THIS POST IS A WIP, PLEASE IGNORE ME FOR NOW!

The attempt of this post is to:

1. Characterize what Anoma is, from first principles, with metaphors/concepts that are relatively accessible, but still convey the key points, and

2. Explain how Anoma relates to the existing Ethereum ecosystem,

– all in less than <> words.

I. What is Anoma?

The word “Anoma” can mean many things. Here we will talk about two: the Anoma protocol

and the Anoma network

. The Anoma protocol

is a distributed operating system for intent-centric applications, and the Anoma network

is the interlinked graph of computers running this protocol.

What does this mean? Let’s break it down into the following series of questions:

1. What is an operating system

?

1. What does it mean for an operating system to be distributed

?

1. What is an application

?

1. What does it mean for an operating system to support intent-centric

applications?

1. What does it mean to participate in the Anoma network

?

What is an operating system?

An operating system:

1. Abstracts diverse hardware behind a universal interface, and

2. Provides an environment in which applications can run.

Three classes of operating systems are predominant today:

1. Desktop operating systems such as Windows, Linux, and Mac OS.

2. Mobile operating systems such as Apple and Android.

3. Web browsers such as Chrome and Safari.

These three classes of operating systems all fit this definition, but differ in what hardware they seek to abstract and what kind of environment they seek to provide. Desktop and mobile operating systems are concerned with abstracting physical

hardware, while web browsers already run on an existing OS and can make use of the hardware abstractions provided by that OS (for which they typically provide even higher-level APIs accessible to webpage code). While all three classes are interested in providing an environment in which multiple applications can run safely, mobile OSs and web browsers are typically more concerned with limiting application permissions and cleanly sandboxing application code.

Similar to web browsers, Anoma is an operating system that runs on top of an existing “base-level” operating system which abstracts differences in physical hardware. Rather than aiming to display websites, however, Anoma seeks to provide an abstraction over hardware that is distributed

, which brings us to our next question:

What does it mean for an operating system to be distributed?

Rather than seeking to abstract the hardware of one computer, a distributed

operating system such as Anoma aims to abstract the hardware of many networked computers so that applications can treat this network as if it were a single abstract machine.

Specifically, Anoma seeks to:

1. Abstract a network of computers which is:

a. dynamic – nodes may enter and leave the system at any time, physical link layer properties may change, etc.,

b. heterogeneous trust – nodes may make different and arbitrary trust assumptions, and

c. scale-free – there is no specific fixed or characteristic scale of either the network itself (in terms of number and connections of nodes) or of applications (in what kind of distribution they will require), and

1. Provide an environment for applications which:

a. Allows applications to treat this distributed network as if it were a single abstract machine with a single state (where the underlying distribution will entail certain restrictions on how that state can be modified),

b. Gives applications the ability to control how a part of that state (the application's state) can change, and

c. Provides appropriate isolation and interfaces so that many not necessarily mutually trusted applications can run simultaneously and interoperate.

Let's talk about state. The state of a single computer can (abstractly) be modeled as a set of registers

- numbered slots - where applications can read and write values to a particular subset of registers which they control, and the operating system tracks which applications own what memory in order to enforce that one application doesn't write to the memory regions controlled by another. Anoma models state using immutable resources

, which can be created and consumed only once. Viewing the network as a single machine, resources are a bit like versioned registers

of that machine, where:

- a given resource represents a specific version of a given register,
- in order to update a register, you must consume the previous resource (old version of that register) and create a new resource (new version of the register, which can have a different value), and
- the resource logic

, an arbitrary program associated with each resource, determines what changes can be made to the state of the register.

Anoma allows applications to program their part of this state by writing resource logics, which have full control over how any resources associated with those resource logics can change.

Resources are distributed over the Anoma network. Each resource has a unique controller

– a consensus (or single node) who must order any changes to that resource – but the resources across the network may have many different controllers, and resources can move between controllers at will. This allows for continuous redistribution of transaction load depending on supply, demand, state codependencies, and trust preferences.

What is an application?

An application

is a program which runs on top of the distributed operating system, governs how a part of state can change, and is typically concerned with providing a specific interface to users which has a certain semantics. For example:

- A token application, governing a set of token resources, would provide a notion of an owner

of each token, where the token can be transferred only with the explicit authorization of the owner.

- A chat application, governing a set of channel resources, would provide a notion of authorized writers

in each channel, where only the writers can post to the channel (and maybe a specific subset of writers have the ability to change the set of writers itself).

Anoma provides a high-level interface so that developers can write applications directly in terms of intents

, bringing us to our next question:

What does it mean for an operating system to support intent-centric applications?

This section has yet to be written

.

Intent-centricity enables universality, because the protocol makes no decisions.

Specific affordances for applications:

- programmable storage
- programmable compute
- programmable disclosure
- programmable permissions

II. How does Anoma relate to Ethereum?

This section has yet to be written.

- Philosophically: a distributed OS for the world computer
- Practically: resource plasma
- Components: node sidecar, protocol adapter

III. How will Anoma be deployed?

This section has yet to be written.