

Anoma the protocol

The Anoma protocol is the logical framework, a meta-logic that can describe and manage various different types of logic systems. Agents use this framework to read, create, and process messages. In particular the Anoma protocol takes inspiration from Conal Elliot's [Denotational Design](#) (DD).

In simple terms. The idea is to specify a higher level abstraction that is intended to be complete, in that enough is said to define precisely what a valid implementation of Anoma must do, and minimal, in that no more is said than that.

Defining the Anoma protocol as abstractly as possible allows for maximum composability of implementations with slightly different subcomponents, since everything is decoupled. For example, architecturally, you could swap out Halo 2 for PLONK in Taiga, swap out the execution engine implementation for a different one, etc. and nothing else in the protocol or architecture would need to change.

-Christopher Goes

Think of the Anoma protocol as a Platonic instruction set which specifies the properties that a valid implementation of Anoma should have, but is agnostic to implementation details. Anoma is completely customizable as long as the specified constraints or valid implementation is achieved. While you can build a valid implementation of the Anoma protocol with Taiga

and Typhon

, this is not a requirement. It is only required that the implementation follows from the specification, adhering to the required properties.

ELI5

Anoma tells you how to make a salad. The salad requires a combination one choice from each category of leafy greens, fruit, vegetables, nuts, and dressing. The ingredients must be fresh. The salad must not contain no synthetic ingredients. How you make the salad is up to you, but the salad can only be an Anoma salad if it follows these instructions.

Architecture

[

1576×882 120 KB

](<https://europe1.discourse-cdn.com/standard20/uploads/anoma1/original/1X/ff59f808f9a71290e2d02b2a7cee500e755111ea.jpeg>)

The diagram attempts to show the separation between the specification of the higher level abstraction and a more concrete specification that suggests a Logic and engine implementation; e.g., Taiga

and Typhon

. This distinction between abstract and concrete is also reflected in the [Zcash specification](#), sections 4 and 5.

In particular, Identities, Resources, and Physical DAG, which are Basic Types, play an important role in describing the Architecture. These basic types map to components of the concrete protocol. Below we'll attempt to describe a few.

Identities

- Identities

are the interface through which agents can create and verify signed and encrypted messages. They comprise two pairs of inverse functions: sign-verify and encrypt-decrypt

- Agents

are non-deterministic participants of the system, stateful entities which can send and receive messages

- [Controllers

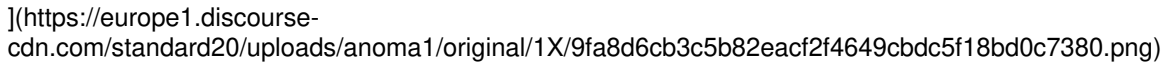
](<https://research.anoma.net/t/controllers-from-first-principles/66>) are the Identities

(e.g. consensus providers) that determine the order of txs

including the given Resource.

[

1222x722 75.2 KB



Resources

- Resources

are the building blocks of the Resource Management System that constitutes the contents of Messages and the substrate on which higher layers (e.g. DAGs representing some kind of State) are built

- Resource_Logic

specifies under which conditions Resources that carry it can be created and consumed. It is defined by its Predicate and its Arguments

```
data ResourceBody = ResourceBody { resource_logic :: ResourceLogic, prefix :: [ContentHash], suffix :: Nonce, quantity :: Natural, controller :: ByteString, TerminalDAG ExternalIdentity
```

[

1128x442 21.9 KB



This diagram aims to show how Resources decompose into their components. All Fields consist of a content hash, which provides content addressing for all elements of all layers.

The Resource management system can be implemented as a UTXO or account model, or both. That is up to the particular developers and what they want.

Physical DAG

- The Physical DAG

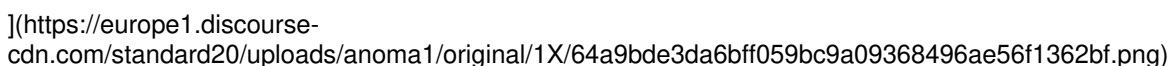
is the layer directly on top of the lowest-layer network, responsible for providing local partial ordering information.

- An Observation

is a type of message which attests to witnessing some data (possibly other messages), and provides a signature along with an external identity.

[

876x850 53.9 KB



The Physical(DAG) is like an information highway where each node can connect to any other node, creating a map of interactions. This system doesn't dictate the sequence or validate these interactions, but it traces when and where they occur, ensuring consistency among users working with the same data

Summary

In this post we present a framing of the protocol and iterated definitions as well as diagrams which fall within the Architecture topic of the specs synthesis concern. This is in Ideas as the diagrams, definitions, and framing all have flaws that should be worked through together

before elevation to a coalescence.