Time

The PBFT line of work

PBFT (1999)	• Tolerates 1/3 Byz faults, • 3-round good-case latency, • O(n^2) comm in each view, • View change requires sending O(n) certificates
Casper (2017)	• Introduced accountability, chaining in BFT
Tendermint (2018)	• View change requires sending O(1) certificates
HotStuff (2018)	• Linear communication within view and during view change, 7-round good-case latency
HotStuff-2 (2023)	• Improving HotStuff to 5-round good-case latency

fast path

Fast path on PBFT line of work

Fab (2006)	• Optimistic latency of 2-rounds when p <= f Byz faults for n = 3f+2p+1, 3-rounds otherwise
SBFT (2018)	• Linear comm within view • Fab+Zyzzyva fast path
Hydrangea (2025)	• For n = 3f+2c+k+1 - 2-round latency w/ < (c+k)/2 faults - 3-round latency with f Byz and c crash faults
Hydrangea++ (2025)	• Lower inter-proposal latency

Improve throughput

Narwhal (2021)	• Introduced DAG-based mempool
Bullshark (2022), Shoal, Mysticeti (2023), Sailfish, Shoal++ (2024)	• Improve latency through consensus or other optimizations
Sailfish++ (2025)	• Signature-free Sailfish

DAG-based protocols

Autobahn (2024), Raptr (2025)	• High throughput through multiple proposers • Robustness against failures

DAG-inspired protocols

The Simplex line of work

ICC (2021)	• Tolerates 1/3 Byz faults, • 3-round good-case latency, • Every view has a block certificate, uses multiple proposers to ensure this
Simplex (2023)	• 3-round good-case latency, • O(n^2) comm in each view • view-change cert => no decision in view
Dispersed Simplex (2024)	• Simplex + erasure coding

fast path

Fast path on Simplex line of work

Kudzu (2025)	• Similar results as Fab, but on DispersedSimplex
Alpenglow (2025)	• Increase fault tolerance (20+20) by delaying voting and making some synchrony assumption

Extension protocols [GP'16, NRSVX'20]	• Use erasure coding techniques to improve throughput • Can be applied to most protocols described above
DA oracles [EspressoDA, EigenDA]	

Using erasure coding