Talk Proposal: Agreement without Coordination

Revisiting Consensus from a Local-First Perspective

Julian Haas
Technische Universität Darmstadt
Germany

By now, CRDTs [14, 15] have become the de-facto standard for data synchronization in local-first software. Several production-ready libraries of CRDTs exist [2, 10], and various CRDT-based data synchronization platforms are under active development. CRDTs provide *eventual consistency* by restricting their updates to operations that ensure convergence. At the same time, it is widely acknowledged that eventual consistency is not sufficient for certain application semantics. Some application invariants are too "strong" to be maintained under eventual consistency because they require agreement between all participants, which is traditionally acquired via consensus/coordination protocols or by designating a *single source of truth* such as a centralized server. This gave birth to a variety of *mixed-consistency* systems [1, 3, 4, 6, 8, 11–13, 16] which combine eventual-consistency with stronger-consistency guarantees for *some* parts of the application. Unfortunately, very few of those systems are suitable for a local-first setting as they often cannot easily handle devices which become unreachable for extended periods of time.

In this talk, we revisit the consensus problem from a local-first perspective and investigate new ways of expressing consensus protocols that work well in a decentralized setting. Our investigation is motivated by an epistemological view of knowledge in distributed systems as proposed by Halpern and Moses [7]: They distinguish *distributed knowledge* which is "distributed among the members of the group", and *common knowledge* which are facts that are "publicly known". Rephrasing this as a consensus problem, we can say that consensus on a query has been reached once the answer becomes common knowledge in the system. Interestingly, common knowledge can often be established without direct coordination or even direct communication: By simply observing a series of messages on the network, participants can learn facts that are known to the other participants and thus these facts can transition from distributed knowledge to common knowledge.

We argue that CRDTs, a common building block in local-first systems, already present an elegant way for reasoning about accumulating knowledge in a system through their monotonic semi-lattice properties. CRDTs do not rely on explicit coordination between participants; instead, the current state of the data type is determined by observing a series of updates that are propagated through the network. In certain cases, this can be used to make decisions that may seem like they would require explicit coordination. For example, in the classic *two-phase set* CRDT [15], elements can be added and removed, but once they have been removed, they can never be added again. In other words, once a participant observes a removal, it can deduce that this particular element will never be in the set again. When all participants have observed this update, the fact that the element has been removed becomes *common knowledge*, and we have reached agreement without explicit coordination. Similarly, participants of a consensus protocol can deduce that consensus has been reached by only observing certain messages over time. From a data type perspective, this allows us to treat consensus as a write-once register that is either in an undefined initial state or has a single value that can never change again [9].

Throughout the talk, we will demonstrate how to use the monotonic properties of CRDTs in a novel strategy for building consensus protocols which allows expressing existing protocols through the composition of simple replicated data types. We will go through several examples and highlight challenges and opportunities of such a programming model. Additionally, we will examine the applicability of existing protocols to the local-first setting: What are the trade-offs regarding offline periods, network size, and delay-tolerance? Additional details about the proposed programming model can be found in a work-in-progress tech report [5].

References

[1] Peter Alvaro, Neil Conway, Joseph M Hellerstein, and David Maier. 2014. Blazes: Coordination Analysis for Distributed Programs. In *2014 IEEE 30th International Conference on Data Engineering*. IEEE, 52–63. https://doi.org/10.1109/ICDE.2014.6816639

[2] Automerge contributors. 2024. Automerge: Build Local-First Software (Website). https://automerge.org/ (Accessed on 2024-06-24).

[3] Kevin De Porre, Carla Ferreira, Nuno Preguiça, and Elisa Gonzalez Boix. 2021. ECROs: Building Global Scale Systems from Sequential Code. *Proceedings of the ACM on Programming Languages* 5, OOPSLA (Oct. 2021), 107:1–107:30. https://doi.org/10.1145/3485484

[4] Alexey Gotsman, Hongseok Yang, Carla Ferreira, Mahsa Najafzadeh, and Marc Shapiro. 2016. 'Cause I'm Strong Enough: Reasoning About Consistency Choices in Distributed Systems. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. ACM, New York, NY, USA, 371–384. https://doi.org/10.1145/2837614.2837625

[5] Julian Haas, Ragnar Mogk, Annette Bieniusa, and Mira Mezini. 2024. Distributed Locking as a Data Type. https://doi.org/10.48550/arXiv.2405.15578 arXiv:2405.15578 [cs]

[6] Julian Haas, Ragnar Mogk, Elena Yanakieva, Annette Bieniusa, and Mira Mezini. 2024. LoRe: A Programming Model for Verifiably Safe

Local-first Software. *ACM Transactions on Programming Languages and Systems* 46, 1 (Jan. 2024), 2:1–2:26. https://doi.org/10.1145/3633769

[7] Joseph Y. Halpern and Yoram Moses. 1990. Knowledge and Common Knowledge in a Distributed Environment. *J. ACM* 37, 3 (July 1990), 549–587. https://doi.org/10.1145/79147.79161

[8] Farzin Houshmand and Mohsen Lesani. 2019. Hamsaz: Replication Coordination Analysis and Synthesis. *Proceedings of the ACM on Programming Languages* 3, POPL (Jan. 2019), 74:1–74:32. https://doi.org/10.1145/3290387

[9] Heidi Howard and Richard Mortier. 2019. A Generalised Solution to Distributed Consensus. https://doi.org/10.48550/arXiv.1902.06776 arXiv:1902.06776 [cs]

[10] Kevin Jahns. 2024. Yjs: Modular Building Blocks for Building Collaborative Applications like Google Docs and Figma (Website). https://docs.yjs.dev/ (Accessed on 2024-06-24).

[11] Mirko Köhler, Nafise Eskandani, Pascal Weisenburger, Alessandro Margara, and Guido Salvaneschi. 2020. Rethinking Safe Consistency in Distributed Object-Oriented Programming. *Proceedings of the ACM on Programming Languages* 4, OOPSLA (2020), 188:1–188:30. https://doi.org/10.1145/3428256

[12] Nicholas V Lewchenko, Arjun Radhakrishna, Akash Gaonkar, and Pavol Černý. 2019. Sequential Programming for Replicated Data Stores. *Proceedings of the ACM on Programming Languages* 3, ICFP (July 2019), 106:1–106:28. https://doi.org/10.1145/3341710

[13] Mae Milano, Rolph Recto, Tom Magrino, and Andrew C Myers. 2019. A Tour of Gallifrey, a Language for Geodistributed Programming. In *3rd Summit on Advances in Programming Languages (SNAPL 2019)*, Vol. 136. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 11:1–11:19. https://doi.org/10.4230/LIPIcs.SNAPL.2019.11

[14] Nuno Preguiça. 2018. Conflict-Free Replicated Data Types: An Overview. *ArXiv* (June 2018). https://doi.org/10.48550/ARXIV.1806.10254

[15] Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. 2011. *A Comprehensive Study of Convergent and Commutative Replicated Data Types.* Research Report RR-7506. Inria – Centre Paris-Rocquencourt ; INRIA. 50 pages. https://hal.inria.fr/inria-00555588

[16] Peter Zeller, Annette Bieniusa, and Arnd Poetzsch-Heffter. 2020. Combining State- and Event-Based Semantics to Verify Highly Available Programs. In *Formal Aspects of Component Software*. Springer International Publishing, Cham, 213–232. https://doi.org/10.1007/978-3-030-40914-2_11