritized
    update_peer_selection_based_on_quality(fanout)

#########################
# Slow blockrate loop (run every 15s)
def blockrate_control_loop():
    global current_interval, hdr_p90_ema, mempool_growth_ema, fork_rate_ema

    # Observe network metrics
    observed_hdr_p90 = measure_hdr_p90()
    observed_mempool_growth = measure_mempool_delta()
    observed_fork_rate = measure_fork_rate()

    # Update EMA
    hdr_p90_ema = ema(hdr_p90_ema, observed_hdr_p90, alpha=0.15)
    mempool_growth_ema = ema(mempool_growth_ema, observed_mempool_growth,
alpha=0.15)
    fork_rate_ema = ema(fork_rate_ema, observed_fork_rate, alpha=0.15)

    # Compute control score S
    w1, w2, w3 = 0.6, 0.3, 0.1
    S = w1*(hdr_p90_ema/target_hdr_p90) + \
        w2*(fork_rate_ema / fork_target) + \
        w3*(mempool_growth_ema / mempool_target)

    # Thresholds for adjustment
```

```
    eps_up, eps_down = 0.08, 0.08
    alpha = 0.12

    if S > 1 + eps_up:
        new_interval = current_interval * (1 + alpha*(S-1))
    elif S < 1 - eps_down:
        new_interval = current_interval / (1 + alpha*(1-S))
    else:
        new_interval = current_interval

    # Apply only meaningful changes (>2%)
    if abs(new_interval - current_interval) / current_interval > 0.02:
        current_interval = max(min_block_interval, min(max_block_interval,
new_interval))
        apply_new_block_interval_locally(current_interval)
```

***Please note that 50 BPS globally is rather unrealistic with current technologies. This is possible regionally and locally. This would require improved connectivity between countries and better networking. Currently, it could potentially be achieved through the use of higher compression, deterministic predictability, and greater shell removal. However, an adaptive dynamic block velocity would find the limit itself.**