ENTERPRISE ETHEREUM VISION (V3.0)

1. Executive Summary

Enterprise use of blockchain technology has evolved at an almost unfathomable rate over the past 24 months. Through early bitcoin experiments, senior bankers joining start-ups, the launch of Ethereum, the creation of industry consortia, and countless conferences, blockchain has emerged as one of the top enterprise IT trends entering 2017.

The market has already moved beyond the incubation phase where innovators effectively build the technology along with their initial applications, and possibly beyond the early adopter phase too. Increasingly, mainstream enterprise IT organizations are not only educating themselves and experimenting with blockchain but also are aiming to tackle novel use cases and complex IT challenges with the technology. More and more frequently, our clients are asking for assistance building MVPs not POCs, or hardening environments for production readiness.

With this whirlwind, dare we say tornado, of adoption, it is also clear that certain key technologies are emerging as potential de facto standards as blockchain platforms, while at the same time IT organizations can be overwhelmed at the complexity and capabilities of the technology.

Ethereum is, arguably, the most commonly used blockchain technology for enterprise development today. With more than 20,000 developers globally (our estimate), the benefits of a public chain holding roughly $1bn of value, and an emerging open source ecosystem of development tools, it is little wonder that Accenture observed 'every self respecting innovation lab is running and experimenting with Ethereum'. Cloud vendors are also supporting Ethereum as a first class citizen: Microsoft Azure, RedHat OpenShift, Pivotal CloudFoundry all feature Ethereum as one of their, if not the, primary blockchain offering.

Why? The software is widely available: simply download an Ethereum client, pick your favorite development environment, and get going. Ethereum is general purpose and easy to program: full stack / web developers can pick-up the Solidity smart contract programming language in a matter of hours and develop initial applications in only a few days. Documentation is plentiful, as are code samples, deployment frameworks, and training. Little wonder that so many companies are using Ethereum as their blockchain of choice.

Today, enterprises are deploying private Ethereum networks in or near production in areas as diverse as supply chain tracking, payments, data privacy, compliance, and asset tokenisation just to name a few. Certainly, we are some time away from seeing fully automated securities clearing and settlement in production on private Ethereum networks, but we already do see private Ethereum blockchain networks in production, even in financial services.

But enterprises adopting Ethereum face a number of challenges, notably:

(1) Ethereum was developed initially for public chain deployment, where trustless transaction requirements outweigh absolute performance. The current public chain consensus algorithms (notably, proof-of-work) are poorly suited for networks with trusted actors and high throughput requirements.
(2) Public chains by definition have limited (at least initially) privacy and permissioning requirements. Although Ethereum does enable permissioning to be implemented within the smart contract and network layers, it is not readily compatible 'out of the box' with traditional enterprise security and identity architectures.
(3) Naturally, the current Ethereum Improvement Process is largely dominated by public chain matters, and it has been previously challenging for enterprise IT requirements to be clarified and prioritised within it.

As a result, many enterprises who have implemented private Ethereum networks have either 'tweaked' or forked open source implementations or have relied on proprietary vendor extensions to meet their deployment requirements. Some of these are extremely sophisticated and are on the cutting edge of computer science: witness the JP Morgan Quorum platform, Raiden, Parity, and HydraChain. While understandable, and in fact until now the only effective approach, the downsides are obvious: lack of application portability, code base fragmentation, and vendor lock-in.

Not surprisingly, this has been a point of conversation for some months between enterprise technology vendors, corporate users and Ethereum startups. These discussions have expanded, with the blessing and involvement of Vitalik Buterin and the Ethereum Foundation, into a dedicated group of enterprise technology vendors, the largest corporate users and Ethereum infrastructure leaders collaborating to define a roadmap, legal structure, governance and initial technical developments to define 'Enterprise Ethereum'.

To some extent this parallels the paths of other significant platform technologies, such as TCP/IP and HTTP and perhaps (from a software perspective) more relevantly Java and Hadoop.

Java was never intended to be a broadly used enterprise development tool; it was in fact developed originally for interactive television (specifically set-top boxes and smart cards ? who remembers Java Card?). However, Java has many advantages for web development with database back-ends (known as web client-server or three-tier architecture): it has comprehensive web and database APIs, it provides 'write-once, run anywhere' platform portability, simplified object-oriented programming constructs with familiar syntax, and a rapidly developing ecosystem. Indeed, it was not even Sun that created Java Enterprise Edition (at that time, J2EE); it was a plucky start-up (WebLogic) and a group of enterprise customers and other vendors. Similarly, Hadoop was originally created to index the web and for advertising serving. And who knew TCP/IP would emerge into a protocol that exists everywhere today?

Ethereum is one of the few, indeed perhaps the only, blockchain technology with a similar trajectory and potential. By banding together the key adopters, supporters and shapers of enterprise usage of Ethereum, we are seeking to provide a platform not only

for the technology, but also to provide the governance and tools to create a standard for 'Enterprise Ethereum'.

If you are a large corporate user of Ethereum and are interested in learning more about this initiative, please feel free to contact me: jeremy.millar@consensys.net.

2. Introduction

The Ethereum [1] protocol is given as a concise, formal specification [2] of a Turing-complete computation executed on a global virtual machine. The robustness of the Ethereum protocol is evident with the success of the public Ethereum network, with over $1 billion dollars of transactions in around 2 years of existence.

As organizations integrate Ethereum blockchain technologies into their technology stacks, they have developed use cases that incorporate a diverse range of enterprise features which are not directly supported by the current Ethereum protocol. Enterprise features includes user and transaction privacy, scalable computation, and network connectivity. For example, a user who wishes to send transactions whose content is visible only to a specific set of users can choose to use J.P. Morgan's Quorum [3], a fork of the Ethereum Foundation's Geth client implementation. Messages sent over a public, permissioned network of Quorum nodes trigger execution of securely communicated [4] transactions that update the private state database of all all specified private parties.

Integrating Enterprise Ethereum protocol features into the Ethereum protocol is challenging for three reasons. First, most core Ethereum protocol changes are hard forks that require consideration, debate, consensus and adoption by the entire Ethereum community. Second, private, permissioned Ethereum-based networks are unable to easily leverage the public Ethereum network without well-defined support for differences in privacy and network connectivity. Third, integration of different block consensus mechanisms to address the Ethereum network's scalability, such as Proof of Work, Proof of Stake, and Proof of Authority, is unclear. These reasons encourage extending the Ethereum protocol and adding support for Ethereum Enterprise protocols in a way that does not rely on the current method of community-based consensus and software support over hard forks.

2.1. **Contributions.**

In this paper, we propose a vision for users and stakeholders to propose, implement, and integrate advances to the Ethereum protocol with support for Enterprise Ethereum protocols. Our contributions in this paper are as follows:

(1) We propose and motivate Ethereum protocol updates that are intended to provide flexibility to Enterprise applications
   (a) network connectivity: consensus algorithms (Simple Consensus, PBFT, HoneyBadger, Tendermint, Ouroboros)
   (b) Configurable privacy involving user confidentiality and tx privacy
(2) We describe features that are relevant in the long term vision for Enterprise Ethereum, including governance frameworks, secure computation, and token management.

3. Short Term Vision

Enterprise Ethereum has four clear goals for 2017:

(1) Develop a sufficiently modular Ethereum implementation to separate and define clear interfaces between networking and storage layers – that is a prototype for *pluggable consensus* that minimizes the code changes required to switch consensus algorithms
   (a) An emphasis on interoperability, including public Ethereum, enterprise and legacy systems, and alternative blockchain protocols
   (b) Stay close to the public Ethereum code and roadmap
(2) Experiment with potential consensus algorithms, along with data privacy and permissioning frameworks
   (a) Replace proof of work - address potential concerns about settlement finality for high value transactions
      (i) Validate digital signatures providing cryptographic proof and therefore enable settlement
      (ii) Very high level of reliability
   (b) Interface to the canonical next block hash
   (c) Canonical design patterns
   (d) Obfuscate private data while retaining overall system consistency
   (e) Observe the required abstractions to enhance protocol
(3) Develop a clear set of capabilities and performance characteristics that suits the needs of enterprises
   (a) E.g. 100 trx per second across a 10 party network
   (b) notably for higher volume or higher value use cases
   (c) Higher complexity / workflows
   (d) High availability / reliability
   (e) Parallelisation and horizontal scaling
(4) Develop a Version 1 specification for Enterprise Ethereum, based on the learnings from the above plus the roadmap and requirements gathered from members, i.e., Produce a reference implementation thereof
(5) Leverage a robust governance process to ensure alignment and agreement on approaches

3.1. **Modularised Ethereum implementation with 'pluggable consensus'.** Most current Ethereum implementations (particularly the 'reference' implementations) are not particularly modular. Some of the vendor implementations do improve on this, but typically also implement 'non-standard' features in part to achieve this. A common, modularized implementation will provide a code based for developing the Enterprise Ethereum specification and also experimenting with consortia consensus algorithms. 'Pluggable' consensus needs a modularized client with a clean interface for networking and between the Ethereum Virtual Machine and the consensus algorithm - it is really these interfaces that make the consensus layer pluggable.

However, our goals here are broader than simply achieving modularity at the consensus layer. For example, implementing configurable privacy (a combination of authorization/authentication and data privacy) also will benefit from a modular and abstracted Ethereum implementation, as will many potential future features. In fact, client modularity, a 'thinned' virtual machine and a streamlined wire protocol are also low level goals for Ethereum more broadly. It just happens that achieving these goals is a priority for Enterprise Ethereum, given our requirement set.

Much of our current development thinking and focus is centered on these issues. We will likely experiment and prototype using the Python Ethereum implementation and also test against Geth. Generally speaking, the Python Ethereum implementation is where many Ethereum features are being prototyped, it is relatively modular (compared to other 'reference' clients), Python itself is well liked by developers and productive for prototyping.

In parallel with these efforts, we have begun reviewing potential consensus algorithms (and indeed low level wire protocols) that could be leveraged by Enterprise Ethereum. There are several considerations and learnings here:

- This is an area of research where extensive progress has already been made: Honeybadger, Tendermint, DFinity, Quorum and many others already have PBFT/BFT or similar implementations
- Enterprise Ethereum should not be prescriptive as to the specifics of the algorithm, but rather should enable various algorithms to be used dependent on the use case (i.e. 'pluggable' consensus).
- However, the Enterprise Ethereum reference implementation should incorporate a 'sample' consortia consensus algorithm
- Enterprise Ethereum will also endeavor to benchmark and continue to research consensus algorithms, to provide its members with recommendations and updates as to latest research, particularly in regards to performance, fault tolerance and malicious actors

The consensus algorithms that are being considered for Enterprise Ethereum are novel, and the abstractions, approaches, and scenarios needed to integrate consensus algorithms will emerge as the consensus algorithms become more prevalent in diverse Enterprise use cases. The current approach of Proof of Work is based on Kademlia [5] without distributed hash table functionality, thereby tying the current consensus algorithm to an arbitrary peer-to-peer network communication protocol. It is important to abstract consensus algorithms from the network layers, as it would allow consensus to work over multiple communication modalities in addition to the current peer-to-peer approach. It would further permit the possibility of running multiple consensus algorithms in environments where multiple permissioned blockchains are working together. Abstracting network layers then allows a reactive approach to processing streams of blocks, akin to moving from batch-processing with Hadoop to stream-processing with Kafka. Our efforts will promote stability and capture a range of settlement finality options that will depend on various Enterprise environments. Our aim is to deliver an abstracted and modularised Ethereum implementation (most likely Python) with a sample consortia consensus algorithm.

The second highest importance areas for investigation and development are privacy and permissioning (authentication/authorization). Certain code bases being evaluated by Enterprise Ethereum, such as Quorum, include implementations of these features. And these features (generally) benefit from modularization. So we anticipate that these features will follow the client modularization and pluggable consensus, but this is for prioritization by the members.

3.2. **Version 1 Enterprise Ethereum Specification.** The goal of Enterprise Ethereum is not simply to develop an implementation but to publish a specification that vendors can implement. Key elements of the specification should include:

- White paper and yellow paper extensions
- Well specified interfaces between the wire protocol, EVM, consensus algorithm, smart contracts and accounts
- APIs for application development, devops/management tools, etc
- Compatibility and interoperability with public Ethereum
- Compliance testing suite

In addition to evolving well defined interfaces from the initial abstraction and modularization work, Enterprise Ethereum aims to accelerate ecosystem development by increasing the availability of APIs for integration with development, devops and management tools as well as legacy system integration. For example, bulk data loading is currently challenging in 'standard' Ethereum, though a number of vendor implementations simplify this.

But perhaps most importantly, developing the Enterprise Ethereum specification demands collaboration between the members and through the working groups to establish clear priorities and architectural approach. In addition to the core specification, we should aim to develop and evolve the road map contained in this document.

3.3. **Robust governance.** We have already invested heavily in developing a governance model for Enterprise Ethereum. At the highest level, this breaks down into:

- The governance of the Enterprise Ethereum group
- The administration and operations of the Enterprise Ethereum group
- Core membership participation in working groups

Enterprise Ethereum will almost certainly be instantiated as a US-based not for profit / foundation. Our legal team is exploring the specific options for this. Legally and practically, the Enterprise Ethereum Foundation will need a lead executive. Initially, we are proposing a volunteer Chair; we anticipate hiring an Executive Director to take on the day to day management functions in due course. The term of the Chair should be strictly limited - likely to one year. In addition, we anticipate a Board of Directors supervising the Chair (and Executive Director) and / or an Advisory Board.

The Enterprise Ethereum Foundation will require administration: legal, financial, IP, PR, membership coordination, etc. We have viewed this administration as living in the Foundation, potentially staffed by Foundation personnel or volunteers. The Chair (and Executive Director) will be responsible for insuring this group is appropriately advancing. There has been a strong indication from members that they will want direct influence on

marketing and communications, but still there are many day-to-day operations that are best managed proactively.

However, the core of Enterprise Ethereum is its members and member participation. Already, a number of areas are emerging as key member interest groups:

- Code group: advancing prototypes, potential specifications, etc - the nuts and bolts of writing EntEth
- Architecture group: overseeing the roadmap, architecture and technical direction
- Industry groups: bringing together interested parties to collaborate on industry specific issues, such as in finance

Combined, these elements represent a strong starting platform for Enterprise Ethereum. Clearly, there is much to achieve in 2017. Fortunately, we have a founding membership base with the motivation, passion and technical expertise to give us a strong belief that this can be achieved.

In our approach, we maintain four guiding principles on what the Enterprise Ethereum community wants to achieve and deliver:

(1) Development of open source standards
(2) Working with builders and doers towards a general purpose system
(3) Maintain compatibility with public ethereum
(4) Not reinventing the wheel on data standards

3.4. **An emphasis on interoperability.** While it may be early to discuss cross blockchain protocol interoperability, already Enterprises have significant interoperability requirements, such as:

- Ability to switch components at various layers of the Enterprise Ethereum while maintaining application portability and network transactions
- Ability to provide non-standard extensions that are interoperable with the core specification
- Inbound and outbound data interfaces, and EVM *hooks*
- Public chain compatibility

Many aspects of interoperability are enabled through abstraction and modularity, with clear interfaces and APIs. Very quickly, these interfaces will become difficult to change, as the code base expands. Therefore, upfront thought and design effort in the aim of interoperability is critical to consider even at these earlier stages.

4. Updating the Ethereum Protocol

We consider two important features for development in the short-term: pluggable consensus and configurable privacy. In this section, we describe why these two features are relevant for the Enterprise Ethereum community.

The Ethereum protocol specifies that nodes choose the next block for the longest-chain based on Proof of Work (PoW), a computation that is expensive in terms of time, electricity, and memory, a mechanism that dissuades invalid block proposals. The PoW mechanism is effective for pacing the growth of the Ethereum network and protocol as it matures;

however, this means that currently blocks can only be committed to the blockchain at intervals on the order of a block around every 10 seconds, thus limiting the number of transactions that each block can accept. With modern-day organizations providing online services to hundreds of millions of users, there is an obvious need to extend the Ethereum protocol to support the scale of existing enterprise technologies.

One way to reduce block time is to reduce the difficulty of the PoW computation, a setting that results in a reduction in the amount of resources needed to control block generation on the Ethereum network. For an attacker to submit an invalid block to the head of the longest chain requires millions of dollars worth of electricity and computational power.

To improve scalability, alternatives to the expensive Proof of Work block consensus protocols focus on providing block consensus using different relevant blockchain properties. For example, both Delegated Proof of Stake [6] and Ouroboros [7] rely on Proof of Stake to choose blocks based on a proportion of stake value, and Proof of Authority [8] relies on the notion that authoritative nodes choose blocks. Such approaches focus less on the network layer and instead rely on communication between distributed nodes as provided by a client implementation. Further dividing the space are where consensus protocols are implemented, such as in the case of Casper [9], a Proof of Stake protocol that is intended to be expressed as a smart contract.

Consensus protocols for the Ethereum blockchain need to perform atomic broadcasts of a total order of state sequences amongst a network of nodes that may experience byzantine failures. Proposals for network consensus protocols for Ethereum include HoneyBadger [10], HydraChain [11], Tendermint [12], which rely on variations of Practical Byzantine Fault Tolerance [13] for network communication with a possibility of failures [14]; one may review a survey [15] on a variety of atomic broadcast protocols.

Approaches that combine block and network consensus protocols, include HydraChain and Ethermint [16]. Ethermint leverages Tendermint's block and network consensus protocols and adding the Cosmos [17] specification of how multiple blockchains can interact with one another.

Other approaches that aim to address the challenge of PoW's scalability include sharding, parallelization, and state channels. Scaling of transaction throughput by executing off-chain transactions between several parties at a fast pace is generally referred to as state channels [18], which leverage optimizations of reduced or non-existent gas costs; state channels generally place a record of the start and completion of off-chain transactions on-chain.

5. Updating the Enterprise Ethereum Protocol

In the sections above, we highlighted several features, currently under development within the broader community, to provide relevant Enterprise features including privacy, network connectivity, and scalability. In this section, we describe an expansive vision for the Ethereum protocol that aims to encompass Enterprise features. Our goal with Enterprise Ethereum is to be able to support deviations from the public chain protocol for

different protocol rules. The ability to more easily swap out different rulesets, by swapping out contract code or performing a hard fork, on a test network or private implementation means that Enterprises can follow a standard to switch between different rules. At the implementation level, supporting modularity and abstraction can lower code complexity and attack surfaces. For instance, distinguising between per-application or per-user configurations such as EIP 86 [19] which describes a use case where some users want their accounts secured by ECDSA, Lamport for quantum-proofness, and the use of specific forms of private, industry, or national standards cryptography.

5.1. **Administration of Shared Digital Infrastructure.** Governance is a big topic across the blockchain and DLT industry. Its applicability includes the consensus algorithms at the protocol level and the operations of industry consortia and standards bodies, such as Enterprise Ethereum itself. Another general area of consideration for governance is the shared administration of permissioned smart contract systems. These systems include both private blockchains as well as smart contracts with restricted access on public networks.

We are all anticipating the enormous benefits of transacting directly between industry partners on shared digital infrastructure enabled by the blockchain, but what happens when those shared programs need to be versioned and updated? The answer to that and related questions will help us develop useful standards for this kind of shared administration, such that all relevant participants could be involved in the vetting of, voting for and deploying of updates to their shared system.

Enterprise Ethereum could look to explore the development of contract interfaces and classes for shared administration, in addition to design patterns applicable to the updating of operational code on already deployed systems. To provide an example, consider the following code used in the uPort identity system:

https://github.com/ConsenSys/uport-proxy/blob/master/contracts/Owned.sol

This contract provides a level of governance over any contract that inherits it. It could be a starting point towards an elaborated and defined standard, which would enable multiple dApp developers to build various Administration Tools, for smart contract systems, which interoperate with one another by way of these standards. Standards for shared administration would likely start on application layer smart contract systems and then be extended from there to include protocol level updates to private chains operating between numerous industry partners.

We can imagine now a future scenario where a new consortium chain consensus algorithm is developed that shows marked improvement in some important performance benchmark; with these shared administration standards that span both the application and the protocol layers, there would be a clear path towards making and tracking updates to the jointly administered system and transitioning to the new consensus algorithm. In this way, no matter which vendors were chosen for smart contract development or post deployment administration, all parties could reliably administer their shared infrastructure together.

In the context of Enterprise Ethereum, whose goal is to coordinate Enterprise requirements, there needs to be a way to coordinate the activities. Currently, the process for integrating improvements to the Ethereum protocol is to submit an Ethereum Improvement Proposal [20] using the git versioning model. In this model, a user who is interested in recommending a change to the Ethereum protocol would first fork the EIP repository as a copy. The user would then create a new EIP document that describes the motivation, specification, and impact of the change; the EIP is committed to the user's own repository. Alongside the EIP document, the user has the option of writing code that implements the EIP. Finally, the user would submit a pull request to the main repository for feedback and review of the EIP and associated code.

In an Enterprise setting, organizations may participate within the community at multiple levels, such as developers and managers, and requiring each to have a github handle is not efficient. Instead, using the blockchain, while leveraging the existing git-based model, provides support for modern Enterprise Ethereum governance mechanisms (an 'eat-our-own-dog-food approach'). An Enterprise extension to the current EIP process onto the Ethereum blockchain is desirable for a few reasons. First, organizational involvement in the changes to the Enterprise Ethereum protocol would be captured as formal records. Second, by moving organizations intentions through smart contract, the process involving multiple organizations can be formalized and incentivized. Thus, by moving the governance architecture onto the Ethereum blockchain, organizations have a clear contractual mechanism by which to conduct their Enterprise Ethereum proposals and activities.

Several blockchain-based governance tools on are already in existence, under active development, with integration in progress. uPort is a self-sovereign identity platform that can support personal and organizational identity. It consists of a mobile app that manages users' keys and credentials to sign transactions, as well as developer libraries for simple integration. uPort released a Developer Alpha version for preliminary testing and intends to support identity needs of the foundation. uPort is currently being incubated at ConsenSys. BoardRoom is an application for making and voting for proposals at an organizational level, committee level and project level. Balanc3 is currently in active development and will be able to create financial assets, transfer, and report all financial transactions.

5.2. **Trusted Computing and Oracles/Data Feeds.** There is increasing interest in leveraging the trusted computing support which is starting to appear in mainstream hardware within decentralized applications to enhance their security characteristics. Intel SGX is the prime example of such support. SGX provides *enclaves*, memory-encrypted on-silicon circuits which can only run signed, attested code, in a way such that other processes cannot access it. These have been characterized as reverse-sandboxes, which don't trust the OS which they are running on top of. Proof of Elapsed Time (POET) within an enclave is Intel's consensus method within their Sawtooth Lake DLT. Microsoft offer code execution within SGX enclaves as an option in their Cryptlets. ARM offer similar functionality to SGX with their TrustZone and TEE (Trusted Execution Environment).

The consumption of data external to the blockchain (for example - forex price feeds) is paramount to real-world applications of smart contract based systems. When considering

important use cases whereby data feeds are automatically shared through the blockchain as a way to trigger contract execution, ensuring that the data being ingested is from a trusted source becomes critical to these applications. Another prominent use case for data feeds is for data integration of legacy systems, such as for reporting based on identity and identifiers already captured in existing systems. There is a close relationship between trusted hardware and data feeds that are ingested into a blockchain. As one of the design goals of the Ethereum virtual machine is to be deterministic, network access is not provided as an instruction thereby making data pushes into the blockchain one of the easiest ways to ingest data. As data feeds are intrinsically external to the blockchain protocol, there is not an immediately apparent protocol-level implementation which can be leveraged by the Enterprise Ethereum project. However, there are best practices and design patterns to take into consideration when considering data feeds for Ethereum smart contracts.

When designing a data feed system designed for blockchains, two important things to consider are: the source of the data, and the channel with which the data is being transmitted. Within a single enterprise, for single purpose (i.e., well trusted) applications, both the source and the channel can be as simple as a periodic task which consumes a legacy data feed and publishes it to a destination contract on the blockchain. However, in environments where multiple counterparties need to agree on the output of data feeds, due diligence should be applied.

There are efforts in the broader distributed ledger ecosystem which are attempting to tackle the data feed challenge. From a research perspective, there are initiatives such as Town Crier from the Computer Science department at Cornell which seek to "present an authenticated data feed system" which "acts as a bridge between smart contracts and existing web sites, which are already commonly trusted for non-blockchain applications". From a commercial perspective, initiatives such as Microsoft's Cryptlet Fabric (as a part of the broader Bletchley initiative) focus on leveraging hardware enclaves as the basis for a hardware enabled data oracles.

While hardware-based approaches provide performant security features, hardware solutions rely on expense and vendor trust. Secure hardware is prevalent in server environments, and its prevalence continues to increase through integration in IoT and network infrastructure; however, due to the myriad of solutions without a standardized API, hardware is more challenging to support. Enterprise needs depend on the level of trust of actors working together in a permissioned enviornment. As a result, there is also a desire to support software-based enclaves as an option; open source enables community review and reduces cost. As security needs in blockchain technology evolves, enterprises will be able to better understand and provide additional offerings.

While the Ethereum Enterprise initiative will focus on protocol level implementations at the onset, with time, efforts to establish meaningful design patterns around oracle data (and perhaps more importantly, how to apply penalties for bad actors publishing malicious data) are expected to emerge from members of the Foundation.

5.3. **Storage.** Many decentralized applications could benefit from decentralized storage in addition to the blockchain. While it is possible to store arbitrary content in the blockchain,

the blockchain is neither cost-effective nor efficient for file storage. Immutable, content-addressable storage systems off-chain are better suited to such needs. These systems seek to decouple the authoring of content from the long-term hosting of that content.

There are two projects with very similar goals which are leading development efforts in this area. The first is IPFS, led by Juan Benet of Protocol Labs. The second is Swarm, funded by the Ethereum Foundation, which has long been envisaged as a central pillar of the Web3 vision, alongside Ethereum and Whisper.

A number of Ethereum projects are already using IPFS, which is in a more mature state than Swarm. Both projects have two conceptual layers - a chunking file system layer and an incentives layer. See IPFS and Swarm summary, written by Viktor Tron. The incentives layers for both systems are both in their early days, but will both be built using Ethereum. The design of the incentives layers will be key to the long-term success of these systems.

5.4. **Performance improvement and optimization.** The Ethereum protocol continues to retain success as a world blockchain-based computer in large part due to the feasibility of executing tens of transactions of per second, which avoids technical challenges associated with computing at scale in a distributed, trustless context. In a setting where many of the parameters involving Enterprise users vary, such as the number of users and their geographic location, and the number of transactions executed amongst participants having varying connectivity (such as due to timezone), close awareness of the performance capabilities of blockchain systems an organization would choose to deploy is of vital importance.

(1) Define useful benchmarks based on various identified parameterizations of features
(2) Sketch how clients could be instrumented to measure various kinds of performance
(3) Relevant Parameters:
    (a) Node connectivity
    (b) Optimal network topologies (num and patterns of connections)
    (c) Reliability of connectivity (maintenance of connectivity among nodes)
    (d) Failure cases: when do nodes stop talking to each other
    (e) Safety
    (f) Liveness
    (g) Min number of healthy honest nodes required for proper operation
    (h) Time to transaction finality
    (i) Total number of network nodes
    (j) Node topologies
    (k) Internode distances in terms of transmission time
    (l) Synchronous / Asynchronous messaging
    (m) Transactions per second throughput
    (n) Transaction / block transmission and processing latency

5.5. **Data management layers, such as through logging and events.** Businesses have the ability to use Business Process Modeling (BPM) tools, such as Activiti [21] that rely on web standards to capture and describe their processes for non-technical users. Often, processes are expressed in XML, such as Business Process Modeling Notation

(BPMN) [22] to describe interactions with web services expressed in Web Service Description Language (WSDL) [23]. Currently, there are no tools that support blockchain applications within their BPM toolset, which then leads to businesses having to rely on experts to integrate their existing processes with the novel blockchain paradigm. Furthermore, the current Ethereum protocol expresses its web APIs in an Application Binary Interface (ABI) format that is incompatible with existing BPM tools. Due to the non-standard ABI format, developers face the challenge of integrating blockchain web APIs into their tool sets without well-defined semantics, resulting in implementations that do not work the same way.

Providing a standard layer to express the web services provided by the Ethereum protocol will allow integration of the Ethereum blockchain into existing BPM tools without modification. By providing this form of integration, the Ethereum blockchain can undergo data integration with a business' existing processes, including monitoring and logging. Furthermore, integration with existing tools provides an effective way to abstract the blockchain paradigm from non-technical users.

References

[1] "Ethereum whitepaper," *https://github.com/ethereum/wiki/wiki/White-Paper*.
[2] G. Wood, "Ethereum yellowpaper," *https://github.com/ethereum/yellowpaper*.
[3] J. Morgan, "Quorum," *https://www.jpmorgan.com/quorum*.
[4] J. Morgan, "Constellation," *https://github.com/jpmorganchase/constellation*.
[5] 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*, IPTPS '01, (London, UK, UK), pp. 53–65, Springer-Verlag, 2002.
[6] BitShares, "Delegated proof of stake consensus," *https://bitshares.org/technology/delegated-proof-of-stake-consensus/*.
[7] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," 2016.
[8] "Proof of authority," *https://github.com/ethereum/guide/blob/master/poa.md*.
[9] V. Zamfir, "The history of casper," *https://blog.ethereum.org/2016/12/06/history-casper-chapter-1/, https://blog.ethereum.org/2016/12/07/history-casper-chapter-2/*, 2016.
[10] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of bft protocols," 2016.
[11] "Hydrachain," *https://github.com/HydraChain/hydrachain*.
[12] E. Buchman, "Tendermint: Byzantine fault tolerance in the age of blockchains," *MASc Thesis*, 2016. https://github.com/tendermint/tendermint.
[13] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI, pp. 173–186, USENIX Association, 1999.
[14] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *J. ACM*, vol. 32, pp. 374–382, Apr. 1985.
[15] X. Défago, A. Schiper, and P. Urbán, "Total order broadcast and multicast algorithms: Taxonomy and survey," *ACM Comput. Surv.*, vol. 36, pp. 372–421, Dec. 2004.
[16] "Ethermint," *https://github.com/tendermint/ethermint*.
[17] "Cosmos," *https://github.com/cosmos/cosmos/blob/master/WHITEPAPER.md*.
[18] "Description of state channels," *http://www.jeffcoleman.ca/state-channels/*.
[19] "Ethereum improvement proposal 86," *https://github.com/ethereum/EIPs/issues/86*.
[20] "Ethereum improvement proposals repository," *https://github.com/ethereum/EIPs*.
[21] "Activiti bpmn platform," *https://www.activiti.org/*.

[22] "Business process model and notation (bpmn)," *http://www.omg.org/spec/BPMN/*.
[23] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web service definition language (WSDL)," tech. rep., Mar. 2001.