# Multiparty Session Types: Paxos Made Easy

Leonid Nosovitskiy

## Abstract

Paxos [1] is a fault-tolerant consensus algorithm that enables distributed systems to agree on a single value despite failure. Paxos and its variants are heavily used in systems like: Google Chubby, Microsoft Azure Cosmos DB, and Apache Cassandra. Paxos presents several challenges, as it is difficult to understand and implement correctly due to its intricate message exchanges and failure-handling mechanisms. Difficult scenarios happen when, for example, an acceptor first agrees to one proposer, then receives an earlier and better proposal request to which other acceptors agreed to. The scenario becomes even more difficult when we need to account for an unreliable network or power cuts.

Multiparty Session Types [2] (MPST) provide a framework for describing and enforcing structured communication among distributed participants, making them well-suited for addressing Paxos challenges. Key properties of the MPST framework, such as static code verification, deadlock freedom, liveness, session fidelity, and correct message ordering, are particularly valuable in tackling these challenges.

To facilitate the representation of Paxos using MPST, we first break the protocol into smaller components, implement them assuming a reliable network, and then refactor the protocol to handle failures. To address this task without introducing new theory, we extend *Scribble* (a DSL used by MPST) with syntactic sugar to make MPST more modular to make global protocols even easier to build and compose. Additionally, we extend *Scribble* and *Effpi* (MPST library for Scala) to support default communications, which present a fine-grained approach to imitate crashed links, originally introduced as optional blocks [3]. Finally, we extend *Teatrino* [4](a toolchain that utilises MPST with crash-stop semantics and generates code) to be able to semi-automatically refactor a reliable global protocol to accommodate for unreliable network, and subsequently adapt associated code to reflect the changes in the global protocol.

With these changes, we aim to produce a toolchain which would make MPST more accessible to creating correct by construction code for industry relevant processes like Paxos.

In this talk, we will present the current state of our project, where we implement our version of Paxos with two proposers and three acceptors. We will discuss the challenges of semi-automatic refactoring, how we addressed them, and how we used our tool to refactor our Paxos implementation. Finally, we will cover the remaining challenges that have yet to be addressed and implemented.

## References

[1] S. Aggarwal, "Raft and paxos : Consensus algorithms for distributed systems," https://medium.com/@mani.saksham12/raft-and-paxos-consensus-algorithms-for-distributed-systems-138cd7c2d35a, 2023, accessed: 4 February 2025.

[2] N. Yoshida and L. Gheri, "A Very Gentle Introduction to Multiparty Session Types," in *16th International Conference on Distributed Computing and Internet Technology*, ser. LNCS, vol. 11969. Springer, 2020, pp. 73–93.

[3] M. Adameit, K. Peters, and U. Nestmann, "Session types for link failures (technical report)," *CoRR*, vol. abs/1607.07286, 2016. [Online]. Available: http://arxiv.org/abs/1607.07286

[4] A. D. Barwell, P. Hou, N. Yoshida, and F. Zhou, "Designing Asynchronous Multiparty Protocols with Crash-Stop Failures," in *37th European Conference on Object-Oriented Programming (ECOOP 2023)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), K. Ali and G. Salvaneschi, Eds., vol. 263. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, pp. 1:1–1:30. [Online]. Available: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ECOOP.2023.1