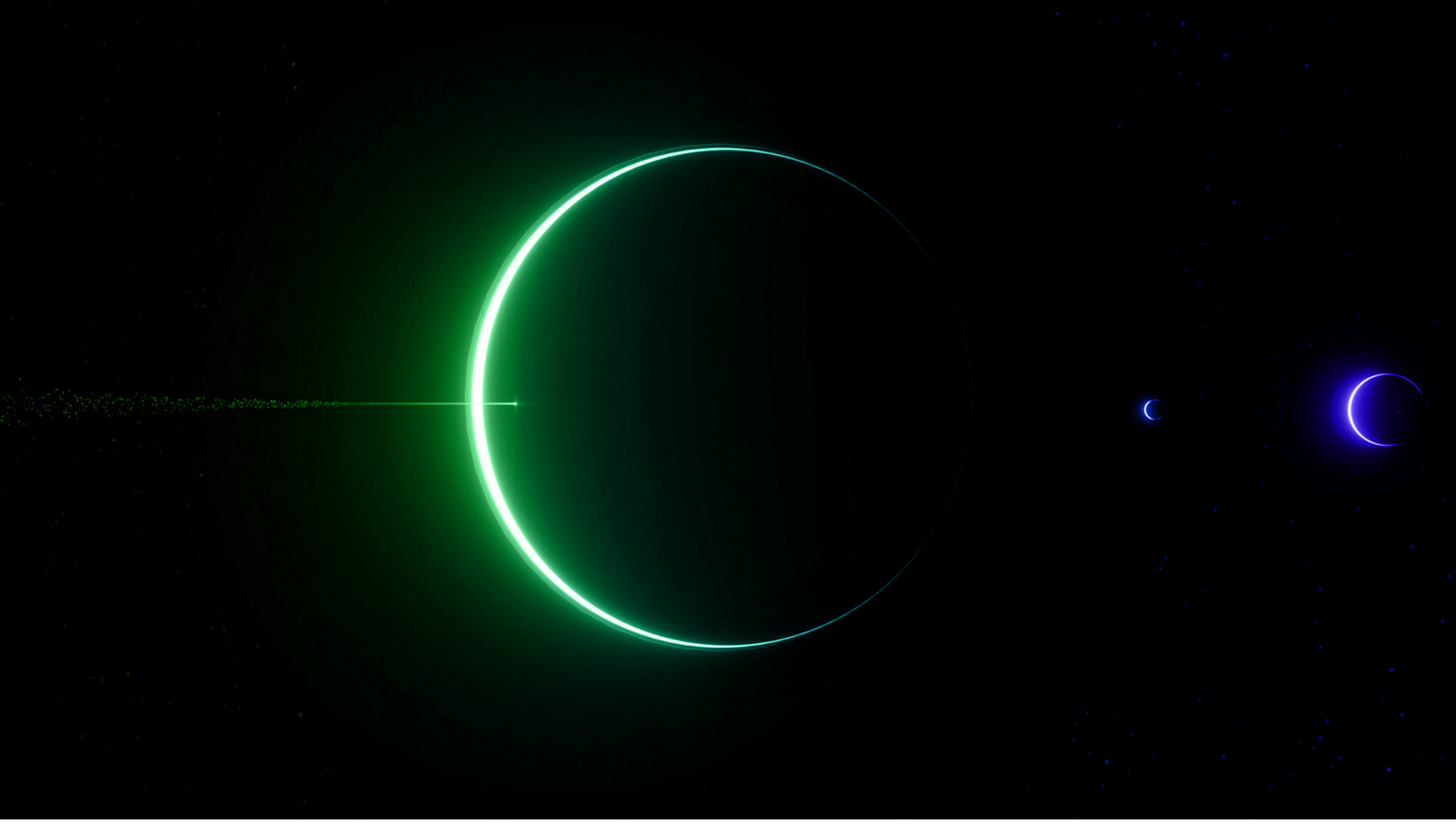
SKY

A Decentralized Video Delivery and Streaming Network Powered by a New Blockchain

SKY

White Paper
Version 1.0



Abstract

This whitepaper introduces the SKY Network, a new blockchain and token as the incentive mechanism for a decentralized video streaming and delivery network.
The SKY Network and protocol solves various challenges the video streaming industry faces today. First, tokens on the SKY blockchain are used as an incentive to encourage individual users to share their redundant computing and bandwidth resources as caching or relay nodes for video streams. This improves the quality of stream delivery and solves the " last-mile" delivery problem, the main bottleneck for traditional content delivery pipelines, especially for high resolution high bitrate 4k, 8k and next generation streams. Second, with sufficient network density the majority of viewers will pull streams from peering caching nodes, allowing video platforms to significantly reduce content delivery network (CDN) costs. More importantly, by introducing tokens as an end-user incentive mechanism the SKY Network allows video platforms to deepen viewer engagement, drive incremental revenues, and differentiate their content and viewing experience from their competitors.

The SKY blockchain introduces three main novel concepts:

- Multi-Level BFT: A modified BFT consensus mechanism which allows thousands of nodes to participate in the consensus process, while still supporting very high

transaction throughput (1,000+ TPS). The core idea is to have a small set of nodes, which form the validator committee, produce a chain of blocks as fast as possible using a PBFT-like process. Then, the thousands of consensus participants, called guardians, finalize the chain generated by the validator committee at regular checkpoint blocks. The name multi-level BFT consensus mechanism reflects the fact that the validator/guardian division provides multiple levels of security guarantee. The validator committee provides the first level of consensus — with 10 to 20 validators, the committee can come to consensus quickly. The guardian pool forms the second line of defense. With thousands of nodes, it is substantially more difficult for attackers to compromise the integrity of the network, and thus provides a much higher level of security. We believe this mechanism achieves a good balance among transaction throughput, consistency, and level of decentralization, the three pillars of the so-called "impossible triangle"

- Aggregated Signature Gossip Scheme: A basic all-to-all broadcasting of the checkpoint block hash could work between guardian nodes, but it yields quadratic communication overhead, and therefore cannot scale to 1,000+ nodes. Instead, we propose an Aggregated Signature Gossip Scheme which significantly reduces messaging complexity. Each guardian node keeps combining the partially aggregated signatures from all its neighbors, and then gossips out the aggregated signature. This way the signature share of each node can reach other nodes at an exponential rate, leveraging the gossip protocol. In addition, the signature aggregation keeps the size of the node-to-node messages small, and thus further reduces the communication overhead.

- Resource Oriented Micropayment Pool: An off-chain " Resource Oriented Micropayment Pool" that is purpose-built for video streaming. It allows a user to create an off-chain micropayment pool that any other user can withdraw from using off-chain transactions, and is double-spend resistant. It is much more flexible compared to off-chain payment channels.

This white paper will describe these concepts and the SKY blockchain in detail. The SKY Network launched with ERC20-compliant tokens and were integrated into the SLIVER.tv platform in December 2017. The SKY blockchain mainnet code has been released, and the first live mainnet implementation is planned to launch on March 15, 2019, at which time each ERC20 SKY token will be exchanged 1:1 for native SKY tokens.

TABLE OF CONTENTS

Vision

Introduction

Video Streaming Market

Live video streaming accounts for over two-thirds of all internet traffic today, and it is expected to jump to 82% by 2020, according to Cisco's June 2016 Visual Networking Index report.1 In the US, millennials between the ages of 18 and 34 are driving the growth of video streaming, and are heavy users of services like Netflix, Youtube, and HBO. Streaming video among this group has jumped 256% from an average of 1.6 hours per week to 5.7 hours per week according to a SSRS Media and Technology survey, and mobile devices are leading the charge in video consumption growing 44% in 2015 and 35% in 2016.2 The top five video streaming players in the US are Facebook, Google/Youtube, Twitter and related properties, Live.ly and Twitch.

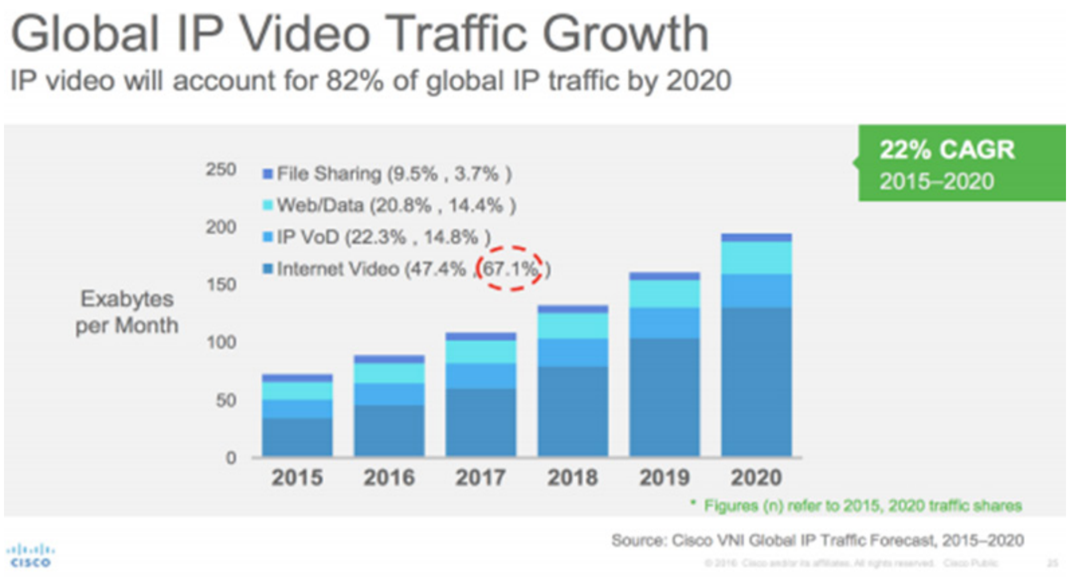


Figure 1. Global IP video traffic growth

At the same time, global virtual reality (VR) traffic including 360 ° video streaming content is estimated to grow 61-fold by 2020, at a staggering 127% CAGR according to the same Cisco report.

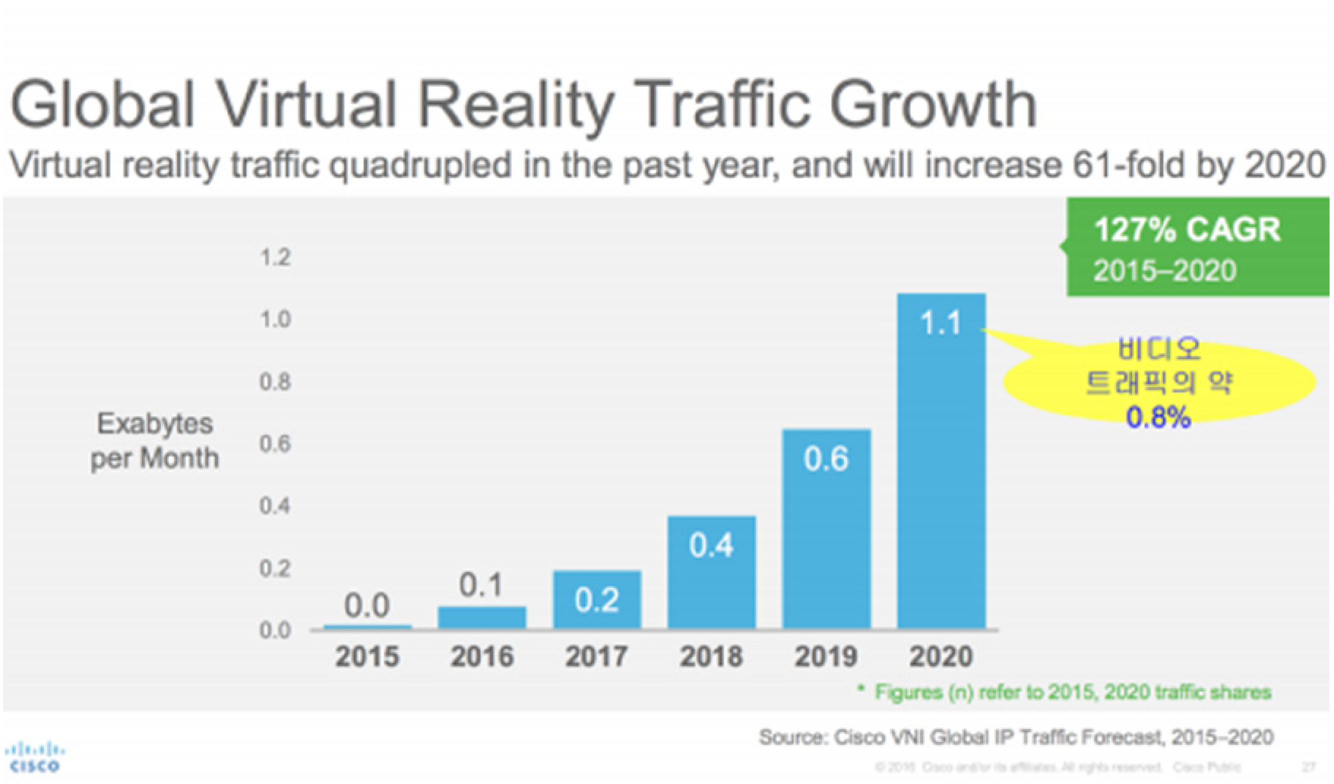


Figure 2. Global virtual reality traffic growth[3]

Video Streaming Challenges

Content Delivery Networks (CDN) play an important role in the video streaming ecosystem. It provides the backbone infrastructure to deliver the video streams to end viewers. One major limitation of today's CDN networks is the so-called "last-mile" delivery problem. Typically, CDN providers build data centers called Point-of-Presence (POPs) in many locations around the world, with the expectation that these POPs are geographically close to the viewers. However, the number of POPs are limited, hence are not near enough to the majority of viewers, especially in less developed regions. This " last-mile" link is usually the bottleneck of today's streaming delivery pipeline and often leads to less optimal user experience including choppy streams, bad picture quality, and frequent rebuffering.

To streaming sites and content platforms, another major concern is the CDN bandwidth cost. For popular sites, the CDN bandwidth cost can easily reach tens of millions of dollars per year. Even if platforms own proprietary CDNs, maintenance costs are often high.

These issues are becoming even more prominent with the coming era of 4K, 8k, 360 ° VR streaming, and upcoming technologies such as light field streaming. Table 1 compares the bandwidth requirements of today's mainstream 720p/HD streams vs 4K, 360 ° VR and future lightfield streams, quickly jumping by orders of magnitude.

Standard	Resolution	Bandwidth / Mbps	Magnitude
720p HD	1080x720	5 to 7.5	1x
1080p HD	1920x1080	8 to 12	1.6x
4K UHD	3920x2160	32 to 48	6.4x
8K 360 ° VR	7840x4320	128 to 192	25x
16K 360 ° VR	15680x8640	512 to 768	100x
Lightfield	---	5000+	1000x

Table 1. Bandwidth comparison: today's 720p/1080p video vs 4K and 360 ° VR streaming, vs future volumetric/lightfield streaming

To solve the VR and light field video delivery problem, the industry has started to explore "foveated streaming" technology. Instead of streaming the entire video in full resolution, this technology reduces the image quality in the peripheral vision (outside of the zone gazed by the fovea) in order to reduce bandwidth requirements. As the viewer turns his or her head to look at a different direction, the system adapts the spatial video resolution accordingly by fetching the high resolution video packets for the viewing direction from the server. For the foveated streaming technology to work well in practice, the round-trip time between the server and the viewer has to be small enough. For viewers that are geographically further from the CDN POPs, their VR stream viewing experience is compromised even with foveated streaming technology.

Background

SLIVER.tv (the "company") has been at the forefront of developing next-generation video streaming technologies for VR and spherical 360 ° video streams since 2015, and is the parent company to SKY Labs, Inc.. SLIVER.tv has raised over $17 Million in venture financing from notable Silicon Valley VCs including Danhua Capital, DCM, Sierra ventures, leading Hollywood media investors including Creative Artists Agency, BDMI, Advancit Capital, Greycroft Gaming Track Fund, and marquee corporate investors including GREE, Colopl, Samsung Next and Sony Innovation funds. Additionally, the company has strong Chinese investors and partners including Heuristic Capital Partners, ZP Capital, Green Pine Capital Partners, and Sparkland.
In a technology derived from "foveated streaming" SLIVER.tv's most recent technology patent granted #9,998,664, " METHODS AND SYSTEMS FOR NON-CONCENTRIC SPHERICAL PROJECTION FOR MULTI-RESOLUTION VIEW"4, specifically addresses the problem of generating highly efficient spherical videos for virtual reality (VR) streaming, highlight, and replay. The technology performs non-concentric spherical projection to derive high resolution displays of selected important game actions concurrently with lower resolution displays of static game environments, thus optimizing tradeoff between visual fidelity and data transfer load.

SLIVER.tv today is the leading next-generation live esports streaming platform with over six million unique visits in July 2018, with a vision to transform the esports engagement experience. As video gaming has grown in popularity to become a $40+ billion market, bigger than Hollywood and Bollywood combined, the rise of multiplayer competitive gaming as a spectator sport has become a major new industry, dubbed esports. Esports is a global phenomenon with major tournaments and major pockets of fans and competitive teams in Europe, Asia and North America. The online gaming and esports ecosystems have exploded over the past five years.
A recent 2017 SuperData research5 put the combined audience for gaming video content on YouTube and Twitch at 665 million, more than twice the US population. This surpasses the viewership of 227 million for HBO and Netflix combined. Today, esports and gaming video content account for a significant portion of all video content streamed over the Internet.
SLIVER.tv additional core patents and technology focus on various applications of cutting edge live streaming to esports content. The company's US Patent #9,573,062 " METHODS AND SYSTEMS FOR VIRTUAL REALITY STREAMING AND REPLAY OF COMPUTER VIDEO GAMES"6, US Patent #9,473,758 " METHODS AND SYSTEMS FOR GAME VIDEO RECORDING AND VIRTUAL REALITY REPLAY"7 and US Patent #9,782,678 " METHODS AND SYSTEMS FOR COMPUTER VIDEO GAME STREAMING, HIGHLIGHT, AND REPLAY"8 pioneer the capture and live rendering of popular PC esports games including League of Legends, Dota2 and Counter-Strike: Global Offensive in a fully immersive 360 ° VR spherical video stream, effectively placing the viewer and audience inside the 3D game through a live video stream, and rendering 360 ° highlights, replays and special effects in real-time.

Since launching in 2016, SLIVER.tv has broadcast numerous global esports tournaments in 360º VR in partnership with premier brands including ESL One, DreamHack and Intel Extreme Masters. At key events in the US and Europe, SLIVER.tv has live streamed top esports games to millions of fans of Counter-Strike: Global Offensive (CS:GO) and League of Legends (LoL).9

SLIVER.tv launched its Watch & Win esports platform in July 2017 and the first virtual token designed around esports content streaming and fan engagement. Since launch, the company has attracted millions of esports fans circulating over 1 Billion virtual tokens by actively participating and engaging with live esports matches. These users viewed over 50 million minutes of live esports streaming, nearly 100 years worth of content in the first few weeks of launch. This positions the company as one of the largest esports streaming sites built around a virtual community today.10

The SLIVER.tv platform is continuing to expand quickly driven by word-of-mouth, referral and social channels.

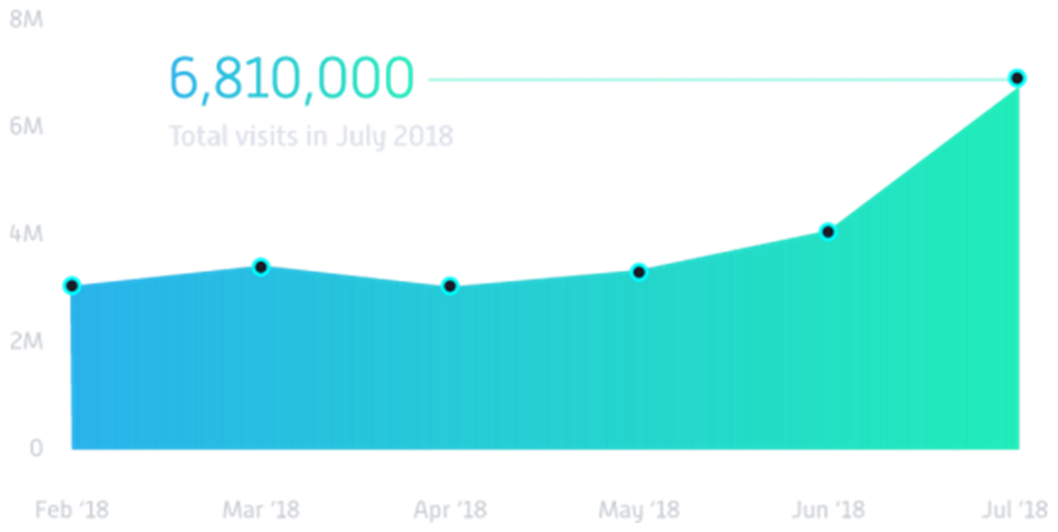


Figure 3. Total visits on desktop and mobile web, in the last 6 months

Opportunity

The company's mission is to leverage blockchain technology to create the first Decentralized Video Streaming and Delivery Network whereby video viewers are incentivized to share redundant computing and bandwidth resources to address today's video streaming challenges. Using the Ethereum EVM "World Computer" metaphor, the SKY Network can be viewed as the "World Cache" formed by the memory and bandwidth resources contributed by viewers.

Specifically, viewers around the globe can contribute their devices as "caching nodes" whereby they form a video delivery mesh network that is responsible for delivering any given video stream to viewers anywhere around the world optimized for local. The SKY Network can effectively address the technical challenges discussed in the previous section. First, viewers' devices are geographically much closer to each other than to the CDN POPs. This reduces packet round-trip time and improves the stream delivery quality, and thus addresses the " last-mile" delivery issue. Second, with a sufficient amount of caching nodes, most viewers will receive the stream from caching nodes, which will help streaming sites reduce their CDN bandwidth cost. Third, caching nodes also reduce round-trip time making foveated and next generation streaming technology practical.

To encourage viewers to contribute their computing and bandwidth resources, we introduce the SKY token as an incentive mechanism. Caching nodes can earn tokens as they relay video streams to other viewers. Not only does the SKY Token motivate viewers to join the network as caching nodes, it also greatly improves the streaming market efficiency by streamlining the video delivery process. We will discuss later in the paper, but within the SKY Network, advertisers can also directly target viewers at a lower cost, viewers earn SKY Tokens for their attention and engagement with their favorite content, and influencers earn SKY Token as gifts directly from viewers. More interestingly, streaming and content platforms can open up new and incremental revenue opportunities with SKY.

The full launch of the SKY protocol introduces a new blockchain and a native token structure where:

- Caching nodes earn tokens for caching and relaying video streams to other viewers
- Viewers optionally earn tokens from advertisers as engagement rewards, and can in turn gift to favorite influencers and content creators
- Streaming sites and platforms can drive incremental new revenues through sales of premium goods and services, and deepen user engagement through SKY
- Advertisers fund advertising campaigns with tokens to support influencers, streaming sites and viewers
- Streaming sites and platforms can offload up to 80% of CDN costs

The SKY protocol builds upon the following novel concepts:

- Multi-Level BFT: A modified BFT consensus mechanism which allows thousands of nodes to participate in the consensus process, while still supporting very high transaction throughput (1,000+ TPS). The core idea is to have a small set of nodes, which forms the validator committee, to produce a chain of blocks as fast as possible using a PBFT-like process. Then, the thousands of consensus participants, called the guardians, can finalize the chain generated by the validator committee at regular checkpoint blocks. The name multi-level BFT consensus mechanism reflects the fact that the validator/guardian division provides multiple levels of security guarantee. The validator committee provides the first level of protection — with 10 to 20 validators, the committee can come to consensus quickly. The guardian pool forms the second line of defense. With thousands of nodes, it is substantially more difficult for attackers to compromise, and thus provides a much higher level of security. We believe this mechanism achieves a good balance among transaction throughput, consistency, and level of decentralization, the three pillars of the so-called "impossible triangle"

- Aggregated Signature Gossip Scheme: A naive all-to-all broadcasting of the checkpoint block hash could work between guardian nodes, but it yields quadratic communication overhead, and so cannot scale to 1,000+ nodes. Instead we propose an Aggregated Signature Gossip Scheme which could significantly reduce messaging complexity. Each guardian node keeps combining the partially aggregated signatures from all its neighbors, and then gossips out the aggregated signature. This way the signature share of each node can reach other nodes at exponential speed thanks to the gossip protocol. In addition, the signature aggregation keeps the size of the node-to-node messages small, and thus further reduces the communication overhead.

- Resource Oriented Micropayment Pool: An off-chain " Resource Oriented Micropayment Pool" that is purpose-built for video streaming. It allows a user to create an off-chain micropayment pool that any other user can withdraw from using off-chain transactions, and is double-spend resistant. It is much more flexible compared to off-chain payment channels. In particular, for the video streaming use case, it allows a viewer to pay for video content pulled from multiple caching nodes without on-chain transactions. By replacing on-chain transactions with off-chain payments, the built-in " Resource Oriented Micropayment Pool" significantly improves the scalability of the blockchain.

SKY Mesh Delivery Network

Peer-to-peer streaming focuses on timely delivery of audio and video content under strict, near real-time parameters. Peer-to-peer livestream delivery works best when many people tune in for the same stream at the same time.   High concurrent user count means more peering resources are available, and thus the peer nodes can pull the stream from each other more effectively. The whole system capacity increases as more peer nodes become available. Moreover, robustness of the system is increased in a peer-to-peer network, as nodes do not need to rely on a centralized server to retrieve content. This is especially important in cases of server failure. In contrast, for centralized CDN-based delivery, high concurrent users instead place scalability pressures on the CDN servers.

However, the shortcoming of pure peer-to-peer streaming is availability. Peers come and go at anytime, which makes it difficult to predict the availability of any given peer node. There are also uncontrollable differences of nodes, such as upload and download capacities. On the other hand, a CDN service is more reliable and robust, and hence it can serve as a reliable "backup" when the stream is not available from peer nodes.
Our goal is to achieve maximum CDN bandwidth reduction without sacrificing the quality-of-service (QoS) which is critical to established streaming platforms such as Netflix, YouTube, Twitch, Facebook and others. This means whenever possible we want the peer nodes to pull the stream from each other instead of from the CDN. To achieve this goal, it's crucial for the peer nodes to be able to identify neighboring nodes effectively. If a node can identify multiple peers in close proximity, chances are that it can find peers that can provide the video stream segments much more consistently. On the contrary, if the identified peers are "further away" in terms of network hops, nodes might not be able to pull stream from peers consistently and cause degraded user experience like stuttering, frequent rebuffering, etc.

To address this problem, SKY has designed and is currently implementing a strategy which combines both a hyper-optimized tracker server and player client-side intelligence. Essentially, the tracker server provides high level guidance (e.g. a list of candidate peers) for the player client, while the player client implements a peer filtering algorithm at a finer granularity based on multiple variables to find the neighboring nodes that can best serve them.

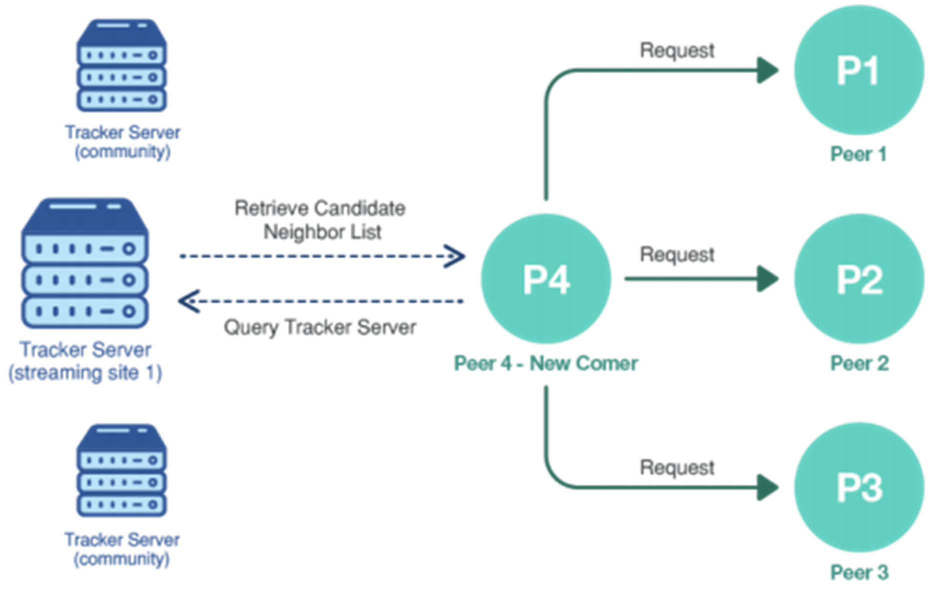

Tracker Server (community)
Tracker Server (streaming site 1)
Tracker Server (community)
Retrieve Candidate Neighbor List
Query Tracker Server
P4
Peer 4 - New Comer
Request
Request
Request
P1
Peer 1
P2
Peer 2
P3
Peer 3

Figure 4. Interactions between the tracker servers and player clients

Geo-Optimized Tracker Server

In order to provide a list of candidate peer nodes to each client, the tracker server records the location information whenever a peer joins the network, including its IP address, latitude/longitude, and a number of other performance parameters. With this information the server can organize the nodes in a spatial database. SKY's "hyper-optimized" spatial database is optimized for storing and querying data that represents objects defined in geometric space. As a peer node joins the network, the server can perform a spatial query to retrieve a list of candidate peers that are in the close proximity very quickly and efficiently, see Figure 4. The tracker servers and the spatial databases can be maintained by video streaming sites that use the SKY network and/or by community peers for content delivery.

As we mentioned earlier, a peer node might leave the network at anytime. Hence the tracker server also needs to be aware of which nodes are active. To achieve this, an active peer node needs to maintain a socket connection with the server and send heartbeat signals consistently. If the server doesn't receive a heartbeat for a certain amount of time, it considers that peer node as having left the network, and updates the spatial database accordingly.

An important distinction is that the "distance" between two peer nodes is measured by the number of router hops between them rather than the geographical distance. Typically network distance and geographical distance are highly correlated, but aren't necessarily equivalent. For example, two computers could sit next to each other physically, but connect to different ISPs so there might be many hops between them. Hence, aside from geographical information, the tracker server also utilizes the connectivity between the IP addresses collected in the past to analyze and select neighbor candidates. For example, candidates returned by the spatial query can go through another filter to exclude those that are not connected to the same ISP as the viewer's.

Intelligent Player Client

Each peer node may act both as a viewer, a caching node or both. As the node launches, during the handshake step, it retrieves a list of candidate peers from the tracker server for the livestream it's playing. Then, it performs a speed and availability test to select a subset that has optimized performance, connectivity and can reliably provide the video stream segments. The client performs the speed and availability tests regularly during a live stream session and continuously refines its neighbor list.

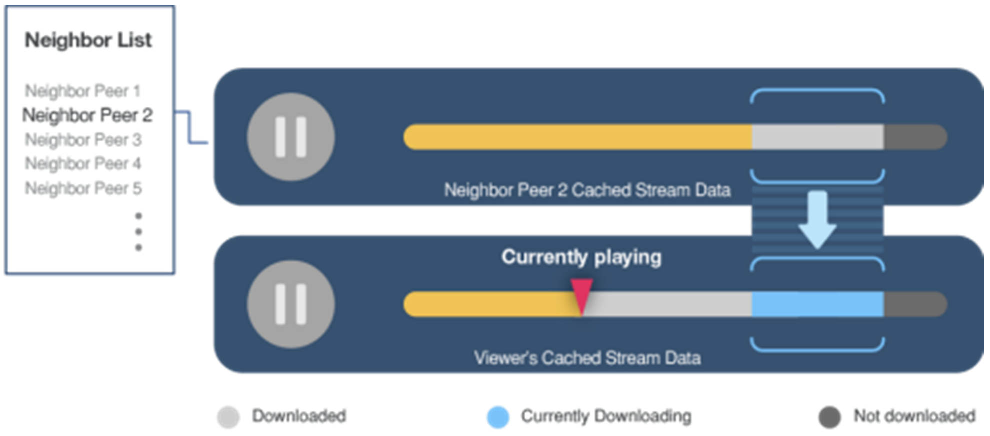


Figure 5. Player stream data buffer handling

To avoid QoS degradation, local buffer management is critical. The client player maintains a local cache to buffer the downloaded stream data as in Figure 5. If the duration of the cached stream data is less than a certain threshold, the player checks with the neighboring peers to see if they have the desired video stream segment. In the event when none of the neighbors has that segment, the player intelligently falls back to the CDN. To achieve the best QoS possible, the player fetches an updated candidate list from the tracker server on a regular basis during the stream session.

The first version of the client video player is a web/HTML5 based player which employs the WebRTC protocol for stream delivery among peers. Deploying web-based players requires minimal effort. Streaming sites and platforms simply embed the player onto their webpages, and it instantly has access and "launches" millions of end user nodes in the SKY mesh network. Thus, the deployment of SKY's mesh streaming technology is very light-weight and frictionless.

SKY also plans to release desktop and mobile client support. The advantage of a desktop client app over the web/HTML5 player is that it can run in the background to facilitate video stream relay (with the consent of the user) even when the end user is not watching any video streams. Further, SKY is investigating dedicated hardware, IOT devices, SmartTVs and related approaches that are specifically designed for stream relay and re-broadcast. Such devices can provide potentially better availability and bandwidth.

SKY Blockchain Ledger

The SKY Ledger is a decentralized ledger designed for the video streaming industry. It powers the SKY token ecosystem which incentives end users to share their redundant bandwidth and storage resources, and enables them to engage more actively with video platforms and content creators. To realize these goals, a number of challenges, many of which are unique for video streaming applications, need to be tackled.

One of such challenges is to support ultra high transaction throughput. Although many blockchain projects are facing transaction throughput problems, scaling for live video streaming is different and possibly even more complex. Typically, video segments are a couple of seconds long. To achieve the finest granularity of a token reward — one micropayment per video segment — even a live stream with a moderate ten thousand concurrent viewers could generate a couple of thousand microtransactions per second, which far exceeds the maximum throughput of today's public chains, such as Bitcoin and Ethereum. Popular live streams like major esport tournaments can attract more than one million viewers watching one stream simultaneously, not to mention multiple concurrent live streams, which could potentially push the required transaction throughput to the range of millions per second.

A byproduct of the high throughput is rapidly growing storage consumption. Storing the micropayment transactions is highly storage demanding. With tens of thousands of transactions added to the ledger every second, the storage space of an ordinary computer could run out quickly.

Video streaming applications typically require fast consensus. For bandwidth sharing rewards, the users that contribute redundant bandwidth typically want the payment to be confirmed before sending the next one. Other use cases, such as virtual gift donations to live stream hosts, also require short confirmation times to enable to real-time interactions between the hosts and audience.

Finally, as in any blockchain, security of the ledger is critical. Security is highly correlated with the level of decentralization. In a Proof-of-Stake (PoS) based consensus mechanism, decentralization means an even stake distribution among consensus participants. Ideally, the consensus mechanism should allow thousands of independent nodes, each with similar amounts of stake and each possessing a local copy of the blockchain, to participate in the block finalization process. To compromise such a system, a significant amount of independent nodes would need to be controlled by the attackers, which is difficult to achieve.

To achieve these goals, we have designed our PoS consensus algorithm based on the Byzantine Fault Tolerance (BFT) protocols, which offers good guarantees such as consistency (a.k.a. safety) when more than 2/3 of nodes running the ledger software are honest. However, the traditional BFT algorithms do not allow a high level of decentralization. They typically incur O(n2) messaging complexity even for the normal (non-faulty proposer) case, where n is the number of nodes participating in the consensus protocol. When we have thousands of nodes, it will take considerable amount of time to reach agreement. In this paper, we present a novel multi-level BFT consensus mechanism that allows mass participation, and still achieves 1000+ TPS throughput with the transaction confirmation time as short as a few seconds.

Such level of transaction throughput, although already much higher than Bitcoin and Ethereum, is still not sufficient to handle the micropayments for the “pay-per-byte” granularity. To further increase the throughput, the SKY Ledger provides native support for off-chain scaling, with a “resource oriented micropayment pool” which further amplifies the supportable throughput by several order of magnitudes.

We note that the off-chain payment support not only boosts the throughput, but also decreases the number of the transactions that need to be stored in the blockchain. On top of that, we introduce the technique of state and block history pruning to further reduce the storage space requirement. Moreover, we have adopted the microservice architecture for the storage system, which can adapt to different types of machines and storage backends, be it powerful server clusters running in data centers, or commodity desktop PCs.

The Consensus Mechanism

Multi-Level BFT

The SKY Ledger is built on a novel multi-level BFT consensus mechanism11 which allows thousands of nodes to participate in the consensus process, while still supporting very high transaction throughput (1000+ TPS).

The core idea is to have a small set of nodes, which forms the validator committee, produce a chain of blocks as fast as possible using a PBFT-like12 process. With a sufficient number of validators (e.g.

10 to 20), the validator committee can produce blocks at a fast speed, and still retain a high degree of difficulty to prevent an adversary from compromising the integrity of the blockchain. Hence, it is reasonable to expect that there is a very high probability the validators will produce a chain of blocks without forks. Then, the thousands of consensus participants, called guardians, can finalize the chain generated by the validator committee. Here "finalization" means to convince each honest guardian that more than 2/3 of all the other guardians see the same chain of blocks.

Since there are many more guardians than validators, it could a take longer time for the guardians to reach consensus than the validator committee. In order for the guardians to finalize the chain of blocks at the same speed that the validator committee produces new blocks, the guardian nodes can process the blocks at a much coarser grain. To be more specific, they only need to agree on the hash of the checkpoint blocks, i.e. blocks whose height are a multiple of some integer r (e.g. r = 100). This "leapfrogging" finalization strategy leverages the immutability characteristic of the blockchain data structure — as long as two guardian nodes agree on the hash of a block, with overwhelming probability, they will have exactly the same copy of the entire blockchain up to that block. Finalizing only the checkpoint blocks gives sufficient time for the thousands of guardians to reach consensus. Hence, with this strategy, the two independent processes, i.e., block production and finalization, can advance at the same pace.

Under the normal condition, finalizing a checkpoint block is similar to the "commit" step of the celebrated PBFT algorithm since each guardian has already stored the checkpoint block locally. Moreover, the checkpoint block has been signed by the validator committee, and hence it is highly likely that all the honest guardians have the same checkpoint. Thus, we only need a protocol for the honest guardians to confirm that indeed more than 2/3 of all guardians have the same checkpoint hash.

To implement this protocol, a naive all-to-all broadcasting of the checkpoint block hash could work, but it yields quadratic communication overhead, and so cannot scale to large numbers of guardians. Instead we propose an aggregated signature gossip scheme which could significantly reduce messaging complexity. The core idea is rather simple. Each guardian node keeps combining the partially aggregated signatures from all its neighbors, and then gossips out the aggregated signature, along with a compact bitmap which encodes the list of signers. This way the signature share of each node can reach other nodes at exponential speed utilizing the gossip protocol. Within O(log n) iterations, with high probability, all the honest guardian nodes should have a string which aggregates the signatures from all other honest nodes if there is no network partition. In addition, the signature aggregation keeps the size of the node-to-node messages small, and thus further reduces the communication overhead.

As mentioned above, the validator committee is comprised of a limited set of validator nodes, typically in the range of ten to twenty. They can be selected through an election process, or a randomized process, and may be subject to rotation to improve security. To be eligible to join the validator committee, a node needs to lock up a certain amount of stake for a period of time, which can be slashed if malicious behavior is detected. The blocks that the committee reaches consensus on are called settled blocks, and the process to settle the blocks is called the block settlement process.

The guardian pool is a superset of the validator committee, i.e. a validator is also a guardian. The pool contains a large number of nodes, which could be in the range of thousands. With a certain amount of tokens locked up for a period of time, any node in the network can instantly become a guardian. The guardians download and examine the chain of blocks generated by the validator committee and try to reach consensus on the the checkpoints with the above described "leapfrogging" approach. By allowing mass participation, we can greatly enhance the transaction security. The blocks that the guardian pool has reached consensus on are called finalized blocks, and the process to finalize the blocks is called the block finalization process.

The name multi-level BFT consensus mechanism reflects the fact that the validator/guardian division provides multiple levels of security guarantee. The validator committee provides the first level of protection — with 10 to 20 validators, the committee can come to consensus quickly. Yet it is resistant enough to attacks — in fact, it already provides a similar level of security compared to the DPoS mechanism if each validator nodes is run by an independent entity. Thus, a transaction can already be considered safe when it has been included in a settled block, especially for low stake transactions. The guardian pool forms the second line of defense. With thousands of nodes, it is substantially more difficult for attackers to compromise blockchain integrity, and thus provides a much higher level of security. In the unlikely event that the validator committee is fully controlled by attackers, the guardians can re-elect the validators, and the blockchain can restart, advancing from the most recent block finalized by the guardians. A transaction is considered irreversible when it is included in a finalized block. We believe this mechanism achieves a good balance among transaction throughput, consistency, and level of decentralization, the three corners of the so-called "impossible triangle"
The multi-level security scheme suits video streaming applications well. For streaming platforms, most of the transactions are micropayments (e.g. payment for peer bandwidth, virtual gifts to hosts, etc.) which typically have low value, but require fast confirmation. For such low stake payments, the users only need to wait for block settlement, which is very fast, in a matter of seconds. For high stake transfers, the user can wait longer until the block

containing the transaction is finalized, which could take slightly longer time, but is still in the range of minutes.

System Model

Before diving into the details of the block settlement and finalization process, we first list our assumptions of the system. For ease of discussion, without loss of generality, below we assume each node (be it a validator or a guardian) has the same amount of stake. Extending the algorithms to the general case where different nodes have different amount of stake is straightforward.

Validator committee failure model: There are m validator nodes in total. Most of the time, at most one-third of them are byzantine nodes. They might be fully controlled by attackers, but this happens only rarely. We also assume that between any pair of validator nodes there is a direct message channel (e.g. a direct TCP connection).

Guardian pool failure model: There are n guardian nodes in total. At any moment, at most one-third of them are byzantine nodes. We do not assume a direct message channel between any two guardians. Messages between them might need to be routed through other nodes, some of which could be byzantine nodes.

Timing model: We assume the "weak synchrony" model. To be more specific, the network can be asynchronous, or even partitioned for a bounded period of time. Between the asynchronous periods there are sufficiently long periods of time where all message transmissions between two honest nodes arrive within a known time bound $\triangle$. As we will discuss later in the paper, during the asynchronous period, the ledger simply stops producing new blocks. It will never produce conflicting blocks even with network partition. During synchronous phases, block production will naturally resume, and eventual liveness can be achieved.

Attacker model: We assume powerful attackers. They can corrupt a large number of targeted nodes, but no more than one-third of all the guardians simultaneously. They can manipulate the network at a large scale, and can even partition the network for a bounded period of time. Yet they are computationally bounded. They cannot forge fake signatures, and cannot invert cryptographic hashes.

The Block Settlement Process

Block settlement is the process in which the validator committee reaches agreement and produces a chain of blocks for the guardian pool to finalize. Inspired by recent Proof-of-Stake research works including Tendermint13, Casper FFG14, and Hot-Stuff15, we have designed and implemented the block settlement algorithm described below. It employs a rotating block proposer strategy where the validators take turns to propose new blocks. Then, the committee votes on the blocks to determine their order using a protocol similar to Casper FFG and Hot-Stuff.

Block Proposal

The validators rotate in a round robin fashion to play the role of block proposer, which is responsible for proposing the next block for the validator committee to vote on. To enable the round robin rotation, each proposer maintains a local logical clock called epoch. Assuming there are m validators, during epoch t, the validator with index (t mod m) is elected as the proposer for that epoch. We note it is important that:

1) The epoch t should not be stalled so the liveness of the proposer rotation is guaranteed; and
2) The epoch t of different validators should be mostly in sync, i.e. most of the time all the validators have the same t value, so they can agree on which node should produce the next block.

Below is our protocol for proposer election and block proposal.

Algorithm 1: Round Robin Block Proposal

```
$t \leftarrow 0$, $proposer \leftarrow 0$
$voted \leftarrow$ false, $received \leftarrow$ false, $timeout \leftarrow$ false

loop begin
  $proposer \leftarrow t \bmod m$
  if ($proposer == self.index$) and (not proposed yet) begin // node elected as the proposer
    propose one block
  end

  $voted \leftarrow$ the node has proposed or voted for a block for epoch $t$
  $received \leftarrow$ the node has received $m/3 + 1$ $EpochChange(t + 1)$ messages
  $timeout \leftarrow$ timeout reached
  if $voted$ or $received$ or $timeout$ begin
    broadcast message $EpochChange(t + 1)$
  end

  if the node has received $2m/3$ $EpochChange(t + 1)$ messages begin
    $t \leftarrow t + 1$ // enters epoch $t + 1$
    $voted \leftarrow$ false, $received \leftarrow$ false, $timeout \leftarrow$ false
  end

  sleep for some time
end
```

Algorithm 1. The round robin block proposal protocol

The protocol defines a message EpochChange(t + 1), which can be viewed as a synchronization message passed among the validators to assist them to advance to the next epoch t + 1 together. Essentially, a validator broadcasts message EpochChange(t + 1) to all other validators if any of the following conditions is met:

1) the node has proposed or voted for a block in epoch t, or
2) the node has received m/3 + 1 EpochChange(t + 1) messages from other validators, or
3) the node timed out for epoch t (the timeout is set to 4 △).

On the other hand, the validator enters epoch t + 1 when it has received 2m/3 EpochChange(t+1) messages from other nodes.

Here we show that this protocol meets the above two requirements.

Eventual Progression: All the honest nodes will eventually enter epoch t + 1. In the worst case, all the honest nodes (at least 2m/3 + 1 nodes) reach timeout and broadcast the Epoch-Change(t+1) messages. Under the timing model assumption, all these messages will be delivered within time $\triangle$ after being sent out. Thus each honest node will receive at least 2m/3 Epoch-Change(t + 1) messages, and it then enters epoch t + 1.

Epoch Synchrony: Intuitively, this means the epochs of all the honest nodes "move together" More precisely, we claim that the time any two honest nodes enter epoch t + 1 differ by at most most 2 $\triangle$. To prove this, we note that since there are at mostf faulty nodes, for the first honest node to enter epoch t + 1, at least m/3 other honest nodes must have broadcasted the EpochChange(t + 1) messages. This honest node then also broadcasts an EpochChange(t + 1) message following the protocol. After at most $\triangle$, any honest node should have received at least m/3 + 1 Epoch-Change(t + 1) messages, which triggers them to also broadcast the EpochChange(t + 1) message. After $\triangle$, all the honest nodes receive 2m/3 EpochChange(t + 1) messages and enter epoch t + 1. Thus, at most 2 $\triangle$ after the first honest node enters epoch t + 1, the last honest node will enters the same epoch.

In practice, when the network latency is small enough, all the honest nodes should enter epoch t + 1 at almost the same time. As a result, they can agree on who is the next proposer. Also we note that for the actual implementation, the EpochChange(t + 1) messages can be combined with other types of messages (e.g. block votes) to improve the efficiency. So that in the normal case (no proposer failure), no additional synchronization overhead is added to the system for epoch changes.

Block Consensus Among Validators

The protocol to settle proposed blocks involves a PBFT-based voting procedure among all validators, similar to Casper FFG and Hot-Stuff. In the SKY Ledger blockchain, the header of each block contains a hash pointer to its parent block (i.e. the previous block in the chain), similar to Bitcoin and Ethereum. Two blocks are conflicting if neither block is an ancestor of the other. If there are multiple, conflicting block proposals for the same epoch, an honest validator keeps all of them until one becomes settled, and then it discards all conflicting blocks.

The block settlement protocol operates epoch by epoch. The proposer for the current epoch sends to all validators a block proposal. A validator reacts by broadcasting a vote for the block. All messages are signed by their senders.

The header of the proposed block might carry a commit-certificate, which consists of at least (2m/3 + 1) signed votes for its parent block. We note that under the assumption that no more than m/3 validators are faulty, at most one block per height can obtain a commit-certificate. A commit-certificate for a block indicates this block and all its predecessors are committed. The proposed block may carry no commit-certificate, if its parent block did not get ≥ 2m/3 + 1 signed votes.

For the validators that are not the current proposer, their job is to vote on the proposed blocks. Once a validator receives the new block, it broadcasts a signed vote to all validators, so it can

be collected by the proposer of the next epoch to form the commit-certificate. If two consecutive blocks A and B both receive a commit-certificate, then the parent block A and all its predecessors are considered settled. To ensure safety, we require that honest nodes never vote for a block that conflicts with a settled block. When there are forks (either due to faulty proposer or asynchrony), the honest nodes should vote for the blocks on the longest fork.

The figure below illustrates the block settlement process. Assume that the proposer for height 101 is faulty, and it proposed two conflicting blocks X101 and Y101, which leads to two branches. Assuming neither block X101 nor Y101 gets ≥ 2m/3 + 1 votes, then, neither the header of X102 nor Y102 contains the commit-certificate (denoted by nil in the figure). However, at some point branch X grows faster, and two consecutive blocks X102 and X103 both obtain ≥ 2m/3 + 1 votes. After that the upper branch X up to block X102 is considered settled. And the lower branch Y can be discarded.

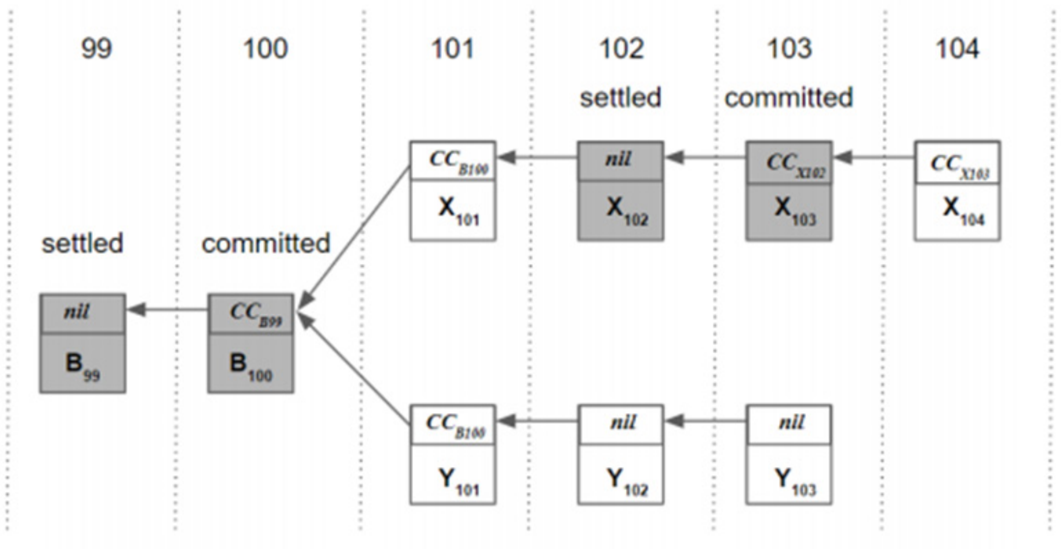


Figure 6. The block settlement process

The above example also illustrates one advantage of our implementation compared to other PBFT based protocols like Tendermint — a block that does not receive a commit-certificate can also be included in the settled chain, as long as one of its successor blocks is settled. For instance, block X101   in the example did not get a commit-certificate, but after block X102 is settled, it is also considered settled. This reduces the waste of computation power and helps increase the transaction throughput.

Analysis

Safety: Safety means all honest validators agree on the same chain of blocks. More precisely, if one honest validator accepts a block A, then any future blocks accepted by other honest validators will appear in a chain of blocks that already contains A. The argument for safety is similar to Casper FFG and Hot-Stuff and is omitted here. We just want to point out that safety stems from the requirement that honest nodes never vote for a block that conflicts with a settled block.

Liveness: Liveness means the validator committee always makes progress, i.e., always able to produce and agree on new blocks. Here we show that under our timing model, during the synchronous periods, the committee can always achieve the liveness goal. First, in the " Block Proposal" section, we have proved that the epoch can always advance, and all the honest

validators march forward together. In an epoch where the proposer is an honest validator, it will propose a new block. For the block settlement process, liveness depends on that during the synchronous periods, there are infinitely many epochs where two proposers in a row are honest, and wait sufficiently long to form the commit-certificate. We note this is guaranteed to happen infinitely often with the round robin rotation, since at least 2/3 of the validators are honest.

Transaction throughput: With ten to twenty validators, the committee can produce and settle the chain of blocks rather quickly. Average block production and settlement time is in the order of seconds, and this leads to high throughput of as much as 1000+ transactions per second.

The Block Finalization Process

In this section, we will discuss the "leapfrogging" block finalization process in detail. As mentioned above, the guardians only need to reach consensus on the hashes of the checkpoint blocks, which are the blocks whose heights are multiple of of some integer r (e.g. r = 100).

To see why it is sufficient to finalize just the checkpoint blocks, we note that the transaction execution engine of the blockchain software can be viewed as a "deterministic state machine", whereas a transaction can be viewed as a deterministic state transfer function. If two nodes run the same state machine, then from an identical initial state, after executing the same sequence of transactions, they will reach an identical end state. Note that this is true even when some of the transactions are invalid, as long as those transactions can be detected by the state machine and skipped. For example, assume there is a transaction that tries to spend more tokens than the balance of the source account. The state machine can simply skip this transaction after performing a sanity check. This way the "bad" transactions have no impact on the state.

In the context of blockchain, if all the honest nodes have the same copy of the blockchain, they can be ensured to arrive at the same end state after processing all the blocks in order. But with one caveat — the blockchain might contain a huge amount of data. How can two honest nodes compare whether they have the same chain of blocks efficiently?

Here the immutability characteristic of the blockchain data structure becomes highly relevant. Since the header of each block contains the hash of the previous block, as long as two nodes have the same hash of the checkpoint block, with overwhelming probability, they should have an identical chain of blocks from genesis up to the checkpoint. Of course each guardian node needs to verify the integrity of the blockchain. In particular, the block hash embedded in each block header is actually the hash of the previous block. We note that a node can perform the integrity checks on its own, no communication with other nodes is required.

Interestingly, the immutability characteristic also enhances the tolerance to network asynchrony or even partition. With network partition, the guardians may not be able to reach consensus on the hash of a checkpoint. However, after the network is recovered, they can move on to vote on the next checkpoint. If they can then reach agreement, then all the blocks up to the next checkpoint are finalized, regardless of whether or not they have consensus on the current checkpoint.

To provide byzantine fault tolerance, an honest node needs to be assured that at least two-thirds of the guardians have the same checkpoint block hash. Hence it needs to receive

signatures for a checkpoint hash from at least two-third of all guardians before the node can mark the checkpoint as finalized. This is to ensure safety, which is similar to the “commit” step in the celebrated PBFT protocol.

Since the guardians only need to vote on checkpoint hashes every T blocks, they have more time to reach consensus. A straightforward implementation of checkpoint finalization is thus to follow the PBFT “commit” step where each guardian broadcasts its signature to all other guardians. This requires each node to send, receive and process O(n) messages, where each message can be a couple kilobytes long. Even with T blocks time, this approach still cannot scale beyond a couple hundred guardian nodes, unless we select a large T value, which is undesirable since it increases the block finalization latency.

Scaling to Thousands of Guardians

To reduce the communication complexity and scale to thousands of guardians, we have designed an aggregated signature gossip scheme inspired by the BLS signature aggregation technique16 and the gossip protocol. The scheme requires each guardian node to process a much smaller number of messages to reach consensus, which is much more practical. Below are the steps of the aggregated signature gossip protocol. It uses the BLS algorithm for signature aggregation.

Algorithm 2: Aggregated Signature Gossip

```
$finalized \leftarrow$ false, $\sigma_i \leftarrow$ SignBLS($sk_i$, $height_{cp} \| hash_{cp}$), $c_i \leftarrow$ InitSignerVector($i$)
for $t = 1$ to $L$ begin
    send ($\sigma_i$, $c_i$) to all its neighboring guardians
    if $finalized$ break
    wait for ($\sigma_j$, $c_j$) from all neighbors until timeout
    verify each ($\sigma_j$, $c_j$), discard if it is invalid
    aggregate valid signatures $\sigma_i \leftarrow \sigma_i \cdot \prod_j \sigma_j$, $c_i \leftarrow \left(c_i + \sum_j c_j\right) \bmod p$
    calculate the number of unique signers $s \leftarrow \sum^{n} I\,(c_i[k] > 0)$
    if $s \geq \frac{2}{3}n$ $finalized \leftarrow$ true
end
```

Algorithm 2. The aggregated signature gossip protocol

The core idea is rather simple. Each guardian node keeps combining the partially aggregated signatures from its neighbors, and then gossips this newly aggregated signature out. This way the signature share of each node can reach other nodes at exponential speed by using the gossip protocol. In addition, the signature aggregation keeps the size of the messages small, and thus reduces the communication overhead.

In the above diagram, i is the index of the current guardian node. The first line of the protocol uses function SignBLS() to generate its initial aggregated signature σ_i. It essentially signs a message which is the concatenation of the height and hash of the checkpoint block using the BLS signature algorithm, with multiplicative cyclic group G of prime order p, and generator g:

$$h_i \leftarrow H\left(pk_i,\ height_{cp} \,\|\, hash_{cp}\right)$$

$$\sigma_i \leftarrow (h_i)^{sk_i}$$

In the first formula above, function $H : G \times \{0, 1\}^* \rightarrow G$ is a hash function that takes both the public key pk_i and the message as input. This is to prevent the rogue public-key attack17.

The protocol also uses function InitSignerVector() to initialize the signer vector c_i, which is an n dimensional integer vector whose jth entry represents how many times the jth guardian has signed the aggregated signature. After initialization, its ith entry is set to 1, and the remaining entries are all set to 0.
After initialization, the guardian enters a loop. In each iteration, the guardian first sends out its current aggregated signature σ_i and the signer vector c_i to all its neighbors. Then, if it has not considered the checkpoint as finalized, it waits for the signature and signer vector from all its neighbors, or waits until timeout. Upon receiving all the signature and signer vectors, it checks the validity of (σ_j, c_j) using the BLS aggregated signature verification algorithm.

$$h_u \leftarrow H(pk_u,\ height_{cp} \,\|\, hash_{cp})$$

$$\text{check if } e\left(\sigma_j, g\right) = \prod_u^n (e(h_u, pk_u))^{c_j[u]}$$

where $e : G \times G \rightarrow GT$ is a bilinear mapping function from $G \times G$ to GT, another multiplicative cyclic group also of prime order p. All the invalid signatures and their associated signer vectors are discarded for the next aggregation step. It is worth pointing out that besides $height_{cp}$, $hash_{cp}$, the above check also requires the public key pk_u of the relevant guardians as input. All this information should be available locally, since when a guardian locks up its stake, its public key should be attached to the stake locking transaction which has already been written into the blockchain. Hence, no communication with other nodes is necessary to retrieve these inputs.

The aggregation step aggregates the BLS signature σj, and updates the signer vector cj . Note that for the vector update, we take mod p for each entry. We can do this because e (hu, pku) ∈ GT , which is a multiplicative cyclic group of prime order p. This guarantees that the entries of vector cj can always be represented with a limited number of bits.

$$\sigma_i \leftarrow \sigma_i \cdot \prod_j \sigma_j \,,\; c_i \leftarrow \left(c_i + \sum_j c_j\right) mod\, p$$

The algorithm then calculates the number of unique signers of the aggregated signature.

$$s \leftarrow \sum^{n} I\,(c_i\,[k] > 0)$$

Here function I: {true, false} → {1, 0} maps a true condition to 1, and false to 0. Hence the summation counts how many unique signers have contributed to the aggregated signature. If the signature is signed by more than two-third of all the guardians, the guardian considers the checkpoint to be finalized.

If the checkpoint is finalized, the aggregated signature will be gossipped out in the next iteration. Hence within O(log(n)) iterations all the honest guardians will have an aggregated signature that is signed by more than two-third of all the guardians if the network is not partitioned.
The loop has L iterations, L should be in the order of O(log(n)) to allow the signature to propagate through the network.

Analysis

Aggregated Signature Gossip Correctness: To prove the correctness of the aggregated signature gossip protocol, we need to prove two claims. First, if an aggregated signature is correctly formed by honest nodes according to Algorithm 2, it can pass the check given by Formula (4). Second, the aggregated signature is secure against forgery. Stated more formally, forging a fake aggregated signature in the context of Algorithm 2 means to find σ ∈ G and integers c1, c2, … cn which satisfy the equation below

$$e\,(\sigma,\; g) = \prod_{u=1}^{n} (e\,(h_u,\; pk_u))^{c_u}$$

for randomly chosen $pk_1 = g^{sk_1}, \ldots, pk_n = g^{sk_n} \in G$, and random message hashes $h_1 \ldots, h_n \in G$. It can be shown that this is as hard as the Computational Diffie-Hellman (CDH) problem. For the proof of these two claims, please refer to our multi-level BFT technical report.
Finalization Safety: Safety of the block finalization is easy to prove. Under the 2/3 supermajority honesty assumption, If two checkpoint hashes for the same height both get aggregated signatures from at least 2/3 of all guardians, at least one honest guardian has to sign different hashes for the same height, which is not possible.

Finalization Liveness: Without network partition, as long as L is large enough, it is highly likely that after O(log(n)) iteration, all the honest nodes will see an aggregated signature that combines the signatures of all honest signers. This is similar to how the gossip protocol can robustly spread a message throughout the network in O(log(n)) time, even with up to 1/3 byzantine nodes. When there is network partition, consensus for a checkpoint may not be able to reach finalization. However, after the network partition is over, the guardian pool can proceed to finalize the next checkpoint block. If consensus can then be reached, all the blocks up to the next checkpoint are considered finalized. Hence the finalization process will progress eventually.

Messaging Complexity: The aggregated signature gossip protocol runs for L iterations, which is in the order of O(log(n)). In each iteration, the guardian needs to send message (σ_i , c_i) to all its neighboring guardians. Depending on the network topology, typically it is reasonable to assume that for an average node, the number of neighboring nodes is a constant (i.e. the number of neighbors does not grow as the total number of nodes grows). Hence the number of message a node needs to send/receive to finalize a checkpoint is in the order of O(log(n)), which is much better than the O(n) complexity in the naive all-to-all signature broadcasting

implementation. We do acknowledge that each message between two neighboring guardians contains an n dimensional signer vector c_i, where each entry of c_i is an integer smaller than prime p. However, we note that this vector can be represented rather compactly since most of its entries are small integers (p) in practice.
To get a more concrete idea of the messaging complexity, let us work out an example. Assume that we pick a 170-bit long prime number p for the BLS signature, which can provide security comparable to that of a 1024-bit RSA signature. And there are 1000 guardians in total. Under this setting, c_i can be represented with about twenty kilobytes without any compression. Since most of the entries of c_i are far smaller than p, c_i can be compressed very effectively to a couple kilobytes long. Plus the aggregated signature, the size of each message is typically in the kilobytes range. Moreover, if we assume on average an guardian connects to 20 other guardians, then L can be as small as 5 (more than twice of $\log_{20}(1000) = 2.3$). This means finalizing one checkpoint just requires a guardian to send/receive around 100 messages to/from its neighbors, each about a couple kilobytes long. This renders the aggregated signature gossip protocol rather practical to implement and can easily scale to thousands of guardian nodes. For further analysis, please also refer to our multi-level BFT technical report.

Reward and Penalty for Validators and Guardians

The token reward and penalty structure is essential to encourage nodes to participate in the consensus process, and not to deviate from the protocol.

Both the validators and guardians can obtain a token reward. Each block includes a special Coinbase transaction that deposits newly minted tokens to the validator and guardian addresses. All the validators can get a share of tokens for each block. For guardians, rewarding every guardian for each block might not be practical since their number is large. Instead, we propose the following algorithm to randomly pick a limited number of guardians as the reward recipient for each block. Denote the height of the newly proposed block by l, and cp is the most recently finalized checkpoint. The proposer should have received the aggregated signature σcp and corresponding signer vector ccp for checkpoint cp. Upon validating (σcp , ccp) , the proposer can check the following condition for each guardian whose corresponding entry in vector ccp is not zero (i.e. that guardian signed the checkpoint)

$$H\left(pk_i,\ \sigma_{cp} \,||\, B_{l-1}\right) \leq \tau$$

where Bl−1 is the hash of the block with height l − 1 , and H : G × {0 , 1}* → G is the same hash function used in the BLS signature algorithm. If the inequality holds, the proposer adds the guardian with public key pki to the Coinbase transaction recipient list. Threshold τ is chosen properly such that only a small number of guardians are included. The proposer should also attach (σcp , ccp) to the Coinbase transaction as the proof for the reward.

The SKY ledger also enforces a token penalty should any malicious behavior be detected. In particular, if a block proposer signs conflicting blocks for the same height, or if a validator votes for different blocks of the same height, they should be penalized. Earlier we mentioned that to become either a validator or a guardian, a node needs to lock up a certain amount of tokens for a period of time. The penalty will be deducted from their locked tokens. The node that detects the malicious behavior can submit a special Slash transaction to the blockchain. The proof of

the malicious behavior (e.g. signatures for conflicting blocks) should be attached to the Slash transaction. The penalty tokens will be pulled from the malicious node and awarded to the node that submitted the first Slash transaction.

In the unlikely event that more than one-third of the validators are compromised, the malicious validators can attempt to perform the double spending attack by forking the blockchain from a block that is settled but not yet finalized. However, this is detectable by the guardian pool, since forking will generate multiple blocks with the same height, but signed by more than two-third of the validators. In this case, the validators that conducted the double signing will be penalized, and the entire validator committee will be re-elected. After the validator committee is reinstated, the blockchain can continue to advance from the most recent finalized checkpoint.

Turing-Complete Smart Contract Support

This SKY Ledger offers a smart contract runtime environment fully compatible with the Ethereum Virtual Machine18. It provides full-fledged support for Turing-Complete smart contracts. Solidity-based Ethereum smart contracts can be ported to the SKY Ledger with little effort. Solidity19 has grown a large developer community and the prospect of allowing that proven talent pool to also contribute to SKY without reinventing the wheel was a prime consideration in enabling compatibility with the Ethereum Virtual Machine.

Smart contracts enable rich user experiences and new attribution models for video platform DApps built on the SKY Ledger. For example, video platforms can write smart contracts for loyalty programs to engage users. Based on users' activity, or the volume of video segments / data they have relayed, platform DApps may promote users to a higher tier, which unlocks certain privileges or exclusive capabilities. As another example, video platforms can issue virtual items backed by the ledger blockchain (e.g. a virtual rose) for gifting to their favorite content creators. To expand on such a concept, built on the "non-fungible token" standard, the virtual items could be rare or entirely unique, such that they are essentially "crypto collectibles", which can be kept as trophies or traded for other sought after collectibles, all without additional permissions from 3rd parties.

Moreover, video platforms are able to write smart contracts that enable more fluid payment-consumption models, such as pay-as-you-go or per-use models. Instead of traditional annual or monthly subscriptions, user consumption can be priced at a bite-sized granularity, such that users only need to pay for what they use. This is a feasible way to allow low-priced, short-form content to be transacted in an economically sensible way, that accrues benefits to both the video platform and user. SKY Ledger's properties of tracking micropayments and video segments enables such smart contracts to be executed.
Smart contracts can also be designed to the benefit of content creators (e.g. user-generated content producers, larger production studios) as a way to fairly and transparently distribute royalties. The traditional royalty settlement processes, with all their complexities and obscurities, can be accommodated with clear smart contract terms that are mutually agreed upon by creators and distributors - and made available to users that consume the content.

Leveraging smart contracts on the SKY Ledger to enable fully digitized item ownership, innovative payment-consumption models, and transparent royalty distributions provide an additional layer of social and economic interactivity that supplements the core functionality of video/content delivery.

Off-Chain Micropayment Support

As discussed in the introduction section, support for high transaction throughput is a must for a video streaming focused blockchain. We build the support for off-chain payment directly into the ledger to facilitate high volumes of transactions.

Resource Oriented Micropayment Pool

We have designed and implemented an off-chain "Resource Oriented Micropayment Pool" that is purpose-built for video streaming. It allows a user to create an off-chain micropayment pool that any other user can withdraw from using off-chain transactions, and is double-spend resistant. It is much more flexible compared to off-chain payment channels. In particular, for the video streaming use case, it allows a viewer to pay for video content pulled from multiple caching nodes without on-chain transactions. By replacing on-chain transactions with off-chain payments, the built-in " Resource Oriented Micropayment Pool" significantly improves the scalability of the blockchain.

The following scenario and diagram provide a comprehensive walkthrough of how the Resource Oriented Micropayment Pool works in application.

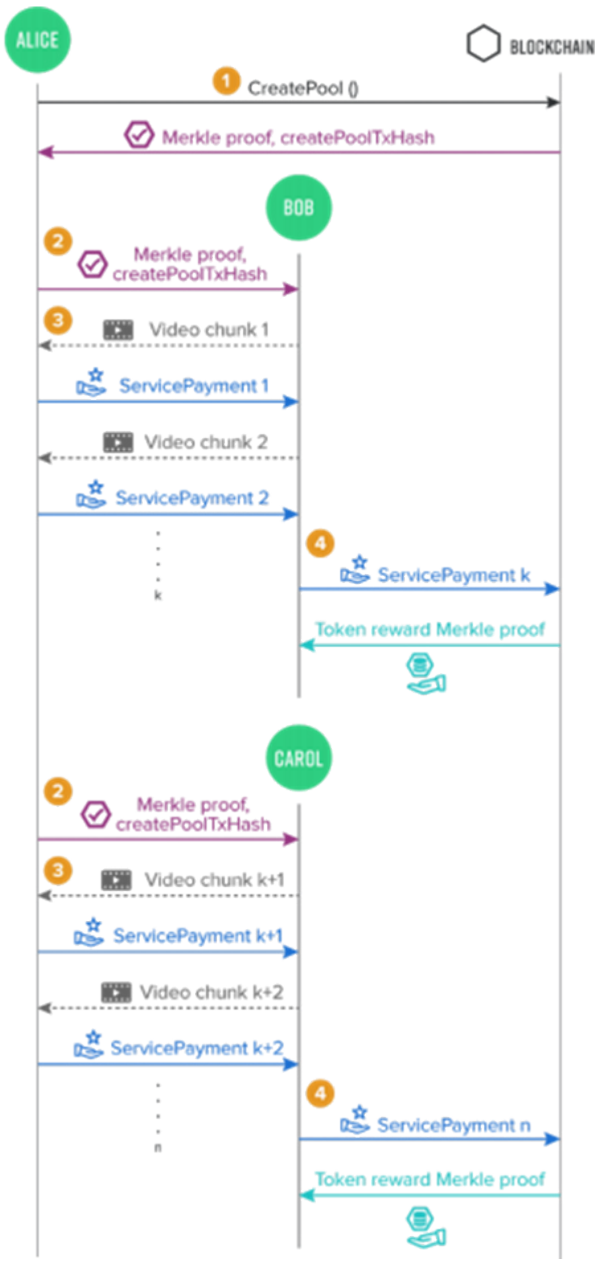


Figure 7. Resource Oriented Micropayment Pool shows viewer Alice making off-chain transactions to cachers Bob and Carol for video chunks

- Step 1. Micropayment pool creation: As the first step, Alice publishes an on-chain transaction to create a micropayment pool with a time-lock and a slashable collateral.

  CreatePool(resourceId, deposit, collateral, duration)

  A couple things to be noted. To create the pool, Alice needs to specify the " Resource ID" resourceId that uniquely represents the digital content she intends to retrieve. It may refer to a video file, or a live stream.
  The deposit amount needs to be at least the total value of the resource to be retrieved. For instance, if the resource is a video file which is worth 10 tokens, then the deposit has to be at least 10 tokens.

  The collateral is required to discourage Alice from double spending. If a double spending attempt from Alice is detected by the validators of the blockchain, the collateral will be slashed. Later in the blogpost we will show that if collateral > deposit, the net return of a double spend is always negative, and hence any rational user will have no incentive to double spend.

  The duration is a time-lock similar to that of a standard payment channel. Any withdrawal from the payment pool has to be before the time-lock expires.
  The blockchain returns Alice the Merkle proof of the CreatePool transaction after it has been committed to the blockchain, as well as createPoolTxHash, the transaction hash of the Create-Pool transaction.

- Step 2. Initial handshake between peers: Whenever Alice wants to retrieve the specified resource from a peer (Bob, Carol, or David, etc.). She sends the Merkle proof of the on-chain CreatePool transaction to that peer. The recipient peer verifies the Merkle proof to ensure that the pool has sufficient deposit and collateral for the requested resource, and both parties can proceed to the next steps.

- Step 3. Off-chain micropayments: Alice signs ServicePayment transactions and sends them to the peers off-chain in exchange for parts of the specified resource (e.g. a piece of the video file, a live stream segment, etc.). The ServicePayment transaction contains the following data:

  targetAddress, transferAmount, createPoolTxHash, targetSettlementSequence,
  Sign(SKA, targetAddress || transferAmount || createPoolTxHash ||
  targetSettlementSequence)

  The targetAddress is the address of the peer that Alice retrieves the resource from, and the transferAmount is the amount of token payment Alice intends to send. The targetSettle-mentSequence is to prevent a replay attack. It is similar to the "nonce" parameter in an Ethe-reum transaction. If a target publishes a ServicePayment transaction to the blockchain (see the next step), its targetSettlementSequence needs to increment by one.

  The recipient peer needs to verify the off-chain transactions and the signatures. Upon validation, the peer can send Alice the resource specified by the CreatePool transaction.

Also, we note that the off-chain ServicePayment transactions are sent directly between two peers. Hence there is no scalability bottleneck for this step.

- Step 4. On-chain settlement: Any peer (i.e. Bob, Carol, or David, etc) that received the ServicePayment transactions from Alice can publish the signed transactions to the blockchain anytime before the timelock expires to withdraw the tokens. We call the ServicePayment transactions that are published the "on-chain settlement" transactions.

  Note that the recipient peers needs to pay for the gas fee for the on-chain settlement transaction. To pay less transaction fees, they would have the incentive to publish on-chain settlements only when necessary, which is beneficial to the scalability of the network.

We note that no on-chain transaction is needed when Alice switches from one peer to another to retrieve the resource. In the video streaming context, this means the viewer can switch to any caching node at any time without making an on-chain transaction that could potentially block or delay the video stream delivery. As shown in the figure, in the event that Bob leaves, Alice can switch to Carol after receiving k chunks from Bob, and keep receiving video segments without an on-chain transaction.
Moreover, the total amount of tokens needed to create the micropayment pool is (collateral + deposit), which can be as low as twice of the value of the requested resource, no matter how many peers Alice retrieves the resource from. Using computational complexity language, the amount of reserved token reduces from O(n) to O(1) compared to the unidirectional payment channel approach, where n is the number of peers Alice retrieves the resource from.

Double Spending Detection and Penalty Analysis

To prevent Alice, the creator of the micropayment pool from double spending, we need to 1) be able to detect double spending, and 2) ensure that the net value Alice gains from double spending is strictly negative.

Detecting double spending is relatively straightforward. The validators of the SKY Network check every on-chain transaction. If a remaining deposit in the micropayment pool cannot cover the next consolidated payment transaction signed by both Alice and another peer, the validators will consider that Alice has conducted a double spend.

Next, we need to make Alice worse off if she double spends. This is where the collateral comes in. Earlier, we mentioned that the amount of collateral tokens has to be larger than the deposit. And here is why.

In Figure 8 below, Bob, Carol, and David are honest. Alice is malicious. Even worse, she colludes with another malicious peer Edward. Alice exchanges partially signed transactions with Bob, Carol, and David for the specified resource. Since Alice gains no extra value for the duplicated resource, the maximum value she gets from Bob, Carol, and David is at most the deposit amount. As Alice colludes with Edward, she can send Edward the full deposit amount. She then asks Edward to commit the settlement transaction before anyone else and return her the deposit later. In other words, Alice gets the resource which is worth at most the deposit amount for free, before the double spending is detected. Later when Bob, Carol, or David commit the settlement transaction, the double spending is detected, and the full collateral amount will be slashed. Hence, the net return for Alice is

$$net_{Alice} = deposit - collateral$$

Therefore, we can conclude that for this scenario, as long as collateral > deposit, Alice's net return is negative. Hence, if Alice is rational, she would not have any incentive to double spend.

We can conduct similar analysis for other cases. The details are omitted here, but it can be shown that in all cases Alice's net return is always negative if she conducts a double spend.

Another case is that Alice is honest, but some of her peers are malicious. After Alice sends a micropayment to one of those peers, it might not return Alice the resource she wants. In this case, Alice can turn to another peer to get the resource. Since each incremental micropayment can be infinitesimally small in theory, Alice's loss can be made arbitrarily small.

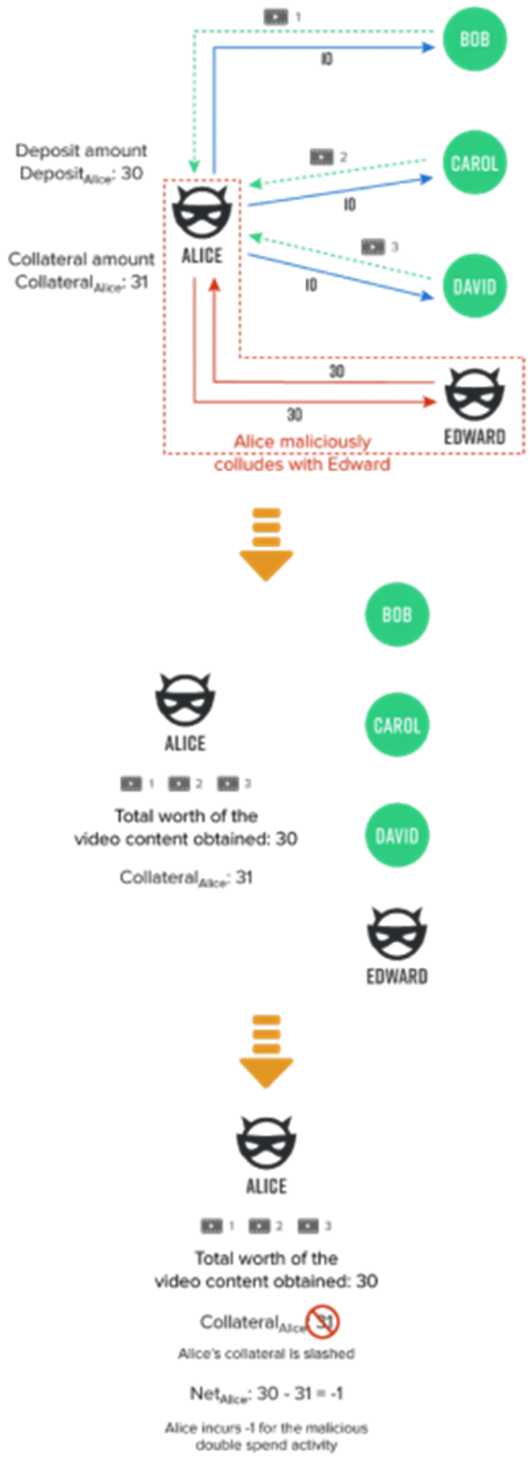

1
BOB
10
Deposit amount
Deposit$_{Alice}$: 30
2
CAROL
10
Collateral amount
Collateral$_{Alice}$: 31
ALICE
3
DAVID
10
30
30
EDWARD
Alice maliciously colludes with Edward
BOB
ALICE
CAROL
1 2 3
Total worth of the video content obtained: 30
Collateral$_{Alice}$: 31
DAVID
EDWARD
ALICE
1 2 3
Total worth of the video content obtained: 30
Collateral$_{Alice}$: 31
Alice's collateral is slashed
Net$_{Alice}$: 30 - 31 = -1
Alice incurs -1 for the malicious double spend activity

Figure 8. Malicious Actor Detection and Penalty shows malicious actor Alice attempting to
make a double spend and the resulting penalty she receives

Ledger Storage System

Using a public ledger to facilitate the micropayments for streaming is challenging, not only because of high transaction throughput, but also for storage space management. To achieve the “pay-per-byte” granularity, each viewer could send out a payment every few seconds. With even a moderate ten thousand concurrent users, it could generate a couple thousands of transactions per second. Even with the off-chain payment pool which already reduces the amount of on-chain transactions dramatically, the block and state data could still balloon rather quickly.

We have designed a storage system that addresses this problem, and can adapt to different types of machines, be it a powerful server cluster running in data centers, or a commodity desktop PC.

Storage Microservice Architecture

To harness the processing and storage power of server clusters, the key design decision is to adopt the popular microservice architecture commonly seen for modern web service backends, where different modules of the ledger can be configured to run on different machines. In particular, the consensus module and the storage module can be separated. Potentially the consensus module can run on multiple machines using the MapReduce framework to process transactions in parallel.

The SKY Ledger stores both the transaction blocks and the account state history, similar to Ethereum. The bottom layer of the storage module is a key value store. The SKY Ledger implements the interfaces for multiple databases, ranging from single machine LevelDB to cloud based NoSQL database such as MongoDB, which can store virtually unlimited amount of data. Thus the ledger can run on one single computer, and can also be configured to run on server clusters.

History Pruning

While the microservice architecture suits the powerful server clusters well, we still face storage space constraints when running the ledger on a lower-end home PC. We have designed several techniques to reduce the storage consumption.

Similar to Ethereum, the SKY Ledger stores the entire state for each block, and the state tree root is saved in the header of the corresponding block. To reduce the space consumed by the state history, the SKY Ledger implements state history pruning, which leverages a technique called reference counting illustrated in the figure below.

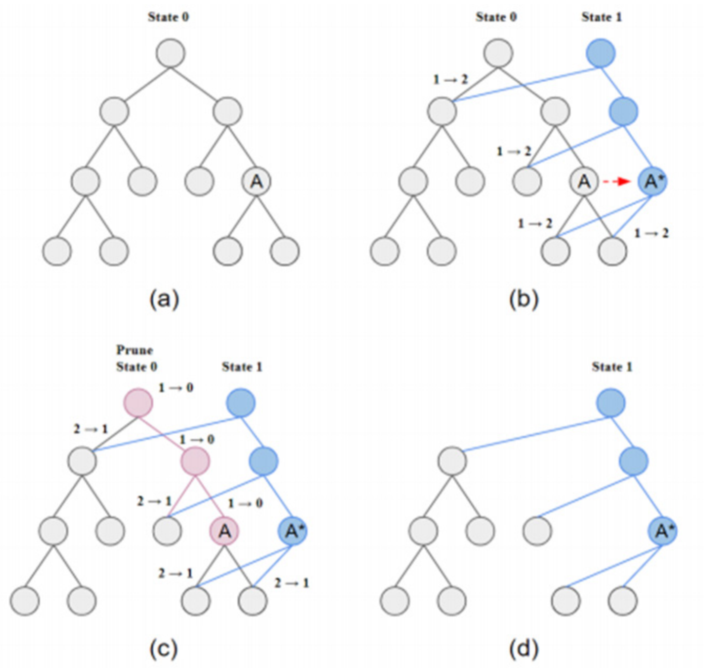


Figure 9. State history pruning with reference counting

The ledger state (i.e. the token balance of each account, etc.) is stored using a Merkle-Patricia trie. Figure 9(a) depicts the initial state tree, whose root is denoted by State 0. Each node in the tree has an attribute called the "reference count", which is equal to the number of parents of the node. In the initial state tree, each node has only one parent, so the reference count are all set to 1.
In Figure 9(b), account A is updated to A* after applying the transactions in the newly settled block. Hence a new Merkle state root State 1 is created, along with the Merkle branch connecting the new root State 1 and A* (the blue nodes and edges). Since new nodes are added, we update the reference count of direct children of these new nodes from 1 to 2.
At some point we decided to delete State 0 to save some storage space. This is done by deleting the nodes whose reference count is zero recursively starting from the root State 0, until no node can be deleted. Whenever a node is deleted, the reference count of all its children will be decremented by one. Figure 9(c) illustrates the process, and Figure 9(d) shows the result of the pruning. To achieve the maximum level to state storage compaction, once a block is finalized by the guardian pool, we can delete all the history prior to that block. The ledger can also be configured to keep a limited history of states, for example, the state trees of the latest 1000 blocks, depending on the available storage space.

It can be shown that with the reference counting technique, pruning a state tree has the time complexity of O(k log N), where k is the number of accounts updated by the transactions in one block, and N is the total number of accounts. Typically, k is in the range of a couple hundreds to a thousand. Hence, pruning a state tree should be pretty efficient and should not take much time.

Managing the space consumed by the transaction blocks is even simpler, after a block is finalized, we can simply delete all its previous blocks, or keep a limited history similar to the state trees.

With these techniques, common PCs and laptops are sufficient to run the guardian nodes.

State Synchronization

One of the pain points using earlier generation blockchains is the state synchronization time. After spinning up a new node, typically it needs to download the full block history all the way from the genesis block. This could take days to complete, and already becomes a hurdle for user adoption. The state and block history stored by the full nodes can help reduce the synchronization time dramatically. After a new node start, the first step is to download all the validator and guardian join/leave transactions and the headers of the blocks that contain these special transaction up to the latest finalized block. With these special transactions and the headers which contain the validator and guardian signatures, the new node can derive the current validator committee and guardian pool. Since the validator and guardian set changes are relatively infrequent, the amount of data need to be downloaded and verified for this step should be minimal.
In the second step, the new node downloads the state tree corresponding to the latest finalized block. And it needs to confirm that the root hash of the tree equals the state hash stored in the latest finalized block. Finally, the new node verifies the integrity of the state tree (e.g. the validity of the Merkle branches). If all the checks are passed, the new node can start listening to new blocks and start participating in the consensus process.

A Dual Currency System and Token Mechanics

In the interest of securing the network, installing proper governance, and managing the usage of the network, the SKY blockchain will use a dual currency system. The SKY token will be used to stake, secure, and govern the SKY Network, while individual operations (video segment transactions, smart contract operations, etc.) will be paid for with the operational token, Gamma.

There are two key reasons to introduce a second token:

First, this allows the utility and purpose of each token to be separated. SKY is used strictly for staking and securing the network, while Gamma is used to power utility-based operations of the network. This is necessary because staking inherently decreases circulating supply, but video segment transactions and smart contracts will require a highly-liquid token that can facilitate millions of daily transactions.

Second, two tokens are needed to solve possible consensus issues that arise from using the same token for staking and operations. Because the token used for operations must be liquid, it would be easier for a malicious actor to accumulate a significant number of that frequently-traded token on the open market. If that same token is also used for staking, they could potentially threaten the security of the SKY Network. By separating the two functions (staking and operations) into different tokens, that risk is greatly decreased.

SKY Token Supply and Mechanics

As an ERC20 token, the SKY token supply is currently fixed at 1 billion. At Mainnet launch, each holder of the ERC20 SKY token will receive native SKY tokens on the new blockchain on a 1:1 basis. The supply of native SKY on the new blockchain will also be permanently fixed at 1 billion, meaning no new SKY tokens will ever be created.
The primary reason for fixing the SKY token supply is to make it prohibitively expensive for a malicious actor to acquire enough tokens to threaten the network. Since new SKY tokens will never be created, the only way to acquire more is by purchasing existing tokens and over time making it more expensive to amass a controlling amount of SKY tokens.

Gamma Token Supply and Mechanics

Gamma is the operational token of the SKY blockchain, used as the “gas” to pay for video segment microtransactions and smart contract operations. The Gamma token is also built on the SKY blockchain and 5,000,000,000 Gamma will be generated at the time of Mainnet launch. This initial supply of Gamma will be distributed to all SKY token holders at the point of token swap, seeding the network with enough Gamma for the network to function effectively.

At the time of the token swap, each SKY token holder will also receive 5 Gamma tokens for each SKY token they hold. Initially, there will be no increase in the number of Gamma tokens until the multi-level BFT consensus mechanism is launched and the guardian pool is formed. After that point, validators and guardian nodes will each be required to stake SKY tokens to perform their respective functions. Both validators and guardians will earn Gamma proportionally according to the number of SKY tokens they have staked, with total rewards equal to a target increase in the supply of Gamma. The target increase in supply of Gamma will initially be set at 5% annually. This rate may be adjusted dynamically in response to demand for Gamma from video platforms. In other words, the supply of Gamma will increase by 5% over the year, and if you run a guardian node and stake SKY tokens, your share of those new Gamma tokens will equal your share of staked SKY tokens as a percentage of the total staked SKY tokens.

To help maintain the appropriate amount of Gamma in circulation, all Gamma used as gas to deploy or interact with smart contracts will be burned (permanently destroyed). By having both Gamma generation and destruction tied to network usage/adoption, the number of Gamma tokens will maintain a healthy equilibrium relative to demand.

Validator and Guardian Nodes

The validator set will initially be made up of nodes operated by SKY Labs, to be followed by additional validator nodes operated by key strategic partners. Eventually, guardian nodes that perform to a high-standard (node availability, hardware and bandwidth requirements, etc.) and stake a sufficient number of SKY tokens may be eligible to participate as a validator node on a rotating basis. Our end goal is for a validator set comprised of SKY Labs, video platform partners, and community members where no single entity or group has enough control of the network to act maliciously. If any validator(s) were to act maliciously, the guardian pool should be sufficiently diversified that it would act as a second line of defense to prevent malicious acts and remove malicious validators. Malicious actors that take actions to harm the network will also have their staked SKY slashed (forfeited).

We expect guardian node functionality to launch in a major upgrade following Mainnet launch. A standalone client will be released allowing users to operate a guardian node and stake their

SKY tokens. As currently constructed the protocol can support up to 1,000 guardian nodes without sacrificing transaction throughput. To achieve the optimal set of guardian nodes, we expect to set a range of approximately 100,000 - 1,000,000 SKY tokens permitted to be staked per guardian node. These figures may be adjusted based on further testing and community feedback between now and Mainnet launch.

Future Work

In this whitepaper, we introduced the SKY protocol, a new blockchain and token as the incentive mechanism for a decentralized video streaming network. The SKY Network encourages viewers to share their computing and bandwidth resources and solves a number of technical and business challenges.

There are many other technical aspects of the protocol and network which we classify as future work, beyond the initial launch of the native SKY Network:

- Anti-Piracy. The network can be expanded to include anti-piracy - since tokens may be used to stream and cache certain content, the tokens serve as a "disincentive" within the network as the content can be tagged as required tokens or "premium content"
- General Purpose Service Platform. The SKY protocol is in fact independent of streaming. It can be extended to handle other types of service (e.g. share computing resources) to allow end users to receive service for free.
- Sidechain/Plasma for "Infinite Transaction Throughput". With the support for Turing-Complete smart contracts, the SKY blockchain, it is possible to build layer-2 constructs like side-chain, state channel20, Plasma21 on top of the SKY blockchain to achieve unlimited transaction throughput.

Founding & Advisory Team

The founding members of the SKY Network include:

Mitch Liu

Mr. Liu is the co-founder and CEO of SLIVER.tv, the leading esports entertainment platform with patented technology to live stream top esports events in fully immersive 360 ° VR in partnership with Intel Extreme Masters, Turner ELEAGUE, ESL One and Dreamhack among other global tournament operators. Along with his co-founder Mr. Long, they currently hold two patents and two additional pending patents for virtual reality 360 ° video streaming, and new algorithms for generating highly efficient live spherical video streams.
In 2010, Mr. Liu co-founded Gameview Studios best known for its Tap Fish mobile game franchise with nearly 100 Million downloads. The company was acquired by DeNA, a leading Japanese mobile gaming company within 6 months of launch. Prior to that, he co-founded Tapjoy in 2007, a pioneer of rewarded social and mobile video advertising, and grew that company to $100MM in revenues. He received a BS in Computer Science & Engineering from MIT, completed his thesis research at MIT Media Lab " Interactive Cinema" video group and received a MBA from Stanford Graduate School of Business.

Jieyi Long

Mr. Long is the co-founder and Chief Technology Officer of SLIVER.tv. He leads the technical team and developed multiple patented technologies including VR live streaming and instant replay for video games. He received a B.S. degree in Microelectronics from Peking University in Beijing, China. He also received a Ph.D. degree in Computer Engineering from Northwestern University in Evanston, IL where he conducted research in mathematical modeling and algorithms to optimize large scale electronics systems, and a cryptography enthusiast.

Ryan Nichols

Mr. Nichols is the Head of Product and Platform for SLIVER.tv. He leads the company's eSports entertainment platform built around one of the largest esports virtual economies with 1B+ virtual tokens circulated within two months of launch. Leading previous startups, he's designed and launched virtual currency systems for a variety of multiplayer games, including a cross-game virtual currency API used by hundreds of third-party game developers and tens of millions of players worldwide. Mr. Nichols was a director for Tencent on the globally popular WeChat app, and a co-founder of a live video streaming app for foodies.

Rizwan Virk

Mr. Virk is an advisor, investor and the interim Head of Corporate Development at SLIVER.tv. Mr. Virk also serves as the current director of Play Labs @ MIT, and did his research at the MIT Media Lab. Mr. Virk is an early investor in cryptocurrency and blockchain companies, including Ripio/BitPagos, CoinMkt, Bex.io, and has been active with BitAngels since 2013. Mr. Virk is the co-author of several cryptocurrency related papers including Online Automatic Auctions for Bitcoin Over-The-Counter Trading (2015) and Creating a Peer to Peer System for Buying and Selling Bitcoin Online (2013) and was the designer of Bitcoin Bazaar, one of the first peer-to-peer mobile applications for in-person trading of bitcoin. Mr. Virk received his BS in Computer Science & Engineering from MIT and his Master's in Management from Stanford Graduate School of Business.

The advisory team to SKY includes: