Bridging the Physical & Blockchain World with Erlang.

By Andrew Thompson

Code BEAM SF 2018

Who am I?

- Writing Erlang since ~2008
- Wrote `gen_smtp`
- Wrote `lager`
- Survived Basho
- 4 year veteran of the IoT wars
- Blockchain skeptic

What this talk is NOT.

- Investment advice
- ‘Expert advice’ – please do your own research
- An ICO pitch
- An in-depth guide

What is a Blockchain?

A blockchain is a way to **forge consensus** given a set of untrusted actors where up to some proportion of the actors are acting maliciously, have crashed or are unavailable.

Blockchains are often, but not always **‘decentralized’**; there’s not a central service/database/arbiter. This can be a very useful property if there’s concern the entity behind the service will get acquired/shutdown or change its behaviour. Decentralized systems also require a majority to accept changes to the protocol or process.

Anatomy of a Blockchain

- Consensus Algorithm
- Peer to Peer Network
- Transactions
- Blocks

Consensus.

Consensus is the way the blockchain comes to agreement if a transaction is valid, a block is valid, a value is being computed correctly, etc.

There are many ways different blockchains achieve consensus with various tradeoffs around speed, power consumption, what percentage of malicious actors are tolerated, how ‘open’ the membership of the system, etc.

Peer to Peer Network.

Every member of the blockchain network needs to communicate with other members of the blockchain. Isolated peers cannot usefully participate in consensus and can be tricked into accepting or generating invalid transactions and blocks.

The modern internet is hostile to peer-to-peer applications; NATs, firewalls, dodgy IPv6 deployments. Most blockchains have extensive support for traversing challenging network topologies.

Most peer-to-peer traffic for a blockchain is not encrypted (most blockchains don't rely on secrecy) but is usually authenticated with cryptographic signatures, or an equivalent.

Transactions.

Transactions are the operations against the shared state embodied in the blockchain. When some actor wants to change the state of something in the blockchain, they submit a transaction. This transaction can add data to the blockchain, trigger code to be executed in the context of the blockchain (smart contracts), change token balances, etc.

Blocks.

Blocks are like checkpoints; they encapsulate some amount of change to the system (transactions), they have a total ordering and they’re tamper proof. Changing any single block will invalidate any further blocks because they’re cryptographically linked together.

Types of Consensus.

- PBFT derivatives – Ripple, Stellar, etc
- Nakamoto Consensus – Bitcoin, Ethereum (now), many others
- Proof of Stake – Ethereum (future), various others
- Many others (Filecoin, Factom, etc)

PBFT Based.

Practical Byzantine Fault Tolerance – many variations. Can have open or closed membership. Open membership is subject to Sybil attacks, so it may be combined with other factors like staking (Tendermint).

Closed PBFT replica sets are often considered to be centralized, because someone or something is gatekeeping membership.

In PBFT the client’s transaction is sent to a ‘primary’ which broadcasts the transaction to its ‘secondaries’, the secondaries evaluate the transaction against the blockchain and return the result to the client. If the client sees more than N agreeing replies, they know it was accepted

Nakamoto Consensus.

Open membership, Sybil attacks are defeated because of the extremely high compute requirements.

Nakamoto consensus works by **computing some hash over the block such that the numerical value of the hash falls under some threshold**. This threshold adjusts regularly to try to make it so new blocks can only be computed every N seconds.

Probably the easiest consensus to implement but also the most wasteful. The proof-of-work has no value outside its difficulty.

Proof of Stake.

Proof of Stake is where actors possessing some predetermined amount of some asset (likely native tokens) are trusted to provide consensus. Typically actors found misbehaving lose some or all of their stake.

Proof of Stake is an area of intense research right now because many blockchains are struggling with the limitations and energy cost problems with Nakamoto Consensus.

Proof of Stake blockchains are just starting to come online, they are relatively untested compared to the other approaches.

Why Erlang?

Most popular blockchains are written in C++ or Go. Rust is also gaining popularity. There are also (at least) 4 blockchains in Erlang: AEternity, Arweave, Ercoin and soon one from Helium (and people working on at least 3 of them are at this conference!). 👋

Erlang is **memory-safe, fault-tolerant, has pretty good cryptographic libraries** and has good tooling for building robust services (Quickcheck, Dialyzer, common_test/eunit, etc). There are also a lot of useful libraries, and the current ongoing blockchain work is adding more.

Libraries we’ve made:

github.com/helium

- **Erlang-libp2p**
- **Erlang-multihash**
- **Erlang-multiaddr**
- **ECC_compact**
- **Kdtree**
- **Merkerl**
- **BEAMCoin (now there’s at least 5 Erlang blockchains!)**

And now, for something mostly different.

What’s up with the IoT?

The IoT has been a thing for a long time now, but it still hasn’t arrived. Why?

- Confusing morass of protocols and transports
- Vendor lock-in & hidden proprietary parts
- Tough to own your own data
- Tough to share infrastructure
- What happens if the vendor or network provider goes out of business?

We at Helium are sick 🤢 of these problems.

What are we going to do about it? How do we build an open-access network with reliable crowd-sourced gateways?

Design a **new radio protocol** from scratch using unpatented , widely available modulation and error-correction schemes

Design a **low-cost software-defined radio based gateway** and open source the hardware and firmware

Design a **reference end-node implementation** and open source the hardware and firmware

Design a new **consensus scheme around 'proof of coverage'** that allows gateway operators to get paid for delivering packets and providing coverage.

Much ado about Coverage.

To build a network of connected, wireless devices, you need gateway infrastructure. This infrastructure should be reliable, ubiquitous and accessible. Today we have several wireless data providers; LoRaWAN operators, Sigfox and the cellular carriers.

Helium doesn't consider these solutions sufficient to answer the demands of the market. Helium's approach to delivering a robust, scalable, public wireless IoT network centers around the network verifying its own integrity, incentivizing useful coverage and responding to user demand.

Proof of Coverage?

1. Verify a gateway is **where** it says it is
2. Verify a gateway is **listening** for radio packets
3. Verify a gateway can **transmit** packets

Verifying these 3 aspects of the network give us a **replacement for hashpower in Nakamoto consensus.** Instead of considering compute power as the scarce commodity, we use location and the physical limits of radio frequency (time of flight, bandwidth, inverse square law) as our scarcity.

This allows us to build Sybil-attack resistant identities for our blockchain.

How does it work?

1

A gateway (the challenger) is assigned (using entropy from the blockchain) another gateway **to verify** (the target)

2

The challenger, using the gateway locations asserted on the blockchain, **constructs a regional view** of the network around the target

3

The challenger then constructs an **'onion routed' challenge packet** that traverses the regional network, intersecting with the target at some point

4

Each gateway **decrypts a layer of the onion**, broadcasts the next layer and sends a receipt (ToA, RSSI, Hash) to the challenger

What happens next?

Once the challenge packet has terminated (reached the final gateway in the chain or hit a routing gap) the challenger assembles the received receipts into a Proof of Coverage.

The Proof of Coverage + a Proof of Time comprise the Proof of Work.

All the blocks published to the network for that block height are ranked according to the consensus algorithm.

Each gateway mines the next block on top of the best candidate for the last height (a vote for consensus around that block).

So what does that get us?

A **self-verifying, decentralized network** of gateways fixed in space and time.

Wide area Time Delay of Arrival **location services** for any basic transmitter.

A way for the network to **profit from its usage** (packet routing fees) and return it to its maintainers (mining rewards/ transaction fees) and, as such, provide for its own sustainability.

Redundant wireless coverage with up-to-date mapping and status.

Cryptographic proof of a packet's context in space and time (where/ when did this thing happen).

How is this better than Cellular/LoRa, etc.?

Everything is open;
no hidden patents, licensing fees, etc

Hybrid coverage model:
seamless mix of user and corporate deployments

Built-in Geolocation:
Native support for device location

Cost Effective;
Prices are truly competitive

Cryptographic proof of location:
Enables a whole new class of use cases

Watch this Space.

We have prototyped most of the components of this system and are currently putting the pieces together.

We will be publishing more details, source code, hardware schematics and updates soon, and on an ongoing basis.

Questions?

Please find me, or one of my colleagues, afterwards for comments/ philosophical discussions.

@potsdamnhacker
andrew@helium.com