Simulation analysis

leios-2025w23
leios-2025w24

Experiments

1. Pseudo-mainnet (`leios-2025w23`)
   a. Transaction lifecycle (Rust only)
2. Mini-mainnet (`leios-2025w24`)
   a. Transaction lifecycle (Rust only)
   b. IB diffusion (comparison of Rust vs Haskell)

Goals

- Compare network topologies
- Re-check Haskell vs Rust
- Realistic scenarios on mini-mainnet

Pseudo-mainnet

This is the first cut at a realistic mainnet-scale topology for Leios, but it likely contain imperfections because several compromises were made during its construction, so as to smooth out inconsistencies in source data.

- Realistic stake distribution
- Realistic number of stake pools
- Two relays for each block producer
- Block producers only connected to their relays
- 10,000 nodes total
- Realistic latencies, generally consistent with the RIPE Atlas ping dataset
- Bandwidth consistent with the low end of what is generally available in cloud data centers
- Node connectivity generally consistent with measurements by the Cardano Foundation
- Geographic distribution (countries and autonomous systems) consistent with measurements by the Cardano Foundation

Finding: *Creating a much more realistic topology would require simulating the p2p algorithm itself at mainnet scale, so that the topology would be an emergent property of the simulation.*

Metric	Value
Total nodes	10000
Block producers	2657
Relay nodes	7343
Total connections	298756
Network diameter	6 hops
Average connections per node	29.88
Clustering coefficient	0.122
Average latency	77.0 ms
Maximum latency	636.8 ms
Stake-weighted latency	0.0 ms
Bidirectional connections	10800
Asymmetry ratio	92.77%

Mini-mainnet

- The “mini-mainnet” was created using the same procedure and data sources as for pseudo-mainnet.
- However, . . .
  - the smaller stakepools were eliminated, and
  - the total number of nodes was reduced to 750.
- The network has a slightly smaller diameter, but its other characteristics are similar to pseudo-mainnet.
- Whereas running simulations on pseudo-mainnet was nearly impractical, running them on mini-mainnet is eminently feasible.

Metric	Value
Total nodes	750
Block producers	216
Relay nodes	534
Total connections	19314
Network diameter	5 hops
Average connections per node	25.75
Clustering coefficient	0.332
Average latency	64.8ms ms
Maximum latency	578.3ms ms
Stake-weighted latency	0.0ms ms
Bidirectional connections	1463
Asymmetry ratio	84.85%

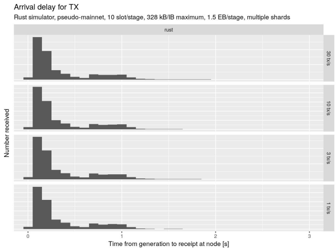
Transaction lifecycle experiments

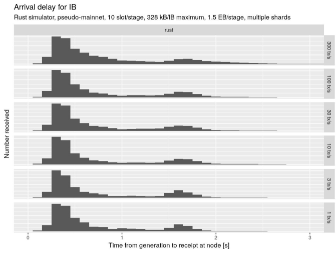
- Rust simulator (since Haskell does not model transactions)
- 100-node topology
- 8 vCPUs / node
- 10 slot / stage
- 3 shard / group
- 10 groups
- 1.5 EB/stage
- 1, 3, 10, 30, 100, 300 tx/s*
- IB generation probability varies with TPS
- Number of shards ⋚ (IB rate) * (shards per group) * (groups)
- 327,680 B/IB maximum
- **No unsharded transactions**

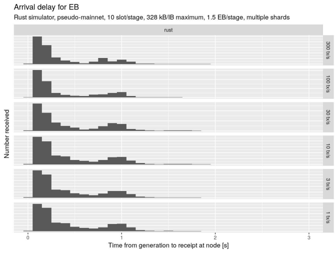
Comparing pseudo-mainnet to mini-mainnet

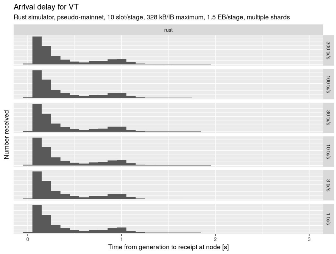
Comparison of message diffusion

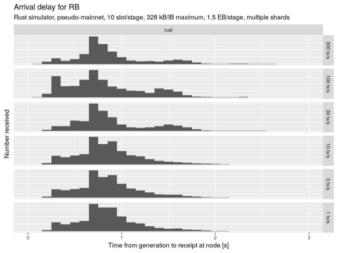
Pseudo-mainnet



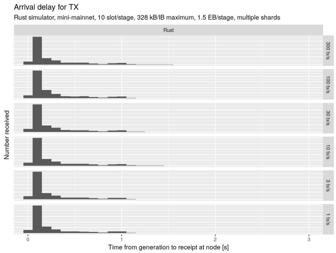


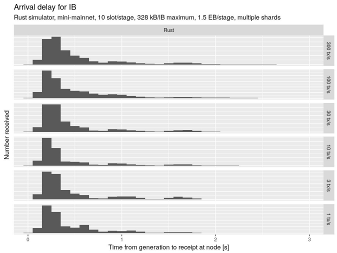


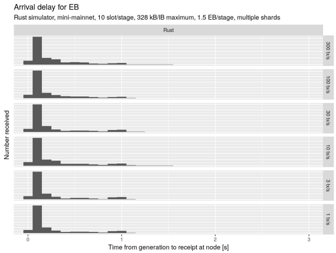


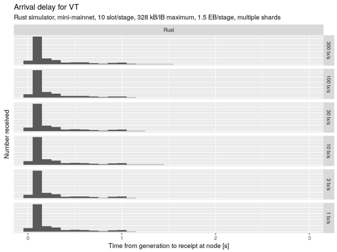


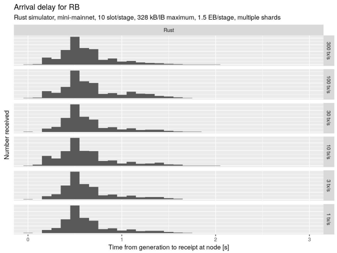
Mini-mainnet







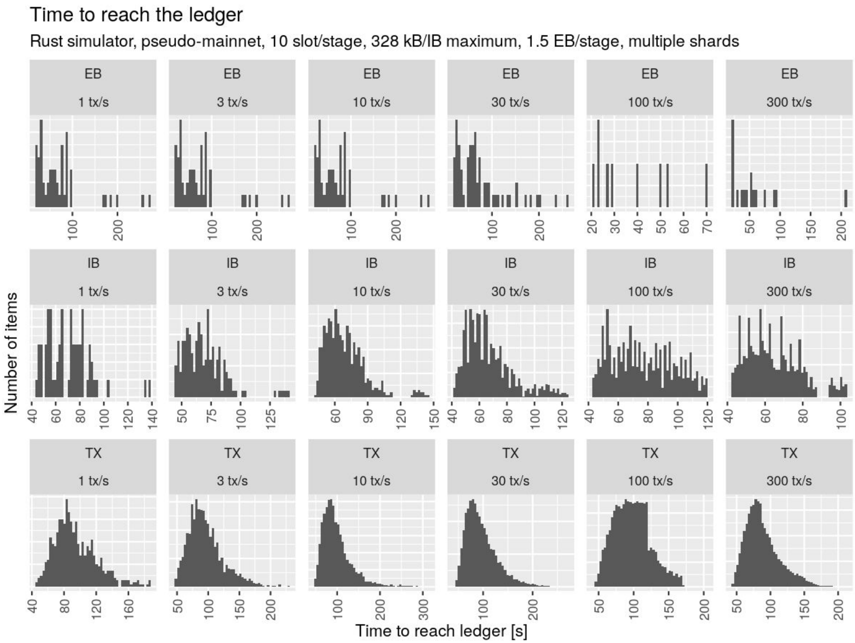




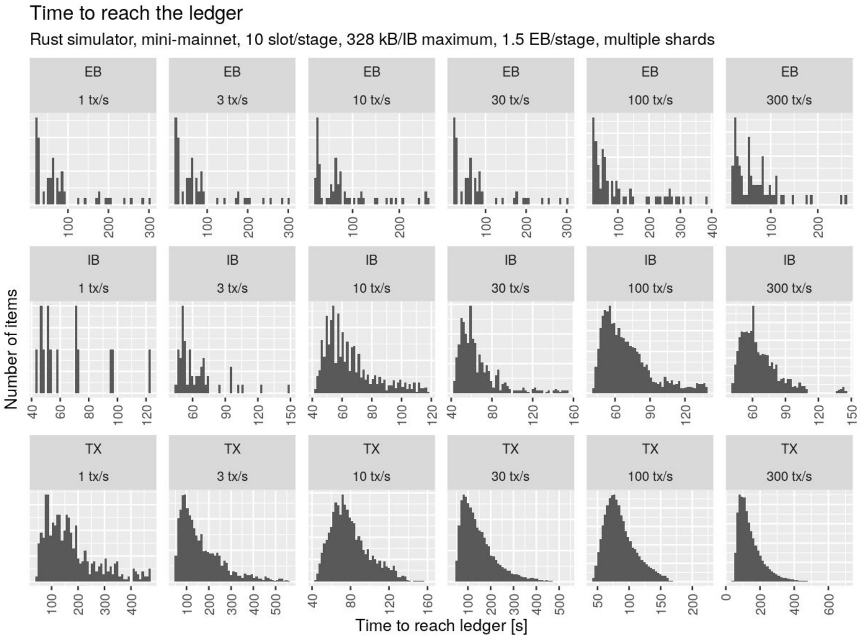
Arrival-time distributions are similar, but pseudo-mainnet has more dispersion.

Comparison of time to reach the ledger

Mini-mainnet

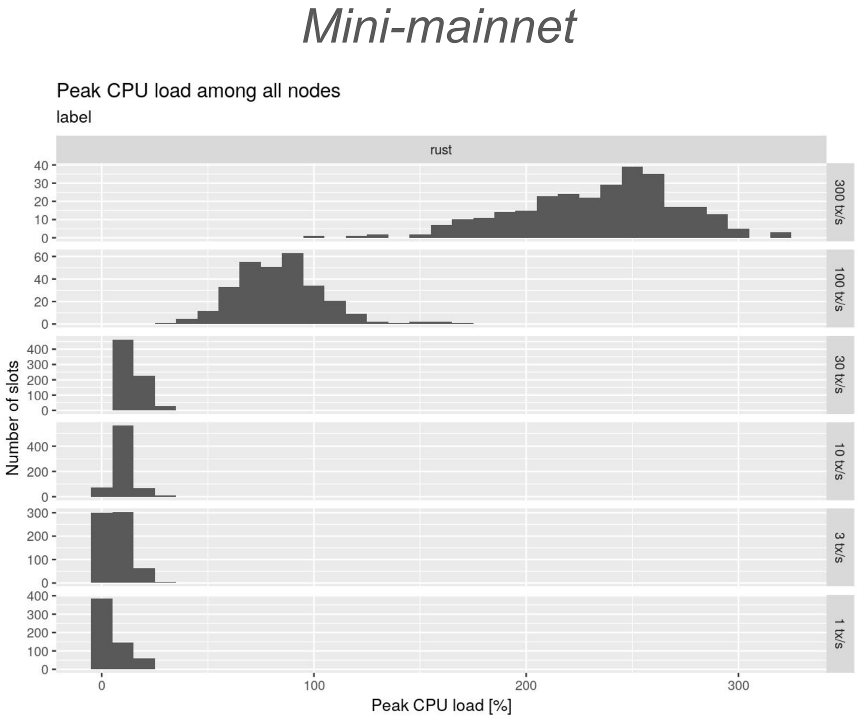


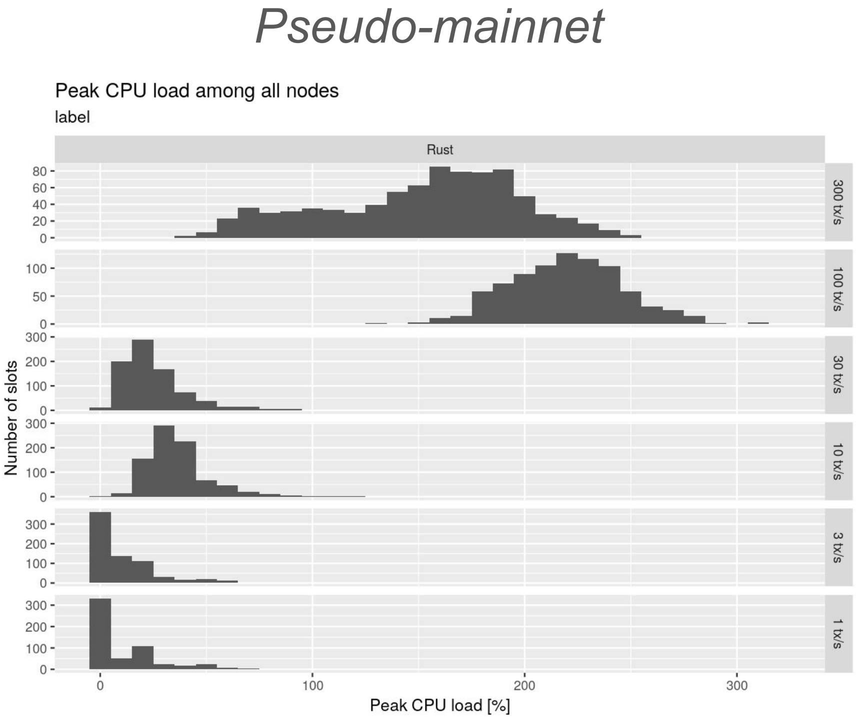
Pseudo-mainnet



Ledger-time distributions are similar, but pseudo-mainnet is slightly faster.

Comparison of peak CPU usage





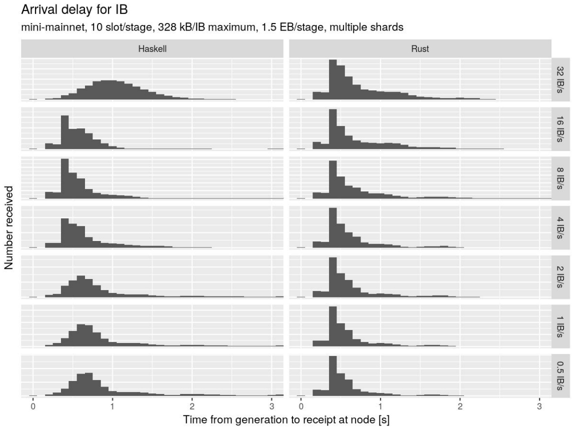
Peak CPU usage is quite similar, except in the 100 TPS case.

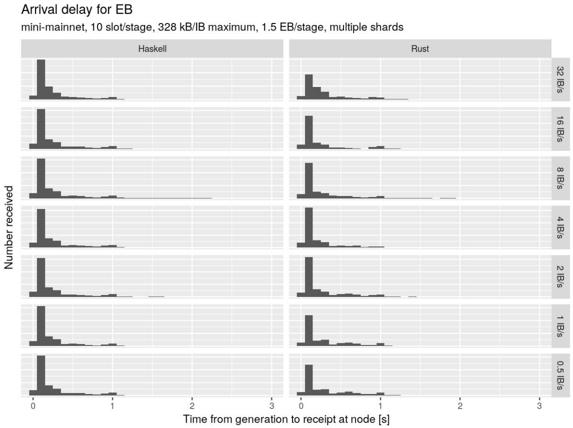
Findings about mainnet variants

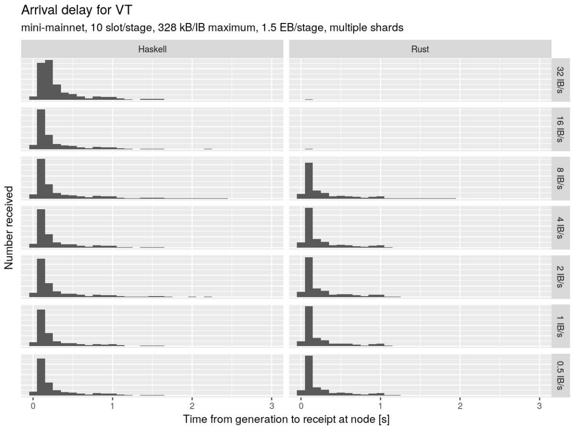
- Qualitative comparison of key metrics indicates little substantial difference between simulation results using pseudo-mainnet vs mini-mainnet.
    - Mini-mainnet might actually be more stressful to Leios than pseudo-mainnet.
- Differences would not affect recommendations about Leios.
    - Similar resource requirements
    - Similar performance metrics
- Future Leios simulations will use mini-mainnet because it is far more computationally tractable.
    - Pseudo-mainnet simulations might be used again at the close of the innovation stream, in order to validate any conclusions based on mini-mainnet simulations.
- Bandwidth assumptions are more important that topology assumptions.
- Next-generation Leios simulators should implement dynamic p2p faithfully, so as to eliminate the expedient of simulation with a static topology.

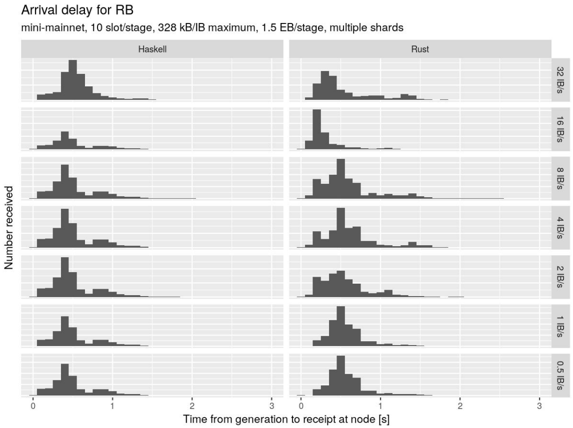
Comparing Rust and Haskell simulators on mini-mainnet

Comparison of message diffusion



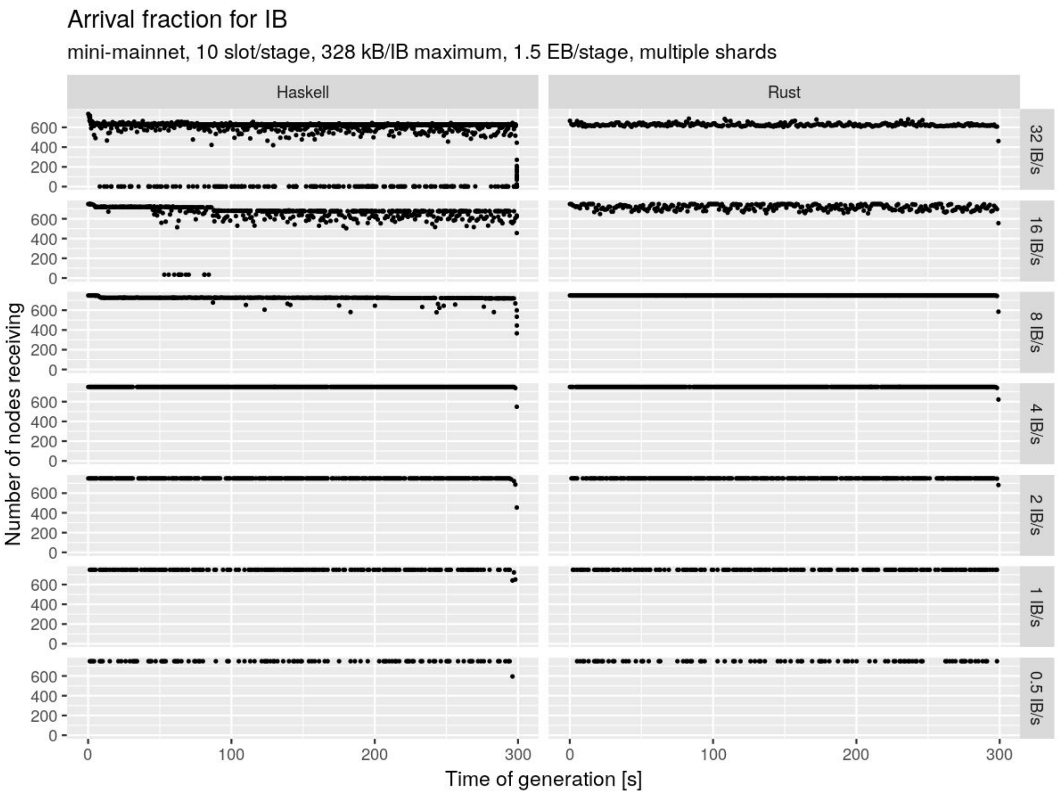






Diffusion of messages is consistent between Rust and Haskell, except for IBs at 32 IB/s, where Haskell is more diffuse.

Comparison of IB delivery



Both Haskell and Rust struggle above 15 IB/s to deliver all IBs, but Haskell struggles more.

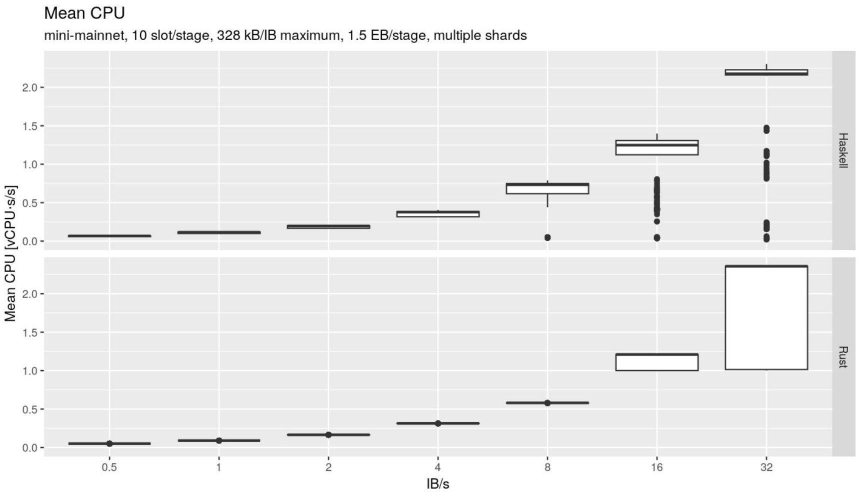
This IB rate corresponds to

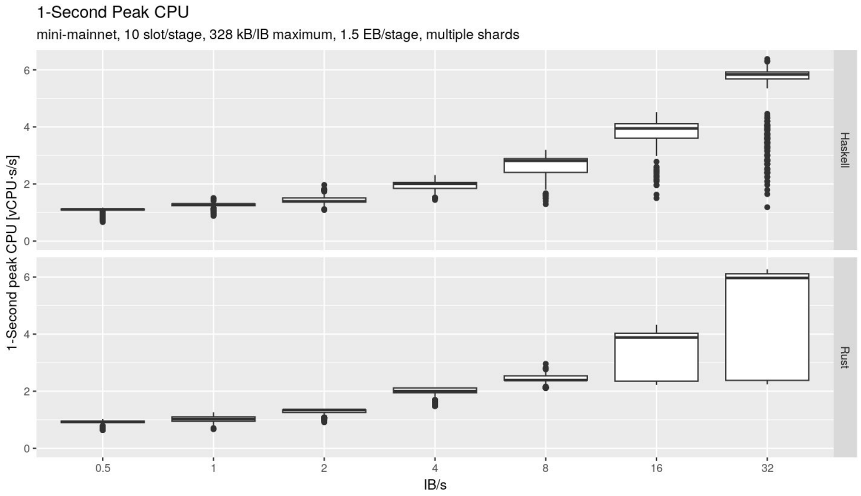
- ~50 MB/s
- ~900✕ Praos
- > 10k TPS

Caveats:

- 1 Gb/s node-to-node links
- IBs are full but TXs are not diffused

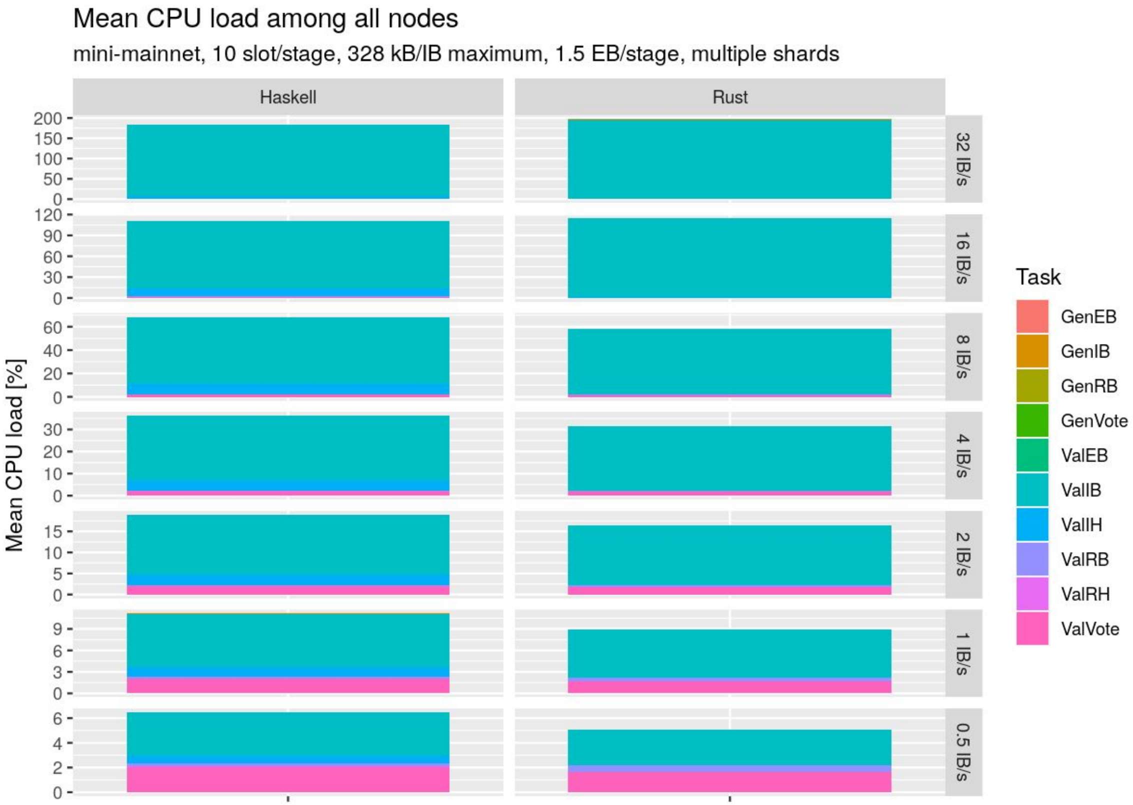
Comparison of CPU usage





Rust shows a broader spread in CPU usage among nodes, but both simulations have similar maxima.

Comparison of CPU tasks



Overall CPU load is comparable between Haskell and Rust, but Haskell appears to spend more time on IB header validation.

Findings about simulator variants

- Haskell and Rust behave similarly on the mini-mainnet topology.
  - Differences are moderately significant only at high rates like 32 IB/s.
- Future simulations studies will rely on the Rust simulator because it is much faster and models transactions.
  - The Haskell simulator might be used again at the close of the innovation stream, in order to re-validate any conclusions based on the Rust simulator.

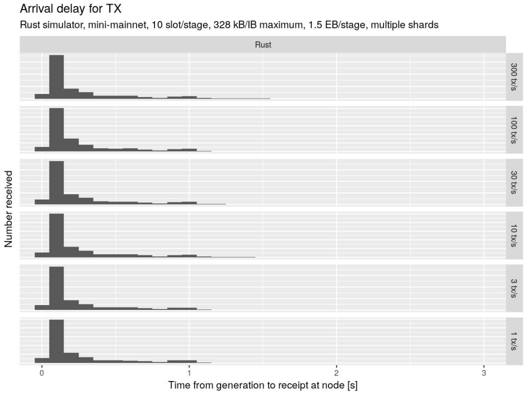
Rust results on mini-mainet at `leios-2025w24`

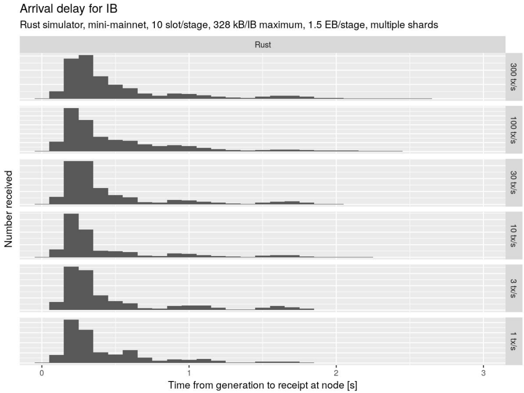
Performance metrics

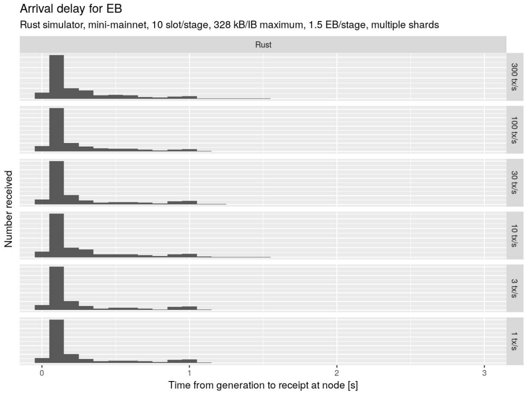
Performance is stable to 300+ TPS.

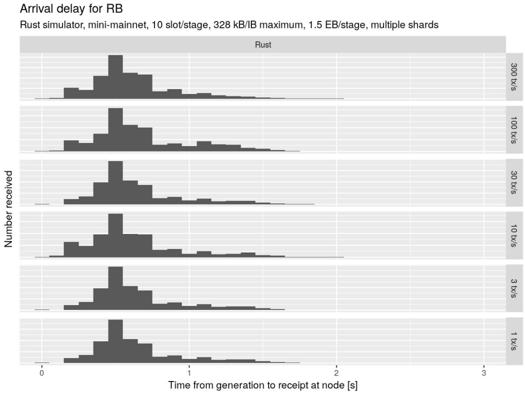
Nominal throughput	TX in IB	TX per IB	IB per EB	IB in EB	Spatial efficiency	TX redundancy	Time to IB	Time to EB	Time to ledger
1 TPS	1.071	31.579	0.851	5.421	67.187%	79.717%	107.129s	131.667s	173.349s
3 TPS	1.034	22.612	3.777	5.506	85.475%	74.333%	94.802s	124.359s	166.205s
10 TPS	1.203	3.002	120.292	5.945	73.887%	81.706%	13.040s	13.040s	79.738s
30 TPS	1.010	22.843	37.397	5.839	94.259%	77.360%	80.816s	109.165s	147.285s
100 TPS	1.262	3.371	1179.405	6.127	72.659%	82.428%	13.336s	41.658s	86.530s
300 TPS	1.027	27.059	386.429	5.692	91.073%	79.845%	86.223s	111.467s	144.033s

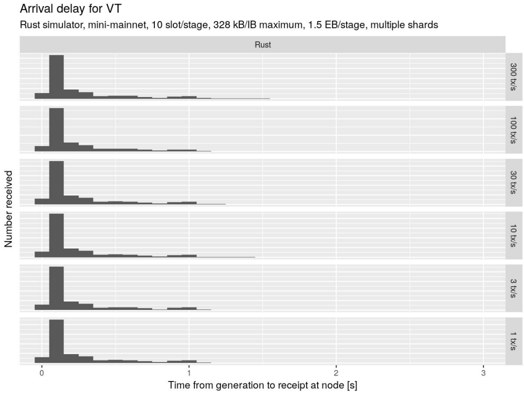
Message arrival (with implications for concurrency)





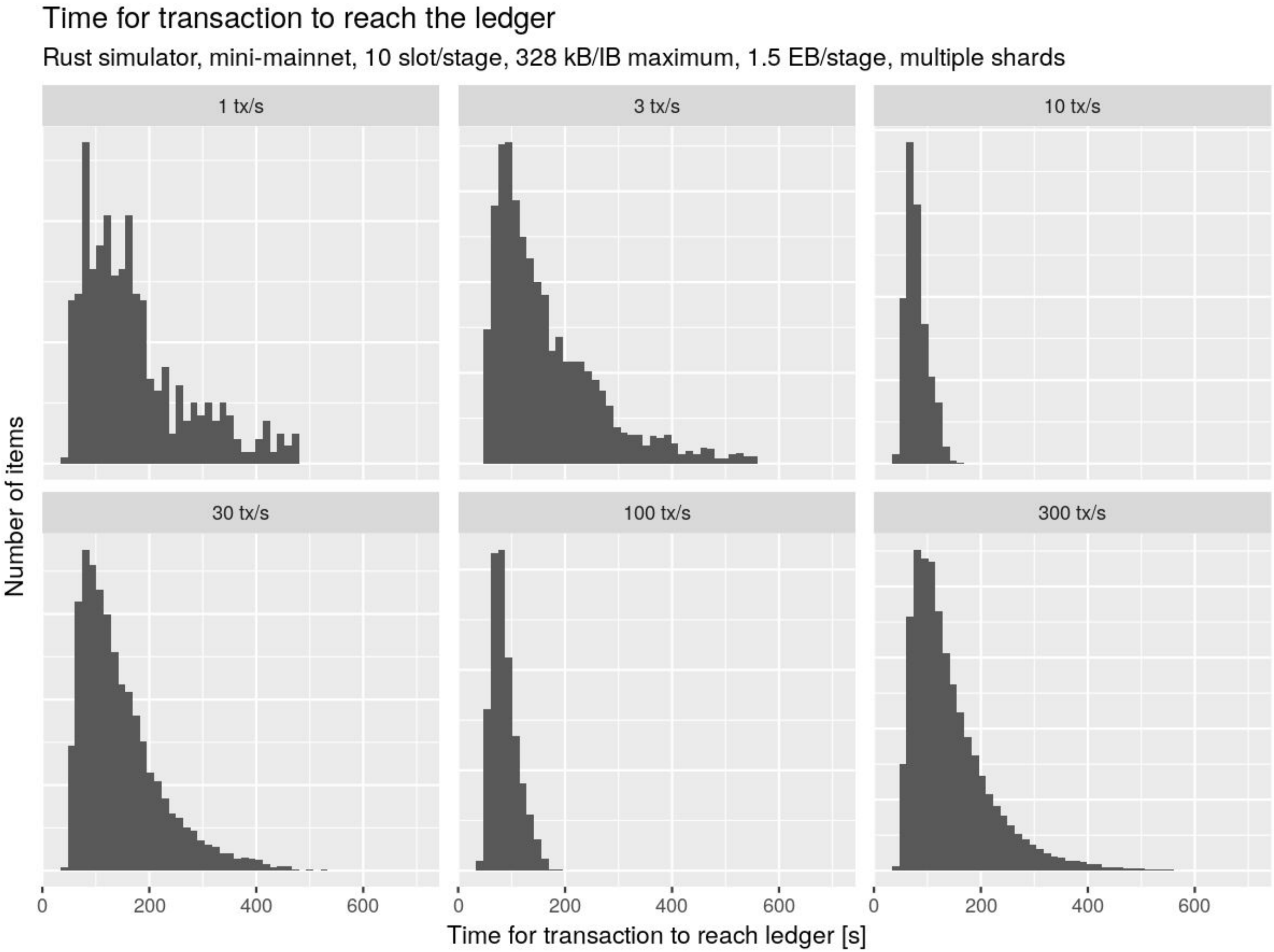




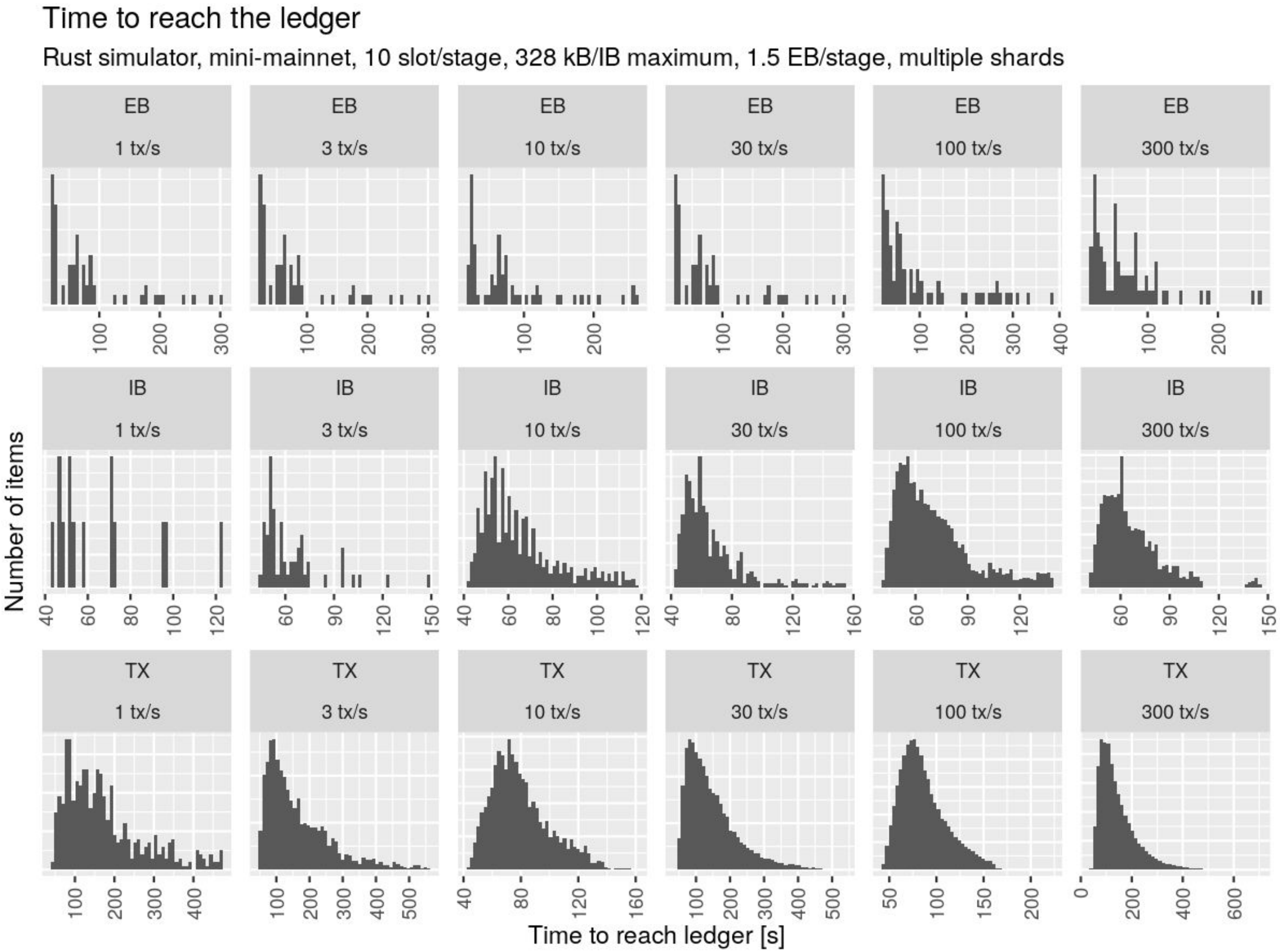


Results are consistent with propagation delays observed in pooltool.io.

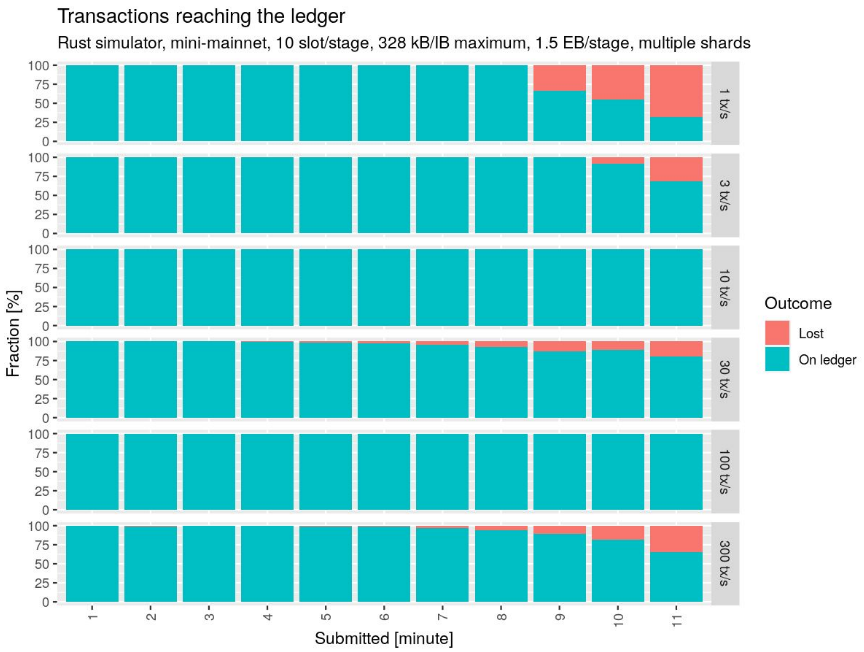
Results are similar to previous simulations, but there is an odd non-monotonicity in the spread of ledger times.

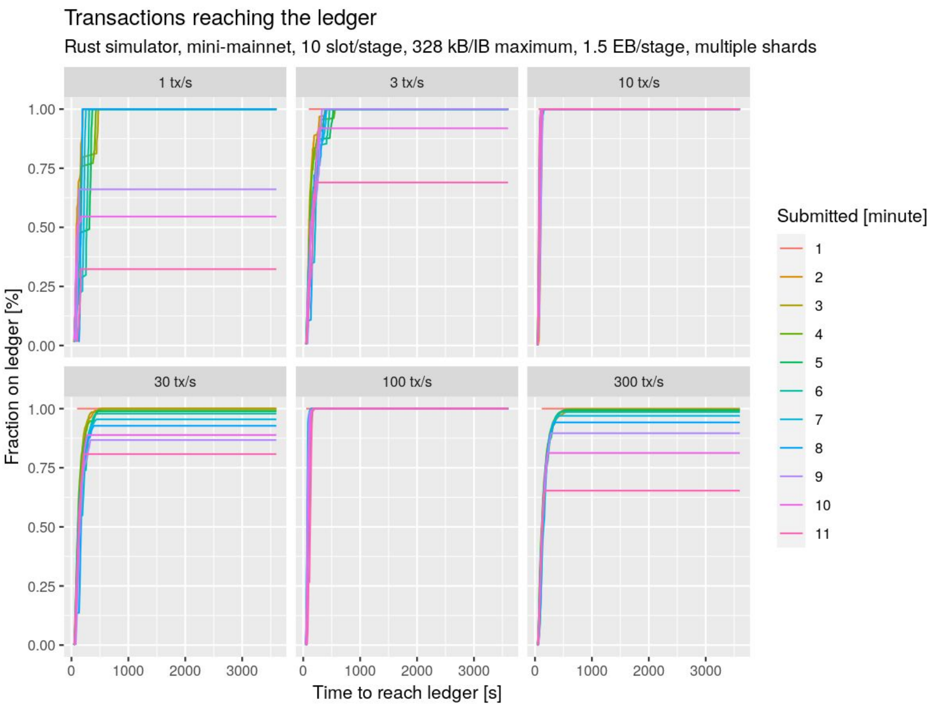


Results are similar to the 100–node topology.



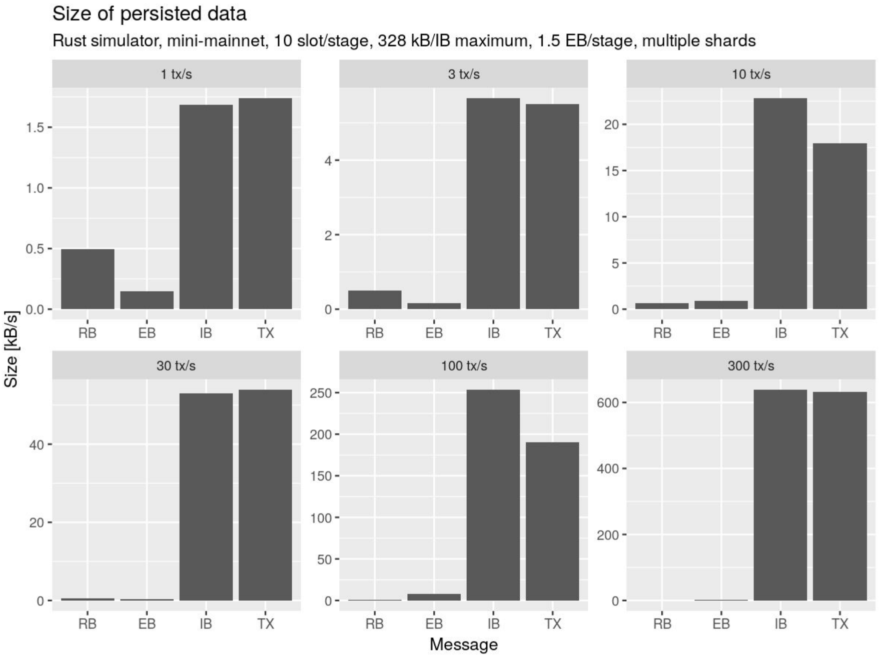
Transactions reach the ledger, except when the simulation was stopped too soon.

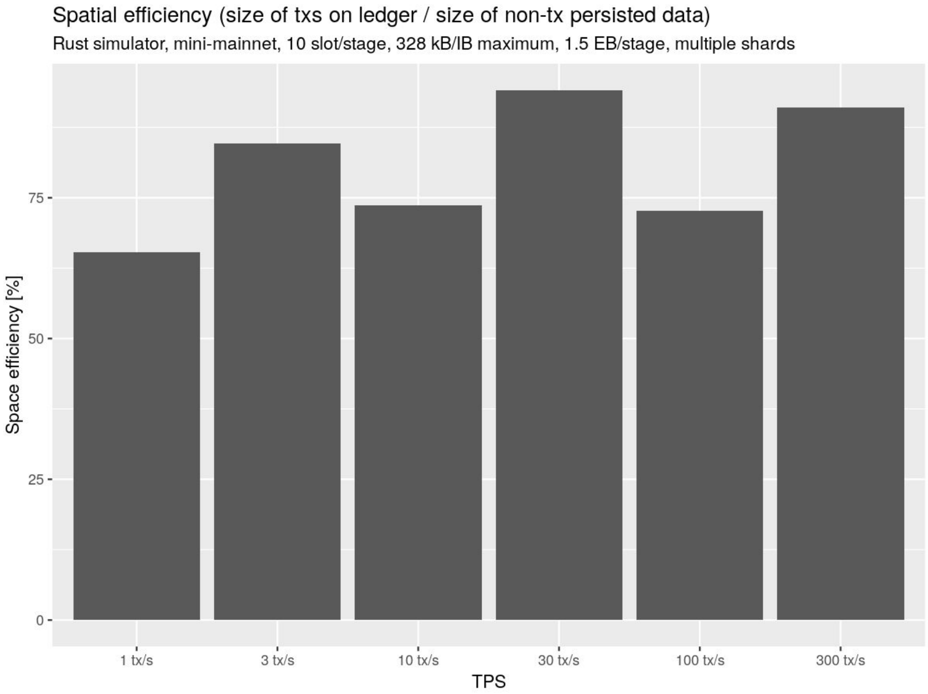




The total size of IBs is on the order of the size of the transactions.

The amount of persistent storage is somewhat larger than the total size of transactions being stored.

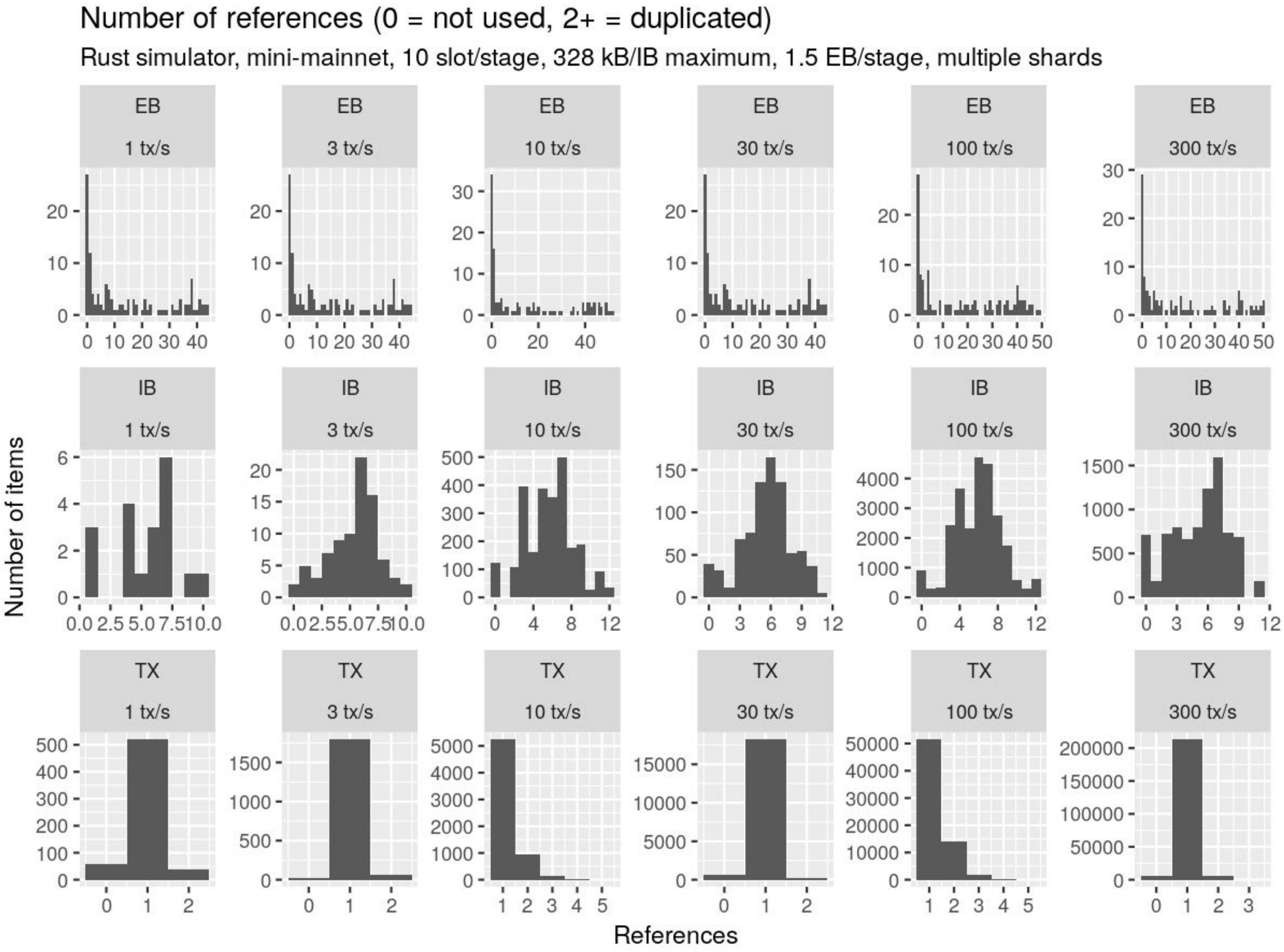




EBs are often referenced by other EBs multiple times.

IBs are often referenced by EBs multiple times.

Transactions are occasionally included in multiple IBs.



Findings from `leios-2025w24` simulations

- The refactored simulation workflow allows rapid processing of large simulation results.
- The mini-mainnet topology balances realism with speed of simulation.
- We need better empirical evidence for the effective bandwidth of node-to-node connections because this sets the fundamental limit on Leios performance.
- Once the candidate Leios variants are finalized, we can run additional experiments:
  - Comparison of variants
  - Optimal protocol parameters
  - Performance measurements for inclusion in the CIP