A Survey of Distributed Trust Mechanisms Suitable for IoT Devices

Morné Pretorius*, Sthembile Mthethwa*
*Modelling and Digital Science (MDS)
Council for Scientific and Industrial Research (CSIR)
[1]mpretorius2@csir.co.za
[2]smthethwa@csir.co.za

***Abstract*—There are challenges and trade-offs associated with building secure decentralised systems that incorporate resource-constrained devices into the internet ecosystem, particularly the internet of things. Introducing distributed ledgers solves some of these problems while introducing new challenges. This paper identifies mechanisms that can be adopted to build distributed trust for smaller devices when attempting to balance the consensus, scalability and decentralisation trilemma. The building blocks of decentralised consensus networks are identified and problem areas with potential solutions discussed to provide proper requirements scope. Newer consensus protocols and cryptographic constructs are discussed that might provide secure, efficient and scalable consensus and avoid excessive computational-, bandwidth- and storage resources. The idea is to limit the conceptual design phase to a resource-constrained operational situation to avoid redesign when the internet of things becomes widely adopted.**

***Index Terms*—consensus, distributed, internet-of-things, cryptographic, risc-v**

I. Introduction and Background

With the Internet of Things (IoT) being a future reality [1], it would be ideal if technologies existed to ensure its safety in terms of an open, permissionless and optimally decentralised arrangement, where any device can join or leave the network without degrading performance or trusting in any central point of control. Distributing control of a system is important because in an IoT context, compromising a central data controller results in the unauthorised manipulation of the digital which is connected to the physical [1]. For example, leaking your baby's sleep patterns through a connected baby monitor is one thing but controlling their sleep pattern is a much different risk equation. Physical and geographical decentralisation of control implies that attackers need to do much more work to compromise a system.

Distributed and decentralised systems have problems with scaling as central points of control are usually required to maintain efficiency at the expense of fault-tolerance which introduces the well know Consistency, Availability and Partition tolerance (CAP) theorem or rather, rule of thumb [2, 3]. A derivative of CAP is used here to view the trade-offs related to distributed and decentralised systems design.

It is important to discern firstly between a classical distributed database and Distributed Ledger Technology (DLT). The key difference lies in the ability to distribute (uniformly or fairly) control and coordination of the system towards anyone willing to participate, whilst using incentive-punitive mechanisms such as cryptography to induce good- and reduce bad behaviour. In contrast, classical distributed databases usually operate in trusted environments and do not necessarily use cryptography to guard against malicious behaviour. This is the most important distinction to make when considering permissioned versus permissionless distributed systems and databases [4].

Achieving a ***safe*** IoT involves a multidisciplinary approach and so this research aims to identify historical and current definitions, trade-offs and challenges related to building these systems while being mindful of resource-constrained environments such as IoT. We aim to use more generic and classical, yet explicit, terminology due to the multidisciplinary nature of these systems when discussing trade-offs and present possible future solutions related to cryptographic primitives and hardware.

Thus, Section II identifies historical definitions that have been used to describe a correctly functioning distributed agreement system. Section III further identifies basic types of distributed agreement technologies that have emerged, including additional terminology that define similar ideas across these types. Section IV merges the previous terminology into more classical definitions from systems and information theory and discusses trade-offs related to the identified consensus types in existence today. Section V evaluates said distributed consensus options in the context of IoT, presents possible solutions at a conceptual level and is followed by lower-level cryptographic and hardware considerations in Section VI and VII respectively.

II. Distributed Systems History

Lamport proposed two properties that need to be present in distributed systems, namely safety and liveness [5]:

1) **Safety** - refers to nothing bad happening when the system executes, such as handling exceptions and not halting if they occur.
2) **Liveness** - refers to things that eventually must happen, such as requests from correct clients that are eventually processed and not getting stuck in an undecided state.

These properties are usually required by most systems to be practical, however, from safety and liveness followed more property definitions to ensure the existence of a correct asynchronous Byzantine Fault-Tolerant (aBFT) system and dates back to the '70s and '80s [6, 7]:

1) **Agreement** - All honest processes/nodes must agree on the same value/output v.
2) **Integrity/Validity** - Any agreed-upon v must originate *once* from honest nodes and thus faulty nodes' values must be filtered out or ignored.
3) **Termination** - Honest nodes must eventually agree and cannot remain undecided. This has to do with the system avoiding open-ended control loops and maintaining state-full transition by handling anomalies or event forks and remaining live.

Combining these requirements into a single sentence to define a correct/safe and live distributed system: Honest nodes are assumed to propose input(s) where all honest nodes have to agree eventually (*Termination*), on the same output (*Agreement*), that has to be equal to the proposed inputs (*Validity*).

Within the Byzantine fault model, an agreement was shown to be possible only if less than a third of nodes are faulty or Byzantine [8] where the behaviour of the Byzantine processes or nodes are restricted and filtered by incorporating message broadcast into the consensus protocol [7]. It was also shown through the Fischer-Lynch-Paterson (FLP) impossibility that agreement is impossible in an idealised non-broadcast, fully asynchronous environment if only one node becomes faulty or Byzantine [9]. Thus, it is important to identify if the system being built needs to be asynchronous or synchronous. Other fundamental properties of fault-tolerant distributed systems include concurrent components that do not rely on a global clock in the presence of delays and other validity faults.

III. Distributed Technologies

After the formulation of the Byzantine general's problem in 1982 [8], various solutions such as Paxos, Raft and Practical Byzantine Fault-Tolerant (PBFT) followed [10] and were eventually adopted by enterprise entities in trusted or permissioned environments.

Whilst Bitcoin is currently the most popular roll-out of a permissionless replicated append-only hash linked list or blockchain [11], earlier experiments have existed [12] and encountered similar trade-offs related to CAP. Blockchains operate in the context of immutable traceability which is not always the predominant requirement for IoT systems as will be discussed.

Another decentralised technology that began to surface along with Bitcoin in 2009 is that of a Conflict-free Replicated Data Type (CRDT) [13] which creates a set of abstract data types that allows for the convergence of concurrent updates with the following properties [14]: *"(i) any replica can be modified without coordinating with other replicas; (ii) when any two replicas have received the same set of updates, they reach the same state, deterministically, by adopting mathematically sound rules to guarantee state convergence."*

To the best of our knowledge, there exist two fundamental types of distributed consensus (asymptotic and convergent) and the aforementioned distributed technologies are currently the three basic implementations in existence today to achieve varying forms of consensus for ordering events.

In other words and a theoretical sense, Proof-of-Resource/Work (PoX) or Nakamoto implementations indefinitely approach consensus whereas aBFT and CRDTs reach consensus eventually, known as finality [15, 16]. The three consensus mechanisms make trade-offs within the CAP spectrum, but before these can be classified, a proper definition of CAP is required.

IV. Trade-offs

Even though there is some confusion regarding the *A* and *P* components within CAP [2], it remains useful as a design tool. Explicit distinctions are required for clarity because a system can be referred to as being *up* or *down* i.e. *vertical-availability* or scaling, by having low algorithmic time-complexity and high performance in the presence of load.

Partition-tolerance can easily be confused with *Availability* and interpreted as *horizontal-availability*, which refers to the ability of the system to uniformly and redundantly synchronise data geographically and to be available in the presence of faulty/malicious/Byzantine behaviour i.e. the fault model. It might be better to refer to this as a *Fault-Tolerance* property as partitions have been modelled as faults using delays, before the CAP theorem's formal inception [2, 17]. To further exacerbate confusion, others have presented a similar Decentralisation, Consensus and Scalability (DCS) theorem [18] although their contextual definitions make more sense than CAP. A less confusing theorem could be postulated as $\mathrm{CA_vA_h}$ or CAF although *Availability* remains an ambiguous term:

1) **Consistency** - Every read receives the latest write or an error.
2) **Availability** - All requests receive non-erroneous responses, with no guarantee that they contain the latest writes.
3) **Fault-Tolerance** - The system continues operating despite an arbitrary number of messages being dropped (or delayed) by the network between nodes.

Whilst considering all previous definitions, it is difficult to place these properties within the CAP spectrum and thus, related terminologies have been classified into Table I in an attempt to find common ground regarding terminology and to induce proper reasoning. Other terminologies encountered within the reading set were also grouped into this table, as these properties represent similar trade-offs within distributed and decentralised systems.

Table I elicits *safety*, *agreement*, *integrity* and *validity* as properties that relate to each other and to how well a particular system meets its contextual requirements. For example, within

TABLE I
TAXONOMY OF TRADE-OFF TERMINOLOGY FROM LITERATURE

GOOD	FAST	CHEAP/COST
Safety [2, 5]	**Liveness** [2, 5]	**Fault-Tolerance** [2]
Agreement [7]	Vertical-time [6]	Horizontal-space [6]
Validity/Integrity [7]	**Termination** [5]	Sharding [3]
Consistency [2, 3]	**Availability** [2, 3]	**Partition-Tolerance** [2, 3]
Correctness [5]	V-Scalability(time) [6, 16]	H-Scalability(space) [6, 16]
Consensus [18]	Scalability [18]	Decentralisation [18]
Atomicity [2]	Finality [15, 16]	Propagation Delay [2]
Linearisability [2]	Latency [19]	Replication [13, 14]
Information [19]	Efficiency [19]	Redundancy [19]

a financial context, the correct ordering of events is of utmost importance to ensure accounting and auditing, whereas reaching a consensus in a swarm of drones regarding their average position could sacrifice some measure of correctness regarding event order.

Liveness and *termination* relate to vertical scaling or performance, as there needs to be a certain flow of information to ensure a frequency at which said information can be considered correctly ordered and agreed upon and is a function of synchronisation and computation.

The least confusing and most generic terms that result from Table I are *Information*, *Efficiency* and *Redundancy*. These stem from information and systems theory where information was defined by Hartley in 1928 as the lack of entropy or uncertainty [19]. Typically, a system or product would lack entropy after it has been tested and qualified to meet contextual requirements which means information and correctness point towards similar entities.

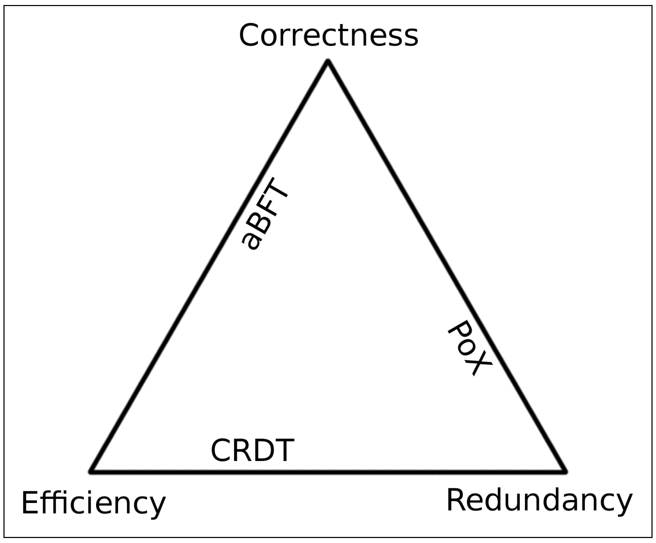

Fig. 1. Distributed Technology Correctness, Efficiency and Redundancy Triangle

In distributed systems, a correct total ordering of events is the primary requirement and therefore, the remainder of this work will promote and adopt *Correctness*, *Efficiency* and *Redundancy* as the terminology to describe the trade-offs within a distributed system. Figure 1 depicts the current three technologies placed in their estimated positions along the spectrum:

1) ***Correctness*** - refers to the system or sub-system meeting its requirement(s) and reducing entropy by trading off against each other: *Efficiency* and *Redundancy*.
2) ***Efficiency*** - refers to the system's optimal usage of resources such as energy, storage and bandwidth to achieve performance or rather: lack of *Redundancy*.
3) ***Redundancy*** - refers to the system using additional resources to achieve a higher level of fault-tolerance or rather: lack of *Efficiency*.

A. Conflict-free Replicated Data Types

Figure 1 shows that CRDTs sacrifice some measure of correctness and settles for various consistency models such as weak-, strong-, strong-eventual-, casual- and just-right consistency. This is done through less synchronisation to provide efficiency and redundancy by grouping commutative operations into data types that possess convergence guarantees and by synchronising only when updates and non-commutative operations occur such as division and subtraction [14].

CRDTs achieve eventual consensus based on lattice joins and guarantee convergence only when all updates are propagated to all nodes [13]. The shared state data is organised in an application-centric manner which results in data not being present on all other nodes and closer to the participants for a particular shared state sub-set. This provides better data access but less shared state redundancy. However, more conflict resolutions are required to ensure correct global consensus or ordering of events.

Although CRDTs do not intrinsically satisfy the Byzantine fault model, attempts have been made to provide this property at the expense of more synchronisation when updates occur [20].

CRDTs are therefore highly efficient, decently redundant but lack correct total ordering of transactions in the context of a global currency. Examples of where CRDTs are used: *Inter-Planetary File System (IPFS), †AntidoteDB, ‡Riak, §Redis and ¶Akka.

B. Proof-of-Resource/Chain-Based

Also from Figure 1, PoX systems sacrifice efficiency and correctness by redundantly synchronising and storing full copies of the shared state information on every "*full node*". They can also scale better horizontally [16] as nodes do not require knowledge of other nodes to tally votes. The implicit resource-based voting mechanism is intrinsically open and permissionless as it randomly selects a participant to process a block of events/transactions and requires no prior knowledge of other nodes when joining the network.

Thus, PoX probabilistically approaches correct ordering through implicit resource-based voting and requires a certain number of blocks to be chained, using hashes, to increase

* https://ipfs.io/ † https://www.antidotedb.eu ‡ https://riak.com
§ https://redis.io ¶ https://akka.io

the likelihood of consensus and therefore sacrifices some correctness [21].

It was shown by [22] that PoX does not provide better fault-tolerance than what is required through the results of [8], implying less than a third of nodes can be Byzantine or faulty regarding resource-based voting. In Proof-of-Work (PoW), less than $1/3$ of mining power needs to be non-Byzantine although there is research suggesting that this threshold is even lower, indicating $1/4$ fault-tolerance in terms of selfish mining [16].

PoX/Nakamoto/chain-based systems are therefore highly redundant, decently correct (asymptotic or probabilistic) but lack efficiency when high event throughput is required. Examples of where PoX is used: Bitcoin's PoW [11], Intel's *Proof-of-Elapsed-Time (PoET) and Ethereum's Proof-of-Stake (PoS) [23].

C. Asynchronous Byzantine Fault-Tolerance

Finally, from Figure 1, aBFT systems sacrifice shared state redundancy and some efficiency by requiring more synchronisation through gossip broadcasts and splits the shared state into shards to scale horizontally [16]. In time-complexity terms, they are better at scaling vertically to achieve higher transactional throughput and lower energy consumption. However, they either require central points of failure to oversee shards or they require additional communications to ensure inter-shard orchestration.

aBFT implementations achieve consensus using explicit voting which induces a definitive time-point when all nodes achieve a consensus of event order, with 100% certainty every r voting rounds. This is convergent as opposed to asymptotic and refers to finality [15, 16]. The downside is that this requires knowledge of all node identifiers which is the limiting factor regarding its horizontal scalability or shared state redundancy.

Many of the existing aBFT protocols are not fully asynchronous which impacts achieving the full 33% Byzantine fault-tolerance [16] and many of them are leader-based which makes them susceptible to follow-the-leader attacks. This can be addressed through randomised aBFT algorithms which allow for much higher levels of asynchrony [16] but introduces latency because of the need for random broadcast or gossip.

aBFT/explicit-voting systems are therefore highly correct, decently efficient but lack shared state redundancy/decentralisation which limits the number of nodes being able to join the network and requires disjoint sections with additional orchestration.

Examples of where aBFT or explicit voting based consensus is used: Paxos [24], Raft [25], PBFT [10], Stellar Lumens/Federated Byzantine Agreement (FBA) [26], Hashgraph [21].

* https://sawtooth.hyperledger.org/docs/core/releases/1.0/architecture/poet.html

V. Recommendations for IoT

PoX systems do not seem like a viable IoT option for various reasons. They consume too much storage and energy resources and have limited formal proofs to converge to a decision. Although PoW does not provide better fault-tolerance than other schemes when a majority of mining power or implicit vote is being compromised, it does seem to present a greater disincentive for attack than other systems, where compromising a majority of the explicit vote is not energy-intensive. However, this requires adoption to reach an acceptable level of security.

aBFT and CRDTs seem more promising in terms of efficiency although the Directed Acyclic Graph (DAG) data structure that they use could prove challenging concerning memory limitations associated with IoT devices as it will limit shard size, however, aBFT doesn't require full state history due to it possessing consensus finality. This means that IoT nodes participating in the consensus protocol do not need to store the full history since the consensus mechanism is not asymptotic. There is also more formal literature available regarding CRDTs and aBFT to further investigate their application.

Another technology worth mentioning to distribute trust is a Distributed Hash Table (DHT) which can adjust the shared state redundancy to a certain amount of copies, uniformly spread out across nodes, which reduces the amount of synchronisation required [27]. This is in contrast to having full state redundancy per node. This uniformity property presents other challenges and trade-offs regarding routing and state efficiency [28] but could be combined with the aforementioned consensus protocols to achieve an autonomous network that could serve IoT devices as a Decentralised Storage Networks (DSN) [29] instead of them persisting the data. The problem with DSNs or any publicly available service is that it suffers from the *"tragedy of the commons"* when free-riders cause an uneven burden to be carried by a minority of network participants.

One idealised solution would be to have highly asynchronous state propagation, correct total ordering of events, adjustable shared-state redundancy and network topology agnostic- and autonomous routing to ensure efficient state propagation. It is also required to have an incentive-punitive mechanism in place when nodes join and leave the network to avoid continuous reconfiguration of routing state information (list of peer nodes) and Sybil attacks. This implies some combination of a DHT and aBFT which is what the team behind Protocol for Asynchronous, Reliable, Secure and Efficient Consensus (PARSEC) [30] known as Massive Array of Internet Disks with Safe Access for Everyone (MAIDSAFE) is attempting [29, 31]. Although the technology is still under construction, there have been interesting developments such as self-authentication and peer-to-peer Public Key Infrastructure (PKI) [32, 33]. To prevent the 33% fault-tolerance from being easily compromised, the MAIDSAFE network implements node-ageing as a function of node service-time, data-retention, -mutation and -distribution. They are also aiming at integration with Prof. Tim Berners Lee's Socially-Linked-Data

(SOLID) which could have future potential as far as Self Sovereign Identity (SSI) is concerned. SSI integration could be a use case where IoT devices are assigned to and from an SSI to always have devices accounted for by human beings.

Regardless of which combination of technology is used from Figure 1, cryptographic primitives are required to apply authentication and incentive-punitive mechanisms to limit attempts by bad actors within the system and to achieve permissionless operation. Apart from state propagation, cryptography remains a fundamental challenge for IoT devices and introduces a bottleneck in achieving low latency, distributed, secure and permissionless consensus. As a security requirement, this implies one signature generation for every message sent and one verification for every message received when IoT devices participate in an asynchronous Byzantine fault-tolerant consensus protocol which is an expensive operation.

VI. Lightweight Cryptographic Primitives

As the fundamental difference between traditional distributed databases and DLTs is the inclusion of cryptography, distributing trust in an IoT context requires that these devices can efficiently do the same cryptographic calculations to reduce latencies and ensure efficiency. In terms of the consensus protocol, three predominant cryptographic primitives are usually used [34]:

1) **Block Ciphers** - ensures *Confidentiality* of information while at rest and in transit and are used to build other functions such as hash-functions or Message Authentication Codes (MAC).
2) **Hash-functions** - induces either uniform distribution of information in key-value pair lookups such as DHTs as well as *Integrity* checks and immutability within systems under the assumption that no hash collisions exist per particular hash function.
3) **Digital Signatures** - ensures source *Authenticity* and provides a measure of identification.

There are two main categories of cryptography namely symmetric and asymmetric, and the same is true for lightweight cryptography. Lightweight symmetric ciphers require less memory, run comparatively faster and are usually combined with asymmetric primitives to establish a session key for a symmetric cipher to provide a confidential communications channel. A wealth of established symmetric algorithms exists; among those, the most prominent representatives are the block ciphers: Advanced Encryption Standard (AES) and Data Encryption Standard (DES). There are several symmetric stream ciphers which prove to be more efficient when compared to block ciphers, where stream ciphers are preferred in some embedded applications even though block ciphers are more secure [35]. Recent primitives are the PRESENT block cipher and PHOTON hash function [36].

Lightweight asymmetric ciphers aim to operate on devices with limited resources, yet require more computational power and are slower than symmetric counterparts [36]. An example from the asymmetric cryptography family is Elliptic Curve Cryptography (ECC), which is considered to be the most effective for resource-constrained devices and is standardised by the National Institute of Standards and Technology (NIST). The Rivest–Shamir–Adleman (RSA) algorithm is the most popular asymmetric primitive, supporting key sizes from 1024 to 4096 bits, whilst ECC offers the same security but with shorter keys and lower computational requirements [37]. In terms of digital signatures, the most popular ECC-based signature scheme is the Elliptic Curve Digital Signature Algorithm (ECDSA).

In summary, the predominant disadvantage of using asymmetric- over symmetric cryptography is the cost of implementation and speed, which remains a challenge for IoT devices. With the growth of IoT, information security is a concern [1], thus cryptographic primitives have been actively proposed and analysed as the NIST aims to develop cryptographic standards that can work within resource-constrained devices. The search for lightweight cryptographic algorithms continues whereby NIST is assessing possible use cases and applications that require lightweight algorithms [38, 39].

In addressing some challenges or limitations associated with using cryptography, the Reduced Instruction Set Computer: Five (RISC-V) has been presented as a possible hardware domain solution which is discussed in the following section.

VII. Reduced Instruction Set Computer: Five (RISC-V)

When resource-constrained devices attempt cryptographic computations, a solution often adopted is to implement cryptographic primitives in hardware, such as an Application-Specific Integrated Circuit (ASIC) or Field-Programmable Gate Array (FPGA) but, these technologies and their associated tooling are often proprietary and expensive; presenting a barrier to entry for researchers and hobbyists [40]. Another barrier is the archaic hardware descriptor languages used to define these hardware accelerators.

RISC-V aims to solve these issues as the first royalty-free, extensible and open Instruction Set Architecture (ISA) specification in contrast to previous patented implementations. With an ISA being the gateway between hardware and software, the intention is to allow for numerous software-defined hardware extensions that leverage common software development [41] to provide an ideal base for specialised accelerators, open to anyone to freely contribute. It is difficult to predict all possible advantages that could result from this technology and so, the following list are some that come to mind:

- Because it is an open ecosystem, it has potential to reduce formal verification overhead so that developers can share and gain more security features such as side-channel attack resistance, which have not been addressed by standards such as FIPS-140 [42].
- Lightweight, quantitative and modular design provides simpler and controllable hardware implementations [43].

- It was designed to implement all ranges of computing devices, such as cloud computing, data centres, network equipment and embedded or IoT devices.
- It can be tailored to address-ultra-low power situations and numerous other use cases such as high-speed compression [43] which is highly beneficial for IoT.
- It could provide backwards compatibility for IoT devices that need to interact with existing systems that use older cryptographic primitives.
- It would be much easier to conduct a security audit due to its royalty-free and open nature.

Although still in its infancy, RISC-V is gaining traction with 48 contributors on GitHub spread out across academia, industry and open-source operating system projects [44, 45]. Western Digital is rolling out more than a billion RISC-V devices per annum and NVIDIA is also adopting the technology for their new graphics accelerator platforms. The introduction of RISC-V has presented the opportunity for the next phase of development in microprocessor hardware architectures [46] and as with any new technology, challenges are expected which might hinder its adoption:

- It runs the risk of producing a non-coherent standard and could result in an explosion of hardware architectures that require support and testing.
- It still lacks a mature and stabilised tooling ecosystem that is generic across all implementations.
- In the big data and server context, it is still unclear whether RISC-V would be faster, more efficient and cost-effective than existing platforms and ecosystems as a whole.

VIII. Conclusion

This research surveyed the basic types of distributed consensus or trust protocols from an IoT perspective, instead of conducting an exhaustive exercise across the numerous distributed ledger projects, whereby trade-offs and recommendations were provided. This research also highlights the security-critical and latency inducing building blocks of existing distributed consensus technologies and two potential paths to follow to achieve secure and performant IoT namely: lightweight cryptography and hardware acceleration using RISC-V. We show that both lightweight cryptography and RISC-V possess much potential, but each presents challenges that are still to be resolved which could take many years. Firstly, the use of cryptography can only address efficiency when lightweight cryptographic primitives that meet the needs of IoT devices and security are identified, accepted, peer-reviewed and standardised, but could lack legacy cryptographic compatibility. Secondly, if the hardware approach is adopted; RISC-V could become to hardware what Linux has become to the software domain. This could allow for highly efficient and context-specific devices to participate directly in a much-needed distributed consensus protocol to avoid central points of failure. It becomes evident that the surveyed technologies are still in the fermentation stage when requiring the permissionless setting where the optimal generic solution might lie in a combination of them. The permissionless requirement is much-needed for IoT to enable decentralised control and is only achievable by using cryptography. Therefore, it might be useful if researchers and developers constrain their thinking to limited resource environments when constructing solutions. Further research and experimentation with technologies such as RISC-V are required by perhaps building an IoT specific distributed ledger test bench using the cheapest and smallest possible embedded devices to demonstrate the effectiveness and practicality of current consensus protocols.

References

[1] B. Schneier, "The internet of things will upend our industry," *IEEE Security & Privacy*, vol. 15, no. 2, pp. 108–108, Mar. 2017. [Online]. Available: https://doi.org/10.1109/msp.2017.39

[2] M. Kleppmann, "A critique of the CAP theorem," *CoRR*, vol. abs/1509.05393, 2015. [Online]. Available: http://arxiv.org/abs/1509.05393

[3] E. Brewer, "CAP twelve years later: How the "rules" have changed," *Computer*, vol. 45, no. 2, pp. 23–29, feb 2012. [Online]. Available: https://doi.org/10.1109/mc.2012.37

[4] M. Rauchs, A. Glidden, B. Gordon, G. C. Pieters, M. Recanatini, F. Rostand, K. Vagneur, and B. Z. Zhang, "Distributed ledger technology systems: A conceptual framework," *SSRN Electronic Journal*, 2018. [Online]. Available: https://doi.org/10.2139/ssrn.3230013

[5] L. Lamport, "Proving the correctness of multiprocess programs," *IEEE Transactions on Software Engineering*, vol. SE-3, no. 2, pp. 125–143, mar 1977. [Online]. Available: https://doi.org/10.1109/tse.1977.229904

[6] ——, "Time, clocks, and the ordering of events in a distributed system," *Communications of the ACM*, vol. 21, no. 7, pp. 558–565, Jul 1978. [Online]. Available: https://doi.org/10.1145/359545.359563

[7] G. Bracha, "Asynchronous byzantine agreement protocols," *Information and Computation*, vol. 75, no. 2, pp. 130–143, nov 1987. [Online]. Available: https://doi.org/10.1016/0890-5401(87)90054-x

[8] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," vol. 4, no. 3. ACM, 1982, pp. 382–401. [Online]. Available: http://people.cs.uchicago.edu/~shanlu/teaching/33100_wi15/papers/byz.pdf

[9] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," vol. 32, no. 2. ACM, 1985, pp. 374–382. [Online]. Available: http://macs.citadel.edu/rudolphg/csci604/ImpossibilityofConsensus.pdf

[10] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, 1999, pp. 173–186. [Online]. Available: http://pmg.csail.mit.edu/papers/osdi99.pdf

[11] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Dec 2008, accessed: 2015-07-01. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[12] S. Haber and W. Stornetta, "How to time-stamp a digital document," *Journal of Cryptology*, vol. 3, no. 2, 1991. [Online]. Available: https://doi.org/10.1007/bf00196791

[13] N. Preguica, J. M. Marques, M. Shapiro, and M. Letia, "A commutative replicated data type for cooperative editing," in *2009 29th IEEE International Conference on Distributed Computing Systems*. IEEE, jun 2009. [Online]. Available: https://doi.org/10.1109/icdcs.2009.20

[14] N. M. Preguiça, C. Baquero, and M. Shapiro, "Conflict-free replicated data types (crdts)," *CoRR*, vol. abs/1805.06358, 2018. [Online]. Available: http://arxiv.org/abs/1805.06358

[15] P. Tasca and T. Thanabalasingham, "Ontology of blockchain technologies. principles of identification and classification," *SSRN Electronic Journal*, 2017. [Online]. Available: https://doi.org/10.2139/ssrn.2977811

[16] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication," in *International Workshop on Open Problems in Network Security*. Springer, 2015, pp. 112–125. [Online]. Available: http://vukolic.com/iNetSec_2015.pdf

[17] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *ACM SIGACT News*, vol. 33, no. 2, p. 51, jun 2002. [Online]. Available: https://doi.org/10.1145/564585.564601

[18] G. Slepak and A. Petrova, "The DCS theorem," *CoRR*, vol. abs/1801.04335, 2018. [Online]. Available: http://arxiv.org/abs/1801.04335

[19] J. Irvine and D. Harle, *Data Communications and Networks: An Engineering Approach*, 1st ed. John Wiley & Sons, Inc., 2002.

[20] W. Zhao, M. Babi, W. Yang, X. Luo, Y. Zhu, J. Yang, C. Luo, and M. Yang, "Byzantine fault tolerance for collaborative editing with commutative operations," in *2016 IEEE International Conference on Electro Information Technology (EIT)*. IEEE, may 2016. [Online]. Available: https://doi.org/10.1109/eit.2016.7535248

[21] L. C. Baird, "The swirlds hashgraph consensus algorithm: fair, fast, byzantine fault tolerance," Swirlds, Inc., Tech. Rep. SWIRLDS-TR-2016-01, 2016. [Online]. Available: http://leemon.com/papers/2016b.pdf

[22] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *Financial Cryptography and Data Security*. Springer, 2014, pp. 436–454. [Online]. Available: http://arxiv.org/pdf/1311.0243

[23] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," 2014, accessed: 2016-08-22. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper

[24] L. Lamport, "The part-time parliament," vol. 16, no. 2. ACM, 1998, pp. 133–169. [Online]. Available: https://www.microsoft.com/en-us/research/uploads/prod/2016/12/The-Part-Time-Parliament.pdf

[25] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 USENIX Annual Technical Conference (USENIX ATC' 14)*. Philadelphia, PA: USENIX Association, 2014, pp. 305–319. [Online]. Available: HTTPs://www.usenix.org/conference/atc14/technical-sessions/presentation/ongaro

[26] D. Mazieres, "The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus," 2015, accessed: 2016-08-01. [Online]. Available: https://www.stellar.org/papers/stellar-consensus-protocol.pdf

[27] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the XOR metric," in *Peer-to-Peer Systems*. Springer Berlin Heidelberg, 2002, pp. 53–65. [Online]. Available: https://doi.org/10.1007/3-540-45748-8_5

[28] J. Xu, A. Kumar, and X. Yu, "On the fundamental tradeoffs between routing table size and network diameter in peer-to-peer networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 151–163, jan 2004. [Online]. Available: https://doi.org/10.1109/jsac.2003.818805

[29] G. Paul, "Secure decentralised storage networks," Ph.D. dissertation, Electronic And Electrical Engineering, 11 2017. [Online]. Available: https://pure.strath.ac.uk/ws/portalfiles/portal/70804098/Paul_2017_Secure_decentralised_storage_networks.pdf

[30] P. Chevalier, B. Kamiński, F. Hutchison, Q. Ma, and S. Sharma, "Protocol for Asynchronous, Reliable, Secure and Efficient Consensus (PARSEC)," 2018. [Online]. Available: https://docs.maidsafe.net/Whitepapers/pdf/PARSEC.pdf

[31] G. Paul, F. Hutchison, and J. Irvine, "Security of the maidsafe vault network," 5 2014, wireless World Research Forum Meeting 32 (WWRF32); Conference date: 20-05-2014 Through 22-05-2014. [Online]. Available: https://strathprints.strath.ac.uk/48569/1/Paul_etal_wwrf32_vault_network.pdf

[32] D. Irvine, ""Peer to Peer" Public Key Infrastructure," 2011. [Online]. Available: https://docs.maidsafe.net/Whitepapers/pdf/PeerToPeerPublicKeyInfrastructure.pdf

[33] ——, "Self Encrypting Data," 2015. [Online]. Available: https://docs.maidsafe.net/Whitepapers/pdf/SelfEncryptingData.pdf

[34] L. Wang, X. Shen, J. Li, J. Shao, and Y. Yang, "Cryptographic primitives in blockchains," *Journal of Network and Computer Applications*, vol. 127, pp. 43–58, feb 2019. [Online]. Available: https://doi.org/10.1016/j.jnca.2018.11.003

[35] A. K. Jadoon, L. Wang, T. Li, and M. A. Zia, "Lightweight cryptographic techniques for automotive cybersecurity," *Wireless Communications and Mobile Computing*, vol. 2018, pp. 1–15, jun 2018. [Online]. Available: https://doi.org/10.1155/2018/1640167

[36] C. Manifavas, G. Hatzivasilis, K. Fysarakis, and K. Rantos, "Lightweight cryptography for embedded systems – a comparative analysis," in *Data Privacy Management and Autonomous Spontaneous Security*. Springer Berlin Heidelberg, 2014, pp. 333–349. [Online]. Available: https://doi.org/10.1007/978-3-642-54568-9_21

[37] B. T. HAMMAD, N. JAMIL, M. E. RUSLI, M. R. Z'ABA, and I. T. AHMED, "Implementation of lightweight cryptographic primitives," *Journal of Theoretical & Applied Information Technology*, vol. 95, no. 19, 2017.

[38] NIST, "NIST Now Accepting Lightweight Cryptographic Algorithm Nominations," www.nist.gov, 2018, https://www.nist.gov/news-events/news/2018/08/nist-now-accepting-lightweight-cryptographic-algorithm-nominations.

[39] K. A. McKay, L. Bassham, M. S. Turan, and N. Mouha, "Report on lightweight cryptography," Tech. Rep., Mar 2017. [Online]. Available: https://doi.org/10.6028/nist.ir.8114

[40] S. D. Mascio, A. Menicucci, G. Furano, C. Monteleone, and M. Ottavi, "The case for RISC-v in space," in *Lecture Notes in Electrical Engineering*. Springer International Publishing, 2019, pp. 319–325. [Online]. Available: https://doi.org/10.1007/978-3-030-11973-7_37

[41] Y. Lee, A. Waterman, H. Cook, B. Zimmer, B. Keller, A. Puggelli, J. Kwak, R. Jevtic, S. Bailey, M. Blagojevic, P.-F. Chiu, R. Avizienis, B. Richards, J. Bachrach, D. Patterson, E. Alon, B. Nikolic, and K. Asanovic, "An agile approach to building RISC-v microprocessors," *IEEE Micro*, vol. 36, no. 2, pp. 8–20, mar 2016. [Online]. Available: https://doi.org/10.1109/mm.2016.11

[42] J. R. Kiniry, D. M. Zimmerman, R. Dockins, and R. Nikhil, "A formally verified cryptographic extension to a risc-v processor," in *Proceedings of Second Workshop on Computer Architecture Research with RISC-V (CARRV 2018)*. New York, NY, USA: ACM Press, 2018, p. 5 pages. [Online]. Available: https://carrv.github.io/2018/papers/CARRV_2018_paper_5.pdf

[43] Z. Cao, Q. Lv, Y. Wang, M. Wen, N. Wu, and C. Zhang, "A compression instruction set design based on RISC-v for network packet forwarding," *Journal of Physics: Conference Series*, vol. 1026, p. 012001, may 2018. [Online]. Available: https://doi.org/10.1088/1742-6596/1026/1/012001

[44] A. Thomas, "Building the risc-v software ecosystem," 2016. [Online]. Available: https://riscv.org/wp-content/uploads/2016/01/Tues1515-riscv_software.pdf

[45] A. Armstrong, C. Pulte, S. Flur, I. Stark, N. Krishnaswami, P. Sewell, T. Bauereiss, B. Campbell, A. Reid, K. E. Gray, R. M. Norton, P. Mundkur, M. Wassell, and J. French, "ISA semantics for ARMv8-a, RISC-v, and CHERI-MIPS," *Proceedings of the ACM on Programming Languages*, vol. 3, no. POPL, pp. 1–31, jan 2019. [Online]. Available: https://doi.org/10.1145/3290384

[46] M. Clark and B. Hoult, "rv8: a high performance risc-v to x86 binary translator," 2017. [Online]. Available: http://rgdoi.net/10.13140/RG.2.2.30957.69601

Morné Pretorius obtained his M.Eng in Computer and Electronic Engineering at the Potchefstroom Campus of the North-West University in 2008 and has worked in the embedded systems industry since 2009, in particular, the cryptographic hardware space since 2013. His current research interests are distributed ledger technologies concerning the internet of things.

Sthembile Mthethwa obtained her MSc in Computer Science (focused in Blockchain technology) from the University of Fort Hare. Currently, she works as a researcher in the field of information security with interests in distributed ledger technologies.