

Implementation of a robust and scalable consensus protocol for blockchain

Raphaël Dunant

DEDIS lab

Supervisors:

Linus Gasser, Eleftherios Kokoris-Kogias

Responsible: Prof. Bryan Ford

Proposal cosigning

- Time or timestamp services
- Certificate Authorities (CAs)
- Directory authorities
- Software update services
- Digital notaries
- Randomness services

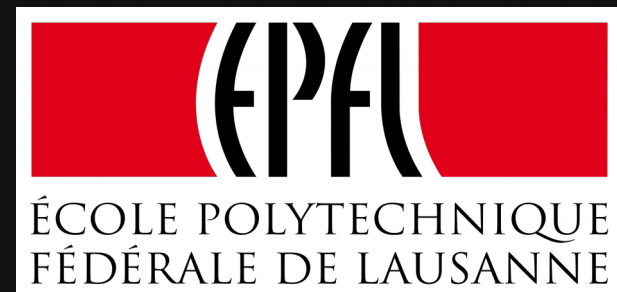


Summary

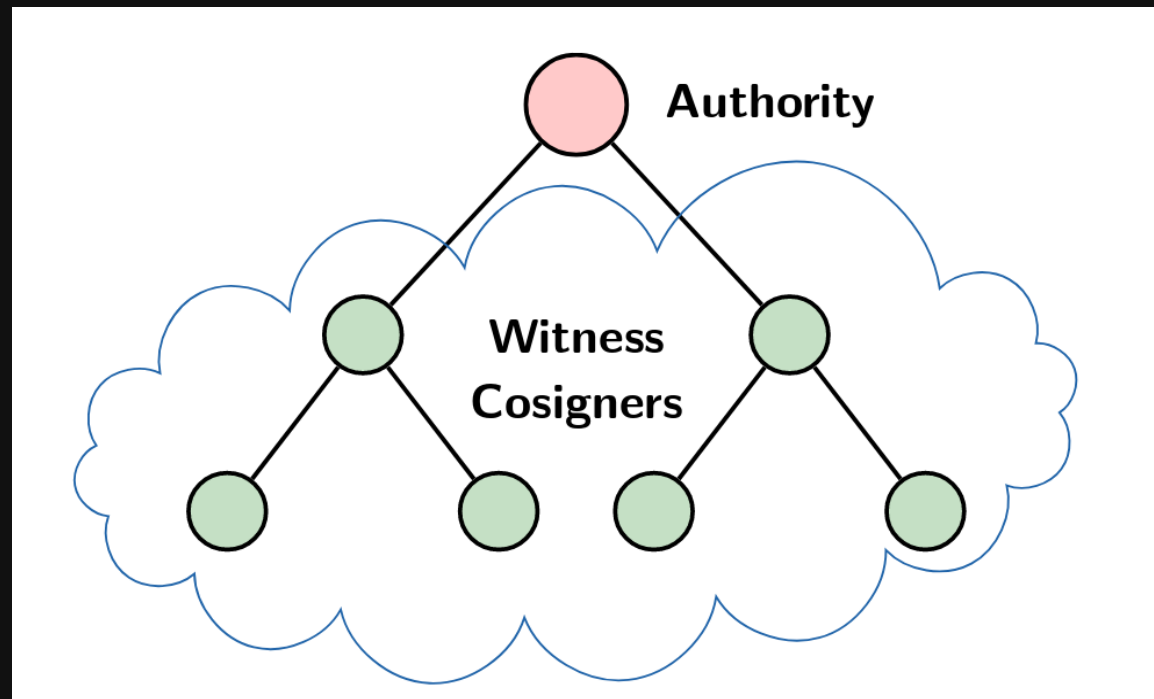
- Introduction (done)
- CoSi protocol
- Work done (challenges and found solutions)
- Simulation results
- Conclusion (results, lessons learned, etc.)

The logo for DEDIS, consisting of the word "DEDIS" in a bold, black, sans-serif font centered within a white square.

DEDIS

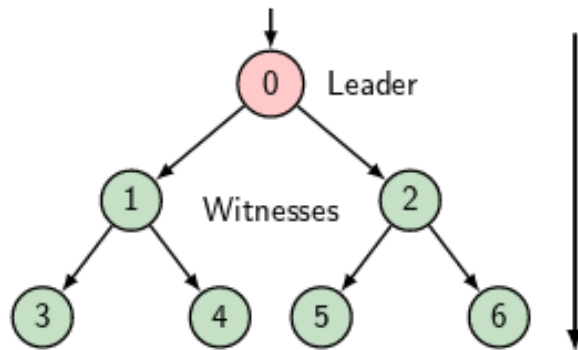


CoSi: Decentralized Witness Cosigning

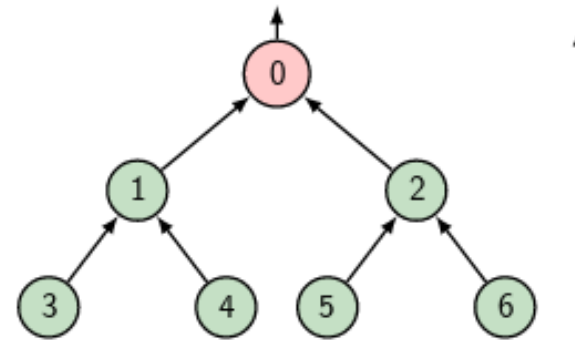


CoSi: Decentralized Witness Cosigning

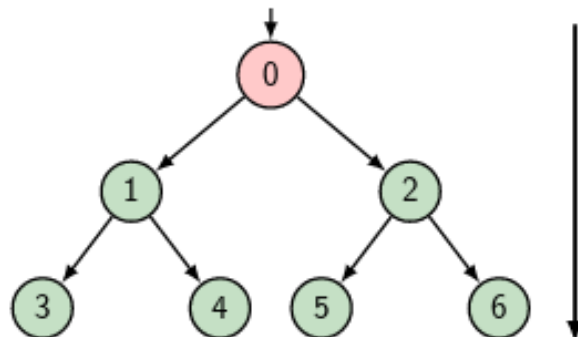
Phase 1: Announcement
(send message-to-witness, optional)



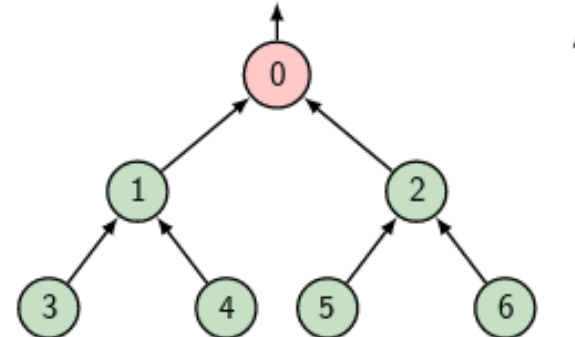
Phase 2: Commitment
(collect aggregate commit)



Phase 3: Challenge
(send collective challenge)



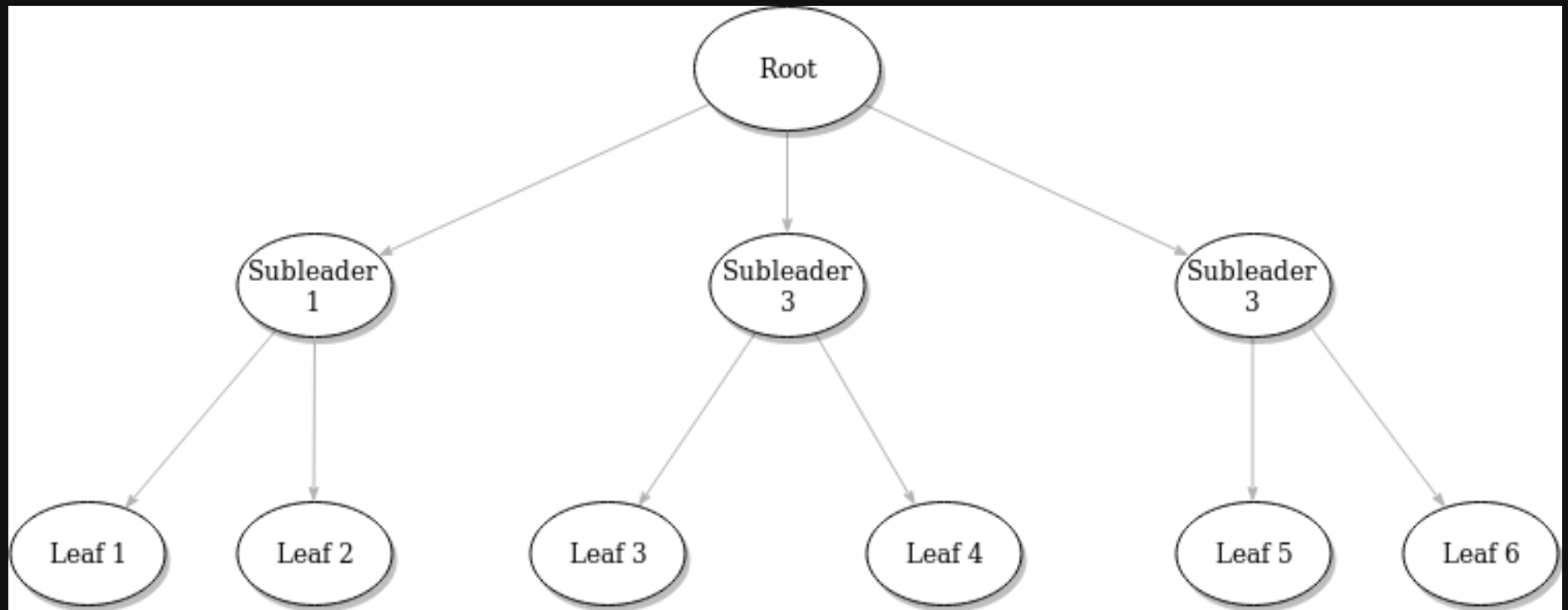
Phase 4: Response
(collect aggregate response)



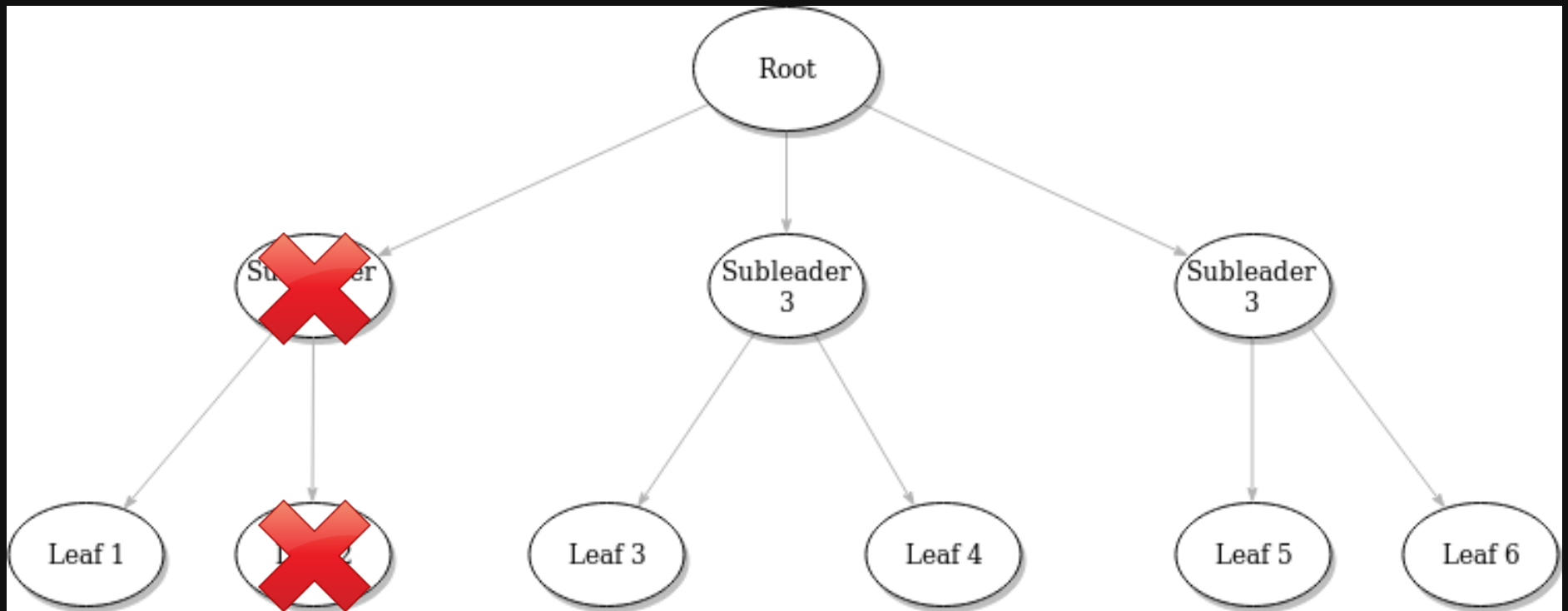
Objectives

- Have a solid implementation of the CoSi protocol
- Compatible with ONet and Kyber libraries
- Handle failing nodes
- Clean, tested and documented code

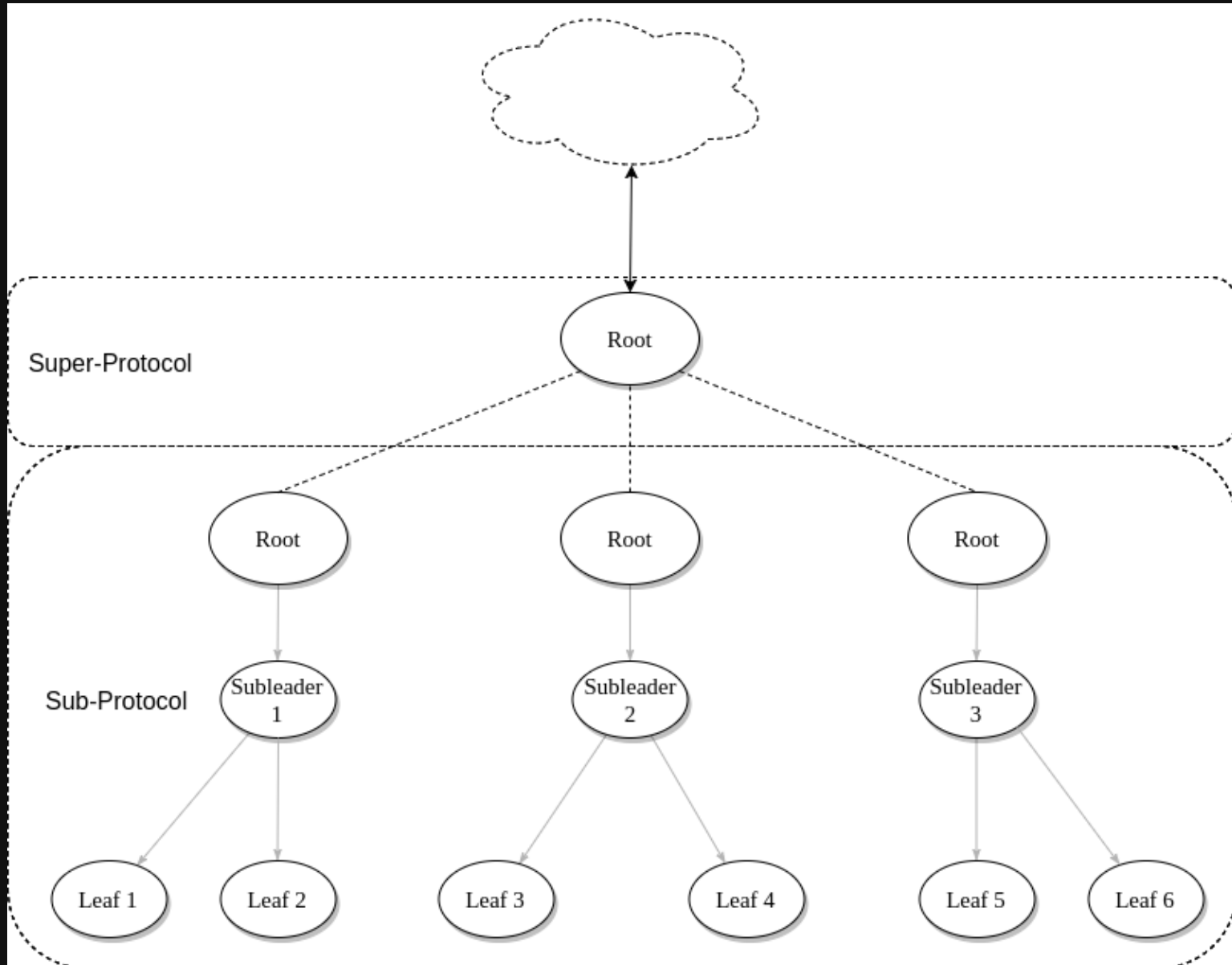
Tree generation



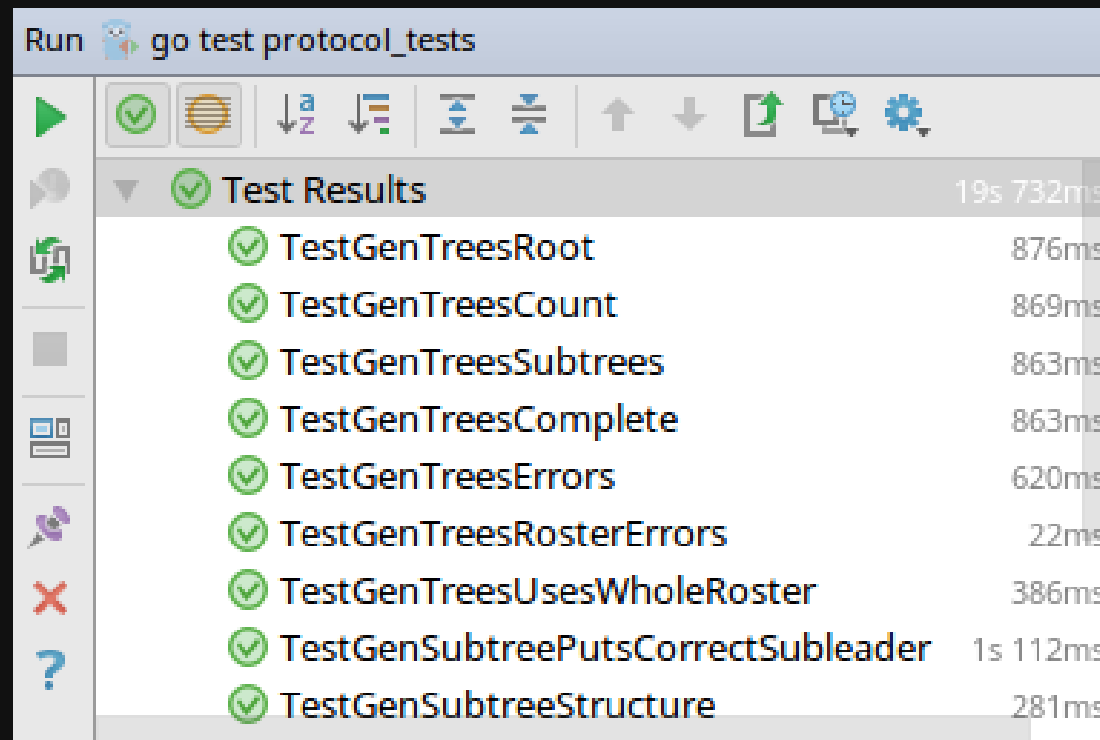
Failing nodes



Multiple sub-protocols



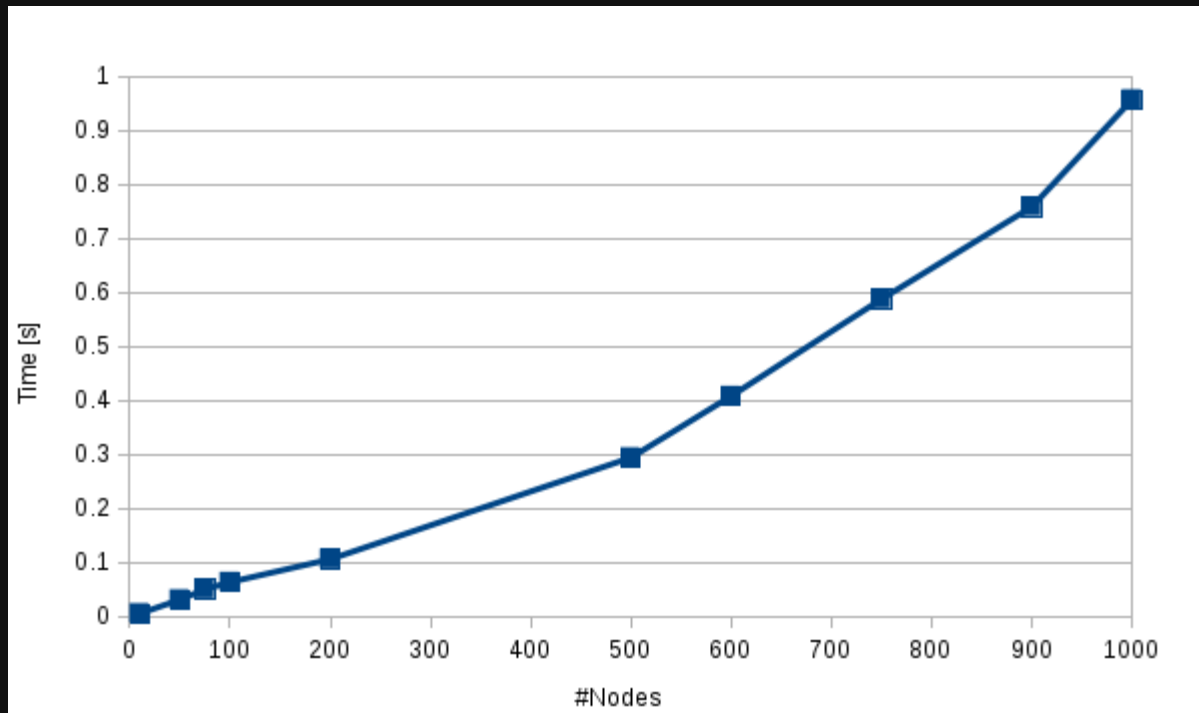
Unit tests and documentation



The screenshot shows the Run window of an IDE with the command `go test protocol_tests` executed. The window displays a list of test results, all of which passed successfully, indicated by green checkmarks. The tests and their durations are as follows:

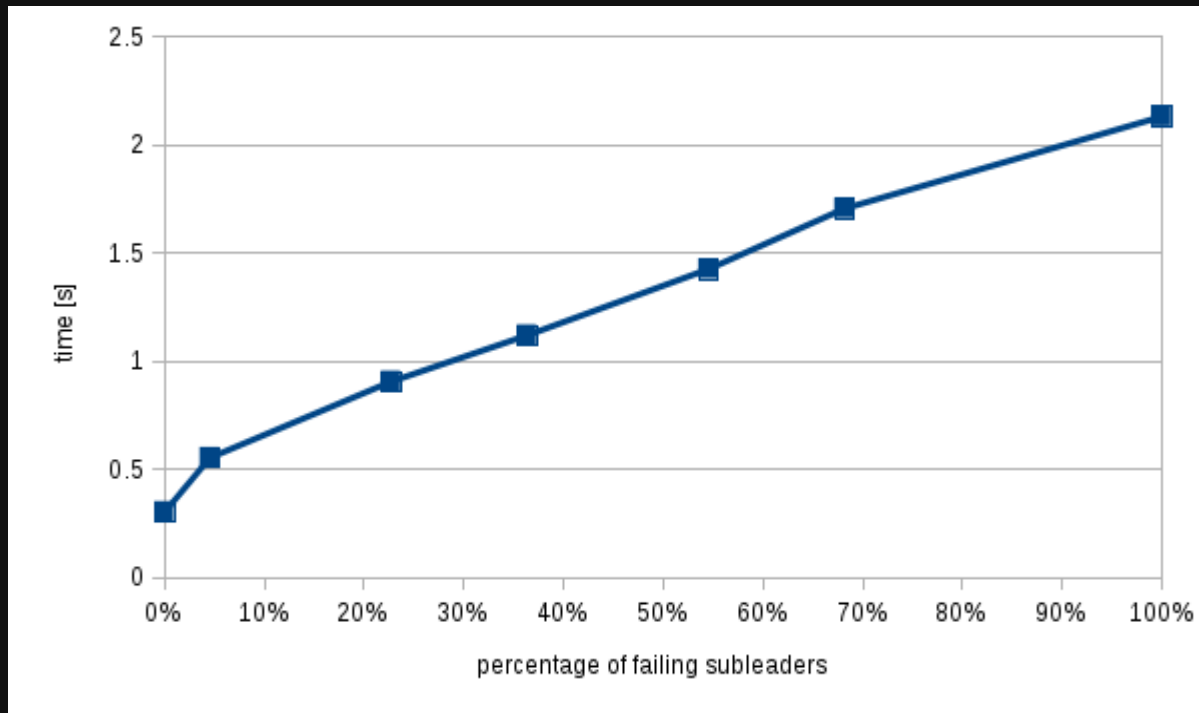
Test Name	Duration
TestGenTreesRoot	876ms
TestGenTreesCount	869ms
TestGenTreesSubtrees	863ms
TestGenTreesComplete	863ms
TestGenTreesErrors	620ms
TestGenTreesRosterErrors	22ms
TestGenTreesUsesWholeRoster	386ms
TestGenSubtreePutsCorrectSubleader	1s 112ms
TestGenSubtreeStructure	281ms

Simulation results : complete working tree



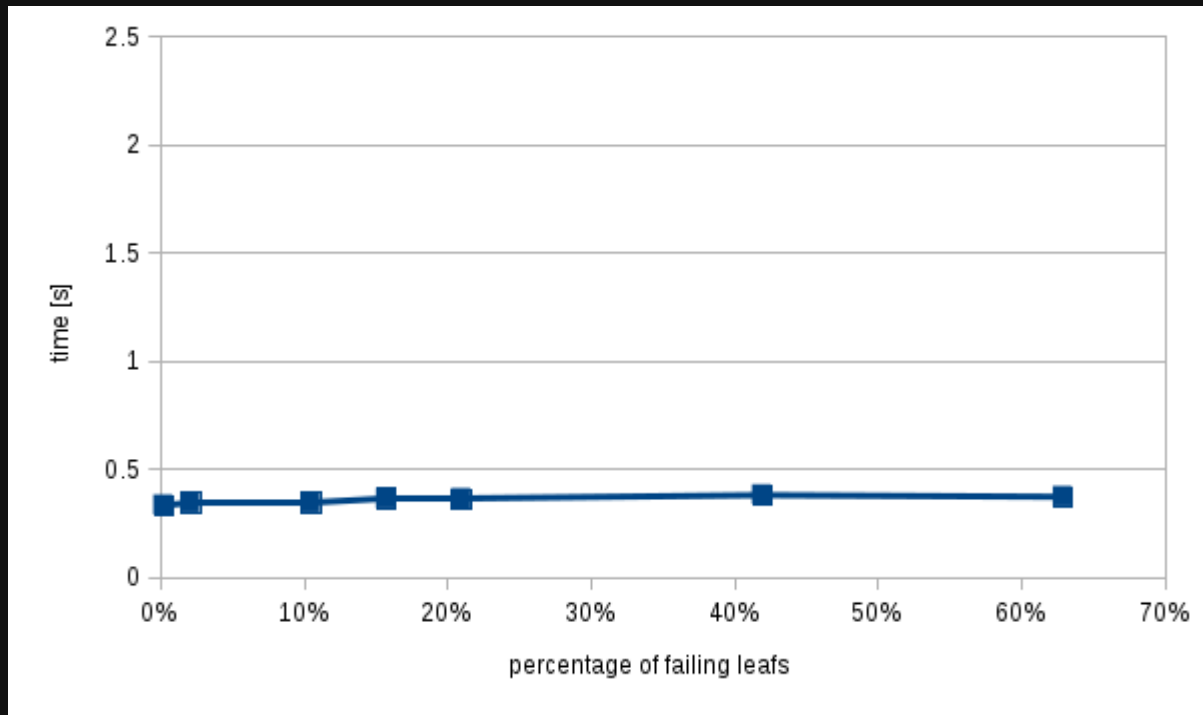
- 50ms delay, 10Mb/s bandwidth
- 4 machines, 4x24 threads, 2.5 GHz, 4x30MB cache, 4x256GB DDR4-2133 RAM

Simulation results : failing subleaders



- 500 nodes, $\lfloor \sqrt{500} \rfloor = 22$ subleaders
- 50ms delay, 10Mb/s bandwidth
- 4 machines, 4x24 threads, 2.5 GHz, 4x30MB cache, 4x256GB DDR4-2133 RAM

Simulation results : failing leafs



- 500 nodes, $\lfloor \sqrt{500} \rfloor = 22$ subleaders
- 50ms delay, 10Mb/s bandwidth
- 4 machines, 4x24 threads, 2.5 GHz, 4x30MB cache, 4x256GB DDR4-2133 RAM

Future work

- BFT-CoSi
- Handle root-node failure
- Handle finely nodes failures during runtime
- Extend unit tests
- Implement on a real blockchain
- Use ONet v2
- Use Omniledger's Sharding Via Bias-Resistant Distributed Randomness



Conclusion

- Complete and working CoSi implementation with node failure
- Easy to use, with documentation and examples
- Lots of interest
- Scalable and tested
- Can still get better
- Personal improvement