Notes from @cwgoes presentation at the modular summit 2023.

- video
- slides

Intent-centric rollups

What is a rollup?

“A rollup is a type of blockchain that offloads some work to another layer 1. Rollups host applications and process transactions. Once those transactions get processed, they are then published to layer 1. It’s layer 1s hob to order those transactions and check that they are available, at minimum.”

– celestia.org

This is a good definition, but there are three words that we will come back to, Type of blockchain

.

Let’s first answer the question of What is an intent?

Anoma has been using this word for a while, and perhaps it remains criminally vague. The term has become popular, which probably has little to do with Anoma. Many people use the word intent

to mean different things, not exactly described by Anoma, but they seem to be correlated. For example;

- Intents for cross-chain bridging UX ala Uma Roy at Research day in NY
- Intents for user wallets ala Penumbra - intents for users thinking about how to declare what they want a transaction to do or not do.
- Intents let you have your cake and it too - the radical takes.

Language is the original-decentralized coordination scheme. We don’t get to pick individually what terms mean. However, if we zoom out and consider words, there is often some commonality. There is a reason the word intent became popular so quickly, so fast. People use the word in a way where others roughly understand what the word means.

What is an intent

?

“An intent is a commitment to a preference function over the state space of a given system.”

– everyone (approximately)

As opposed to a transaction which specifies a specific imperative execution path to A then B then C. An intent says I want an execution path that satisfies these constraints. I have these preferences that I want executed. The state space of a given system is what varies. For example, when @Uma

is talking about cross-chain bridging, the system is like the chains you are interested in bridging between.

When Anoma is talking about who knows what, the system is like the information flow. The bounds of what the system is that the intent refers to can change. But in all cases, intents are these credible commitments to preference functions.

Let’s zoom out. In the modular ecosystem, everything is organized around this concept of rollups. In our world (Anoma) everything is organized around this concept of intents. These are two interesting concepts because they come at the problem from two different approaches.

- Rollups are bottom up - you start with a modular thesis - start with DA as the base layer, then you build rollups on top, and we see a proliferation of execution environments, different approaches to sharding, etc.
- Intents come at thing top down or user down - users have intents. They are going to start with intents and the system better figure out how to do something reasonable. As system architects, we don’t get to choose, users will have their intents, and we have to try to build something that can satisfy that as credibly and fairly as possible.

In particular, let’s ask; can intents help out the modular ecosystem?

Mustafa in his talk earlier brought up a bunch of challenges and open problems. We have a slightly different list, but it shares several components

Three challenges for modular

- Inefficient sharding
- Application lock-in
- User-facing complexity

By challenges, we don't mean flaws. As the ecosystem is moving presently, if we try and foresee what might happen, and we try and avoid a few problems while we still have a chance to steer things, here are a few things to be cognizant of that perhaps intents can help with.

Challenge 1: inefficient sharding

- Atomicity is expensive
- Users want cross-rollup
- Applications
- Transfers
- Interactions
- Applications
- Transfers
- Interactions
- Rollups implement static sharding
- No response to "atomicity demand"
- No response to "atomicity demand"

Atomicity is the most expensive thing in distributed systems. It requires that you send messages to one place and order them there. That implies N^2 consensus communication. The typical problem of sequential behavior in concurrent systems.

Users want everything. They want cross-rollup applications, transfers, and interactions. Users are not going to think about - I should put all of my state on rollup one to make it most efficient. Users will say I have assets on rollup one and want assets on rollup 2, it's our job to make it work. In making it work, we want to give as much freedom as possible, when properly constrained, to the operators of the system to make it more efficient.

If we envision a world of rollups and all the rollups have different applications and the applications are bound to rollups and users hope to do lots of cross-application interactions, then we have added this weird constraint. There is a demand for atomicity with shared state. If we tie applications to rollups with separate sequencers, then we get this static sharding system - where we can't change the topology of which different rollups are settled atomically. If we view the demand for atomicity as varying dynamically over time, this seems inefficient.

Challenge 2: application lock-in

- Apps could get locked-in to rollups
- Heterogeneous protocols/VMs make life hard
- Apps may pay for more atomicity than necessary

One of the great things about the modular stack is that you can build heterogeneous execution layers very cheaply because it doesn't require deploying a whole layer one. One challenge of heterogeneous protocol execution layers is that they make applications less portable.

The EVM for example, is an interesting VM, but it changes slowly. It would be easy to fork the EVM and launch a rollup with new EVM precompiles. One disadvantage, from the application or ecosystem perspective, is that then if other rollups don't adopt this new opcode then there is application locked in.

For example, if the Sequencer of the rollup starts charging higher fees and users can't switch, if there is a bunch of application state that gets tied to this specific execution system then apps get locked in. This means that applications will pay for more atomicity than strictly speaking is necessary.

Challenge 3: user-facing complexity

- Modular component selection entails complex security assumptions
- Different parties for solving, execution, DA…
- Different parties for solving, execution, DA…
- How can users reason about what's safe?
- Different interactions require different safety levels…
- Different interactions require different safety levels…
- How can users switch if something goes wrong?

One modular component selection and modular component construction adds a lot of clarity to the design construction of blockchains and allows teams to work on different parts, which is helpful from a coordination perspective. If these different parts are operated by different sequencers or validator sets, it tends to entail complex security assumptions. Requires that we create good standards

We think about this from the perspective of the user, what they have to reason about to know whether their interaction is safe. If there are different parties for solving, DA, or execution. Every time the user wants to send an intent or transaction, they are not going to reason through the crypto-economic calculus to determine if their interaction is safe.

We would always like to maintain sovereignty - users can switch if something goes wrong. Communities own the system. They bring value by bringing their applications or users to the blockchain. In particular, communities need to be able to credibly threaten to switch if something goes wrong and not actually do it because it's cheaper.

What if…

… we define applications not on top of rollups, but as intents?

Applications are kind of defined on top of rollups in the modular stack. There is a DA layer, execution, rollups have state formats and instruction sets, VMs and applications are defined on top of the rollups

Intent representation in Anoma

- Intents specify what parts of state they must modify atomically
- State referenced can be help on heterogeneous security domains
- Each piece of state has a party who must sign

If we think about the whole system as having a sharded state where the state is sharded by concurrency domains and security domains, intents specify in an explicitly include fashion which part of the state they must modify. The state can be held in different domains. If you require atomic settlement between two completely validator sets, that is not possible. But you can specify in the intents which things you need to be atomic and the custodians of that state (validators, sequencers) must sign.

We have a concept called partial solving. Solving means matching intents. Solving can be done fully - you take a bunch of intents, you match them completely and form a fully balanced transaction. You can solve intents partially.

github.com

anoma/taiga/blob/main/book/src/images/exec_3_party.png

This file is binary. show original

In this example, we take Alice's intent and Bob's intent. Bob already has something Alice wants. We take Bob's dolphin and send it to Alice. Now we craft a new intent that requires we get a tree and give a star. We can do this by combining two intents doing some type of simplification and creating a new intent that we then send elsewhere to do more solving later, e.g.,

A⇒ B, B⇒ C, A⇒C

What is partial solving? Partial solving is a rollup. If we think about type of blockchain, what does type of blockchain mean? There is some hash linking, there is some history, and we need to be able to verify this later. Partial solving satisfies all of those properties, its just on demand.

- We look at the intents,
- do some type of partial state change,

- we have some state changes we need to do,
- we commit to those,
- we can make them private,
- roll them up in a ZKP,
- and send the intents onwards.

Intent-Centric Rollups

- Partial solving = rollup creation
- Local liveness, global compositionality
- This requires standardizing on a state format
- Luckily, it’s a universal one - no UTXO/account dichotomy
- This requires standardizing on a state format
- Luckily, it’s a universal one - no UTXO/account dichotomy
- Heterogeneous instruction sets are fine
- But need the ability to verify all of them
- Not convinced this matters anyways - compilers problem
- But need the ability to verify all of them
- Not convinced this matters anyways - compilers problem

Perhaps the difference with the current modular stack is that rollups are created on demand. This has some advantages. It allows for this kind of global compositionality and determining the actual topology of sharding at runtime (when you are processing intents), preserves local liveness, and perhaps standardizing on a state format which maybe controversial. We can probably do this without constraining choices.

One nice thing about intents is that they specify verification conditions, not the execution method. Which means you can have different instruction sets - preserve heterogeneity of execution systems - as long as you can verify. For example, If everything is a ZK rollup then intents specify conditions for verifying the other guy’s ZK rollup. If the other guy uses some other opcodes internally, you don’t care as long as the condition is eventually satisfied. It is a standard which allows you to agree on as little as possible,

Dynamic Sharding

- Shard assignment is chosen at runtime
- Intents can specify many valid consensus providers
- if I’m paying for coffee, most anything is fine
- if I’m paying for coffee, most anything is fine
- Enables the network to dynamically sort into independent atomic bundles
- Cheaper for users!

Intents can specify which consensus providers they are okay with. You don’t need to fix sending your intent to one specific rollup. You can say I want the cheapest settlement subject to these conditions and these are the security assumptions I’m okay with. This enables the network to dynamically sort into independent atomic bundles, so it should be cheaper for users.

Application portability

- Applications defined by intent formats
- Applications can move freely across rollups
- The same application can shard its state
- according to what users want!

- according to what users want!
- Credible threat of forking out extractive operators

Defining applications by intent formats, if done well, should help with application portability because applications then are not tied to a specific rollup, they can move freely across rollups. Maybe heterogeneous instruction sets become specialized solving algorithms for different domains. The same application can shard its state depending on what users want, so you can move code and data across chain. Application portability gives you a credible threat to fork out extractive operators because it's easy to move your application code and logic somewhere else. Everything is standardized to a sufficient degree. If someone is extracting MEV you have a credible threat to leave.

Information Flow Control

- Intents in Anoma specify declarative constraints on information flow
- "C must be revealed to A&B"
- "Y must be revealed to C at block 123"
- "C must be revealed to A&B"
- "Y must be revealed to C at block 123"
- Enables things like:
- Cross-rollup private bridging
- New auction designs
- Privacy-preserving governance
- Programmatic disclosure of aggregate data

Specifically in Anoma we've been trying to craft a good framework for describing declaratively what information flow users want to allow for intents. This looks like declarative constraints. Intents can say "in conjunction with this atomic settlement, this value x must be revealed to A and B" X could be a note or a key.

These kinds of declarative information flow constraints allow things like cross-rollup private bridging, new auction designs, privacy-preserving governance, programmatic disclosure of aggregate data.

… so wtf is a blockchain?

- With intent-centric rollups, blockchains are created on-demand
- They live ephemeral lives
- L1? L2? L3?
- Observer-dependent!
- Observer-dependent!

A blockchain is a data structure. If you take a piece of data and you hash it with another piece of data and you include it, you've just gotten a partial ordering relation. This is the essential thing and everything else can perhaps be separated.

With intent-centric rollups we just create blockchains on demand. They live ephemeral lives where a blockchain exists for a moment when two intents are matched then its rolled up and it can be verified later. You still need somewhere for the data to be stored. The blockchains are quite ephemeral. Weather something is an L1, L2 or L3 is just an observer dependent finality choice.

Memes

h/t @jon_charb

Rollups are L2s

Rollups are L1s

Rollups aren't real

Blockchains aren't real

Intent-centric modular economics

- Service provider DAOs
- Data availability DAOs (Celestia, Avail, Eigen DA, Ethereum)
- Execution DAOs (current rollups?)
- Solver DAOs (SUAVE)
- … competing on the basis of liquidity + role-specific optimization
- Data availability DAOs (Celestia, Avail, Eigen DA, Ethereum)
- Execution DAOs (current rollups?)
- Solver DAOs (SUAVE)
- … competing on the basis of liquidity + role-specific optimization
- Assets people want
- BTC, ETH
- Ye Almighty American Empire Bucks
- … competing on the basis of distribution and PGF
- BTC, ETH
- Ye Almighty American Empire Bucks
- … competing on the basis of distribution and PGF
- (h/t @zmanian

)

If we conceive of a modular world of intents;

There are two classes of things people will want in an intent-centric modular world. The first class is service provider DAOs. A group of operators who are working together to provide services as a collective, coordinating together to provide that service efficiently and reliably. Users or applications see it as a service provided as a whole.

One kind of DAO in an intent centric world is a data availability DAO, for example. Service DAOs are competing on liquidity and role-specific optimizations.

Then there are assets people want. The assets are competing independently of protocols based on distributions and how good they are at public goods funding.

Three concluding thoughts

- Intents <3 Modular
- the last architectural convergence
- the last architectural convergence
- Ethereum meets Cosmos
- Also maps to the social structures!
- Also maps to the social structures!
- Make your rollups private now!
- avoid mistakes you don’t need to make
- avoid mistakes you don’t need to make

Intents and modular are like a match made in heaven. They come at the problem from opposite directions, they can kind of help solve each other’s problems. One challenge we’ve had in building Anoma is that we don’t have specialized primitives.

We don’t have efficient DA sampling or individually optimized things, so there is a nice synergy here.

One thing to like about the modular blockchain world is that it seems like a sort of fusion between Ethereum and Cosmos. It is a fusion of the polycentric or self-sovereign political ideology with the kind of clear architectural thinking of the Ethereum ecosystem.

Let’s please not remake the mistake of making transparent rollups, this is not going to work.