Vegvisir: A Partition-Tolerant Blockchain for the Internet-of-Things

Kolbeinn Karlsson, Weitao Jiang, Stephen Wicker
School of Electrical and Computer Engineering
Cornell University, Ithaca, New York

Edwin Ma, Robbert van Renesse, Hakim Weatherspoon
Department of Computer Science
Cornell University, Ithaca, New York

***Abstract*—While the intersection of blockchains and the Internet of Things (IoT) have received considerable research interest lately, Nakamoto-style blockchains possess a number of qualities that make them poorly suited for many IoT scenarios. Specifically, they require high network connectivity and are power-intensive. This is a drawback in IoT environments where battery-constrained nodes form an unreliable ad hoc network such as in digital agriculture. In this paper we present Vegvisir, a partition-tolerant blockchain for use in power-constrained IoT environments with limited network connectivity. It is a permissioned, directed acyclic graph (DAG)-structured blockchain that can be used to create a shared, tamperproof data repository that keeps track of data provenance. We discuss the use cases, architecture, and challenges of such a blockchain.**

***Index Terms*—blockchain, CRDT, tamper-proof logging, ad hoc networks, IoT, gossip protocols**

I. Introduction

Blockchains have emerged as an exciting new paradigm for distributed systems. From Bitcoin's cryptocurrency [1] to Ethereum's Turing-complete smart contracts [2], blockchains are being explored as a means to solve problems across a wide range of industries, including banking, energy, transportation and accounting. Blockchains could have important uses in IoT and Edge Cloud environments as well [3]. Such systems are often used for applications that require tamperproof logging of events for accountability. They are deployed under varied administrative domains, and so the decentralized ownership of blockchains make them a good match. However, current blockchain designs require high network connectivity and are power-intensive, both of which can detract from their utility in such environments.

A blockchain is simply a tamperproof log of transactions. Blockchain implementations use a distributed trust model, removing the need for centralized control and single-point-of-failure designs. As long as a large enough fraction of participants execute the protocol (usually half or one-third), its security properties will be enforced. This provides a system that is both strongly consistent and highly available. It is not, however, tolerant of network partitions. Partitions cause branches (aka forks) in the blockchain and branches must be resolved, meaning only one branch gets to stay a part of the blockchain while all others are discarded. With network partitions, such branches may stay around for a long time and lead to undesirable behaviors, even if branches are eventually resolved. For example, people who have conducted business with Bitcoin may find that the bitcoins they were paid are now back in the hands of the original owner or have been spent otherwise. Additionally, most current blockchain designs are very energy-intensive, requiring vast amounts of computation solving cryptopuzzles. Bitcoin alone is estimated to use tens of terawatt hours per year, enough to power a mid-sized country [4], [5].

These two characteristics, the need for high network reliability and high power consumption, make Bitcoin and most other existing blockchain designs unsuitable for deployment in ad hoc IoT networks or edge cloud systems. We present *Vegvisir*, a blockchain specifically designed for the low-connectivity, low-power IoT setting. It tolerates network partitions well and uses a low-power consensus mechanism. Instead of resolving branches, it permits them, resulting in a Directed Acyclic Graph (DAG) structure of the blockchain rather than a linear one. The cost of this partition tolerance is that the types of applications that can be implemented with the blockchain are limited to ones that only require a partial ordering of logged events. To this end, Vegvisir supports applications based on Conflict-free Replicated Data Types (CRDT) [6].

To motivate the need for Vegvisir, we present two use cases in Section II. We go on to survey related literature in Section III and develop the architecture of the blockchain in Section IV. Section V describes our initial implementation of the blockchain and Section VI discusses the implications and challenges of this line of research.

II. Motivation

The blockchain became popular in the financial technology sector. However, it is generally believed that the blockchain may have far-reaching consequences for many other industries. But many industries require interaction with the physical environment, which—unlike the fintech industry—is not always well-connected. Below we give two examples.

A. Use-based Privacy During Disaster Response

The 2017 Atlantic hurricane season was one of the worst on record. Three major hurricanes devastated the Caribbean, Florida, and Texas. Hundreds of people lost their lives and the property damage is estimated to be over $300 billion [7]. The loss of lives, limbs, and property had undoubtedly been greater if not for the valiant efforts of thousands of emergency first responders. If first responders could leverage more information and communication technology to aid and coordinate their efforts, further lives could potentially be saved. In an ideal world, first responders have a strong communication network and a robust cloud infrastructure that enables information to flow to the right places at the right time and that eases coordination of rescue efforts. Natural disasters, however, can render communication infrastructure such as cell towers and Land Mobile Radio System (LMRS) repeaters inoperable. First responders must in those cases deploy their own communication infrastructure as well as take advantage of every possible means of communicating, forming heterogeneous ad hoc mobile networks to make up for lack of connectivity. Existing communication and cloud infrastructure is not built to operate in such environments. We need a new infrastructure design to enable first responder applications.

One of the problems medical personnel face both in and outside of emergency situations is the need for accessing electronic health records promptly while safeguarding their security and privacy. We propose that blockchains can be used to implement a use-based privacy solution that gives emergency first responders ready access to sensitive patient health records but enforces strict accountability. Use-based privacy is an approach to privacy that focuses on uses (and abuses) of sensitive records, rather than access [8]. In recent years, use-based privacy has been proposed as a framework under which to design privacy policies [9], [10]. Patients generally will not object to a physician or paramedic accessing their medical records in order to help save their lives (a valid use) but they would object to the same physician accessing their records without a medical reason. As such, during emergencies, paramedics and physicians could have all their access requests to sensitive records granted under the condition that the request has been recorded in a tamperproof log. Once the state of emergency is over, the log is reviewed. If frivolous access has occurred, such as a medical worker accessing an ex-spouse's or a celebrity's health record, the worker could be sanctioned, providing incentive to only access health records when necessary.

Our approach with Vegvisir presents a good avenue to implement a tamperproof log in such an environment. It consists of an unreliable network between many low-power IoT devices (first responder smartphones), some of whom cannot be fully trusted. Our solution can ensure that no health record is accessed without an explicit request for access being persistently stored on the blockchain. It does not require proof-of-work and is therefore easy on the batteries, and its opportunistic gossip-style protocol for spreading blocks is well-suited for a mobile ad hoc network.

B. Digital Agriculture

The blockchain is a promising technology that could bring transparency and accountability to the food supply chain. There are many participants in the food supply chain, including farmers, brokers, packers, traders, distributors, food processors, retailers, regulators, and ultimately, consumers. Today, record keeping is mostly on paper and various centralized databases, while many negotiations are purely verbal. This is prone to mistakes and simplifies fraud. Farmers do not know how much profit is made on the food they supply, and consumers cannot easily track where their food comes from. Blockchains could potentially create a shared and tamperproof data repository in which all information is readily shared, available, and auditable.

Blockchains could make it straightforward for a consumer to check the source of a food product. In the case of a meat product, information of interest might include the animal's date of birth, place of origin, vaccinations, and use of antibiotics. Food safety is a related application. For example, Walmart (which sells 20% of all food in the U.S.), IBM, and Tsinghua University are looking into using the Hyperledger blockchain [11] for food supply chain traceability and authenticity. Today, if a pathogen is found in a food product, it takes several days to weeks to trace it back to the supplier. Using a blockchain, Walmart hopes to reduce this to seconds, potentially saving lives. From the farmer's perspective, blockchains could make it easier to find consumers for their products, potentially reducing waste from either unused land (if too little is produced) or from overproduction. Farmers could check to see what retailers sell their products for, so they may negotiate a better price for their produce.

Ideally, recordkeeping with the blockchain would reach all the way to the farm and to the distribution channels. Tagging of animals, pallets, shipping containers, and so on with RFIDs or related technology enable tracking. But farms and distribution centers have intermittent if any connection to the Internet, and must rely on a system consisting of small fixed and mobile IoT devices for sensing. While such systems can be used to create a sensor network, there is no integrated blockchain that can securely store the history of sensed data. Again, Vegvisir allows deploying a low-energy tamperproof log in this environment.

III. Related Works

Blockchains were first introduced in 2008 as part of the then novel Bitcoin cryptocurrency system [1]. Since then, the blockchain field has seen explosive growth with many variants and use cases proposed. One of the more notable variants is Ethereum, which replaces the basic scripting language implemented in Bitcoin with a Turing-complete one, paving the way for so-called smart contracts [2]. Both of these blockchains have a linear structure and rely on a proof-of-work (PoW) consensus mechanism which requires solving a computationally expensive cryptopuzzle, making them poorly suited to our use cases.

Some variants of the blockchain use a DAG structure instead of a linear chain. The GHOST protocol is a modification to the Bitcoin blockchain to a DAG structure to improve security [12]. The point of this modification is to enable a more robust method of selecting which fork to keep and which to discard. By keeping track of all forks, a node can choose a fork based on the heaviest-subtree-wins rule (the subtree with the largest

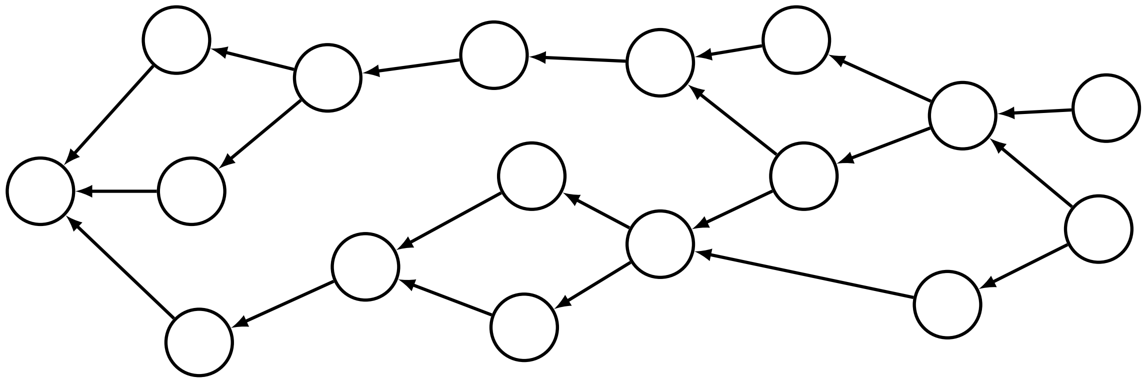

Fig. 1. Sample DAG. Branches are reined in by making every known leaf a predecessor of your new block. As our applications are based on CRDTs, there is no need to determine a total order between the blocks.

number of blocks) as opposed to the longest-chain-wins rule, and thus eliminating certain forms of attacks.

The Byteball blockchain platform proposes a new type of cryptocurrency with a DAG structure [13]. Byteball eliminates the distinction between blocks and transactions. Each 'block' is a single transaction and can have multiple parents. Double spending is prevented by determining a total order on the DAG through the behavior of a set of privileged users called 'witnesses'. The total order is used to determine which transactions to keep as which to declare invalid when double spending occurs.

Iota is perhaps the best-known implementation of a blockchain with a DAG structure [14]. Iota is a transaction fee-less cryptocurrency where double spends are resolved by a consensus algorithm that determines which transaction to keep based on the number of descendant transactions. In both Iota and Byteball, the DAG structure is designed to compensate for the slow rate of block propagation relative to block generation rather than to tolerate network partitions.

In 2011, Mark Shapiro formalized the types of data structures that can be replicated across multiple hosts and updated concurrently and independently, while still providing strong eventual consistency [6]. These data structures, known as Conflict-free Replicated Data Types (CRDT), have been shown to include versions of registers, counters, sets, graphs, and maps [15]. These basic data types can be combined and composed to create more sophisticated data structures such as key-value stores [16] and JSON documents [17]. Applications include collaborative editing [18] and distributed databases [19].

Our blockchain uses a gossip-style protocol. Gossip protocols originated in the field of distributed databases [20], but has seen a resurgence coinciding with the proliferation of cloud computing [21], [22], [23] and more recently in blockchain protocols such as Bitcoin [24]. While a variant of the gossip protocol has been shown to work well in unreliable networks [25], most gossip protocols assume full network connectivity and can therefore not be directly applied in IoT environments with low connectivity.

IV. Architecture

A. Design Requirements

The blockchain is essentially a log of records that are generally called *transactions* in the blockchain literature. The blockchain is maintained by a group of *users*. We would like the Vegvisir blockchain to have the following informal properties:

- *Tamperproof*: Once a transaction has been stored on the blockchain, it cannot be removed or modified, and neither can transactions that precede it in the blockchain.
- *Provenance*: If a user can read a transaction on the blockchain, then the user can read all transactions that precede it on the blockchain.
- *Authenticity*: Every transaction on the blockchain is identified by the user that created the transaction and placed it on the blockchain.
- *Transitivity*: If one user learns of a transaction on the blockchain, then eventually all users do.
- *Access Control*: There should exist control over which users are allowed to append which types of transactions to the blockchain.
- *Partition Tolerance*: The blockchain is available even when not all users can physically communicate with one another for some unspecified length of time.
- *Storage Efficiency*: IoT devices may have limited storage. They do not have to store all of the blockchain—some of it may be stored elsewhere.

These requirements, and in particular partition-tolerance, stipulate that the blockchain maintain a partial order of transactions. The transactions within a block are totally ordered, but a block may have multiple "parents." Nonetheless, Vegvisir will make an effort to reduce branching as much as possible. In particular, when a user appends a new transaction, all transactions known to the user must become ancestors of the transaction. Thus the Vegvisir blockchain essentially maintains the causal history of all transactions.

B. Adversary Model

We assume that among the k closest network neighbors of a user (which may be malicious), at least one user correctly follows the Vegvisir protocol. The parameter k can be set according to need. Malicious peers want to change or remove blocks from the blockchain. Adversaries cannot forge signatures from other users, but they can remove blocks from their local version of the blockchain and they can choose not to propagate new blocks they receive.

C. Design Overview

Like most blockchains, Vegvisir consists of a series of interlinked blocks containing a block header and one or more transactions. Unlike most blockchains, each block can point to multiple other blocks as its predecessors. Thus blocks in Vegvisir form a DAG rather than a linear chain (see figure 1). The DAG has a single block, the genesis block, as the ancestor of all blocks.

The DAG encodes a partial ordering on transactions. The DAG nature of the chain is what makes it partition-tolerant but limits the types of applications that can be implemented with the blockchain compared to a blockchain that provides a total ordering. When interpreting a DAG of transactions, we require that transactions that are not ordered with respect to one another in some sense commute. For this reason, we limit usage of Vegvisir to CRDT-based applications. The commutativity of CRDT operations removes the need for imposing a total order on transactions. Using CRDTs, any total ordering consistent with the partial ordering will produce the same interpretation on the state produced by the transactions. Below we will assume CRDT-based applications.

Vegvisir is a so-called *permissioned* blockchain and has a membership (for example, emergency first responders). It has an *owner* who generates and signs the genesis block. The genesis block contains a self-signed certificate of the owner, who will act as a certificate authority (CA) on the blockchain. Each authorized user must have a certificate signed by the CA placed on the blockchain. Certificates specify the role of each user, and access control is determined based on those roles.

D. Blocks and Transactions

Each block is composed of block header, zero or more transactions, and a digital signature. The block header contains the user ID of the block creator, a timestamp, if possible a physical location, and a list of hashes of its parent blocks (see figure 2).

Transactions specify operations on CRDTs. For example, in our emergency first responder use case, a user might want to add an access request for a health record on the blockchain. Vegvisir could have an add-only set (which is a CRDT) of health record access requests. Call this CRDT $\mathcal{H}$. Then the user would add a transaction r, containing the request, to $\mathcal{H}$.

A transaction must specify the name of the CRDT, the type of operation to perform, and any arguments that operation requires. Transactions do not carry a digital signature—a transaction is implicitly signed by the block that contains the transaction. In Vegvisir, the creator of a block is the originator of all transactions in the block, so the block signature also establishes the authenticity and integrity of the transactions.

The set of valid users can also be thought of as a CRDT. Specifically, it is a *2P set*, which is a set representation composed of an *add set* A and a *remove set* R. When adding an element, it is added to A and when removing an element, it is added to R. The elements that are said to exist in the 2P set are $A \backslash R$. If the elements of A and R are public key certificates, then certificates can be added to A, while revocations amount to adding the same certificate to R. Every Vegvisir blockchain must have a 2P set of users, $\mathcal{U}$, and so it is implicitly created when a new blockchain is formed.

Other CRDTs, such as the add-only set $\mathcal{H}$ mentioned above, can be created as needed. Each new CRDT must have a unique name. To avoid collisions, names can be a randomly generated string of length n, where n is high enough that the probability of a naming collision is negligible. A collection of CRDTs is a CRDT itself. We will refer to the set of user-created CRDTs as Ω from now on.

E. Separation of Concerns

The software of each Vegvisir user has two main components. The first component is the blockchain. It maintains the local copy of the DAG, checks the validity of the blocks, and passes the transactions to the other component, the CRDT state machine (CSM). The only CRDT the blockchain component concerns itself with is $\mathcal{U}$. The following checks are performed to assess if a new block is valid:

- The user must be member of blockchain (specified by $\mathcal{U}$);
- Parent blocks must be in the blockchain already;
- Timestamp must be higher than the maximum of the timestamps in the parent blocks but lower than the current time at the user;
- Signature must be valid and match user ID.

The CSM in turn checks the validity of the transactions themselves and makes the appropriate updates to Ω and $\mathcal{U}$ once it has verified that the transaction satisfies the following:

- The CRDT must exist (i.e., it must be U, Ω, or an element of Ω);
- The specified operation must be valid for the CRDT;
- The argument to the operation must pass type checks (e.g. we cannot add an integer to a set of strings);
- The user must have permission to perform the operation.

When creating a CRDT, one must specify which roles can perform which actions. For example in the case of $\mathcal{H}$, it could be specified that only users with the role 'medic' can perform the add operation. Users' roles are specified in their public key certificates.

F. Public Key Certificates

A public key certificate contains the user ID, the public key of the user, the user's role, and a digital signature from the CA (the blockchain owner). When performing block validation, the user ID in the block header must match a user ID of one of the certificates in $\mathcal{U}$. Elements in the remove set of $\mathcal{U}$ act as certificate revocations. Similarly, when performing transaction validation, the CRDT indicates whether the user's role is permitted to perform the specified operation.

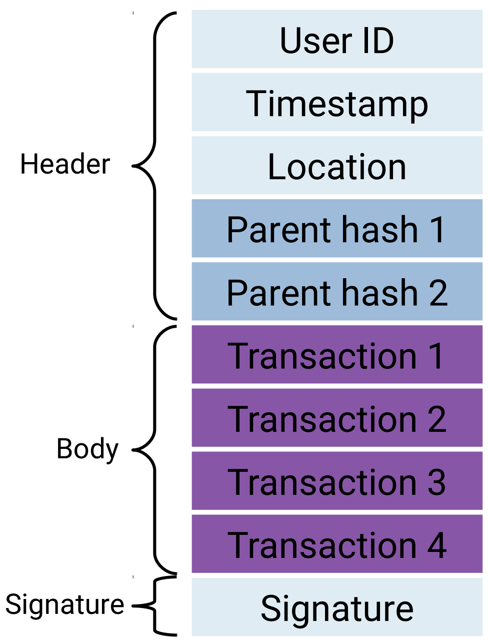

Fig. 2. A layout of a block. The header contains a user ID, a timestamp, and if possible, a location. It also contains a variable number of parent hashes. The body consists of zero or more 'transactions'. Every block is signed by its creator, which is also the creator of all transactions in the block.

G. Opportunistic Reconciliation

Blocks are spread throughout the network via a protocol that resembles gossip. Periodically, a node picks a physical neighbor at random (if it has any). The initiator then asks the other node, the selected neighbor, for its *frontier set*. The frontier set is the set of blocks on the DAG that have no successors (sources, given that blocks in the DAG point to their parent blocks). If the neighbor's frontier set is identical to the initiator's, then their blockchains are identical too and the process stops. If, however, the frontier set contains blocks unknown to the initiator, it adds the frontier set to its own replica of the DAG. That operation will fail if the DAG does not contain all parents of all blocks in the frontier set. In that case, the initiator requests to see the level 2 frontier set, which is the frontier set plus the set of all parent nodes. In general, a level N frontier set is defined as the union of the level $N-1$ frontier set and the parents of all blocks in the $N-1$ frontier set. The base case of this recursive definition is the level 1 frontier set, which is the frontier set described above (see figure 3). The initiator continues to ask for higher levels of the frontier set until it is able to bridge the gap between its blockchain and the neighbor's. That must happen eventually assuming they have the same genesis block (which is the unique sink of the DAG and identifies the Vegvisir blockchain).

H. Persistence through Proof-of-Witness

Persistence and immutability are primary advantages of blockchains. Since malicious nodes may attempt to remove

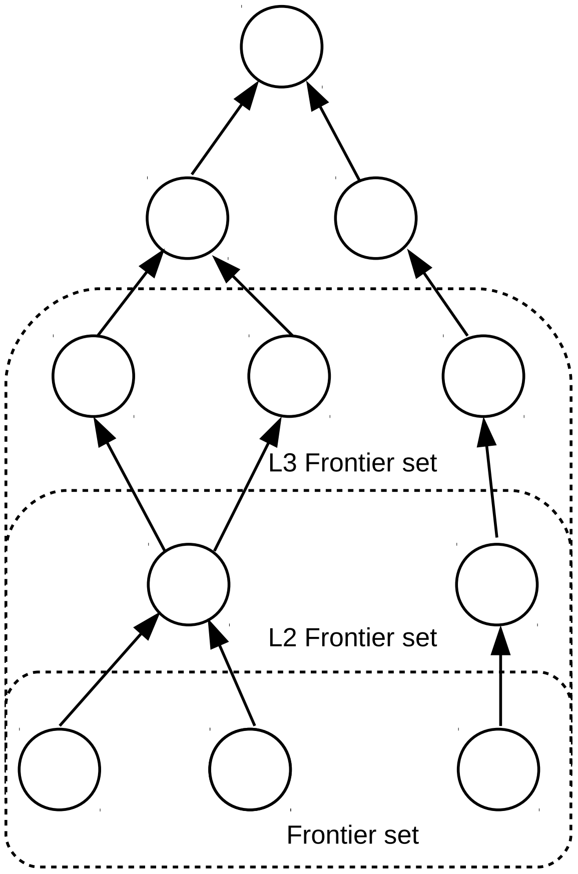

Fig. 3. The (level 1) frontier set is the set of blocks without successors. The level 2 frontier set contains the level 1 frontier set plus their parents. More generally, the level n frontier set is the union of the level $n-1$ frontier set and the parents of its blocks.

Algorithm 1 DAG Reconciliation Pseudocode

```
1: procedure ReconciliateDAGs(S)                    ▷ $S$: local DAG
2:     $B \leftarrow$ getRandomNeighbor()
3:     if $B$ is not empty then
4:         $n \leftarrow 1$
5:         $S_{B,n} \leftarrow$ getNthFrontierSet($B, n$)
6:         if parents($S_{B,n}$) $\subseteq S$ then
7:             $S \leftarrow$ merge($S_{B,n}, S$)
8:         else
9:             $n \leftarrow n + 1$
10:            goto 5
11:        end if
12:    end if
13: end procedure
```

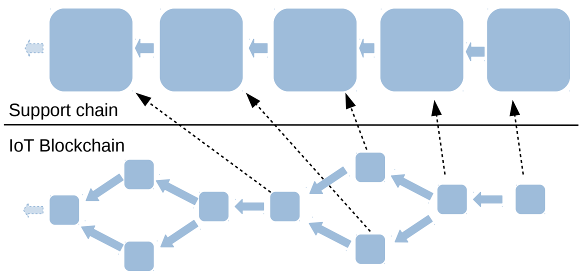

Fig. 4. The IoT blockchain has periodic access to a support blockchain.

newly added blocks, we cannot be confident that a block will persist once it is added by one user of the blockchain. In particular, an application may not be able to take action until it has some guarantee that a particular transaction and the transactions that causally precede it are persistent. To solve this problem, an application may require confirmation from users in some quorum that they have a copy of the block. The choice of quorum is up to the application. Because the Vegvisir blockchain is a DAG rather than a linear chain, there is no requirement that quorums overlap.

For example, if a user requests access for a health record by adding a transaction to $\mathcal{H}$, an application may require that k additional nearby users have stored the block containing the transaction before counting it as a persistent part of the blockchain. One way to obtain the desired effect is as follows: A user may indicate that it has stored a block by adding an ancestor block to the blockchain, signed by that user. Once a block has ancestor blocks signed by at least k different nearby users, the block may be considered persistent by the application. These blocks need not contain any transactions. Their sole purpose is to signal that a user has a copy of the ancestor blocks. We say that a block has a *proof-of-witness* once it has reached this condition.

I. Support Blockchain

IoT devices may be constrained by the amount of storage they have for the Vegvisir blockchain. We allow such devices to offload parts of their DAG to a more traditional blockhain that we call the *support blockchain* (see figure 4). This would be applicable to environments where the low-power, battery-constrained IoT devices that make up the partition-tolerant blockchain have occasional access to higher-powered servers. The higher-powered servers can function as superpeers, taking blocks from the partition-tolerant blockchain and placing them on the support blockchain, which operates between the superpeers as well as in the cloud. Once a block is placed on the support blockchain, the IoT device can drop the block. Typically, IoT devices would only do so when running low on storage, and would only offload their oldest blocks on the blockchain.

As superpeers get new blocks, they in turn add new blocks to the support blockchains. The body of a block on the the

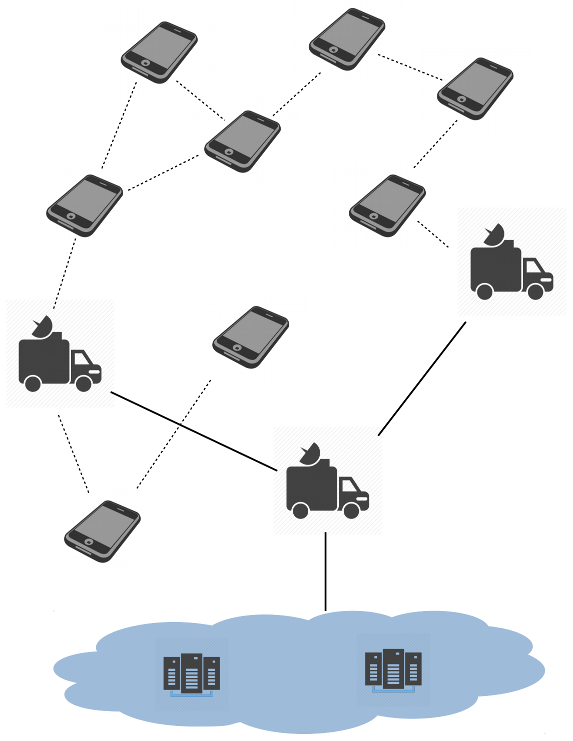

Fig. 5. The network can consist of both battery-constrained IoT devices (depicted as smartphones) as well as relatively high-powered deployable servers (depicted as trucks) that may be connected to the rest of the Internet. The high-powered nodes relay blocks to the support.

support blockchain is a Vegvisir block. Support blocks must be added in a way that preserves the topological order of the Vegvisir DAG.

V. Implementation

We have an Android implementation of Vegvisir under construction. The Android prototype is designed specifically for the emergency-first responder scenario. As such, it allows users to place requests for health records on the blockchain. It uses Bluetooth and Google Nearby (which uses a combination of Bluetooth and WiFi Direct) to communicate opportunistically with anyone in its neighborhood. Additionally, the implementation has to have a mechanism to deliver the health records to the requester once the request is stored securely on the blockchain.

The simplest way would be to have the user present a proof-of-witness that their request has been placed on the blockchain to a centralized database server. But a key assumption of Vegvisir is that devices operate in a environment with unreliable access to each other and the public cloud, so the device might not be able to connect to such a database for extended periods of time. An alternative could be to have each user carry an encrypted version of the database in secondary storage. The key would be kept in a trusted execution environment (TEE) and only through a certifiably correct program can a health record be decrypted and made available to the user and only once that program has determined that the request is on the blockchain and has obtained the proof-of-witness.

VI. Conclusion

Providing partition-tolerance is vital if blockchains are to be adapted to IoT environments. IoT devices operate with strict constraints on power and often limited network connectivity. Vegvisir extends partition tolerance to blockchains, although at the cost of limiting the classes of applications that can be implemented with CRDTs. While that prohibits applications that need a unique total ordering, like cryptocurrencies, it can nonetheless be used to implement a persistent, tamperproof distributed ledger suitable as a building block in many useful applications such as digital agriculture and support for emergency first responders during disaster response. There is nonetheless room for improvement in Vegvisir. The opportunistic reconciliation method, while considerably more efficient than exchanging entire DAGs, still incurs a considerable communication overhead. More efficient DAG reconciliation algorithms could make blocks propagate faster through the network while using less bandwidth. At this time, Vegvisir is still in the early stages of development. Its advantages and challenges will become more apparent when a prototype is built and more extensive simulations have been performed to evaluate it.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Tech. Rep., 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] G. Wood, "Ethereum: A secure decentralized generalized transaction ledger," Tech. Rep. EIP-150, Apr. 2017.

[3] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[4] K. J. O'Dwyer and D. Malone, "Bitcoin mining and its energy footprint," in *25th IET Irish Signals & Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014)*. IET, 2014.

[5] A. Beall, "Bitcoin mining uses more energy than Ecuador but theres a fix," *New Scientist*, Oct. 2017. [Online]. Available: https://www.newscientist.com/article/2151823-bitcoin-mining-uses-more-energy-than-ecuador-but-theres-a-fix/

[6] M. Shapiro, N. Preguia, C. Baquero, and M. Zawirski, "Conflict-Free Replicated Data Types," in *Stabilization, Safety, and Security of Distributed Systems*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Oct. 2011, pp. 386–400.

[7] M. P. a. M. G. CNN, "Hurricane season is finally ending," Nov. 2017. [Online]. Available: https://www.cnn.com/2017/11/30/weather/hurricane-season-2017-recap/index.html

[8] F. Cate, "Principles for protecting privacy," *Cato Journal*, vol. 22, no. 1, pp. 33–58, 2002.

[9] F. Cate, P. Cullen, and V. Mayer-Schonberger, *Data Protection Principles for the 21st Century*, ser. Books by Maurer Faculty, 2013, no. 23.

[10] E. Birrel and F. B. Schneider, "A reactive approach for use-based privacy," Nov. 2017.

[11] C. Cachin, "Architecture of the Hyperledger blockchain fabric," in *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*, 2016.

[12] Y. Sompolinsky and A. Zohar, "Secure High-Rate Transaction Processing in Bitcoin," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Jan. 2015, pp. 507–527.

[13] A. Churyumov, "Byteball: A Decentralized System for Storage and Transfer of Value," Tech. Rep., 2015. [Online]. Available: https://byteball.org/Byteball.pdf

[14] S. Popov, "The Tangle," Aug. 2017. [Online]. Available: https://www.docdroid.net/mWTNlgd/iota1-2.pdf

[15] M. Shapiro, N. Preguia, C. Baquero, and M. Zawirski, "A comprehensive study of Convergent and Commutative Replicated Data Types," Inria Centre Paris-Rocquencourt ; INRIA, report, Jan. 2011. [Online]. Available: https://hal.inria.fr/inria-00555588/document

[16] S. Sanfilippo, "Redis," 2009. [Online]. Available: https://redis.io/

[17] M. Kleppmann and A. R. Beresford, "A Conflict-Free Replicated JSON Datatype," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 10, pp. 2733–2746, Oct. 2017.

[18] X. Lv, F. He, W. Cai, and Y. Cheng, "A string-wise CRDT algorithm for smart and large-scale collaborative editing systems," *Advanced Engineering Informatics*, vol. 33, pp. 397–409, Aug. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474034616301811

[19] C. B, "Getting Started with Active-Active Geo-Distribution for Redis Applications with CRDTs (conflict-free replicated data types)," Oct. 2017. [Online]. Available: https://redislabs.com/blog/getting-started-active-active-geo-distribution-redis-applications-crdt-conflict-free-replicated-data-types/

[20] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic Algorithms for Replicated Database Maintenance," in *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, ser. PODC '87. New York, NY, USA: ACM, 1987, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/41840.41841

[21] J. Lim, K.-S. Chung, H. Lee, K. Yim, and H. Yu, "Byzantine-resilient dual gossip membership management in clouds," *Soft Computing*, pp. 1–12, Mar. 2017. [Online]. Available: https://link.springer.com/article/10.1007/s00500-017-2553-3

[22] L. Chuat, P. Szalachowski, A. Perrig, B. Laurie, and E. Messeri, "Efficient gossip protocols for verifying the consistency of Certificate logs," in *2015 IEEE Conference on Communications and Network Security (CNS)*, Sep. 2015, pp. 415–423.

[23] R. Chandra, V. Ramasubramanian, and K. Birman, "Anonymous Gossip: improving multicast reliability in mobile ad-hoc networks," in *Proceedings 21st International Conference on Distributed Computing Systems*, Apr. 2001, pp. 275–283.

[24] P. Koshy, D. Koshy, and P. McDaniel, "An Analysis of Anonymity in Bitcoin Using P2P Network Traffic," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Mar. 2014, pp. 469–485.

[25] D. B. Terry, M. M. Theimer, K. Petersen, A. J. Demers, M. J. Spreitzer, and C. H. Hauser, "Managing update conflicts in bayou, a weakly connected replicated storage system," in *Proceedings of the fifteenth ACM symposium on Operating systems principles (SOSP '95)*, vol. 29, no. 5. ACM, 1995.