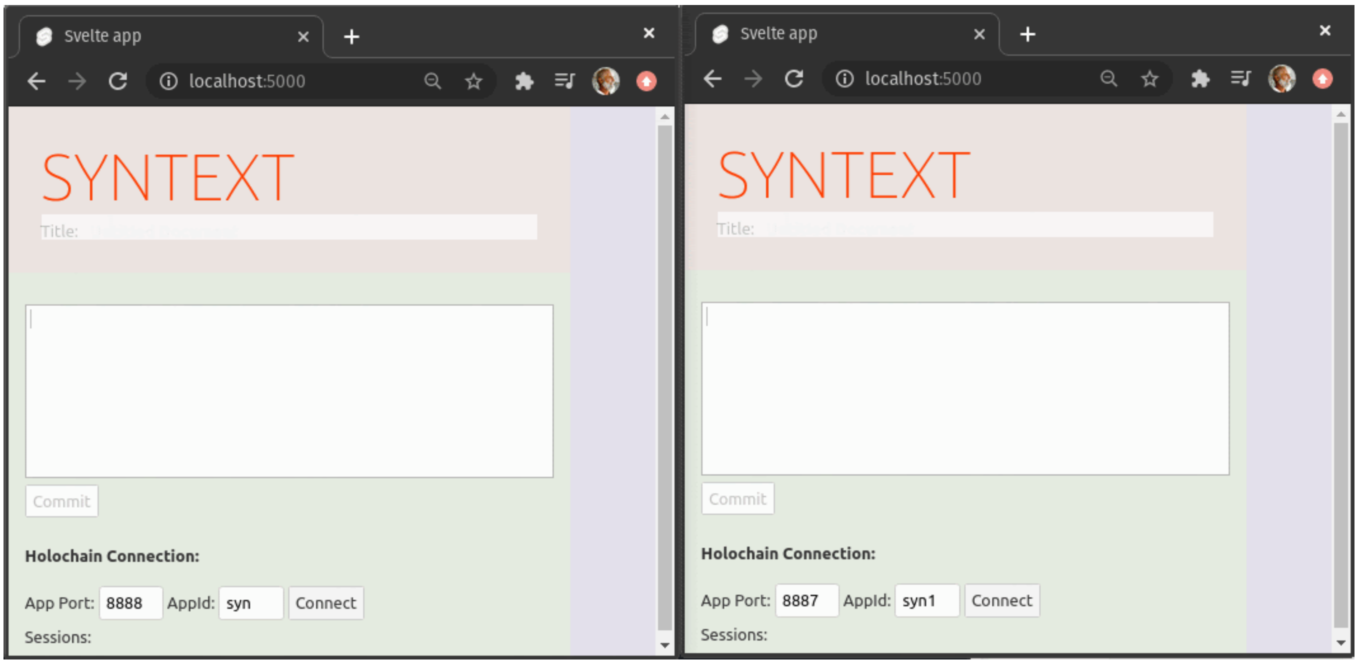
11 March 2021, 15:10 (CET)

Decentralized Next-level Collaboration Apps with Syn

Usually I am more energized by building tech than by talking about it, but I am so excited about what my son and I did over winter break, that I just have to share about it here. In an odd kind of busman's holiday, I spent a good chunk of my time off writing a Holochain application. Coding with my kid is just pure pleasure for me, but I have to describe the additional incredible experience of having spent 4 years building a tool, and now suddenly being able to use that tool to build what it was meant for: creating collaboration applications.

My passion for Holochain has always been sourced in upgrading our collective intelligence, which means making it easier and more joyful to collaborate together. In early December, Art and I sketched out, in an afternoon, a generalized pattern on Holochain for real-time collaborative apps like Google docs where you can type in the same document as others simultaneously, and see their cursor moving around.

Over the holiday break, it took qubist and me just one week to prove out SynText, a peer-to-peer collaborative text editing application.



Not only will it be incredibly easy (see the postscript) to add other such applications on top of the UI pattern and Holochain DNA that we built, but this pattern will allow git-like branching, forking, and merging, in a user-friendly way for any type of collaborative work, not just code.

My deep excitement in this achievement is two-fold: it's both practical and also deeply philosophical:

1. Syn demonstrates the technical threshold that we've crossed in landing Holochain. It was just easy and fast to build it, and things are working like a charm.
2. The patterns in the app bring a level of awareness and embodiment of the deep principles of Holochain's architecture to the forefront. These patterns are all about creating thrivable social-coherence. And by that I mean the kind of social-coherence that arises not from structural coercive centralization, but from peers collaborating by consent. This matters to me.

So, here's the story about Syn, and bear with me through some back-story because it's important.

Back-story: Agent-Centric Data Enables Collaboration

If you've been following the Holochain/Holo story at all, you've probably heard that Holochain is "agent centric". Strictly speaking this is true, but unfortunately it has hidden a deeper truth about Holochain. Let me explain: we used the term **agent-centric** to distinguish Holochain from other software architectures (especially in the blockchain world) that are philosophically **data-centric**. The prime concern of blockchain ledgers is to create a single "consensus" view on a data reality in the distributed peer-to-peer world, where that's a known hard problem. In the case of Bitcoin, the idea is to create a data reality of the location of digital coins: who spent them and with whom. In the case of Ethereum, the idea is to create a data reality of not just coin locations, but also the state of a global computer that you can write and run programs on, i.e. "smart-contracts".

The reason creating these data realities is a hard problem in the decentralized network world is that it's impossible to determine the absolute ordering of events. This is a simple fact of physics. In the client–server paradigm this isn't even a problem, because we start from a centralized server where there is a canonical notion of what the data is and a canonical time. The central server stores the data reality. The brilliance of blockchain is that it provides a protocol for distributed peers to hold data reality together, be it the order of transaction of Bitcoins, or the ordering of computation steps on Ethereum. But it does this by enforcing a single ordering of events. The stupidity of blockchain is that the protocol it uses to enforce this single ordering of events is incredibly expensive, extremely wasteful of computation power, and generates obscene amounts of greenhouse gases. On top of all that, for almost all distributed collaborative applications it's not even necessary to keep one single global ordering of events. It's easy to make this mistake of thinking so if your purpose is a trustless system that's trying to track the location of digital coins, but it turns out that you can still solve that same problem a different way.

	Where is true data?	**Ordering of….**	**Merge of Perspectives?**
***Data-Centric,* Centralized client-server**	Central server, canonical version	...**content** is absolute as performed by the central server.	Forced (There is only one central perspective.)
***Data-Centric,* Bitcoin**	All peers hold copies of data processed via a validation protocol	...**transactions** is based on randomized selection of one miner's sequencing for each block to construct Consensus of token reality.	Forced (One perspective is kept for each block, all others discarded)
***Data-Centric,* Ethereum**	All peers hold copies of data processed via validation protocols	...**content** is based on randomized selection of one miner's or staker's sequencing for each block to construct Consensus of the state of a dApp or smart contract.	Forced (One perspective is kept for each block, all others discarded)
***Agent-centric,* Holochain**	Peers hold data they've authored, as well as data they've validated from other peers. Agents can see or create different realities/times	...**content** is based on its actual **local** order, since data is only truly sequential in the experience of an agent. Relative ordering across many people's local state orderings can be established by explicit protocols and agreements to create shared reality in a case-appropriate way.	Merges only when needed, functions like git. All perspectives are preserved.

Enter Holochain. The ****agent-centric**** view point starts from the understanding that data is derived from agency. It does not have “primary” ontological status.

Data is essentially and fundamentally a record of a particular agent’s experience, be that a human, a sensor, or a device. In reality, agents see/sense/detect different things and receive feedback in different orders representing different realities. If any record’s provenance is separated from it, then the “data” represented by that record is fundamentally broken. We can only make sense of data if we know who sensed the data. This is why absolute ordering of massively simultaneous computation is impossible.

But herein lies the "unfortunate" part that I mentioned before. I have seen people jumping on Holochain's agent-centricity as if it were an ideological stance about the supremacy of individual rights. It is true that the agent-centric approach will help achieve some crucial goals in individual rights, especially around privacy, and preventing inappropriately centralized entities (corporate or governmental) from abusing data privilege. But this is only the first part of the story.

A Holochain application's architecture is better described as ****collaboration-centric****. Start with the reality of agency. Don't dis-empower these agents. Instead fully empower them to enter into non-coercive play together with agreed upon rule-sets. Then what you get is thrivable social-coherence. This is what Holochain is designed to embody. And it's working!

So, enough back-story on the philosophy and infrastructure, now for the “front-story”.

Front-Story: Combining the two mother apps of collaboration

So here’s the big deal for me about Syn: It creates the possibility of merging the two most powerful aspects of collaboration software and doing so in the fully decentralized world. This merge, will, I believe have some really interesting consequences, but first the aspects are:

1. Real-time multi-user “document” editing
2. Alignment of multiple realities: i.e. branching and merging

I’m assuming that point #1 above isn’t controversial, as I will bet that upwards of 75% of everyone reading this has switched from a local word-processor to Google Docs or HackMD for almost all of their text editing.

Point #2 probably ought not be controversial as indicated by the prevalence of git in software development, but it might need a little explanation for those readers that don't do team coding and therefore aren't making code commits and branching, forking, and merging many times a day. Even for those of you who are, if you grew up doing so always with your git repos centralized on GitHub, you may be missing a key part of the decentralization story.

For me, the first time I understood what distributed software and peer-to-peer really meant as an architectural reality was when I switched from Subversion to Git for version control. Subversion was based on the mental model of a centralized repository (a database of changes to the code's text files) from which coders would 1) make local check-outs of that source code 2) edit the files, 3) and then make a commit (which is storing back into the database a set of such changes), but 4) you might have to merge your changes before you could make the commit if someone else had committed first, which entailed resolving any conflicting changes. Coders could also make a branch (which was a copy of the repository) and create a bunch of commits on that branch and merge those in too, but the mental model was still of some central place with a single "canonical" reality that the group of coders was updating.

Git was different. Git assumes no center point. It assumes that each coder has their own copy of the code repository and can pull and push changes to any other distributed copy of the repo they have permission to access. It assumes that all changes are made to local branches and that branches will be created and merged all the time. It embraces locally divergent realities and eventual group consistency.

If you think that your git repository on GitHub is somehow special, well from git's perspective it isn't. It's just a repo that a bunch of agents have agreed to use in a hub-and-spoke synchronization pattern. Your "git push" could be directly to your friends repos if they gave you access and you added them as a remote. In that case you would just be agreeing to a more mesh-like collaboration pattern.

Of course github has loaded lots of collaboration value-add on top of just being a place to embody the hub-and-spoke pattern, with issues, pull requests, kan-ban boards, integrated CI, etc. This is an important part of the story which is part of a pattern that repeats: decentralized public-domain protocols spawn centralized proprietary value-add silos built on top of those protocols. The detailed story of another example of this—about how AOL, GEnie, and Prodigy created chat-rooms, news services and all kinds of proprietary value-add on top of the primary feature of the internet's SMTP protocol that provided the core service, email—is one I will tell some other time. But the key there is that proprietary value-add was erased by the arrival of another set of open protocols: http and HTML. And I think that Holochain/Syn may be yet another example of this story of the orders-of-magnitude value level-up that happens in this evolutionary process.

I'll show how in a bit, but first it's important to understand what it takes to build Syn apps.

To make a Syn app you have to do just a few things:

- Define a **state model** that represents your application.
- In the case of a text document this state model might be just two strings, the title and the document body.
- Create **patch grammar** that defines all the transformations that you can do to the state. In our text doc example that might be just three things: insert a character at a position, remove a character from a position, and set the value of the title.
- Create a **delta function** which, given a state and a transformation, produces a new state. This is pretty easy to do with simple patch grammars.
- Create a **UI** that:

a. renders the state when it gets notified that the state has changed, and
b. creates Syn transformations when the user would like to make a change.

Once you've done that, Syn takes care of the heavy lifting of coordinating between all the agents making and *syn*chronizing changes, all without a central server.

That's aspect 1 from the consequences of collaboration software above:real-time multi-user collaboration. With one additional app-design task, Syn's architecture also handles aspect 2, Alignment of multiple realities, in a general way:

5. Define a **merge strategy** for resolving differences between states and transformation sets.

This last task is where things get tricky to do well, and in general can't always be done in a fully automated way. However, the idea is that when necessary, a merge strategy would include human intervention: manually (i.e. using human judgement) resolving the differences between conflicting change sets. This is nothing new to git users who handle merge conflicts all the time! Note that it's important that merge strategy is not generalized and specified per-app. This is part of the power of the Syn approach.

Once this fifth task is done, Syn becomes a general framework for adding branching and merging to any collaborative app. At any given time, Syn has the notion of a session, which is just like a branch in git. It's a commit to treat as a starting point for a series of changes. Unlike Google Docs which just has a single revision history, Syn provides, at the low level, a space for multiple version histories based simply on which session users join and start making changes. With merging, the realities these branches represent can become aligned. Because Syn supports generalized UI for creating/viewing/switching sessions and can insert the merging strategy defined by the particular app, this is a powerful pattern that used to be available only to coders using git. Now, suddenly, it can be applied to arbitrary collaboration apps. In a way, it's two-level collaboration, synchronously on the "same" state, and asynchronously on divergent/convergent state.

To me this is a big deal. If collaboration is about empowered agents entering into non-coercive playing together by agreed upon rule-sets, then this pattern in Syn is a massive level up meta-rule-set for doing so.

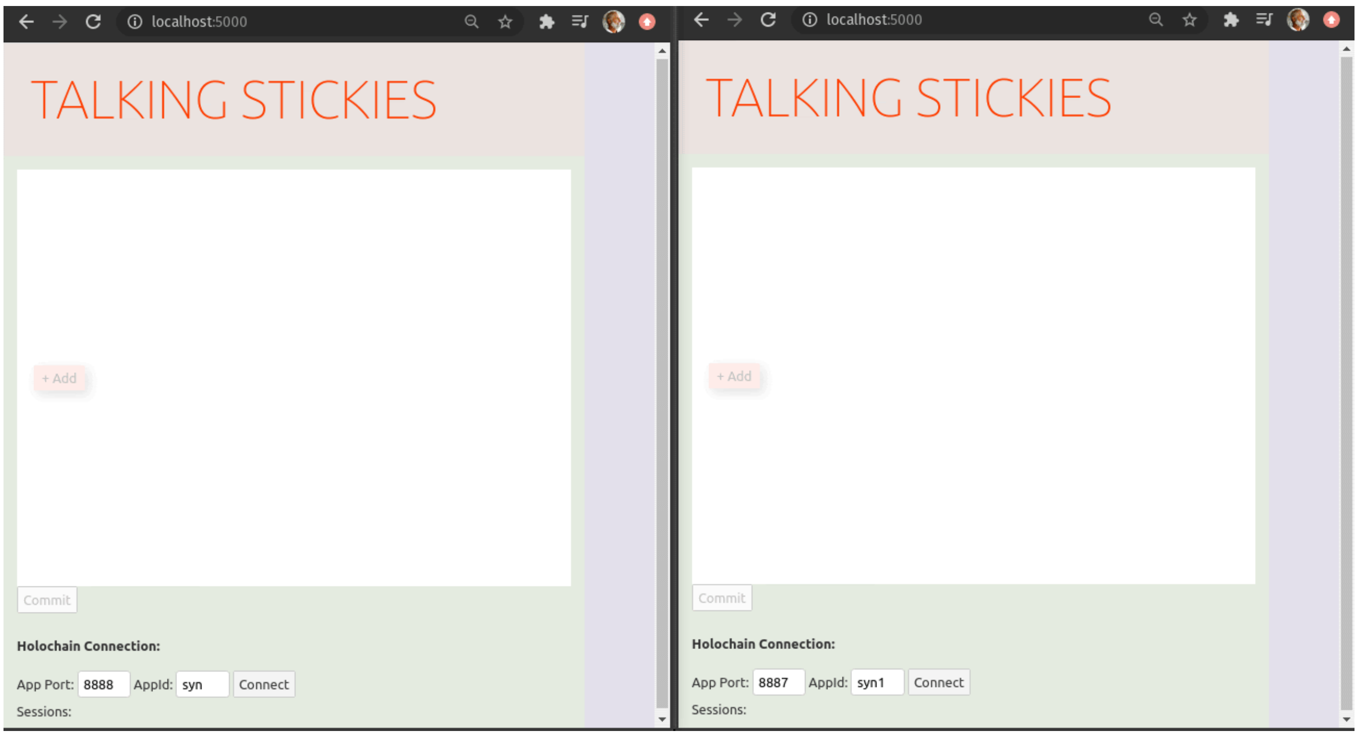
Imagine Wikipedia articles with multiple branches that reflect the true differences we have rather than just being fights in the talk pages, frozen around the assumption that there is a single, neutral point-of-view. And because of their many branches, they can actually merge when people find a way to represent those divergences in ways that can be understood by those who were arguing. That's next-level collaboration.

Imagine creating branched spreadsheets where each branch models entirely different scenarios off of a core set of assumptions that can then get merged together as reality shows us what happens. That's next level collaboration.

Imagine the concrete vocabulary of conflict resolution that emerges as we develop new and interesting merge strategies for conflicting change sets in different application contexts. What does it mean to resolve two different color choices applied to the same object by different people in a drawing program? I can imagine an ad-hoc Syn app spinning up that lets people on the fly make proposals and vote, or do rock-paper-scissors, or whatever… all as part of resolving a merge conflict in another Syn app! That's next level collaboration.

Do you see the implications? I'm excited.

P.S. The holo-hosting team has been using an online-stickies tool to manage choosing topics for our virtual retros. On seeing Syn one of our team members dove right in and got this working in no time:



Holochain Connection:

App Port: 8888 AppId: syn

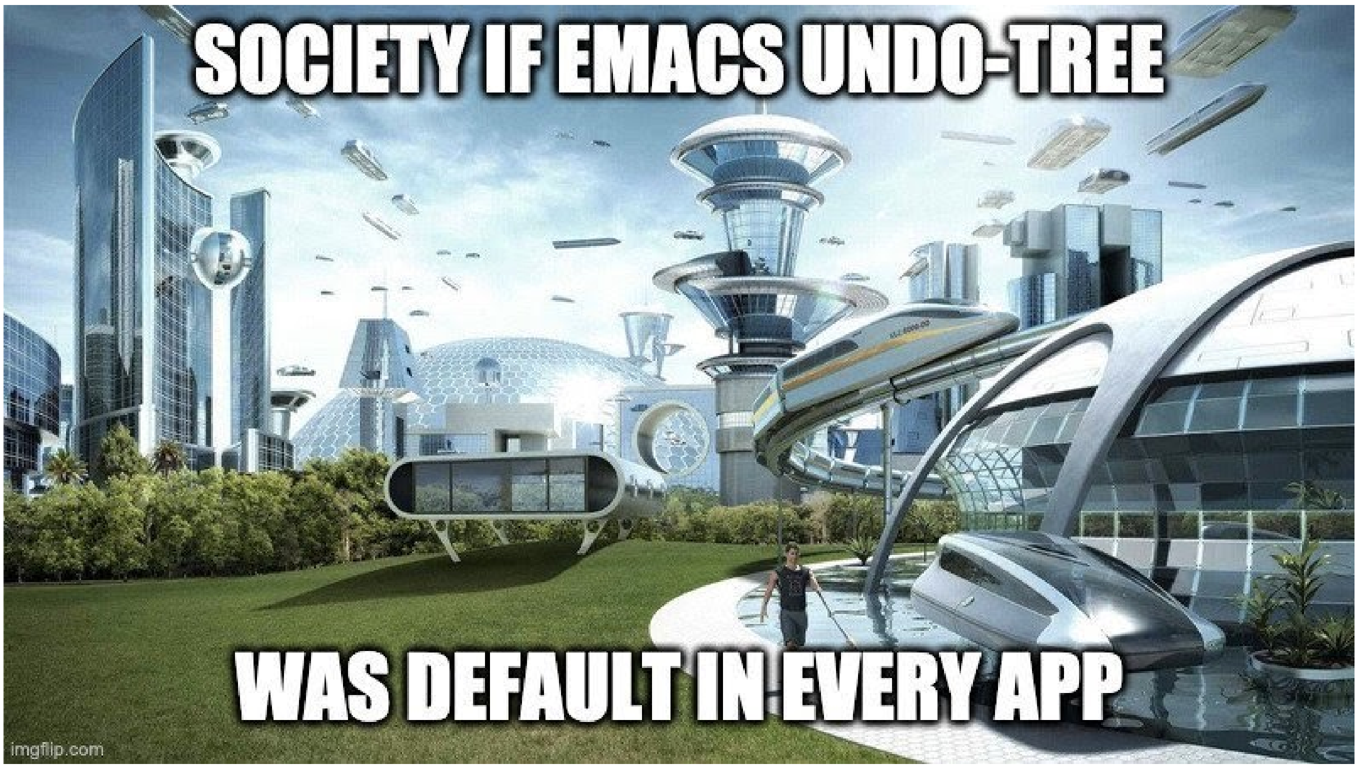
Sessions:

Holochain Connection:

App Port: 8887 AppId: syn1

Sessions:

P.P.S. a little teaser for the hyper-nerdy reader



P.P.P.S A number of folks helped with this article: Thanks first of all to Will for being a co-conspirator-in-syn-itself and for detailed edits and clarifications here. Thanks Jean Russell for supporting and pushing me to get this out, and especially for the initial draft of the comparison table. Thanks to Pospi, Guillem Cordoba, Siddharth Sthalekar, Hedayt Abedijoo, Emaline Friedman for comments edits and suggestions that made this article much better. And finally thanks to Art Brock for the usual co-creative sketching out of things that we can then turn into reality.

Holo