Scalable blockchain platform for practical solutions.

Gorki Network
contact@gorki.dev

November 1, 2023
v1.0

Abstract

Gorki's vision is to create an advanced L1 blockchain platform that addresses the challenges currently faced by Web3 developers and users while also preparing for future trends in hardware and software, namely massive parallelism. A process calculi-based programming language has been developed as the computational model for the blockchain's Virtual Machine (VM) to achieve this. This programming language explicitly exposes concurrency and offers tight resource tracking at a calculus level.

This approach grants Gorki fine-grained control over concurrent execution and conflict resolution between shard validators, as well as within a single transaction. It eliminates the need for external concurrency management techniques like Software Transactional Memory (STM).

Gorki's primary objective is to build a fully decentralized blockchain that minimizes synchronization requirements, aiming to solve the well-known blockchain trilemma. Concurrent block production and validation enable scalability at the execution layer, while process calculi provide an elegant solution for resolving dependencies and conflicts.

Research and development of the consensus protocol are derived from the CBC-Casper branch to maximize resilience and decentralization while remaining a Proof of Stake (PoS) protocol. The development of this novel, leaderless consensus algorithm affords Gorki the ability to create software that maximizes the utilization of existing hardware and paves the way for the efficient use of concurrent hardware in the future. Ultimately, this should lead to the lowest possible transaction costs for each degree of decentralization.

One aspect that sets Gorki apart is attention to data, based on the premise that concurrent access to data stored on the blockchain is essential. This approach allows Gorki to maintain the ownership of data, which is crucial in concurrent settings. Gorki's smart-contracting language facilitates the seamless storage and exchange of data blobs between platform users.

1 Introduction

The last 15 years of development of online services have all been about digital asset management platforms from large corporations. They help billions of people upload, disseminate, and manage zettabytes of data. While many online digital asset management platforms began with more open models, they are becoming less open daily, still firmly rooted in the web 2.0 paradigm[1].

The emergence of a decentralized global data network represents a significant shift, yet the potential of the blockchain to completely upend the market continues beyond that. With the development of the consensus algorithm known as proof-of-work, the Bitcoin network used it to store a ledger recording the balances at the Bitcoin holder's addresses. This choice has obvious but limited utility. A more sophisticated choice for what to store with a consensus algorithm is the state of the virtual machine. This choice, originally conceived and developed by Ethereum, turns the global data store into a global computer.

More generally, the market has been exploring decentralized alternatives[2], like blockchains, while building online services that have sprung up and insinuated themselves into modern life over the last few decades.

At the core of every blockchain is an economically-secured, leaderless consensus algorithm. Essentially, algorithms of this type, proof-of-work, proof-of-stake, or some other kind, allow computer programs that do not trust one another to agree on a value. Since they agree on the value, they can store a local copy for easy access and only run the algorithm if there is a change to that value. Such a capacity, if it were scalable enough, makes it possible to deploy a global decentralized data network.

2 Current runtimes for dApps

Current proof-of-work runtimes for dApps (decentralized applications) are inherently wasteful, spending inordinate compute cycles. The protocol uses extensive energy resources to secure the network[3].

Not only are they energy-intensive, but current proof-of-work consensus algorithms, like Bitcoin, only process less than 11 transactions per second. Further, even current proof-of-stake consensus algorithms have problems with realizing their advertised transaction rates. Chains with higher transaction numbers, like Solana [4], rely on linear block production. Their virtual machines are usually sequential in their computation, in that all transactions must be processed sequentially through the validators, or if some level of parallelism is enabled on the platform, typically there is a requirement to define dependencies upfront, leading to difficulties in developing and implementing smart-contracts and apps[5]. In some cases techniques like ”optimistic execution” are leveraged, however this has known limitations, either from hiding complexity from the developers and pushing the complexity to the validators, This can lead to issues in transaction validity, or even having to reschedule transactions[6].

3 Gorki Protocol

The Gorki protocol is being designed to minimize the amount of synchronization required in order to avoid the infamous blockchain trilemma (liveness, safety, and fault-tolerance) to ruin user experience. To do this, we are targeting four major innovations and principles.

- The mathematical model backing the state of a computer is derived from process calculus, which uses process names as the fundamental element of that computer's state, and computation is described through the process of exchanging data between names. To give concurrent access to this state names are stored inside a tuple space[7], inheriting from coordination language Linda. For practical reasons, this can be seen as a map that stores the content of the channels (the data passed between the names), which allows Gorki to easily compute the proof of state by maintaining this map in the form of a Merkle tree. Another benefit of this model is that it allows easy access to sharding through namespaces. Since each name can belong to one or more namespace, transferring a value from one shard into another is a matter of transferring records through the states.

- Rholang, the programming language of the platform, is an exact description of the state of the computer. In essence, it is the WYSIWYG principle. Rholang supports an object-capability security model that unlocks an enormous amount of value when deployed to a shared execution environment like a blockchain. Rholang is the API to a powerful concurrent state machine which prevents the user from making mistakes, restricts access and manages resources. Since process calculus has a history of usage proving concurrent programs (CCS)[8], this opens a path towards a proof system for on-chain smart-contracts. However, unlike existing proof assistants - Rholang allows for proof on concurrent ones.

- It is well known that systems that rely on a leader for consensus are prone to attacks, but also, there are no systems to date presented that are entirely asynchronous. One breakthrough is the Nakamoto consensus used in PoW systems, which employs economic incentives to make finality probabilistic and make attacking unprofitable. Gorki is developing a consensus algorithm inspired by the Casper CBC research branch to enable a genuinely leaderless PoS consensus. A synchronous version of such a consensus algorithm (codename: Weaver) has been developed and deployed on the Gorki devnet.

- Gorki employs programming principles inherited from Rholang, building composable modules under concurrent settings. Essentially, all software manipulates data in memory, but finding the right compromise between simplicity and efficiency is crucial. Concurrency and composability are the main requirements that software should match in the upcoming decade to fully utilise hardware resources and reach the levels of performance required to facilitate the next generation of decentralised applications.

Each of these components and the innovations they unlock are designed to meet specific market needs. In general, Gorki would much rather follow state-of-the-art practices and engineering where it makes sense. At the same time, Gorki recognises that if we are going to build a platform sound enough to rebuild the world's data and financial networks, it needs to be of a completely different kind of quality than what we find in most internet software.

4 Technology

The Gorki technology stack is built in a layered structure. The GSpace layer is a persistent layer for the storage of data and execution of smart-contracts. On top of this is Rholang layer, the by-design fully concurrent language for decentralised networks. Additional layers are implemented above this, as smart-contracts, written in Rholang that describe the consensus protocol.

4.1 GSpace - A new kind of store

In the last decade, technical communities, especially those involved in big data, have seen a rethinking of storage and retrieval. In particular, a dialectic around the no-SQL alternative to relational data stores has been developed. First, a wave of storage systems based on the key-value pair, together with the map-reduce paradigm, emerged. A backlash followed this, levying critiques of the key-value store paradigm regarding the semantics of queries and transactions. GSpace threads the needle, offering a no-SQL store but with precise query semantics and clear transactional semantics. It goes beyond this by offering a critical feature necessary to support user-controlled concurrency in queries: the ability to store code and data. Putting code and data on equal footing in the storage layer derives from a consistency constraint in Gorki's design; this inherent treatment of both smart-contract code and decentralised data generates Gorki's powerful data handling capabilities.

Meredith devised a version of this for Microsoft's BizTalk process orchestration [1], a business process automation platform, then updated the idea to use so-called delimited continuation in SpecialK, and finally proposed the GSpace design as a further refinement, this design is at the core of what allows Gorki to scale.

Given the need to organise and handle the world's data in a decentralised way, Gorki begins with a store that builds and improves upon what we have learned from the last decade of big data. Components like this sit as private assets inside all the major digital asset management platforms, from Google to Facebook. One key difference, however, is that our store is open-source and fits into a decentralised public infrastructure. Its features and functions are derived from specific concurrency semantics embodied in Rholang.

4.2 Rholang - A new concurrent programming language

When designing a new language one has to look at the requirements needed for writing smart-contracts for distributed systems.

A careful analysis of the various models of computation, from Turing machines [2] to lambda calculus [3], from Petri nets [4] to the π-calculus [5], shows that there are four properties we are interested in relative to this market's requirements.

- Completeness - can we say everything we need to say?
- Compositionality - can we build more complex programs out of simpler ones?
- Concurrency - can we build programs that have parts that run simultaneously?
- Complexity - can we measure the cost of computational resources

Table 1 shows that the π-calculus, and more generally, the family of models of computation known as the mobile process calculi, is the only one that has all four features necessary. Analysing this table also makes it possible to compare different technologies that other decentralised networks use.

It is also worth mentioning that it is not just the blockchain that demands concurrency as the model of computation. The programming model for Internet-scale programs has been under considerable pressure to move to a concurrent model of computation for quite some time. For the last two decades, two trends have been putting pressure on the programming model from below and from above. From below, we see that Moore's law ended in the early 2000's [6]. When Moore's law was driving processing power on computational platforms, i.e. increasing with time, it was, therefore, possible to see performance gains in software purely from the increase in capacity of the hardware year on year. This trend ended when limits to sequential processing speeds were hit in the early 2000s. Recently, the predominant way to scale applications and computational power has been to scale horizontally, adding more resources. This has been seen in industry-scale supercomputers, the dawn of GPU-based processing and in data centres around the globe. A derivative of this horizontal scaling is that code that does not take advantage of resources on offer concurrently does not scale. Likewise, from above, the commercialisation of the Internet has resulted in user demand for programs that are globally accessible by millions of concurrent users.

Reaching the apex of how many transistors can be placed in a processing unit and its resulting halt in processor speed increase means that the programming model used for programming web applications, whether they are on the blockchain or not, must evolve to be concurrent to have sufficient use. The mobile process calculi form the basis for that evolution, not only because they represent significant advances in language design but because of a sound basis for static analysis of programs. The importance of this feature is hard to state. Concurrent programming is many times harder than sequential programming, both in terms of design and debugging potential errors. Without significant support from static program analysis, the bugs in concurrent code will become overwhelming as the number of programmers writing concurrent code increases.

4.2.1 Rho-calculus vs π-calculus

The π-calculus is just one example of a family of models enjoying all the features necessary to address this market. Since Turing Award winner Robin Milner put forward the model[9], several models of computation sharing many of the π-calculus features have been identified and studied, including the join calculus, the blue calculus, and the ambient calculus. Each of these has exciting properties but ultimately fails in one way or another to map and program the internet as the π-calculus. There is one model, however, that derives from the π-calculus but fixes a small lacuna

[1] https://en.wikipedia.org/wiki/Microsoft_BizTalk_Server
[2] https://en.wikipedia.org/wiki/Turing_machine
[3] https://en.wikipedia.org/wiki/Lambda_calculus
[4] https://en.wikipedia.org/wiki/Petri_net
[5] https://shorturl.at/douvW
[6] https://interestingengineering.com/no-more-transistors-the-end-of-moores-law

	Completeness	Compositionality	Concurrency	Complexity
Turing machines	✔	X	X	✔
lambda calculus	✔	✔	X	X
Petri nets	✔	X	✔	✔
CCS	✔	✔	✔	X
π-calculus	✔	✔	✔	✔

Table 1: Comparison of different models of computation

and at the same time adds some powerful features that are common in programming the internet, namely Meredith and Radestock's Rho-calculus [10].

The Rho-calculus plugs a hole in the π-calculus by making names first-class elements of the model. The π-calculus is parametric in a theory of names; that is, given a theory of names, the π-calculus will produce a theory of processes that get computation done in terms of communications that use those names as channels. The π-calculus is agnostic as to precisely what these names represent, e.g. telephone numbers, email addresses, blockchain addresses, or all of the above. However, the pure π-calculus can only exchange names between processes. This is a powerful capability; however, it can limit the efficiency of the work that can be carried out, for example, creating a program that performs a task purely by exchanging phone numbers as pieces of data. It may be possible to complete any task in this manner. However, it is likely to lead to a lot of wasted resources. What happens on the internet, however, is that not only data but code gets distributed from process to process, and the rho-calculus supports this feature. The Rholang language specification paper describes in detail the grammar and semantics of this language.

4.2.2 Namespaces

The Rho-calculus achieves this ability to ship processes by making names be the codes of processes. Once names are code, it is possible to encode all the different kinds of common telecommunications notions of addresses into the Rho-calculus setting, everything from email to blockchain addresses embed nicely, and thus it is straightforward to embed the most common addressing scheme on the Internet: URI's and URL's. This latter is important not just because it is the way the entire World Wide Web is organized, but also because it identifies a very powerful feature that the Rho-calculus refines: namespaces. URI's organize the web into a tree of resources. Each URI is a path from the root of the tree along the branches to the leaf, or endpoint that holds the resource. Because of this path structure, it is possible to indicate entire groups or spaces of resources using only partial paths. This allows developers to organize and search spaces of resources in terms of the tree and path structure. The Rho-calculus enhances this paradigm by identifying these spaces programmatically.

The reason this feature is so critical in this market is, because most transactions are isolated. A programmatic way to segment, organize, and reorganize transactions is needed so that they are grouped in terms of the resources they have in common. This is commonly called sharding in the blockchain space. The Rho-calculus namespace capability provides an extremely powerful approach to sharding. Specifically, the same approach can be used to rejoin forks, interoperate with other networks, as well as give speed-ups on transactions that are isolated.

4.2.3 Operational semantics and correct-by-construction language design

The synergy presented by the correct-by-construction methodology and the type system used by Rholang is an important property of the Gorki technology. The Rho-calculus provides a Turing complete operational semantics of Rholang's core features. Each of Rholang's additional features are defined in terms of a mapping back to the core calculus. This means that the language is correct-by-construction. Contrasting this with other programming languages e.g. Solidity [1] [2], where there is no guarantee that a specific compiler has preserved the security and integrity of the code that was created when generating the EVM byte code. As of the publication of this paper, Solidity does not yet have a formal semantics.

Rholang cannot suffer a number of differnet types of attack e.g. byte-code manipulation or arithmetic manipulations, because the language and the execution mechanism are both derived directly from the Rho-calculus.

It is also worth noting that a clean operational semantics is necessary to identify when one program is substitutable for another, in other words, when is it safe to put a different program in place of another in a wider execution context. It is common practice to use these semantics to ensure the maintenance and stability of a system in the long term, this is a critical feature, when swapping out upgrades or applying fixes for old or failing components. In software development, this is predicated upon having a semantics of the programming language, and operational semantics are by far the most widely used form of programming language semantics in theory and practice.

[1] https://en.wikipedia.org/wiki/Solidity

[2] https://medium.com/mycrypto/the-ethereum-virtual-machine-how-does-it-work-9abac2b7c9e

This principle of substitutability is closely connected to static analysis and especially the form of static analysis, commonly called type checking. All programs inhabiting a given type are substitutable in any context requiring that type. This is the basis of correct plug-n-play software and is used in all serious production settings. However, the type systems for most popular modern languages are relatively weak, only ensuring that data structure supplied matches the data structure expected. There is usually no verification of the program structure, or program behavior. Over the last two decades, a new class of type-systems (behavioural-types) has emerged, but have yet to find their way into a mainstream programming language.

4.2.4 Object capabilities

Further, Rholang provides an elegant solution to enable 'object capabilities' (OCAPs). The OCAP security model is a superior model to 'access control lists' (ACL). ACL is now almost ubiquitous, and used almost all blockchains, despite its weakness to hacks. There have been a number of attempts to correct for this, for example linux's SELinux, AppArmor and many more, these take the place of an enhancement to ACL, but not the solution. Since Rholang has unforgeable names as first-class citizens, users can generate an unforgeable name and use it as a capability (in this example the OCAP becomes like a key to a secure room) - it is then possible to pass this capability to anyone, or restrict further sharing of this capability. So for a decentralized computer with lots of users and no trust, having OCAPs huge benefits and facilitates fine-grained security models to be created to manage and secure decentralised data.

4.3 Consensus algorithm

Consensus algorithms, especially in Proof-of-Stake protocols, are the most controversial topic in blockchain. Projects claiming thousands or hundreds of thousands of transactions per second but, in reality, demonstrating very modest results are a common practice. The Gorki network is built on the belief that it is not possible to distance consensus from execution; any attempt to do this is misleading. When a protocol focuses on straightforward arithmetic operations on lists or maps of balances, like in the case of digital money - the model for blockchain VM can be simplistic enough and then relatively high TPS can easily be demonstrated. A decentralised computing infrastructure is a different class of problem; it is a multi-vector optimisation problem: ease of use, composability, access control, data storage, throughput, latency, and more. On this basis, bold statements regarding simplistic throughput metrics do not make sense. Scalability is the key to unlocking actual adoption - having more useful compute when adding more hardware, whereby the actual meaning of useful is driven by the user, not by the protocol.

Many leading blockchain protocols currently spend the vast majority of their throughput on their consensus algorithms [1]; the design and implementation of this can be driven by security, providing confidence for their users. However, there are more sustainable solutions. Consensus inevitably has a cost, and different use cases require different degrees of decentralisation. The Gorki platform is being designed as a solution that is adaptable and user-friendly enough to enable a wide range of applications. Most blockchains also have a notion of a leader; this is expected since it makes the protocol much faster and more attractive to the general public. Nevertheless, Bitcoin (and many other PoW which are leaderless by default) still has the highest rank across all projects. If it were not for the overwhelming waste of resources to solve the cryptographic puzzles ensuring the network, the Nakamoto consensus might be the best protocol for distributed computing. Unfortunately, most energy is spent just to maintain the protocol itself, which leads to several problems in itself. However, the payoff is that there is no leader, therefore less risk of censorship and a higher attack cost. Gorki is developing a consensus algorithm that takes advantage of the low energetic cost of a PoS blockchain with the security of a leaderless consensus, with all validators being equal. The consensus algorithm is derived from the CBC-Casper research branch to develop a protocol robust enough and lightweight enough to support a scalable execution engine, which Rholang provides.

Main rules that put a constrain on a message DAG are:

1. **Continuity**. Message has to see everything that the self parent sees.
2. **Frugality**. Message should not add new messages to the view of self parent from offenders detected in the view of self parent.
3. **Integrity**. Message disagreeing with offences declared by an ancestor should record that ancestor as an offence.
4. **Unambiguity**. Having multiple parents from the same sender is the proof of an offence. Messages providing the proof should record an offence.

Linear logic, developed by Girard in the late 80s and early 90s, revolutionised our understanding of logic and proof. Unlike classical or intuitionistic logic, which considers the proposition "A A" to be the same as "A," linear logic also considers the resources required to establish a proposition. This resource-sensitive logic is similar to establishing properties of chemical compounds where many assays require modifying or even destroying a quantity of the compound. Linear logic is balanced, meaning all resources must be carefully accounted for, and double-spend is prevented.

Linear logic shows promise in achieving consensus, especially economically secured consensus. Traditional BFT consensus with propose-commit logic looks similar to this level of calculus. Among the different semantics for linear

[1]https://solana.com/news/network-performance-report-july-2023

logic, game semantics stands out as intuitive and pluripotent, with many variations illuminating a wide variety of features of linear logic.

Hyland and Ong's game semantics[11] provide an extremely faithful interpretation of the logic. In their semantics, each move made by a player or opponent must be justified by previous moves. This justification structure is used to establish strategies, which must be single-threaded. Meredith proposed using this structure to secure network protocols in communicating between instances of an earlier version of GSpace. Following these insights, our proof-of-stake consensus algorithm imposes and exploits a justification structure on blocks to detect equivocation and provide liveness constraints and fairness constraints. These properties are ensured by imposing specific communication patterns, and the justification structure gives a view into just enough of the history of communication that these properties or their violations can be detected.

4.4 Optimistic concurrency and block merge

Optimistic concurrency is a widely discussed topic in modern blockchain platforms[12]. The state models that distinguish between shared and isolated objects, such as in protocols like Sui, and software transaction memory are used to run concurrent computation and join parallel threads. This process of joining parallel threads is called "block merge". It allows for the computation to be joined using the unique properties of the mathematical model behind Rholang. Conflicts are defined by the process calculus, and two branches of execution can conflict for two reasons: race conditions for the same event and having the tuple-space in non-normal form when composing two transactions. Since transactions can be executed upfront without ordering, execution decides on the execution path inside the smart contract call. Users are not required to provide all dependencies for their program upfront, as the program finds out dependencies during the execution process. With dependencies and logs of execution at hand, conflicts between two execution traces can be easily identified.

The consensus layer plays a crucial role in determining how a block is treated. Each block makes a local decision about finality; having this functionality in the protocol allows for speculative conflict resolution. We propose using rounds, similar to Blocklace[13], but more finely-grained, operating with fringes. A fringe is a set of descendants of another fringe, akin to a Tetris game. The final fringe, defined by the block view, determines the blockchain state below which all futures will agree. The part above is the conflict set, which remains uncertain. A block can be recorded as B(finalScope: Set[B], finalState: State, conflictScope: Set[B]), where B is a block and State is the blockchain state being built while the network progresses. Each 'next final state' is computed by merging the next round of already executed blocks into the previous final state, observed by the block parents. Each pre-state for a block is the conflict set merged into the final state. By advancing the fringe in a way that ensures safety, all possible futures will agree on the same final state. Blocks can thus be produced concurrently, providing scalability.

4.5 Sharding

Since Rholang executes by adjusting the content of channels, and the state of the computer (GSpace) is effectively a map with a proof of state, it is possible to have multiple instances of GSpace that enable database sharding. This also allows for thinking about a part of a shard (e.g. a submap containing only channels for a specific dApp) as a separate shard. So, given a dedicated root in a hash tree, a dApp developer should be able to validate only the part of the state that is related to their application. In essence, sharding is enabled by the composable nature of Rholang, which is exposed in the language and reflected in GSpace. Compute sharding is enabled by the fact that the execution of each block is isolated and defined only by its view.

4.6 Garbage collection

It has been discussed previously that the structure of blocks in the DAG is effectively recursive, where each block is essentially a reference to its own view, which also consists of previous blocks. This can be illustrated with the following figure.

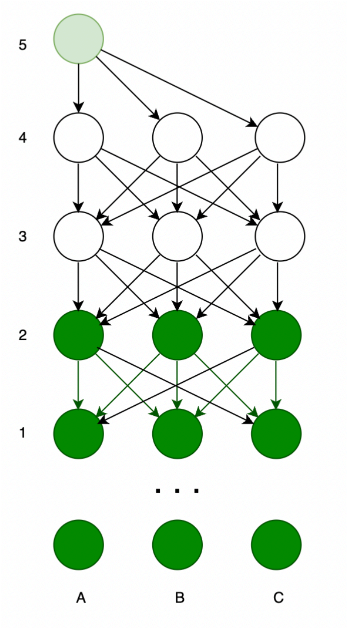

Figure 1: Local view of the message

Block A5 here observes the DAG through:

$$A_4, B_4, C_4$$

To be able to be merged, and therefore finalised, the view of the block transactions from blocks:

$$A_3, B_3, C_3 \ \& \ A_4, B_4, C_4$$

have to be merged into the final state of the current fringe:

$$A_2, B_2, C_2$$

This means that the oldest messages that are required to be available for the merge algorithm are:

$$A_3, B_3, C_3$$

To be able to merge two blocks the same local view has to be available, as it is shown for A5 on Figure 1. To be able to merge, all views of

$$A_3, B_3, C_3$$

have to be reconciled. Given views of finality (or in DAG terms, final fringes) of these blocks can be nothing more then prefix or suffix of each other, it is enough to have blocks for the view that is the oldest one across:

$$A_3, B_3, C_3$$

So given the local DAG that a validator has and rules for the DAG formation - we can easily identify where the tips of the DAG are and what are possible views of new valid messages. Therefore everything older than these views require - can be remove from the consensus state. Truncating the state of a blockchain is more tricky than truncating a DAG (consensus state), because one piece of data might belong to many states represented by different blocks. So even if blocks can be removed from the historical DAG, the whole state corresponding to this block cannot be removed, yet. But, as it has been mentioned earlier, the process of merging starts with the state of the final fringe. So its enough to have the state of the lowest final fringe that potentially can be used for future valid messages.

5 Governance

When discussing a globally secure and decentralised computer and data storage network, questions regarding network governance cannot be avoided. In Gorki, one solution is to handle forks as shards. Sharding is a sophisticated method of establishing an economic connection between multiple networks, ensuring the network remains open and scalable even in the event of governance disagreements. The Gorki token will be central to establishing the structures in place for governance.

5.1 Jurisdiction

It is crucial to explore more scalable governance structures because blockchain technology transcends geographical boundaries. Just like the commercialisation of the internet allowed for global commerce and markets, blockchain technology goes beyond that. It is possible to create a smart contract that locks a digital asset in a way that all parties involved in the contract find unfavourable or unfair.

Geopolitical boundaries that are based on jurisdiction are not very useful in certain situations. Communities that may be diverse either geographically, culturally, and/or politically, however, share common interests and goals are better aligned with the basic organisational components of the blockchain. The emergence of blockchain technology will pose fundamental challenges to global jurisprudence. The UAE, being open to decentralised technologies, provides an ideal environment for Gorki to grow in a fast-changing and dynamic landscape. Leading decentralised/centralised exchanges and layer-1 platforms are moving to the UAE due to regulatory support for these emerging technologies.

5.2 Tokenomics

The Gorki token is the native utility token for the Gorki network and is needed for paying for the execution of smart-contracts by the validators. Additionally, the Gorki token secures the network by rewarding validators automatically through a smart-contract (proof-of-stake smart-contract) if they behave honestly. Otherwise, they are slashed. The reward is dynamically adjusted by the proof-of-stake consensus algorithm based on network participation. The reward is not predefined, and there is no fixed guarantee of a reward.

There will be 10.000.000.000 tokens minted at genesis. The tokenomics paper by Gorki references the amount of tokens released to participants in the token distribution. All other remaining tokens will be held in the staking reward pool at genesis, and distributed as rewards once external validators are enabled on the network. One of the core values of Gorki is that inflation will be very low at the inception of the network. The inflation rate at which validators will get rewarded in addition to transaction costs will be regulated by a smart-contract. The inflation rate depends on the number of transaction blocks over time. Substantial growth in network users will result in increases in circulation supply over time.

6 Ecosystem

The following dApps show, how smart-contracts are currently written for running on the Gorki network. They show the vast possibility for new ecosystem participants.

6.1 Atomic transactions - On-chain data storage

Smart-contracts governing transactions in the blockchain can create governance structures and ensure transparency in systems. A powerful use case for a blockchain solution is to allow atomic transactions, small transactions with data that would be uneconomical to track and monetise in traditional systems. In this paradigm, even a simple post on a social media platform, or a photograph taken on a smartphone and transfered to cloud-storage can be a transaction on Gorki. Opening the way for unique pay-per-use models in data storage and content delivery.

The data handling capacity of Gorki is second to none in the Web 3.0 space. As a comparison, the Ethereum virtual machine and, by derivation, most 'EVM compatible' architectures are built to store the state of bits and web assembly code – this design is resource-led and meant to be as close to the bare metal of the servers as possible to maintain speed and efficiency. Handling data is not in their design. In the Gorki network, data and code for smart contracts are treated in the same manner. This treatment of data, in channels, is part of the design and core platform that allows us to create a concurrent computational model. So not only are there several unique aspects to what we can do with data on Gorki, but there are substantial potential use cases in almost every sphere of the "cloud data world".

The data handling capacity and unique positioning of Gorki can be summarized through the following features:

- Interoperability - connecting and being the communication/orchestration layer between systems and shards.
- Concurrency - Maximising throughput of transactions by employing concurrency mitigating network latency and avoiding lengthy bottlenecks.
- Security - fine-grained control over security and data ownership.
- Composability - powerful computationally intensive smart-contracts can be executed on data.
- On-chain data storage -no requirement for an external solution, enhanced decentralisation with fewer attach vectors.

On-chain data can either be decentralised, and secured by the Gorki network, or centralised on existing systems and the movement and ownership of that data can be orchestrated by Gorki. The possibilies are endless.

6.2 Atomic transactions - Content delivery and marketing

In today's digital landscape, the freemium business model has emerged as a powerful strategy to attract and engage users. By offering core functionalities at no cost and reserving advanced features for a premium, businesses are effectively lowering the barriers to entry, encouraging mass adoption, and fostering user loyalty. This approach has been notably successful in various web-based and mobile applications, allowing companies to establish a broad user base quickly. When executed precisely, the freemium model can be a win-win: users benefit from free access to essential tools and services, while businesses generate revenue from those who see value in upgrading. Leveraging this model for newer technologies can revolutionise how consumers interact with platforms, especially in sectors where upfront costs traditionally deter user engagement. With that in mind, a significant pain point for blockchain adoption is the need to continually load wallets with tokens to perform even rudimentary activities. In many cases, this leads people to centralised exchanges to on-board tokens and then have to perform numerous steps to get tokens to individual wallet accounts. Creating a portal for the freemium model in the hands of app developers is a methodology to lower this barrier to entry. Integrating marketing APIs into a native blockchain is a path to "kill two birds with one stone". Marketing and engagement models will need to be able to bridge from existing systems and databases, security and privacy being some of them, as well as integration into existing content systems and marketing channels. The interoperability of Gorki is critical for this. Interoperability will allow us to onboard marketing paradigms that were not previously possible in the blockchain space, moving from Web 2.0 to Web 3.0, where content can be directly loaded on-chain. Imagine a decentralised application allowing the user to turn on/off advertising through a wallet app function. Alternatively, think of an application allowing users to choose which information they would be willing to share based on tokenised incentives. Gorki opens up the possibility for data-intense platforms to monetise through marketing or utility token payments, depending on the user's preference for privacy. All this requires the need to connect and interoperate. This will also require smart-contract execution to be done concurrently because there will be latency and compute involved. The objectives here would be to integrate this into the core Gorki APIs – creating two specialised entry points:

- Sponsored content API - This API is used to create and curate content on-chain, providing the correct metadata to the content and allowing the consumption model that it is designed for to be specified.
- Self-disclosure API - This API also allows users to divulge information about themselves with regards to content they would like to see, information that they are willing to share, and perhaps if they are willing to pay for services and app usage that would mean that they see less sponsored content, or none.

6.3 Decentralised healthcare - IoT and data privacy

Many medical institutions have historically depended on local infrastructure to store patient records, and a significant number do not have robust backup systems. This approach can lead to potential data breaches or data degradation. As a rising number of these institutions shift towards cloud-based storage, they still face threats. For instance, during the COVID-19 crisis, healthcare systems were notably targeted by ransomware and distributed denial-of-service attacks[1]. These digital attacks hampered critical emergency services, rendering numerous patients without medical care. In addition to these challenges, there are instances where traditional data storage methods incorrectly process patient records, which can result in grave health implications. Simultaneously, there's been a surge in the development of compact health sensors that monitor patients' vital statistics. This progression has bolstered the overall quality of the healthcare ecosystem[14]. The proliferation of these portable medical devices has amplified the efficiency of health tracking. They serve dual purposes:

- User - advanced health and fitness monitoring.
- Professional - a richer dataset, a wealth of information that can potentially speed up diagnoses and shape treatment modalities[2].

However, with the increased frequency of reporting this has brought an equal increase in the volume of the data that is required to be distributed and stored. In the quest for more secure storage, there's growing interest in technologies like Blockchain[15]. However, many existing system designs merely leverage the trust capabilities of blockchain and utilise other layers of technology to overcome the lack of data handling capacity on-chain. Gorki does not have this limitation, an encapsulated system with sovereign data can be built entirely on the platform, eliminating dependencies and vulnerabilities.

6.4 GorDrive - On-Chain file storage

GorDrive is a utility to mount a Gorki shard as a filesystem. All NFTs, domain names, tokens, etc. are viewable as regular files. This allows even a casual non-technical user to interact with the blockchain and perform basic web3 tasks such as uploading and transferring ownership of assets with a simple drag and drop. The ultimate use case for GorDrive has been demonstrated on the server side, turning most off-the-shelf legacy applications into web3-enabled hybrid dApps to create viable alternatives to existing Big Tech for purchasing and streaming movies and music from chain, sharing files, or decentralising data.

6.5 Game Engine SDK

Gaming has become an important part of our lives. It's what we grow up with, how we spend time with friends and discover new ones. This is why every time a large AAA publisher introduces blockchain into their games, the community rejects it, creating a massive backlash. Players tend to see it as a predatory monetary scheme rather than improving their experience.

Gorki believes that the gaming community is more than just simple NFTs, players need to feel cared for and not exploited. The Gorki network's ability to host decentralised data on-chain, and integrate with legacy systems will enable new and existing gaming ecosystems to leverage the network to share and distribute content, unlocking new platforms and integrating communities. At the same time, fine-grained security models will allow for safe and secure interactions to unlock thriving economies in new and existing gaming worlds. Building gaming SDKs for game engines with interaction with the Gorki network is of high value for dApps that want to use it for players. The community is planning to organise several game jams to let game developers discover new ways of using our technology within games.

6.6 Self-sovereign identity, data privacy, and decentralization

The dominance of a few centralised companies in the digital asset management platform capabilities has been a concern for both the market and the public sector. To address this issue, the market has pursued key alternatives such as self-sovereign identity, data privacy, and decentralisation. All of these pursuits have converged on the technology known as blockchain.

6.6.1 Data privacy

There is a growing concern among people about large online companies hoarding personal information, particularly after the security breaches they have experienced in recent years. This has led to new terms to describe the public's increasing anxiety about protecting their data. Many individuals and organisations, including Sir Tim Berners-Lee, the inventor of the World Wide Web, have called for personal data vaults. In response, governments have introduced regulations like the European Union's GDPR. Gorki is a platform that allows people to control their data by creating their own smart-contracts, thus providing data privacy and self-sovereignty.

7 Summary

Gorki offers a grounded and unique combination of technical innovation and economic opportunity compared to existing decentralised networks or blockchains. The mobile process calculi have dominated protocol design and protocol analysis for decades. Gorki's rho-calculus-based language, Rholang,

[1]https://healthitsecurity.com/features/the-threat-of-distributed-denial-of-service-attacks-in-healthcare
[2]https://www.igi-global.com/gateway/chapter/www.igi-global.com/gateway/chapter/219853

allows developers to enter the modern world where programming language semantics meets protocol design. By putting math first but making it available for everybody to use, we can bring scalable smart-contracts to users and bridge enterprises from Web 2.0 into the decentralised world becoming the communication, orchestration and trust boundary for all.

References

[1] Miltiadis D Lytras, Ernesto Damiani, and Patricia Ordóñez de Pablos. Web 2. 0: The Business Model. Springer, New York, NY, 1 edition, 2008.

[2] Akanksha Kaushik, Archana Choudhary, Chinmay Ektare, Deepti Thomas, and Syed Akram. Blockchain - Literature survey. pages 2145–2148. IEEE, 2017. Book Title: 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT).

[3] Alex de Vries. Bitcoin's Growing Energy Problem. Joule, 2(5):801–805, 2018. Publisher: Elsevier Inc.

[4] Anatoly Yakovenk. Solana: A new architecture for a high performance blockchain. `https://solana.com/solana-whitepaper.pdf`, Accessed: 2023-01-14.

[5] Jian Liu, Peilun Li, Raymond~Cheng, N. Asokan, and Dawn Song. Parallel and Asynchronous Smart Contract Execution, June 2023. arXiv:2306.05007 [cs].

[6] Alex Valtchanov, Lauren Helbling, Batuhan Mekiker, and Mike P. Wittie. Parallel Block Execution in SoCC Blockchains through Optimistic Concurrency Control. In 2021 IEEE Globecom Workshops (GC Wkshps), pages 1–6, Madrid, Spain, December 2021. IEEE.

[7] Enrico Denti and Andrea Omicini. An architecture for tuple-based coordination of multi-agent systems. Software, practice experience, 29(12):1103–1121, 1999.

[8] Howard Bowman. Concurrency Theory Calculi an Automata for Modelling Untimed and Timed Concurrent Systems / by Howard Bowman, Rodolfo Gomez. Springer London, London, 1st ed. 2006. edition, 2006.

[9] Robin Milner. Elements of interaction: Turing award lecture, 1993. ISSN: 0001-0782 Issue: 1 Pages: 78–89 Place: New York, NY Publication Title: Communications of the ACM Volume: 36.

[10] L.G. Meredith and Matthias Radestock. A reflective higher-order calculus. Electronic Notes in Theoretical Computer Science, 141(5):49–67, 2005. Proceedings of the Workshop on the Foundations of Interactive Computation (FInCo 2005).

[11] J. M. E. Hyland and C.-HL Ong. On Full Abstraction for PCF: I, II, and III. Information and computation, 163(2):285–408, 2000. Place: San Diego, CA Publisher: Elsevier Inc.

[12] Cheqing Jin, Shuaifeng Pang, Xiaodong Qi, Zhao Zhang, and Aoying Zhou. A High Performance Concurrency Protocol for Smart Contracts of Permissioned Blockchain. IEEE Transactions on Knowledge and Data Engineering, 34(11):5070–5083, November 2022. Conference Name: IEEE Transactions on Knowledge and Data Engineering.

[13] Idit Keidar, Oded Naor, Ouri Poupko, and Ehud Shapiro. Cordial Miners: Fast and Efficient Consensus for Every Eventuality, September 2023. arXiv:2205.09174 [cs].

[14] Faisal Jamil, Shabir Ahmad, Naeem Iqbal, and Do-Hyeun Kim. Towards a Remote Monitoring of Patient Vital Signs Based on IoT-Based Blockchain Integrity Management Platforms in Smart Hospitals. Sensors, 20(8):2195, January 2020. Number: 8 Publisher: Multidisciplinary Digital Publishing Institute.

[15] Shivansh Kumar, Aman Kumar Bharti, and Ruhul Amin. Decentralized secure storage of medical records using Blockchain and <span style="font-variant:small-caps;">IPFS</span> : A comparative analysis with future directions. SECURITY AND PRIVACY, 4(5):e162, September 2021.

THIS WHITEPAPER PROVIDES AN INITIAL SUMMARY OF CERTAIN TECHNICAL AND BUSINESS ESSENTIALS UNDERLYING THE GORKI PROTOCOL. THIS DOCUMENT IS EXPECTED TO EVOLVE OVER TIME, AS THE PROJECT PROCEEDS. THE GORKI TEAM MAY POST MODIFICATIONS, REVISIONS AND/OR UPDATED DRAFTS FROM TIME TO TIME, INCLUDING BEFORE, DURING, AND AFTER THE CREATION OF ANY TOKENS, AND WHILST NETWORK(S) BASED ON THE GORKI PROTOCOL ('GORKI NETWORKS') ARE IN OPERATION.

THIS DOCUMENT SETS FORTH A DESCRIPTION OF THE GORKI PROTOCOL, REFERENCE SOFTWARE IMPLEMENTATION, AND POTENTIAL GORKI NETWORKS. THIS INCLUDES DESCRIPTIONS OF THE PROTOCOL ITSELF AND THE USE OF TOKENS SUCH AS THE PROPOSED GORKI TOKEN, GORKI TOKEN. THE POTENTIAL GORKI NETWORK IS A DECENTRALIZED, PUBLIC BLOCKCHAIN UPON WHICH PEER TO PEER TRANSACTIONS CAN BE CARRIED OUT BY USERS AND DEVELOPERS. DEVELOPERS HOLD AND CONSUME UNITS OF NETWORK CAPACITY ON THE PROPOSED GORKI NETWORK TO BUILD AND MAINTAIN DISTRIBUTED APPLICATIONS AND USERS STAKE, USE AND TRANSFER UNITS OF NETWORK CAPACITY ON THE PROPOSED GORKI NETWORK. THESE UNITS OF NETWORK CAPACITY ARE EXCLUSIVELY REPRESENTED BY CRYPTOGRAPHIC UTILITY TOKENS.

THIS DOCUMENT IS PROVIDED FOR INFORMATION PURPOSES ONLY AND IS NOT A BINDING LEGAL AGREEMENT. ANY SALE OR OTHER OFFERING OF GORKI TOKENS WOULD BE GOVERNED BY SEPARATE TERMS & CONDITIONS. IN THE EVENT OF CONFLICT BETWEEN APPLICABLE TERMS & CONDITIONS AND THIS DOCUMENT, THE TERMS & CONDITIONS GOVERN. THIS WHITEPAPER IS NOT AN OFFERING DOCUMENT OR PROSPECTUS, AND IS NOT INTENDED TO PROVIDE THE BASIS OF ANY INVESTMENT DECISION OR CONTRACT.

Legal disclaimer

As of the date of publication, the Gorki team have no plans to launch any public Gorki Networks, and Gorki Tokens are a proposed token with no known potential uses outside of Gorki Networks, and no such use is intended. This document does not constitute advice nor a recommendation by the Gorki team, its officers, directors, managers, employees, agents, advisers or consultants, or any other person to any recipient of this document on the merits of purchasing, otherwise acquiring, or holding Gorki Tokens or any other cryptocurrency or token. The purchase and holding of cryptocurrencies and tokens carries substantial risks and may involve special risks that could lead to a loss of all or a substantial portion of any money invested. Do not purchase tokens unless you are prepared to lose the entire amount allocated to the purchase. the purchase. Gorki Tokens, if and when they are created and made available, should not be acquired

for speculative or investment purposes with the expectation of making a profit or immediate re-sale. They should be acquired only if you fully understand the intended functionality of the Gorki Tokens, and you intend to use the Gorki Tokens for those purposes only, and it is legal for you to do so. No promises of future utility or performance or value are or will be made with respect to Gorki Tokens, including no promise any Gorki Networks will be launched, no promise of inherent value, no promise of any payments, and no guarantee that Gorki Tokens will hold any particular value.

Gorki Tokens are not designed and will not be structured or sold as securities. Gorki Tokens will hold no rights and confer no interests in the equity of the Gorki business or any future Gorki Networks. Gorki Tokens are designed and intended for future use on public Gorki Networks that may be created using the Gorki protocol, for the purposes of trading and governance transactions, or for the operation of a node. Proceeds of any sale of Gorki Tokens may be spent freely by Gorki for any purpose, including but not limited to the development of its business and underlying technological infrastructure, absent any conditions set out in this document.

This whitepaper is not a prospectus or disclosure document and is not an offer to sell, nor the solicitation of any offer to buy any investment or financial instrument or other product in any jurisdiction and should not be treated or relied upon as one. Any distribution of this whitepaper must be of the complete document including the cover page and this disclaimer and the accompanying boilerplate in their entirety.

All information in this document that is forward looking is speculative in nature and may change in response to numerous outside forces, including technological innovations, regulatory factors, and/or currency fluctuations, including but not limited to the market value of cryptocurrencies.

This whitepaper is for information purposes only and will be subject to change. The Gorki team cannot guarantee the accuracy of the statements made or conclusions reached in this whitepaper. The Gorki team does not make and expressly disclaims all representations and warranties (whether express or implied by statute or otherwise) whatsoever, including but not limited to: any representations or warranties relating to merchantability, fitness for a particular purpose, suitability, wage, title or non-infringement; that the contents of this document are accurate and free from any errors; and that such contents do not infringe any third party rights. The Gorki business, Gorki team, and operators of any Gorki Networks shall have no liability for damages of any kind arising out of the use, reference to or reliance on the contents of this whitepaper, even if advised of the possibility of such damages arising.

This whitepaper includes references to third party data and industry publications. The Gorki team believes that the information reproduced in this whitepaper is accurate and that the estimates and assumptions contained herein are reasonable. However, there are no assurances as to the accuracy or completeness of this data. The information from third party sources contained herein has been obtained from sources believed to be reliable; however, there are no assurances as to the accuracy or completeness of any included information. Although the data is believed to be reliable, the Gorki team has not independently verified any of the information or data from third party sources referred to in this whitepaper or ascertained the underlying assumptions relied upon by such sources.

Please note that Gorki is in the process of undertaking a legal and regulatory analysis of the functionality of the protocol, proposed Gorki Tokens, and the operation of its business. Following the conclusion of this analysis, the Gorki team may decide to amend the intended functionality of Gorki Tokens in order to ensure compliance with any legal or regulatory requirements to which it is subject, which may affect the utility, fungibility, or any other properties of Gorki Tokens.

Any Gorki Tokens could be impacted by regulatory action, including potential restrictions on the ownership, use, or possession of such tokens. Regulators or other competent authorities may demand that the mechanics of the Gorki Tokens be altered, entirely or in part. Gorki may revise the Gorki protocol or Gorki Token mechanics to comply with regulatory requirements or other governmental or business obligations. Nevertheless, Gorki believes it has taken all commercially reasonable steps to ensure that the design of Gorki Tokens is proper and in compliance with currently considered regulations as far as reasonably possible.

No regulatory authority has examined or approved any of the information set out in this whitepaper. The publication, distribution or dissemination of this whitepaper does not imply compliance with applicable laws or regulatory requirements.