m-ld: Realtime Information Sharing with RDF

George Svarovsky[0000−0002−7480−2888]⋆

m-ld.io Ltd., Lyndale House, 24 High Street, Addlestone, KT15 1TN, UK
https://m-ld.org/ info@m-ld.io

Abstract. Users of information systems increasingly expect information to be available to edit from multiple devices and by multiple users, online and offline. Strategies exist for shared data types with strong eventual consistency guarantees. These can be complex and fault-prone to implement *de novo* for application data, and library implementations do not present standard APIs. To improve data interoperability, portability and extensibility, these strategies can be applied to a standard and self-describing data format, RDF. We introduce **m-ld**, a component providing eventual consistency for RDF data, showing how it can be used to create a collaborative message board program.

Keywords: Realtime collaborative editing · CRDT · RDF.

1 Introduction

Real-time collaborative editing of documents is now a well-established pattern in groupware programs, popularised by Google Docs. Other notable *ad hoc* implementations include Figma [14] and Wikidocs [7]. Evan Wallace (Figma) justifies their investment with the comment "it just felt wrong not to offer multiplayer as a tool on the web".

However, collaborative editing features are particularly complex to implement from scratch. Reasons for this relate to ensuring *strong eventual consistency* in the face of concurrent edits for a non-trivial data structure [10]; implementing the consistency algorithm in the face of network and compute failures; and testing the implementation. Haymo Meran (Wikidocs) comments, in relation to his subsequent work integrating collaborative editing into Atlassian's tools, "you need a *lot* of endurance" (https://youtu.be/EgCYd6ei7QI?t=21).

To address this, re-usable software components have been developed that provide abstract shared data types. Active open-source projects include Yjs [9] and automerge [5]. These tools alleviate the implementation complexity of real-time collaborative editing for stand-alone programs.

If two *different* programs are to support collaborative editing on the same information, it is necessary for the shared data types' behaviours to be correctly implemented in both. This will generally require changes to source code, whether to re-implement an *ad-hoc* data type, or to include a library.

We can conceive improvements to this integration complexity. First, if each program were to publish the identities of the shared data types in use, other programs would be able to dynamically select and apply a suitable implementation. However, this would still require a specific mapping of the program's presentation layer to each of the supported data types' interfaces. This could be addressed with a common representation such as the Resource Description Framework (RDF).

A key feature of RDF is that it is extensible, including with new data structures, which can be strongly identified in the data so consuming applications can adapt to them. Extensions to RDF use vocabularies of known Internationalised Resource Identifiers (IRIs), given meanings by specifications or conventions. An extension may define an entailment regime, that is, a set of logical consequences of the statements in an RDF graph. It may also impose syntactic conditions or restrictions.

The **m-ld** project is exploring how to use these properties for *shared* data types. The goal is to facilitate the re-use of shared information in different contexts and by different applications.

2 Related Work

m-ld builds directly on prior work on the theoretical basis for real-time collaborative editing of RDF [3,6,15]. Besides implementing strong eventual consistency for RDF graphs, this project also seeks to consider the automatic maintenance of extended semantics in real-time.

Other work has focused on a Distributed Version Control System (DVCS) approach [2,13], such as for improving the quality of Linked Open Data [4]. The DVCS model supports extended semantics by requiring an external intervention if concurrent operations give rise to an inconsistent state. So, this approach is better suited to systems without a real-time constraint, or with a central authority such as a database.

Consistency can also be achieved in a decentralised system using distributed consensus, often used with blockchains. RDF is being considered in this space, for example, for indexing [12]. Consensus does not itself address the merge algorithm for concurrent or offline edits, but is a means for an observer to know that a state has been agreed. This property is likely to be useful in future work on **m-ld**.

3 Approach

The **m-ld** project (`https://m-ld.org/`) approaches RDF shared data types by:

1. Implementing the RDF graph itself as a shared data type.
2. Establishing a pattern for the composition of extended shared data types within the RDF graph, using vocabularies.

3.1 Shared RDF

The realisation of an RDF graph as a shared data type is a direct implementation of SU-Set, a Conflict-free Replicated Data Type (CRDT) [3]. Sharing is effected by making a copy of the graph (a *clone*), with a new process identity. Upon mutation, clones publish *operations* which are delivered to every peer clone and merged into their graphs. The SU-Set permits these mutations to be concurrent, while guaranteeing that all clones will eventually converge on the same graph.

As a CRDT, the SU-Set does not require central coordination for a total ordering of operation messages. Instead, it requires *causal* ordering. This is realised in **m-ld** using a logical clock and re-ordering of incoming messages, as necessary, in each clone. The clock chosen is a simplification of an Interval Tree Clock [1], which is more space-efficient than a vector clock.

The SU-Set also does not require "tombstones" (markers for deleted data). However, this means that it is not possible to arbitrarily merge clone graphs without knowledge of the operations applied to each since they diverged. For this reason, **m-ld** clones maintain a *journal* of operations, to allow clones to *rev-up* from a peer if they have missed operation messages.

3.2 Shared Data Type Vocabularies

The realisation of the RDF graph as a shared data type ensures eventual consistency for all graph mutations. However, it does not guarantee correctness of the graph content according to any applicable extended semantics.

The **m-ld** project has explored the embedding of ordered lists in the shared graph, using a sequence CRDT inspired by LSEQ [8]. The CRDT behaviour is not core to **m-ld** but rather implemented using a *constraint*, which encapsulates the list syntax and semantics. The input into a constraint is the current state and the proposed operation, whether local or received from a remote clone, and it is able to *reject* an invalid local operation, *rewrite* the operation, *entail* consequences of the operation, or even *assert* new data. The constraint is dynamically selected and executed based on the vocabulary used in the data (or also, in this case, as the default list implementation in **m-ld** [11]).

In principle the same process, of defining a vocabulary and providing a constraint implementation, can be used to implement other shared data type extensions.

4 Demonstration

The **m-ld** website provides two web applications that demonstrate the component, using an *engine* running in the browser. In both cases the Javascript of the web application loads and uses the engine as a library. The dynamic RDF graph is stored locally in the browser; services are only used for website content delivery and operation message delivery.

The *demo* web application (`https://m-ld.org/demo/`; Fig. 1, left) presents a "message board" which can be edited concurrently by multiple users worldwide.

The *playground* (`https://m-ld.org/playground/`; Fig. 1, right) is a utility for interrogating a **m-ld** graph using the API syntax. In the figure, the playground has been directed to share a graph with the message board, and the representation of the one of the messages is visible, as JSON-LD.

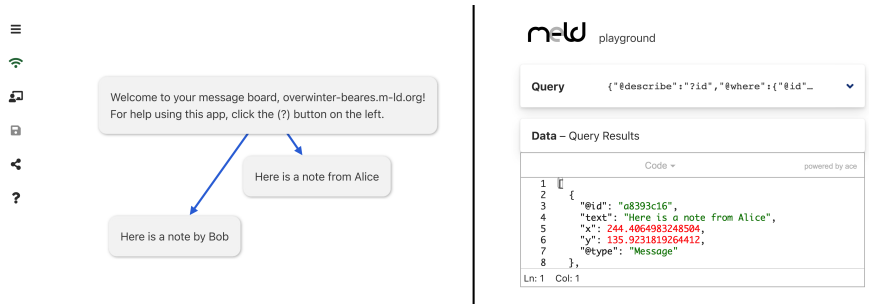

Fig. 1. The **m-ld** demo and playground web applications

5 Evaluation & Ongoing Work

The **m-ld** project has built a shared data types engine using RDF as its foundational data representation. **m-ld** supports real-time collaborative editing of information, including maintenance of extended semantics for ordered lists. Further work is needed to validate the extensibility pattern against other shared data types, and in realistic use-cases.

Other ongoing and future work in the **m-ld** project includes:

- Publication of the **m-ld** protocol specification.
- Research into supporting strong cryptographic assurance of data integrity and traceability, with authority assignable to identified users or groups.
- Research into tunable strategies to truncate or compress the clone journal to balance availability against storage consumption.
- Performance characterisation and tuning.

References

1. Almeida, P.S., Baquero, C., Fonte, V.: Interval Tree Clocks. In: Baker, T.P., Bui, A., Tixeuil, S. (eds.) Principles of Distributed Systems. pp. 259–274. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-92221-6_18
2. Cassidy, S., Ballantine, J.: Version Control for RDF Triple Stores. ICSOFT 2007 - 2nd International Conference on Software and Data Technologies, Proceedings p. 12 (Jan 2007)

3. Ibáñez, L.D., Skaf-Molli, H., Molli, P., Corby, O.: Live linked data: Synchronising semantic stores with commutative replicated data types. International Journal of Metadata, Semantics and Ontologies **8**(2), 119–133 (Jan 2013). https://doi.org/10.1504/IJMSO.2013.056605
4. Ibáñez, L.D., Skaf-Molli, H., Molli, P., Corby, O.: Col-Graph: Towards Writable and Scalable Linked Open Data. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C. (eds.) The Semantic Web – ISWC 2014. pp. 325–340. Lecture Notes in Computer Science, Springer International Publishing, Cham (2014). https://doi.org/10.1007/978-3-319-11964-9_21
5. Kleppmann, M., Beresford, A.R.: Automerge: Realtime data sync between edge devices. In: 1st UK Mobile, Wearable and Ubiquitous Systems Research Symposium (MobiUK 2018) (2018), `https://mobiuk.org/abstract/S4-P5-Kleppmann-Automerge.pdf`
6. Mechaoui, M.D., Guetmi, N., Imine, A.: Towards Real-Time Co-authoring of Linked-Data on the Web. In: Amine, A., Bellatreche, L., Elberrichi, Z., Neuhold, E.J., Wrembel, R. (eds.) Computer Science and Its Applications. pp. 538–548. IFIP Advances in Information and Communication Technology, Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-19578-0_44
7. Meran, H.: Wikidocs - Real time collaborative editing for HTML (Oct 2013), `https://www.slideshare.net/draftkraft/wikidocs-real-time-collaborative`
8. Nédelec, B., Molli, P., Mostefaoui, A., Desmontils, E.: LSEQ: An adaptive structure for sequences in distributed collaborative editing. In: Proceedings of the 2013 ACM Symposium on Document Engineering. pp. 37–46. DocEng '13, Association for Computing Machinery, New York, NY, USA (Sep 2013). https://doi.org/10.1145/2494266.2494278
9. Nicolaescu, P., Jahns, K., Derntl, M., Klamma, R.: Yjs: A Framework for Near Real-Time P2P Shared Editing on Arbitrary Data Types. In: Cimiano, P., Frasincar, F., Houben, G.J., Schwabe, D. (eds.) Engineering the Web in the Big Data Era. pp. 675–678. Lecture Notes in Computer Science, Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-19890-3_55
10. Shapiro, M., Preguiça, N., Baquero, C., Zawirski, M.: Conflict-Free Replicated Data Types. In: Défago, X., Petit, F., Villain, V. (eds.) Stabilization, Safety, and Security of Distributed Systems. pp. 386–400. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24550-3_29
11. Svarovsky, G.: Truth and Just Lists (Feb 2021), `https://codeburst.io/truth-and-just-lists-67c0e0e22a9d`
12. Third, A., Domingue, J.: Linked Data Indexing of Distributed Ledgers. In: Proceedings of the 26th International Conference on World Wide Web Companion. pp. 1431–1436. WWW '17 Companion, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE (Apr 2017). https://doi.org/10.1145/3041021.3053895
13. van Otterdijk, M., Mendel-Gleason, G., Feeney, K.: Succinct Data Structures and Delta Encoding for Modern Databases (Jan 2020), `https://terminusdb.com/t/papers/terminusdb-git.pdf`
14. Wallace, E.: How Figma's multiplayer technology works (Oct 2019), `https://www.figma.com/blog/how-figmas-multiplayer-technology-works/`
15. Zarzour, H., Sellami, M.: srCE: A collaborative editing of scalable semantic stores on P2P networks. International Journal of Computer Applications in Technology **48**(1), 1–13 (Jan 2013). https://doi.org/10.1504/IJCAT.2013.055562