# P2P Networking Interop Breakout Notes

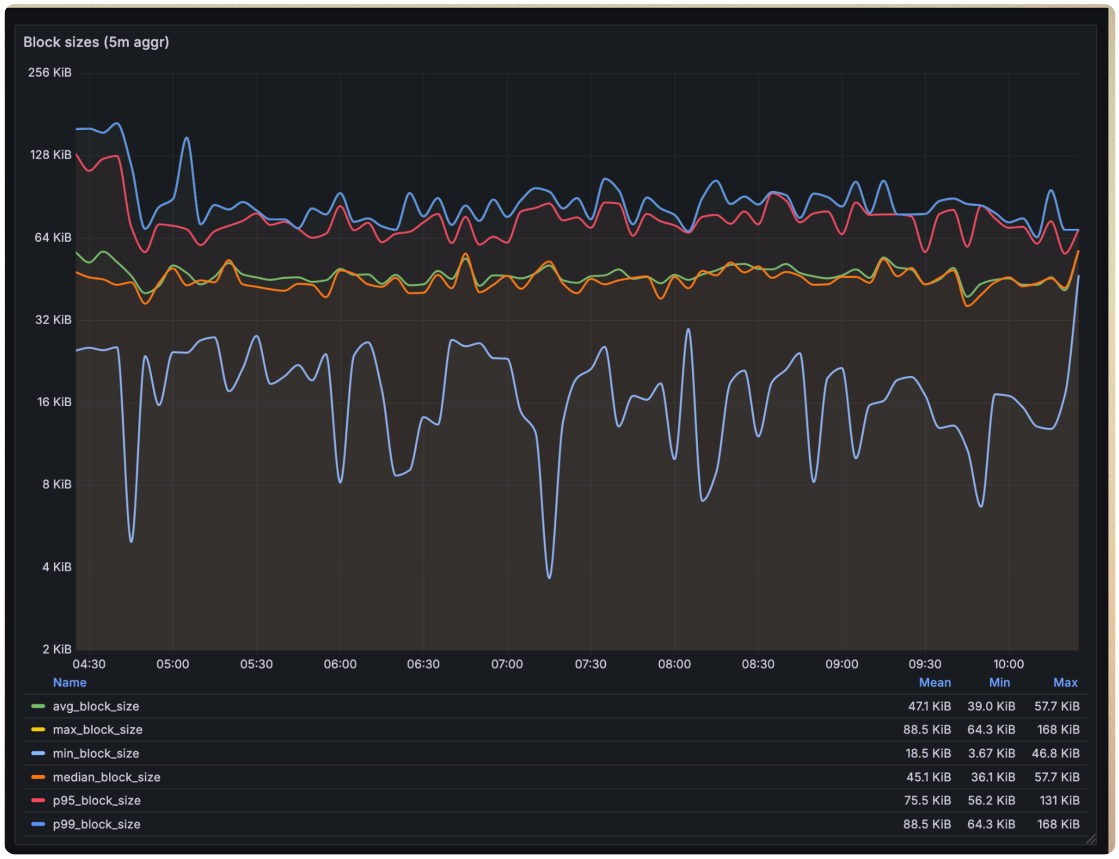
## Overview

Goal of discussion

- libp2p was sufficient five years ago but now there are new requirements for Ethereum like faster block propagation, slot time compression, increased gas
- Goal is to analyze current state of libp2p and the network, discuss propagation, transport, and protocol layers, and discuss about ways to improve libp2p (or even rearchitect from first principles)

What’s going on today

- Focus on getting insight on what is happening at the network layer.
    - Block Propagation CDF visualization: Blocks propagate relatively quickly (within 500ms) likely because MEV publishing
    - Attestation Propagation CDF: Different profile from block. Different payload characteristics from blocks. Many small payloads (200-250 bytes) with high message volume. Different clients attest differently, some attest immediately others wait up to 4 seconds. 4 seconds to get to 80%. Note, different measurements show that attestations arrive in < 1 second.
        - Note: inconsistencies may be due to logging at not at the right time
    - Block sizes: Average block size through beacon block topic 47KB. Maximum current window is 168KB maybe up to 256KB uncompressed.
        - Note: Historical experiments showed max size blocks can reach 2MB, blocks with > 1MB significantly slow down propagation



| Name | Mean | Min | Max |
|---|---|---|---|
| avg_block_size | 47.1 KiB | 39.0 KiB | 57.7 KiB |
| max_block_size | 88.5 KiB | 64.3 KiB | 168 KiB |
| min_block_size | 18.5 KiB | 3.67 KiB | 46.8 KiB |
| median_block_size | 45.1 KiB | 36.1 KiB | 57.7 KiB |
| p95_block_size | 75.5 KiB | 56.2 KiB | 131 KiB |
| p99_block_size | 88.5 KiB | 64.3 KiB | 168 KiB |

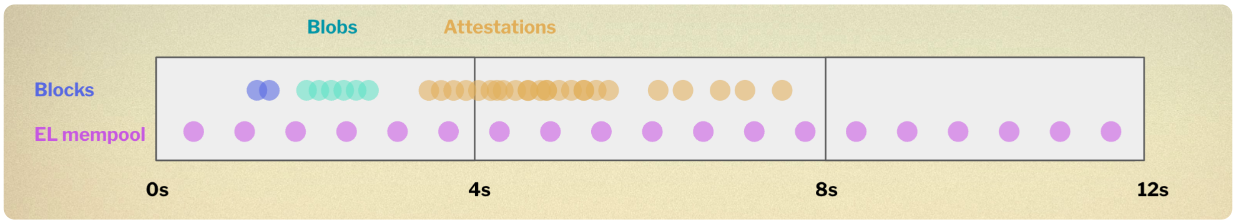
- Blob sidecars: Take around 1s on average to propagate

Note: Attestation propagation curve may not really be attestation arrival, many nodes queue attestations and only verify signatures later which can skew perceived propagation times

Slot activity timeline

- 0 - 2s: Block propagation, 2 - 6s: attestation propagation and aggregation, 6 - 12s: relatively idle or EL mempool traffic

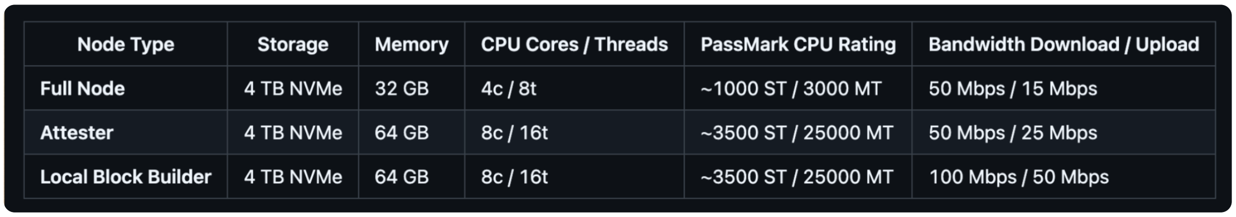
- High burstiness during attestations which means opportunities for optimization by spacing out traffic.



New pressures means an improvement is needed

- Protocol changes such as slot restructuring, faster slot times, increased gas means more load on network → may need to rethink network stack

- Make these improvements while working under constraints of consumer hardware and network infra



| Node Type | Storage | Memory | CPU Cores / Threads | PassMark CPU Rating | Bandwidth Download / Upload |
|---|---|---|---|---|---|
| **Full Node** | 4 TB NVMe | 32 GB | 4c / 8t | ~1000 ST / 3000 MT | 50 Mbps / 15 Mbps |
| **Attester** | 4 TB NVMe | 64 GB | 8c / 16t | ~3500 ST / 25000 MT | 50 Mbps / 25 Mbps |
| **Local Block Builder** | 4 TB NVMe | 64 GB | 8c / 16t | ~3500 ST / 25000 MT | 100 Mbps / 50 Mbps |

Immediate next steps

- Profile clients for underperforming implementations, these can noticeably bottleneck network
- Leverage features like QUIC, parameter tuning with new transports
- Resolve client disparities

## Propagation Layer Planning

Problem

- Gossipsub is designed for small messages but we are sending big messages
    - Duplicate messages become a problem

Sending fewer duplicate messages

- Eager Push: Node sends message diretly to a peer
- Lazy Push: Sends a IHAVE and node can request message
- Overview of solutions
    - IANNOUNCE
    - Message Preamble + IAMRECEIVING
    - Generalized Gossipsub
    - PPPT
    - Choke Extensions

Choke Extensions

- New control messages: Choke and Unchoke
    - Control whether a peer lazy pushes or eager pushes
- Allows protocols for implementation optimizations
    - Enables receiver controlled lazy push
    - Enables sender controlled lazy push
- Requires minimal spec changes
- This is a proposal to add these new control messages, the strategies using these control messages need to be fleshed out and solidified
- Client’s don’t have to implement these optimizations, there will be suggestions and guidelines on how to implement them though

Currently disparity between implementations

- Teams working on impls are isolated from each other
- Before adding more features, reduce disparities to clean things up

Where to go next

- Network coding (RLNC)
    - Chunk large payloads and distribute in parallel over network
    - Dumb gossip layer and spraying chunked packets around to maximize network bandwidth usage
    - Existing exploration for using RLNC. Existing PR in prysm and research post
- Streaming
- Structured topologies
- Transport possibilities

## Transport Layer Planning

Context

- Propagation relies heavily on the transport layer, currently it is limited by legacy TCP stack

Shift to new transport protocols like QUIC

QUIC

- UDP packets but better
- 0-RTT connections. After initial TLS session connection, future connections can be 0-RTT
- Avoid TCP head of line blocking,
- Larger gossip messages
- Persistent sessions lowering overhead
- However, not supported by all clients, need to coordinate across teams to implement and testclie

Basically

- Push for QUIC adoption across all clients
- Maintain support for legacy TCP + TLS stack
- These new protocols will allow things like larger message sizes and reduced handshake latency

## Benchmark and Simulation

Goal

- Test interoperability, performance across clients
- Define certain scenarios and expected behaviors during these scenarios so that implementations can test consistently
- Create these reusable scenarios to test protocol changes and cross client behavior
- Essentially, build a mainnet replica (or as close as possible) in a local machine to test

Shadow for network simulation

- Discrete event simulator
- Allows testing between different client implementations, language agnostic
- Describe situations and test these situations on shadow locally

Example use case

- Nagle's algorithm and delayed ACKs are TCP optimizations that can improve latency when used in isolation. When used together, can have unintended performance degradation
- Use a simulation tool to simulate interactions like these

## Telemetry and Analysis

Currently

- Data collection is fragmented, client-specific. Want to centralize telemetry data and correlate between datasets to understand network bevavior
- Needed for benchmark and simulation and for making data driven decisions

Infrastructure

- Probes, tracers, crawlers, and node level metrics pushed up to EthPandaOps clickhouse

Goal

- Take this data and map network topologies, understand propagation bottlenecks, etc…
- Use this data to feed into simulation, protocol design changes, etc…
- Telemetry → Simulation → Evaluation → Deployment
- Data driven, tested, protocol changes

## Group Discussion

Couldn't really catch everything but what I got was

- Do we have to rethink the entire stack now that workload and technologies have changed
- Research in mix networks over gossipsub. Trade off between latency and privacy
- Unify EL/CL networking layers: Intention is to unify or simplify network responsibilities, particularly as the execution and consensus layers have differing needs.
- Client an devx: Better developer experience with devp2p and rlpx
- Cross client coordination for long term alignment, keep implementations close to spec