MONET: Mobile Ad Hoc Blockchains

Martin Arrivets
www.mosaicnetworks.io

v1.0 (14/05/2018)

Abstract

The peer-to-peer economy, where individuals are both consumers and providers of goods and services, is paradoxically built on centralised client-server infrastructure. Attempts at moving these services on public blockchains, to mitigate the trust factor, are very inefficient because they require every node, across the world, to process everybody's commands. Part of the problem can be addressed by designing more efficient consensus protocols, but the idea of implementing localised services on a global public network is fundamentaly flawed. We suggest a system, based on mobile *ad hoc* blockchains, whereby groups of people involved in any task or activity can effectively form temporary networks, with their mobile devices, and coordinate themselves without delegating full responsibility to trusted third parties. Users only connect to people they need to be connected to, based on service and location, for the duration of their interaction. The system initially relies on Babble, a consensus module suitable for mobile deployments, and the MONET Hub, a public blockchain providing infrastructure services.

This is a conceptual document, likely to evolve with community feedback and experimental evidence. The definitions of interfaces and protocols are not intended to be comprehensive and final. Where tokens and Proof-of-Stake are mentioned, more details will be provided based on community ideas and critiques.

Contents

Glossary

Babble: A pluggable software component that brings fast Byzantine fault-tolerant consensus to distributed applications. It produces the Babble blockchain.

Blockchain: A continuously growing data structure representing a list of records, called blocks, which are linked and secured using cryptography. We sometimes refer to Blockchains as peer-to-peer networks that use this data structure as the underlying support for a consensus protocol.

BFT: Byzantine Fault Tolerance. The characteristic which defines a system that tolerates the class of failures that belong to the Byzantine Generals' Problem [1]. Byzantine Failure is the most difficult class of failure modes for a distributed system as it implies no restrictions, and makes no assumptions about the kind of behavior a node can have.

Consensus Algorithm: A protocol for achieving agreement among distributed processes or systems.

DAG: Directed Acyclic Graph. A directed graph that has a topological ordering, a sequence of the vertices such that every edge is directed from earlier to later in the sequence.

FIP: Full Information Protocol. A communication protocol where processes tell each other everything they know at every step.

Hash: The result of applying a cryptographic hash function that maps data of arbitrary size to a bit string of a fixed size and is designed to be a one-way function (impossible to invert).

Merkle Tree: A tree constructed by hashing paired data (the leaves), then pairing and hashing the results until a single hash remains.

Mobile: We generally refer to mobile devices as modern smartphones, tablets, or wearable devices, capable of connecting to the Internet, equipped with a processor and memory.

MONET: An open network architecture based on mobile ad-hoc blockchains and an interoperability protocol.

MONET Hub: An *always on* utility blockchain providing optional infrastructure services for MONET applications and ad-hoc blockchains.

NAT Traversal: A computer networking technique of establishing and maintaining Internet protocol connections across gateways that implement network address translation.

Node: In this context, a device participating in a consensus protocol. We use the terms *node*, *participant*, and *validator* interchangeably as they are usually one and the same. Generally, however, *validators* are special nodes that play an active role in the protocol, whereas, some nodes can be passive and only consume the output. *Participant* also refers to the person or entity manipulating the device.

Light-Client: A program that does not fully participate in a blockchain network but is capable of verifying blocks.

Peer-to-Peer: A distributed application architecture that partitions tasks or workloads between peers. Peers are equally privileged, equipotent participants in the application.

Proof-of-Stake: A category of consensus algorithms for public blockchains that depend on a validator's economic stake in the network.

Server: A computer program or a device that provides functionality for other programs or devices, called "clients".

Smart Contract: A program deployed across a distributed, decentralized blockchain network. Smart Contracts permit trusted transactions and agreements to be carried out among disparate, anonymous parties without the need for a central authority.

Introduction

Many online services, that we use on a daily basis, only involve people in relatively close range, and rely almost exclusively on trusted third parties acting as coordinators. Mobile devices, that keep us all connected, only operate as clients to the servers maintained by these intermediaries. While the system works well, it suffers from the inherent weakness of any centralised model; users are forfeiting privacy, transparency and control over their data and computation. Furthermore, the cost of mediation is ultimately passed on to the end-user.

Systems built on peer-to-peer networking, consensus algorithms and cryptography (blockchains like Bitcoin and Ethereum) provide a framework for individuals to enforce contracts amongst themselves without outside help. However, as currently implemented, they are not fulfilling their potential because most users still interact with public blockchains in the same way they interact with other online services; the actual computation, data, and work involved in securing the service is left to somebody else.[1]

[1] At the time of writing, the Bitcoin network is composed of 11732 nodes while Coinbase alone, the global market's largest bitcoin brokerage and wallet platform, serves more than 13

What is needed is a system for serverless mobile applications whereby people can interact directly with one another without reverting entirely to a third party or an extended set of participants. This involves three fundamental operations:

1) **Distributed Consensus**: reaching agreement among a group of peers, connected by an unreliable communications network, without reverting to a third party.

2) **Interoperability**: enabling information exchange across consensus networks, online or offline.

3) **Peer-discovery**: finding other peers to connect with, based on location and activity.

MONET is an open network architecture, materialized by a set of software tools and protocols, designed to address this matter.

We are building a free and open-source Software Development Kit (SDK) which enables developers to add blockchain consensus to their applications. It introduces a new paradigm whereby users dynamically join or form local blockchains for the duration of their interaction, and eliminates the need for servers. Ephemeral by nature, these mobile *ad hoc* networks only exist for the duration of a group's activity; following this temporary collaboration, participants are not required to stay online to support other users. The SDK implements the Babble consensus system which is particularly suited for mobile deployments due to its low messaging complexity.[2]

Thanks to the blockchain design, each user can record an immutable and auditable record of a group's correspondence, providing a base layer for accountability and interoperability. The Babble blockchain is theoretically compatible with existing Inter-Blockchain Communication Protocols (IBC) which generally enable verifying on one blockchain that a transaction happened on another blockchain. This facilitates transferring assets and data across application boundaries, and goes in the sense of an open network architecture.

Peer-discovery services are deliberately under-specified; application developers are ultimately responsible for deciding how users discover each other. Any attempt at providing a general solution for peer-discovery would invariably fall short of its goal, and would introduce an element of unwarranted centralisation. This principle, known as the *end-to-end* argument, heavily influenced the design of the Internet protocols themselves (TCP/IP); they involve a "dumb" network with "smart" systems connected to the network.

A permanent master blockchain, the MONET Hub, optionally supports the first applications on the network with services such as ID, peer-discovery, NAT-traversal and cross-chain communication. It is a *Smart Contract* platform where anyone can deploy programs with non-transient logic and benefit from a secure

million active users.

[2]https://github.com/mosaicnetworks/babble

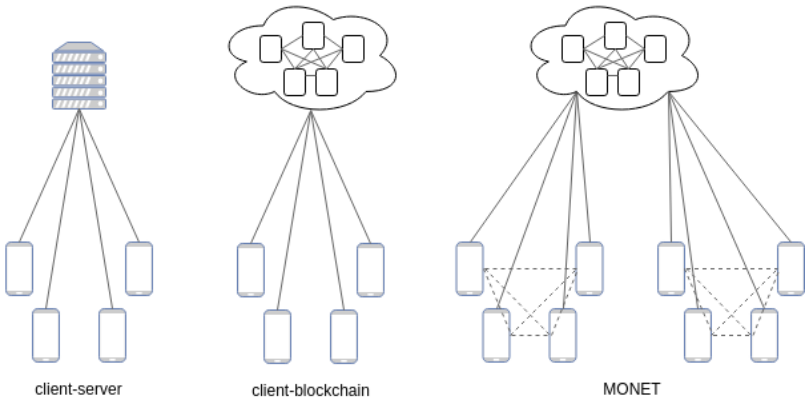

Figure 1: From Client-Server to MONET

distributed computing network. Validators are rewarded for operating nodes and securing the Hub by earning transaction fees in *Tenom*, the Hub's native token, which also serves as the backing asset for the Proof-of-Stake protocol built atop the underlying Babble blockchain.

Nodes

We refer to *nodes* as the physical devices equipped with the software for participating in MONET. Nodes are active blockchain participants; they run a p2p application that connects to nearby peers and performs the consensus functions that ultimately allow a group to coordinate itself without resorting to a central authority. One of MONET's distinctive traits is the ability to operate nodes on *off-the-shelf* mobile devices.

Anyone can run a node, provided they have a connectable computing device with an application that uses a MONET-compatible consensus module. One such module is Babble; it was designed to work with any application, written in any programming language. Hence, we can provide SDKs for all mobile platforms (Android, iOS and Windows Phone). Most mobile devices are already capable of connecting to the Internet through cellular networks (3G, LTE, 4G, etc.) or Wi-Fi. They also have enough memory and processing power to support demanding software like feature-rich applications or the functionality presented in this paper for mobile *ad hoc* blockchains.

When a set of nodes creates a new *ad hoc* blockchain, they agree on an initial list of participants which they each copy locally. The list is subsequently updated dynamically by the blockchain, depending on the application, as individuals request to join or leave. *Join Requests* are processed by the blockchain, and go through consensus as a special type of command. The application contains

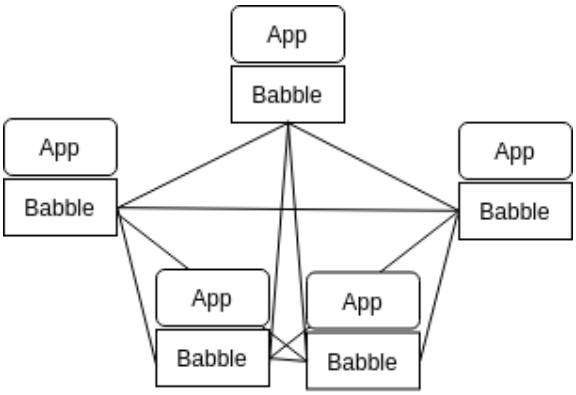

Figure 2: MONET blockchain with 5 nodes

the logic that every node runs to cast its acceptance vote on the request; the membership protocol is thus built-in to the consensus protocol. In practicality, MONET blockchains will perform effectively with anywhere from 1 to 100 nodes.

Identities

Nodes are identified by an *address*; the hash of a public key, created with a asymmetric cryptosystem, whose private key is also controlled by the node. To form or join an *ad hoc* blockchain, nodes must be aware of the addresses of the other participants. This information is necessary for two reasons:

1. To know the exact number of nodes in the *permissionned* consensus system.
2. To verify messages signed by other members using public key cryptography.

Nodes must communicate with one-another in a fully connected p2p network. The consensus algorithm, which replaces the trusted third party, requires each node to be aware of all the other nodes in the group, and each member maintains a list of peers to which it refers regularly as part of the consensus protocol. Public keys are used to verify the authenticity of messages received from other peers.

In the consensus algorithm itself, nodes are identified by the hash of their public key, as it remains unchanged when they roam around and change IP realms. This also makes for a NAT-friendly consensus layer by avoiding addressing in payload [2].

Programming Model

MONET applications are fundamentaly different from the applications we use on a daily basis; instead of using centralised servers, these apps resort to Babble to broadcast messages directly to other participants, and come to consensus on an order of events. However, the Babble SDK makes it easy to adopt this model thanks to a simple API which completely abstracts the details away from developers, and lets them focus on the core business logic and user interface.

Babble and the application are loosely coupled. The application submits transactions for consensus ordering via the *SubmitTx* function, while Babble asynchronously processes transactions and eventually feeds them back to the application, in consensus order and bundled into blocks, through a *CommitBlock* handler. To the consensus module, transactions are just raw bytes and encoding/decoding is left to the application.

The SDK leverages local storage to manage configuration files and persist Babble's underlying key-value database. The most important item is the *peers.json* file which contains a blockchain's list of participants (public key, IP:Port). Both the application and Babble have read/write access to this file; the application is responsible for initially populating the list but all further updates should go through Babble, enforcing consensus on all changes.

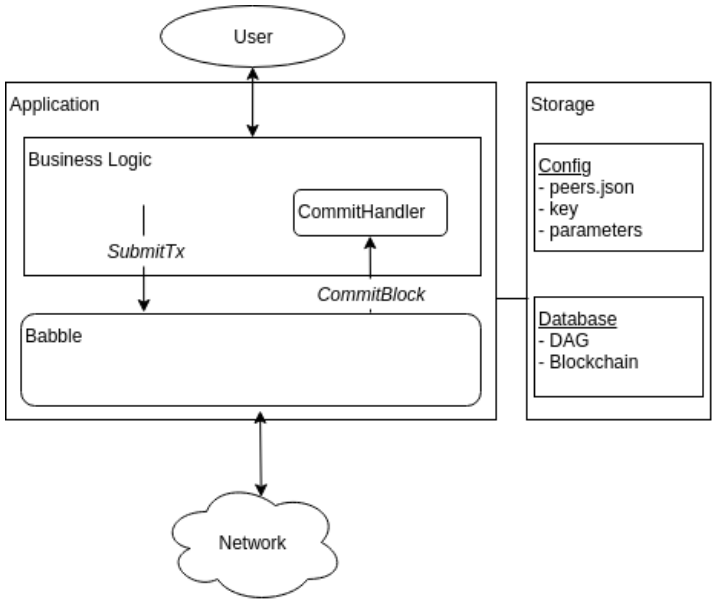

Figure 3: Programming Model

Bootstrapping a new *ad-hoc* blockchain can be achieved in many ways. Fundamentaly, an initiator is expected to create a first version of the *peers.json* file,

containing at least itself as one of the participants. It can then start receiving *Join Requests* from other users, through the application, and eventually communicate the resulting *peers.json* file to new entrants. The questions of how peers are discovered and how public keys and the *peers.json* file are exchanged, are left to the application developer. It falls under the responsibility of peer-discovery, which we expressly leave open.

Consensus

A consensus algorithm enables multiple computing devices to process the same commands in the same order, even in the presence of faulty participants. Coupled with an application that actually processes the commands, it makes for a group of devices performing the same activity as one, without outside help. Different deployment contexts require different types of algorithms: permissioned networks, where all the validators are known, rely on protocols where nodes dialogue among themselves to come to an agreement, whereas public networks, like Bitcoin or Ethereum, use another class of algorithms where a subset of nodes acquires the right to define the order of the next batch of commands, in such a way that the choice is fair, secure and verifiable (Proof of Work, Proof of Stake, random sortition, etc.).

In MONET, mobile *ad hoc* blockchains are formed by small, localised, groups involved in a common activity. Participants are directly connected but do not necessarily trust each other and are most likely using mobile devices. Hence, we are in the *permissionned* scenario with the additional constraints of potential malicious behavior and network failures. Because these blockchains are used to coordinate real-time activities, the output of the consensus algorithm must be final and directly usable; a concept known as deterministic finality.

Babble is based on an asynchronous Full Information Protocol (FIP) which is fitting for this scenario [3], [4]. It is a leaderless, asynchronous, Byzantine fault tolerant system, capable of processing thousands of transactions per second with sub-second latencies. The interested reader is invited to consult the open-source repository, but in short, it is based on the engaging notion of *gossip about gossip*, which intuitively yields enough information to compute a consensus order of events.

The algorithm leverages a distributed data structure that records the history of gossip; It is a Directed Acyclic Graph (DAG) which cannot be tampered with because all its parts are cryptographically signed and chained together. Using this distributed history of gossip, participants continuously compute the group's common knowledge and agree on an order of events through a system of *virtual voting*. Hence, like all FIPs, the core of Babble involves three operations:

1. Gossiping and constructing the Communication Graph.
2. Determining which Events will eventually be common knowledge.
3. Sorting the Events with a deterministic function.

To facilitate interoperability, we added an extra step which consists in projecting the output of the ordering function onto a blockchain. Transactions are mapped against a linear data structure composed of blocks; each block containing an ordered list of transactions, a hash of the previous block, a hash of the resulting application state, a hash of the validator set, and a collection of signatures from the set of validators.[3] This method enables DAG-based systems to implement any Inter-Blockchain Communication protocol and integrate with an Internet of Blockchains [5].

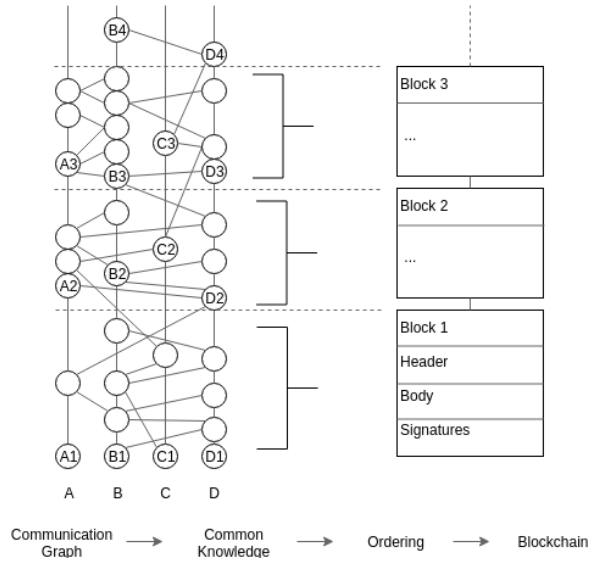

Figure 4: From Communication Graph to Blockchain

Inter-Blockchain Communication

There are several examples, in the traditional client-server model, of one server needing to communicate with another server; this plays a fundamental role in enabling interoperability between services (e.g., Uber interacting with payment systems). In a similar vein, services deployed on blockchain networks require a mechanism to transact outside of their respective circles. This presents at least two challenges:

- Input/Output operations - communication between a state-machine and the outside world - are non-deterministic and could undermine consensus.
- Data integrity - verifying that the item of communication went through consensus on the originating chain - needs to be preserved.

What is needed is an Inter-Blockchain Communication (IBC) protocol.

[3] http://babbleio.readthedocs.io/en/latest/blockchain.html

Although *ad hoc* blockchains are closed and ephemeral networks by design, they can represent immutable and permanent information. Even long after the last node of a blockchain is shutdown, persisted blocks still contain irrefutable cryptographic evidence of the events that took place in the network. This evidence, packaged in a digestible format, provides the basis for an IBC protocol. How the evidence is passed from one chain to another, is where the protocol is extensible.

Inter-Blockchain communication is about verifying on one chain that a transaction happened on another chain. In permissionned algorithms, like Babble, the verification process involves counting signatures. Validators sign block hashes, which are themselves obtained by recursively hashing together, in a simple Merkle tree, the various components of a block, which include: the *state-hash*, the *validators-hash*, the *transactions-hash*, and the *previous-block-hash*. Cryptographic hash functions are such that tampering with any of these intermediary hashes yields a completely different *block hash*.

To certify that a transaction **t** happened on a given chain, we retrieve the block containing **t**, and generate a proof object - an IBC packet - with the Merkle proof that **t** is consistent with the block's transactions-hash, along with the block's header and signatures. Light-clients recursively check the Merkle proofs all the way to the block hash and finally check that the block obtained a majority of signatures from the validator set registered with the light-client upon initialization.[4]

Light-clients can be implemented in *Smart Contracts* and deployed on blockchains themselves. Each light-client is initialized with the public keys belonging to the validators of the remote blockchain. For example, suppose we wanted to verify on chain B that a transaction **t** happened on chain A. We would implement a light-client for chain A in a *Smart Contract* deployed on chain B, initialized with the public keys of chain A validators. We would then submit an IBC packet for **t** to chain B, which would get gossiped and processed independently by each chain B validator by a call to the light-client. Changes to the remote validator set are communicated through IBC packets themselves.

Because the IBC packets cannot be tampered with, we may rely on trusted or untrusted entities to act as relays. The most a blockchain can do in the way of initiating communication is to include, within the chain itself, a form of signal of the intention to communicate some data. This can either be held as part of the state or within an exclusive "sub-state" for logging purposes (e.g., Ethereum's logging trie). An IBC packet can then be formed and passed to the other chain, outside of the consensus mechanism, by any entity observing the chain.

[4] When the validator set doesn't change, the list of signatures is enough, but since the block header only has a fingerprint of the validator set, it is necessary to establish a mechanism to track it.

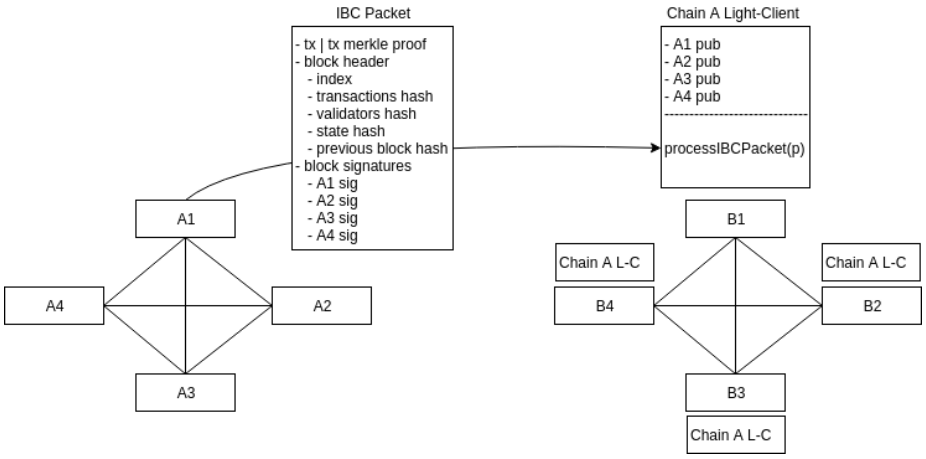

Figure 5: IBC

Network

MONET nodes form an overlay network on top of the Internet, and communicate over TCP/IP. Point to point communication between mobile devices, however, is not easy to implement with the current network infrastructure. The three challenges for mobile p2p applications are:

1. **Peer-discovery**: Supporting the discovery of local peers in a network where nodes may come and go.
2. **Mobility**: Maintaining a mapping between identities and public IP addresses as peers roam around and change IP realms.
3. **NAT traversal**: Working effectively even as nodes are located between multiple layers of NAT.[5]

A peer-discovery service enables nodes to learn which other peers can offer a required service, and how these peers are reachable. Different solutions are merely different approaches to providing a mapping between the identifiers and IP addresses of devices, along with information about the services they are offering [6]. Ultimately, some entities are responsible for managing this mapping in a file or a database, potentially in a distributed manner. Modern p2p applications use Distributed Hash Tables (DHT), such as Kademlia [7], to coordinate lookups on the network with no central instance to store the mapping. However, this is not applicable to our network without introducing special types of nodes because of the transient nature of mobile *ad hoc* blockchains.

On the Internet, end-user devices are often operating behind several layers of NAT. Hosts do not have globally unique IP addresses but share one with multiple other hosts. Devices behind a NAT cannot be reached without participation

[5] We generally refer to Network Address Port Translation (NAPT) as NAT in this document.

from the NAT for at least two reasons: the NAT must agree to forward incoming traffic, and it must redirect packets to the destination host according to a *port mapping*. Hence, to be reachable from the outside, a host must learn the public IP of its top-level NAT and configure all the NATs in between with the appropriate *port mappings* and firewall settings. There are two families of solutions for NAT traversal: communicating directly with the NAT to control port mappings (UPnP, PCP) [8], or *hole-punching* techniques which use third party services, residing outside the NATs, to echo or relay messages and create implicit mappings in the NATs along the way (ICE, STUN, TURN) [9].

Because the elementary components of MONET - nodes - are ephemeral by design, they cannot be relied upon to provide peer-discovery and NAT traversal services. In the spirit of building a truly decentralized and fault tolerant system, MONET is agnostic to how these services are implemented. It is left to application developers to define how they facilitate peer-discovery and NAT traversal for their users. However, as an optional utility for the first MONET applications, we will setup a master blockchain, providing a secure *Smart Contract* platform for long-lived data and programs. Eventually, each sector of the network could be supported by regional autonomous hubs, potentially tied to one another.

Hub

The MONET Hub is an *always on* master blockchain that provides infrastructure services for the first applications on the network. It is composed of multiple independent nodes operated by a group of validators engaged in a Proof of Stake consensus mechanism. Each node runs an instance of the Ethereum Virtual Machine (EVM) coupled with Babble, where *Smart Contracts* can be deployed to provide services such as peer-discovery, ID directories, financial ledgers, light-clients for other blockchains, and much more. Each validator also operates STUN and TURN servers to facilitate NAT traversal.

The Hub is implemented on a blockchain, rather than privately operated servers, because this affirms MONET's decentralised nature; it is a distributed and fault-tolerant network with no single point of failure. Anyone can become a validator on the MONET hub, provided they participate in the consensus algorithm and the *Smart Contract* layer. Moreover, anyone is free to create their own hub or deploy any other solution to support their applications.

Each validator runs an instance of the Babble consensus engine, coupled with EVM-Babble; a wrapper around the Ethereum Virtual Machine specifically designed to plug into Babble.[6] The EVM can accommodate arbitrary stateful logic and has a built-in cryptocurrency, which we label *Tenom* in the context of MONET to differentiate from Ether.[7] The workflow is well known to blockchain enthusiasts: one can create and call *Smart Contracts* through Ethereum-like

[6] https://github.com/mosaicnetworks/evm-babble

[7] Ether is the built-in cryptocurrency of the Ethereum public blockchain.

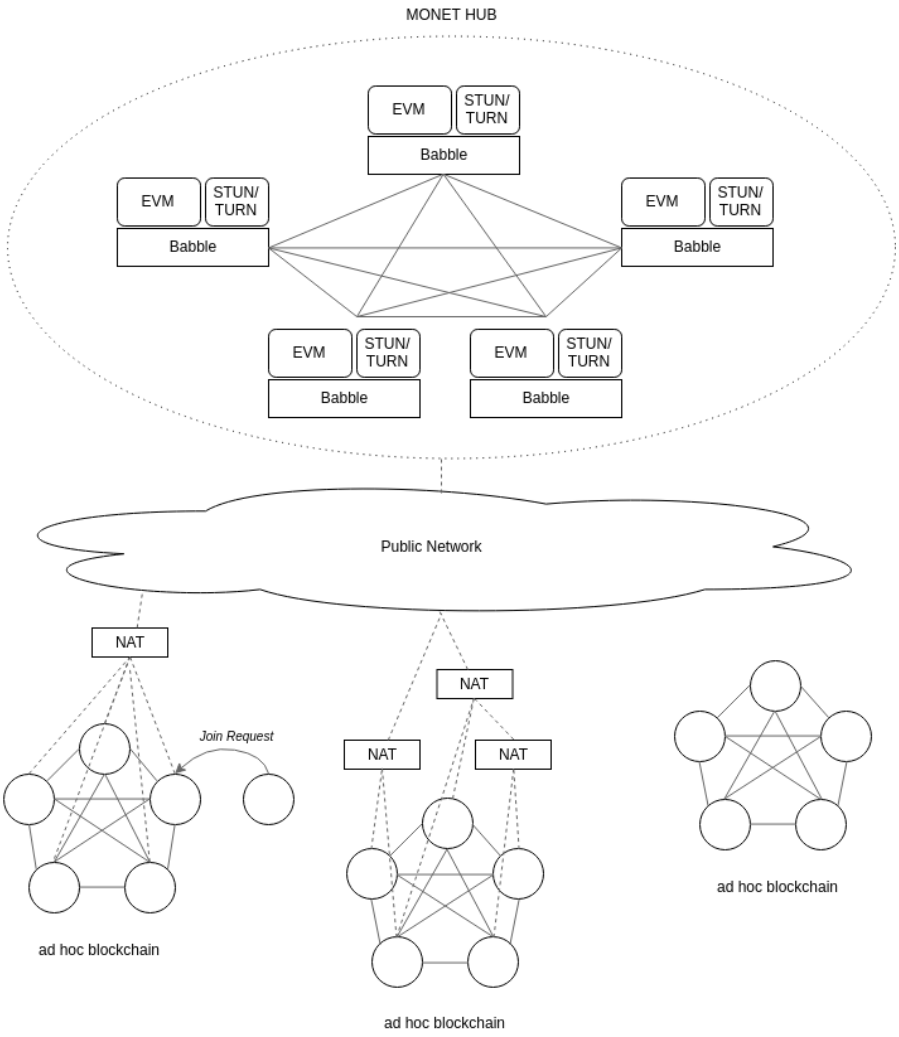

Figure 6: MONET HUB

transactions. Each operation has a cost and validators get rewarded for running the underlying computation. *Smart Contracts* can also charge a *Tenom* fee for the services they provide.

We will start by implementing free *Smart Contracts* for what we believe are important utilities to support the network:

- **ID directory**: a global system for identification whereby users may dynamically register relevant personal information to be used across different applications.
- **Peer-Discovery**: a generic registry mapping public keys or identities to public network addresses, location and application.
- **Light-Clients**: a template for registering Babble light-clients for any *ad-hoc* blockchain, serving as a a base for cross-chain communication.

Each validator on the Hub will be encouraged to operate STUN and TURN servers to support the underlying *ad hoc* blockchains. STUN and TURN servers enable *hole-punching* techniques for NAT traversal. On cellular networks and some public Wi-Fi hotspots, users are prohibited from explicitly creating port-mappings in the NATs that connect them to the Internet. It is necessary to resort to protocols like STUN [10] and TURN [11], relying on servers on the public network, to echo or relay packets from end-user devices.

Unlike mobile *ad hoc* blockchains, the hub is intended to be a stable utility providing pivotal services for other participants in the network. To incentivise validators in supporting the blockchain, the consensus algorithm will be augmented with a Proof of Stake (PoS) mechanism relying on *Tenom* - the native token of the Hub. Hence, the token is also a license to participate in the consensus algorithm or to delegate decision power to other validators. As a reward for contributing to the functioning of the network, validators earn a proportional share of the transaction fees processed by the Hub.

Security

Babble consensus supports up to one third of faulty nodes, including malicious behavior. This is actually a theoretical limit for Byzantine agreement protocols in general [12]. Above that limit, dishonest nodes could collude to subvert a blockchain by creating network partitions or manipulating the order of messages. Selective membership protocols, bolted on top of the consensus layer, mitigate this attack vector by drastically enforcing strong conditions for joining a group.

Securing mobile *ad-hoc* blockchains is more challenging because, being relatively small by definition, they are not formed of many nodes, and are more vulnerable of being controlled by a colluding group of malicious actors. The strongest line of defense is to implement rigorous membership criteria to approve *Join Requests* and to carefully select the initial set of participants. Hence the motivation to create a global public ID registry, where identities (MONET addresses) are

mapped to real-life data, such a user rating, name, endorsements, or any voluntarily submitted relevant information. Users keep complete control over their ID and selectively expose different subsets to different applications. Programs could then query the registry to approve or refuse *Join Requests* based on the satisfaction of certain queries against the requester's ID.

On the Hub, Babble is augmented with a Proof-of-Stake algorithm which incentivizes validators to behave correctly. Babble voting power is denominated in *Tenom* and validators are forced to "bond" their currency holdings in a security deposit that can be destroyed if they are found to misbehave in the consensus protocol. This adds an economic element to the security of the protocol, and quantifies the cost of violating the assumption that less than one-third of voting power is Byzantine.

Alternative Networking: Mobile ad hoc Networks

Mobile devices are also equipped with hardware that would allow establishing direct point-to-point links between nodes. In theory, nothing prevents mobile devices from establishing direct radio links, and there are a few available technologies that enable it (eg., Wi-Fi-Direct, Bluetooth or LTE-Direct). However, this is limited by several factors: range and power consumption, regulated or congested parts of the spectrum, operating system support.

Wi-Fi and Bluetooth can be used in *ad hoc* mode to establish infrastructureless p2p communications. Their range is realistically in the order of 30 to 50 meters and degrades quickly in the presence of physical obstacles. Android and iOS have built-in support for Wi-Fi-Direct, a complementary technology to traditional Wi-Fi ad hoc, where scalability and stability are improved by introducing the role of Group Owner, which acts as an access point. The group owner provides the communication infrastructure for all the other devices within its range. In the context of MONET blockchains, however, such a scheme would undermine the BFT properties of the consensus algorithm, which is designed to work without a leader or central point of failure.

Theoretically, one could choose any radio frequency and modulation scheme to communicate directly with other devices. But, to prevent interference between different users, the generation and transmission of radio waves is strictly regulated by national laws, coordinated by an international body, the International Telecommunication Union (ITU). Furthermore, it is unclear whether smartphones have full capabilities for Software Defined Radio (SDR), which would enable them to tune to arbitrary radio frequencies and process any modulation scheme.

LTE-Direct is a promising development in device-to-device (D2D) communication. It operates in a licensed spectrum, allowing it to provide high communication performance and reliability. Two scenarios are provided to support D2D communication: in-coverage, and out-of-coverage/ partial-coverage. The in-coverage

scenario is activated when both devices can be reached by one or more LTE cell towers, which have the role of timing, radio resource allocation (to LTE Direct), as well as user authentication. The out-of-coverage (and partial-coverage) scenario supports non-mediated D2D communication, meaning that the LTE cell-tower does not need to be available. However, it is only permitted for public safety applications. LTE-Direct is highly scalable and can support some 1000 devices in a 500 meter range, but D2D communication for commercial applications has not been standardised in 3GPP yet. For now, only communication for public safety applications has been standardised.

Use Cases

MONET introduces a new paradigm for building mobile applications, whereby individuals are directly connected to each other without fully depending on centralised servers. Even current mainstream applications could benefit from switching to this model. For example, MONET could provide the backbone for a fully decentralised Uber. Users and drivers in close vicinity would form *ad hoc* blockchains and coordinate themselves without relying on an expensive service provider to run the orchestration. Shared state, like a user rating system, could be implemented on the Hub's master blockchain, supported by a trustless and secure consensus algorithm.

Every MONET blockchain records an immutable and irrefutable log of events, which is highly relevant to financial ledgers; *ad hoc* blockchains can be used to represent movements of money. Even if they are transitory networks, blockchains record a provable history of the transactions that took place within them. Inter-Blockchain Communication allows blockchains to transact outside of their closed circles, by proving on one chain that a transaction happened on another chain. Hence, MONET also allows creating sharded cryptocurrencies where the shards are mobile. Groups of people can withdraw from a main ledger into a temporary *ad hoc* blockchain, transact among themselves in isolation, and later come back to the main ledger.

MONET could also be used for more trivial matters such as gaming. Today, multi- player games are usually implemented with a central server to coordinate participants; alternatively, when they adopt a p2p design to cut down on the costs of maintaining such servers, game developers are forced to implement their own solutions from scratch, which is a relatively important investment. MONET provides a free framework that game developers can plug into and access a network of users already equipped for p2p coordination, in the form of mobile *ad hoc* blockchains.

Conclusion

We presented the concept of mobile *ad hoc* blockchains which enables groups of people, in relatively close range, to engage in a common activity without resorting to an extended set of orchestrators. This makes for a truly decentralised network topology, intuitively appropriate for the peer-to-peer economy. The challenge is to provide a layer of trust, supported by hardware and software, within dynamically formed groups, where individuals do not necessarily know each other, and to provide mechanisms for mobile devices to discover one another and establish point-to-point communications, where current infrastructure does not necessarily allow for it. Our solution relies on a free and open-source software module, Babble, which provides BFT consensus suitable for mobile deployments. Peer-discovery and NAT traversal are left to the users, but will initially be supported by an optional utility blockchain, the MONET Hub, which will itself be operated by a number of independent validators. A new generation of mobile applications will be able to leverage this new infrastructure, which adapts organically to the urban nature of the p2p economy, and addresses the disadvantages of the current client-server model.

Aknowledgements

Many thanks to Duarte Aragão, Ivan Dikov, Yussuf Dirie, Daniel Grimes, Abdi Hersi, Kevin Jones, Michael Petidis, Giacomo Puri Purini, and Stanislav Stoyanov for participating in the many exciting conversations that led to these ideas. Concepts relating to blockchain networks were heavily inspired by the fantastic work done by the Tendermint team and the Cosmos foundation.

References

[1] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982.

[2] D. Senie, "Network Address Translator (NAT)-Friendly Application Design Guidelines." RFC 3235; RFC Editor, Jan-2002.

[3] J. Y. Halpern, Y. Moses, and O. Waarts, "A Characterization of Eventual Byzantine Agreement," *SIAM Journal on Computing*, vol. 31, no. 3, pp. 838–865,

2001.

[4] Y. Moses and M. R. Tuttle, “Programming Simultaneous Actions Using Common Knowledge,” *Algorithmica*, vol. 3, nos. 1-4, pp. 121–169, Nov. 1988.

[5] J. Kwon and E. Buchman, “Cosmos, a Network of Distributed Ledgers.”.

[6] R. Wakikawa, Z. Zhu, and L. Zhang, “A Survey of Mobility Support in the Internet.” RFC 6301; RFC Editor, Jul-2011.

[7] P. Maymounkov and D. Mazières, “Kademlia: A peer-to-peer information system based on the xOR metric,” in *Revised papers from the first international workshop on peer-to-peer systems*, 2002, pp. 53–65.

[8] D. Wing, S. Cheshire, M. Boucadair, R. Penno, and P. Selkirk, “Port Control Protocol (PCP).” RFC 6887; RFC Editor, Apr-2013.

[9] J. Rosenberg, “Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols.” RFC 5245; RFC Editor, Apr-2010.

[10] P. Matthews, J. Rosenberg, D. Wing, and R. Mahy, “Session Traversal Utilities for NAT (STUN).” RFC 5389; RFC Editor, Oct-2008.

[11] P. Matthews, J. Rosenberg, and R. Mahy, “Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN).” RFC 5766; RFC Editor, Apr-2010.

[12] M. Pease, R. Shostak, and L. Lamport, “Reaching Agreement in the Presence of Faults,” *J. ACM*, vol. 27, no. 2, pp. 228–234, Apr. 1980.