Blue Paper - SpaceComputer

Daniel Bar, Dahlia Malkhi, Matej Nemcek, Filip Rezabek

November 2024

Abstract

SpaceComputer leverages satellite infrastructure to deliver a robust and secure blockchain environment resistant to traditional cyberattacks and tampering. It builds on top of Cryptosat, providing TEE-like functionality with physically tamper-proof satellite nodes. SpaceComputer is an ideal solution for high-security applications, such as cryptographic key management and secure computations. The system benefits from the Iridium distributed network of satellites, making it resilient to DoS attacks and censorship attempts. To manage the nodes and state among them, we introduce various design considerations for a suitable two-layer approach, relying on off-loading to Earth Layer-2 solutions. In addition to consideration of classical rollup based on approach, we introduce an innovative L2 design allowing for fast finality, combining both on-orbit and Earth-based validators. This hybrid model ensures low-latency processing while preserving the scalability and security needed for complex applications, such as smart contracts and decentralized marketplaces for satellite services. The paper further explores several use cases of SpaceComputer. This document shares our rationale and design considerations with the community. With continued development, SpaceComputer aims to eliminate dependency on Earth-based intermediaries and continuously improve the performance of the L1.

1 Why Use SpaceComputer?

We are building a decentralized platform called SpaceComputer in orbit. The first question we address is **why satellites.**

Satellites can address certain security vulnerabilities on Earth, such as storing sensitive data that remains leak-proof for the next one thousand years, that even Trusted Execution Environments (TEEs) do not solve. Consequently, SpaceComputer offers many security and longevity benefits on multiple levels due to the infrastructure it relies on.

More specifically, SpaceComputer nodes run within satellite-cubes in orbit that are physically tamper-proof by any administrator or powerful actor and do not suffer from side-channel leaks that are prevalent here on Earth. Once safe software is deployed on SpaceComputer, it is virtually impossible to penetrate it, leak information through side-channel attacks, or bribe node operators to modify their behavior. Thus, less trust needs to be placed in the entities that operate the network, and the presumption that a threshold is failure-free does not rest on their honesty or game-theoretic rational behavior. The Hardware (HW) components offer physical redundancy, allowing software to provide crash-fault tolerant guarantees.

The SpaceComputer network is far more resilient to denial of service attacks from Earth than Internet-based networks, and likewise, taking over nodes is much harder. These two factors combined make it difficult to bring the network completely down, even by a state-level actor. The Iridium network offers redundant paths among its satellites, contributing to communication's overall robustness. This positions SpaceComputer as a robust main or backup path for deployed systems.

Notwithstanding the above, it should be noted that communicating and utilizing SpaceComputer from Earth currently relies on Iridium terminals, which are subject to such attacks and might effectively cause

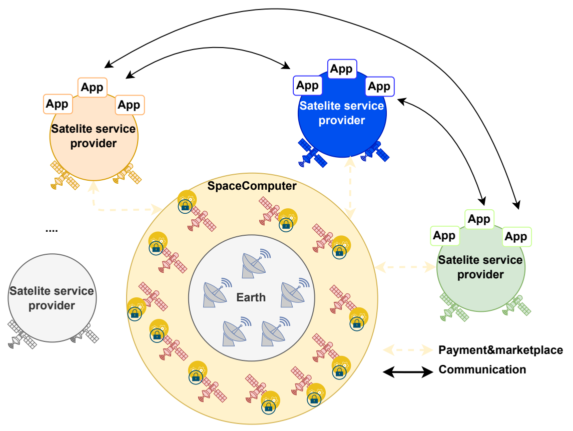

App
App
App
Satelite service provider
App
App
App
Satelite service provider
SpaceComputer
....
Earth
App
App
App
Satelite service provider
Satelite service provider
Payment&marketplace
Communication

Figure 1: SpaceComputer serving as a marketplace and payment layer for other satellite service providers

SpaceComputer to become unusable. In the future, we will aim to have direct consumer device communication to satellites without depending on censorship-prone intermediaries.

1.1 What can SpaceComputer be used for?

Broadly, we can think of two use-case categories that leverage the tamper-proof nature of SpaceComputer and its resilience, SpaceComputer as a celestial commerce marketplace and SpaceComputer as a security service. We elaborate on each one below.

Commerce marketplace for celestial services. There have been more satellite launches in the past few years than the total number of launches in the past decades. This indicates a flourishing space-based services sector, including imaging, communication, secure storage/communication, and others. SpaceComputer can facilitate payments and other market infrastructure tools to serve other applications running in orbit. This use-case may never (or very seldom) need to communicate back with Earth; it will be entirely among celestial applications. Figure 1 shows such interactions, where various satellite service providers host applications that can either communicate internally in the same network or across different satellite networks. SpaceComputer provides a settlement layer allowing to form payment instruments, and to tokenize and to trade new digital assets that represent and hold value in orbit.

We want to employ a blockchain for such a marketplace in orbit for several reasons. The first is that smart contracts enable software upgrades and customization while making all changes transparent. This allows the platform to evolve while deterring and detecting any rogue software updates.

The second reason is decentralization. SpaceComputer will be operated by a network of nodes deployed on cube satellites, governed by separate administrative entities, and transcending the control or limits of any single company. In some cases, entities (initially only a few) will be able to fully control and launch their cube satellite and participate in SpaceComputer's operation. In other cases, entities can lease hardware resources on cubes, similar to how one can lease bare-metal resources on today's clouds. Communication from the Earth will also be completely decentralized, initially through a decentralized collection of ground stations, and later allowing anyone with a satellite-enabled device to initiate communication to SpaceComputer.

Security services. To reach massive adoption, decentralized platforms must be able to handle sensitive information such as compliance data and trade secrets with confidentiality. Additionally, in some cases, they must be able to *delete* sensitive information with provable erasure guarantees [1]. In many cases, decentralized applications and services can utilize SpaceComputer to implement critical security components, taking security to new heights with tamper-proof guarantees. Below, we elaborate on several potential use cases.

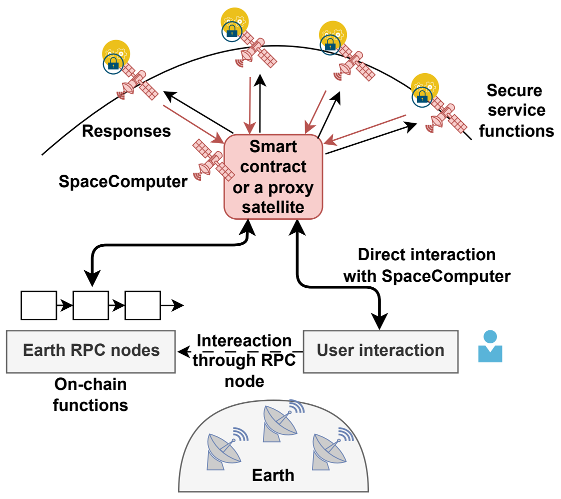

Secure service functions
Responses
SpaceComputer
Smart contract or a proxy satellite
Direct interaction with SpaceComputer
Earth RPC nodes
Intereaction through RPC node
User interaction
On-chain functions
Earth

Figure 2: SpaceComputer provides security services for Earth users and RPC nodes

Co-processor. SpaceComputer can be utilized as a trusted co-processor for off-loading secure computations over encrypted data with two key advantages.

First, since SpaceComputer is a full-fledged smart contract chain, it supports programming with a familiar and full-featured language (e.g., Solidity), allowing to easily express and invoke arbitrary computations.

Second, SpaceComputer security remains leak-proof forever and confidential information sent to it is not subject to future attacks.

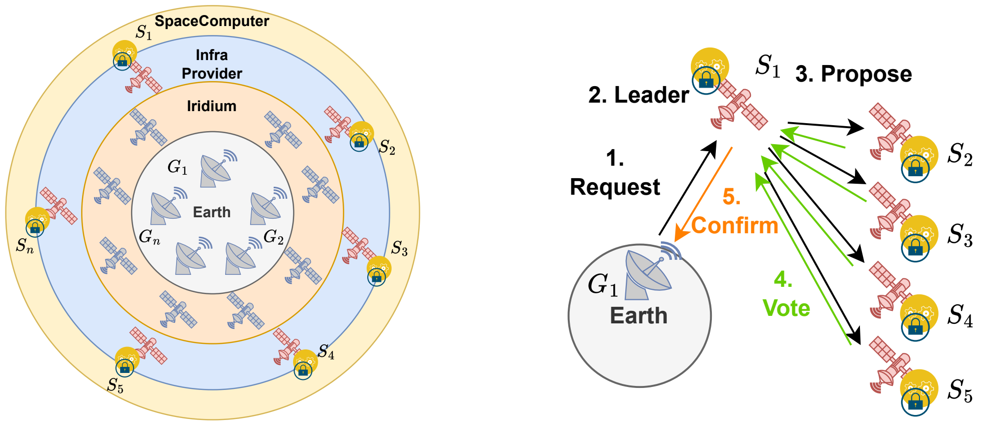
The second advantage provides a unique guarantee of forward secrecy offered by SpaceComputer. By comparison, consider two prevailing approaches for secure computation, each carrying its own risks:

- Trusting committees for secure computations is a double-edged sword: on the one hand, trust is decentralized and raises the bar for hackers and misbehavior. On the other hand, as time goes by, a threshold exceeding the committee resilience may likely become compromised and then all past secrets may be exposed.
- TEEs can boost trust across geographic boundaries through remote hardware attestation but entail their risks concerning confidential information: they are the target of hackers and governments alike, and eventually, someone with physical access to a TEE may obtain their secret key. Once again, all past secrets would become exposed in that event.

Off-loading secure computation to SpaceComputer is depicted in Figure 2. The SpaceComputer network serves as a TEE-service that allows client applications to store encrypted values and compute over secret values through a fully programmable API. For added security and high availability, secrets can be split into shares which are distributed (encrypted) across SpaceComputer satellites. A secure computation over a network of TEE-held shares can be done efficiently.

Secure custodian. Another principal use case, owing to the leak-proof nature of SpaceComputer, is a secure custodian of cryptographic secrets and a source of space/cosmic randomness. Figure 2 presents a setting with SpaceComputer hosts multiple key shares distributed among the satellites, actively used as a robust custodian solution. The user sends a request through the available network and receives back a signed transaction. Similarly, combining individual information can provide randomness by using key shares as a randomness beacon or by using space entropy.

Bulletin-board. SpaceComputer may be used as an eternal bulletin board that is forward-secure against alterations, shutdown, and bribery. For example, say that a particular hash chain, e.g., Bitcoin, stops



(a) Layers Overview (b) Form of Interaction

Figure 3: SpaceComputer Network

growing, which makes it vulnerable to a powerful actor to mounting a long-range attack and forking it. SpaceComputer can come to the rescue, recording with finality the chain's tail periodically.

Secure deletion. Many services require the disposal of information securely, either for confidentiality reasons or due to compliance requirements (e.g., with [1]. On Earth, deleting information is a very hard problem: some bits of information can almost always be extracted from physical storage media. SpaceComputer can store data with an optional delete operation that can be activated without worrying about physical tampering.

2 Introductory Concepts

Any node equipped with an Iridium terminal can interact with SpaceComputer nodes for easy communication and openness. Figure 3 provides an overview of the communication layers and peer interaction. Figure 3a shows the terminals communicating with the Iridium network from the Earth. The Cryptosat[1] serves as spaceTEE [2] offering a tamper-proof environment. It relies on cube satellites, which are connected to the Iridium network and communicate with each other. SpaceComputer leverages the Cryptosat satellites as a physical infrastructure. At any moment, certain SpaceComputer nodes are within range of Iridium[2], and others are not. Similarly, the Cryptosat satellites communicating with each other will have limited connectivity. This sporadic reachability between Iridium and certain satellites is a primary consideration in our design. Figure 3b illustrates a SpaceComputer network with five satellite nodes, $S_1, ..., S_5$, and a single ground station G_1. The node S_1 is currently within the range of G_1, while the rest are in the dark. Since the nodes have a location beacon, the end terminal on the ground can choose which node to talk to.

As both Iridium and Cryptosat networks develop independently of each other, the performance of the networks gets better over time. Using the layered design, the Cryptosat satellites can contain different transceivers that communicate not only via Iridium but also other networks such as Kuiper[3] or Starlink[4]. In the beginning, we envision a latency within seconds in the best-case scenario (nodes can see each other) to tens of minutes, where Cryptosat and Iridium satellites are out of sync. This latency applies to scenarios where we need to send data to all satellites in the network. If only a subset is required, latency may be lower if the position of satellites is favorable. For throughput, we foresee data constraints to be sent by each node daily in the order of KB with an early transition to MB.

[1]https://www.cryptosat.io/
[2]https://www.iridium.com/network/
[3]https://www.aboutamazon.com/what-we-do/devices-services/project-kuiper
[4]https://www.starlink.com/

Based on the current state, the satellites $S_1, ..., S_5$ run comparable hardware with ARM v8 64-bit, 8GB memory, storage up to 1TB, GPU Speed up to 8 TOPS, and HW crypto acceleration. The system will run hardened Linux OS. Each satellite has two identical sets of HW for redundancy and resource usage. Each board runs with two identical ARM CPUs, allowing better parallelization. The resources might be provided as multi-tenant solutions.

3 Two-Tier Architecture

The special infrastructure of SpaceComputer brings unique design considerations and leads us to pursue a two-tier architecture.

Network constraints impact the design of the SpaceComputer core consensus protocol in several principal ways. We expect Earth's storage and processing capacity to be higher and cheaper than in orbit in the foreseeable future. Therefore, we treat SpaceComputer as a (relatively) capacity-constrained chain and offload as much work as possible to Earth. This leads to the design of SpaceComputer with a two-tier approach, an Layer-1 (L1) in orbit (Celestial) and an Layer-2 (L2) on Earth (Uncelestial).

We can borrow various existing two-tier scaling solutions that were developed to address capacity limitations on Ethereum (see *"A rollup-centric Ethereum"* [3]). Two main scaling techniques are known, commit-chains and state-channels; Frame 3 provides a refresher on them.

Frame 1: Refresher on Scaling Approaches

Generally, off-chain scaling solutions allow distrustful parties to deposit funds into a bridge-contract on L1 and then operate an off-chain engine (known as "layer-2", L2) which sequences updates to the contract state, including withdrawals.

There are two principal L2 approaches: commit-chains (shown in Figure 4a) and state-channels (adopted and improved in SpaceComputer, as shown in Figure 4b). Both enable throughput scaling by off-loading compute and (possibly) storage resources, but only state channels can help with fast finality, which is crucial in our settings: whereas on Ethereum, throughput is the main scaling challenge, whereas latency (12 s) may be tolerable, in SpaceComputer, latency from Earth may be as high as tens of minutes in the early phases. Therefore, off-chain fast finality is extremely valuable in the SpaceComputer setting.

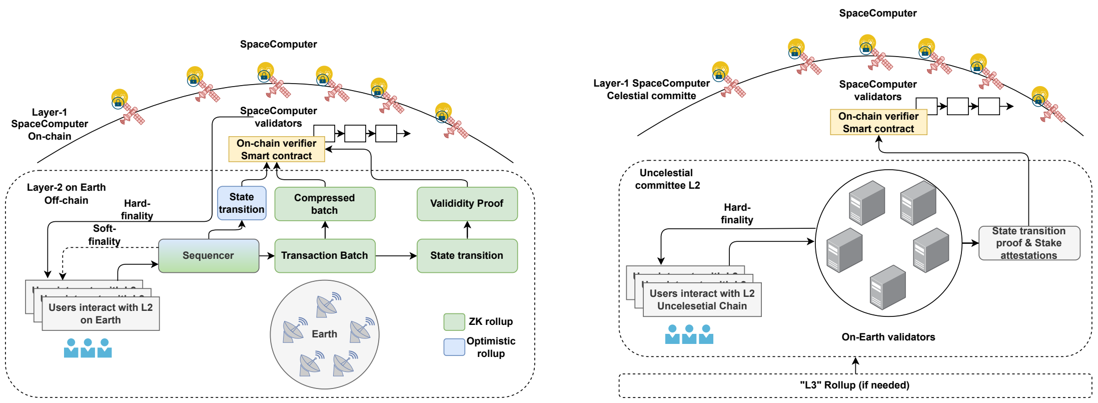
Existing state-chain approaches are constrained, predominantly supporting two-way payment channels (e.g., Lightning network [4]). A different kind of state-chain is Plasma [5], but it did not get much traction (it has been noted that "the details of how complex fraud proofs could actually be constructed inside a Plasma contract are lacking. Plasma contracts need to somehow specify all of the consensus rules and ways to prove fraud on a newly defined blockchain, which is a complex and currently unsolved problem inside an Ethereum contract.

What is the difference between commit-chains and state-channels?

Commit-chains have to settle state-updates on-chain for transactions to be considered finalized; until they are committed on L1, transactions are considered tentative only (softly finalized). Primary examples of commit chains are rollups and Plasma chains. The biggest drawback of commit chains is deferred settlement.

State-channels do not need to settle transactions on-chain and support immediate finality. Transactions are posted to L1 only in case of a dispute or during channel set-up and shut-down. Payment networks like Lightning [4] and Nitro [6] are primary examples of state channels. The biggest drawback of state channels is that they require dispute and punishment mechanisms, which are quite complex and do not easily support multi-peer settings as needed, e.g., Decentralized Finance (DeFi) applications.

SpaceComputer can support two types of L2s, commit-chains and a new variant of state-chains introduced here, referred to as a fast pre-finality chain.



(a) Rollup-centric approach for SpaceComputer network with soft finality [7]

(b) Uncelestial and Celestial SpaceComputer network with hard finality

Figure 4: SpaceComputer Network Designs

In both options, a Celestial chain is deployed as a consensus network in orbit and serves as the root of trust. An Uncelestial sequencer on Earth requires less security, can aim at high transaction per second (TPS) and low latency finality, and is open to many design choices.

An Uncelestial commit-chain (depicted in Figure 4a), can be materialized by any rollup protocol, e.g., optimistic, fraud-proof, validity and ZK rollups. In principle, all of them can synchronize off-chain sequencing decisions with a bridge-contract on L1, so long as they can deploy the bridge-contract on the Celestial Virtual Machine (VM). Additionally, to adhere to current storage and bandwidth practical limitations on Celestial, the commit-chain should provide a Data Availability (DA) functionality.

As discussed in Frame 3, the main drawback of commit-chains is hard-finality slowness, especially in the SpaceComputer settings. Therefore, SpaceComputer goes a step beyond by introducing a novel fast pre-finality approach (depicted in Figure 4b). In some sense, this is the most interesting part, since it combines the scaling benefits of rollups with the latency benefits of state channels.

Technically, the idea is to introduce a state-agnostic pre-finalization step that can work atop of any consensus sequencing protocol, and works roughly at the speed of the underlying sequencing protocol. Crucially, this step simplifies dispute resolution and thus alleviates some of the complexities associated with managing disputes in state-chains.

The fast pre-finality approach is described in more detail in Section 5.

4 Celestial Layer-1

Powering the SpaceComputer Celestial L1 consensus requires satisfying challenging technical considerations, which were elaborated above, regarding low bandwidth and sporadic communication, especially to/from Earth, as well as limited storage capacity. Our design for the core consensus protocol, therefore, emphasizes communication efficiency and separating data availability and execution from sequencing.

First, due to bandwidth and connectivity constraints among the Celestial nodes themselves, protocols that employ a linear communication pattern, e.g., HotStuff [8], are advantageous in our settings. Not only do they consume lower bandwidth and messages, they require connectivity with only one node designated as *leader*. In addition to linearity, the streamlined nature of HotStuff is very important in our setting. Each leader broadcast carries a new block of proposed transactions and thus effectively utilizes bandwidth.

Another challenging aspect that requires particular attention is uploading transaction requests from Earth to the Celestial network due to connectivity and latency constraints. Initially, a handful of ground stations on Earth will serve to interact with Celestial nodes through the Iridium network. At any moment,

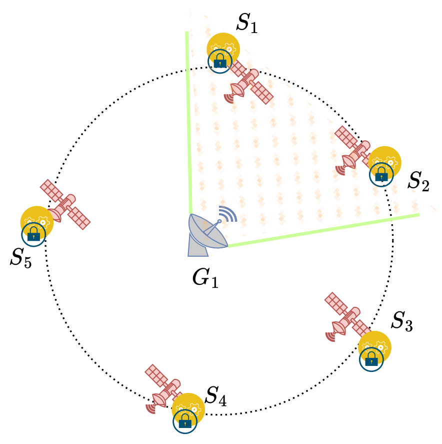


Figure 5: Reachability to a subset of satellites

only certain nodes are within range of some Iridium satellites, and others are not. Figure 5 illustrates a simplified setting with SpaceComputer network with five nodes, $S_1, ..., S_5$, and a single ground station G_1. Nodes S_1 and S_2 are currently within range, while S_3 to S_5 are "in the dark".

We adapt HotStuff to utilize an upward link effectively as follows. Whenever an upward link from a ground station is used, the station sends user messages (e.g., transactions, or L2 updates) to a SpaceComputer node within range. The node can try to become the next leader and additionally disseminate transactions to other nodes, so that the next designated consensus leader can propose to sequence them.

Last, we expect the latency to be an important consideration overall in our architecture. Therefore, SpaceComputer adopts several recent improvements to HotStuff latency, including HotStuff-2 [9], which reduces latency to finality under all circumstances by one third; and HotStuff-1 [10], whose speculative confirmation path reduces the number of exchanges between the ground station and Celestial nodes under common conditions.

We remark that there exists an open-source implementation of HotStuff-2, HotShot [11], that separates DA and execution from sequencing, and can serve as a basis for implementing the Celestial consensus protocol.

An Illustration. Figure 6 shows how the individual steps of HotStuff-1 may work in our settings, depicting a flow with two consecutive proposals and their votes with the streamlined approach:

- In ①, G_1 sends a request (say, T_1) to all satellites, but S_1 and S_2 receive it with S_1 being the leader for the view. S_1 as a leader picks up the request and disseminates a first proposal (in black). The leader broadcasts the request to all of the SpaceComputer nodes. Of note, this step might take some time, as not all nodes are available right away and must properly align with the Iridium network.

- Nodes verify the validity of the leader proposal and send a signed vote on the proposal to be routed back to the ground station G_1. The communication might happen directly or indirectly depending on the connectivity between the satellites and the Iridium network. Sometimes, the nodes can send votes directly to ground stations within range. In other cases, nodes may send a copy of signed votes to future candidate-leaders who will shortly become within range from ground stations.

- The ground station G_1 aggregates votes into a certificate on T_1 to be used when forming the next request.

- Continuing with ②, S_3 and S_4 are now within reach; therefore, they directly vote on the first proposal. G_1 sends a second request, T_2, along with an aggregated certificate of votes on the first request, to S_3.

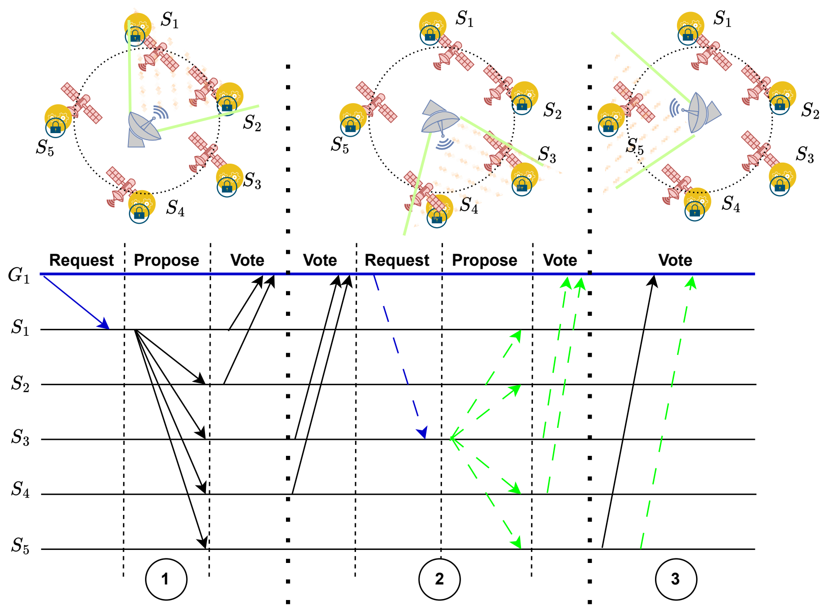


Figure 6: Communication flow of two requests between Uncelestial and Celestial

- S_3 picks up the request and sends a second proposal (in green). Both S_3 and S_4 immediately return speculative responses on T_1 and send them with their votes on the T_2 directly to G_1.
- In ③, the remaining satellite S_5 responds to both proposals as it is reachable via the G_1 and the Iridium network.
- At this point, one more response is needed on T_1 for it to become committed. Likewise, one more vote on T_2 is needed in order for it to become certified and indirectly commit T_1. Either or both should happen directly or indirectly when S_1 or S_2 are reachable. Once this happens, T_1 becomes committed.

Figure 7 shows the certificate generation steps for a stream of requests and commits such as above. Once a quorum of votes is collected on an immediately preceding request for which a certificate has been formed, the double-certified request becomes finalized. Speculative results can be computed upon forming a certificate for the request.

We note that to further offset load from the Celestial network, the ground station G_1 can delegate to an execution network to speculatively process requests and provide a proof of a tentative result. The speculative result can be carried out as additional information in the next proposal.

5 Uncelestial Layer-2 with Fast Pre-Finality

The Uncelestial network sequences transactions outside L1 by a consensus protocol operated as L2 by an Uncelestial committee.

We incorporate a consensus enhancement mechanism into Uncelestial that allows users to trust a consensus decision on Uncelestial as *pre-final*, similar to the state-channel capabilities. In this way, users can get extremely fast pre-finality. However, for users to trust the Uncelestial fast pre-finality mechanism, it must be able to detect forking and punish the Uncelestial committee members if they misbehave. Then, the punishment may be utilized to **repair** any damage caused by the bad event, and furthermore, it would normally deter participants from ever misbehaving in the first place.

In order to be able to effectively detect and punish misbehavior in a sequencing consensus decision, we borrow a trick from Ebb-and-Flow [12]. We ask parties to publish a signed attestation backed by stake

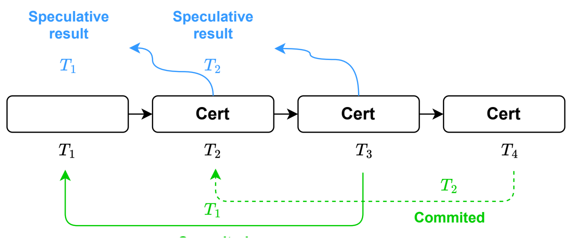


Figure 7: Streamlined HotStuff-1 with speculative results and commitments

confirming the decision. When a quorum of signed attestations is received, it is considered pre-final. This extra step can be added to any consensus protocol employed by Uncelestial as shown in Figure 4b.

The extra round of attestations achieves two things. First, a L1 contract can easily validate it without getting into the complex details of an off-chain consensus protocol. Second, we can detect and punish the misbehaving parties.

More specifically, to detect the root cause of forking and punish relevant parties, the idea is to add one more round to a consensus protocol. When a validator learns a decision, it broadcasts a vote to finalize it. When finality votes are cast by $\frac{n+f+1}{2}$ validators, it guarantees a unique decision unless f are detectably misbehaving. Should that happen, their misbehavior can be easily detected and proven, independent of complex and internal protocol logic. In this way, they can be punished, which usually suffices to deter cheating in the first place.

There are three cases of misbehavior to consider:

1. Failing to post an update
2. Confirming a flawed update
3. Confirming two conflicting updates (a fork)

We now explain how each one is addressed.

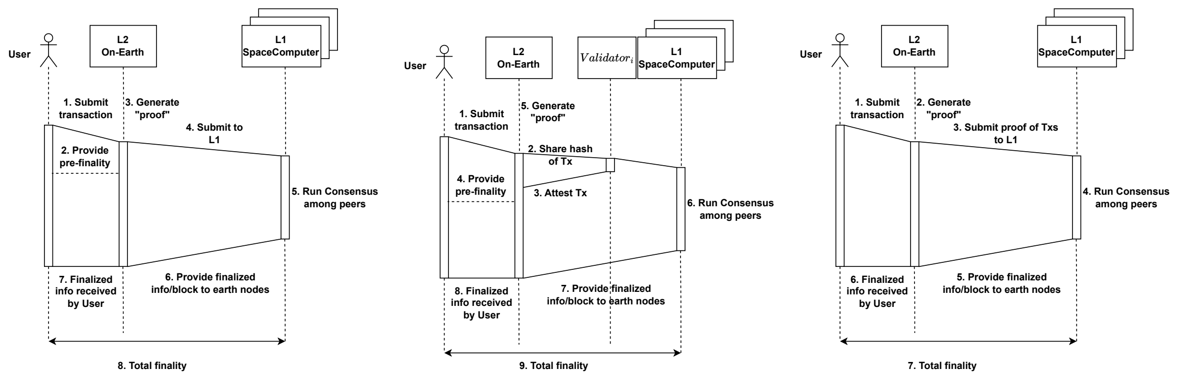
The first case of misbehavior is when Uncelestial publishes an update but fails to post it (or maliciously avoids posting it) to the rollup contract on Celestial. We can allow anyone to post such published decisions to Celestial, and furthermore, due to the signed attestations, we can require that fees are included in the signed update so anyone can post them without holding a balance on Celestial.

The second case of misbehavior by the Uncelestial committee is committing an invalid state. There are several battle-tested rollup methods, including fraud proofs and optimistic dispute management, which may be harnessed for preventing invalid updates from being committed to the bridge-contract on L1.

Naturally, users should not to have to wait for L1 to trust a published decision with pre-finality. Rather, anyone can independently check that validity of published updates on Uncelestial. If an invalid update is published, they would **not** consider it as pre-final. If such an invalid update gets posted to Celestial, the published update is invalid and will be rejected by the L2 contract because either it contains an invalid transaction or the result of processing a transaction is incorrect.

Furthermore, our unique pre-finality method allows a challenging entity to prove to a bridging contract on L1 that the Uncelestial chain validators signed off a bad update. In this case, Uncelestial stakers attesting to a flawed update may be punished, and the punishment may suffice to re-enact any rejected transaction correctly). We remark that a prevention/remediation mechanism (always) needs to employ properly incentivized watchers to monitor the Uncelestial chain for misbehavior and to alert if a bad state update gets finalized.

To handle settlements and disputes, we need to allow anyone to post a finality claim to L1. The claim remains undisputed for a certain amount of time until it is considered settled. If a user believes that the Uncelestial chain has confirmed an erroneous state update, they can submit fraud-proof to the bridging



(a) Pre-Finality by L2 (b) Pre-Finality by L2 with a backup (c) L1 finality

Figure 8: End-to-end flows with three security grades (depicted left to right) among a user, the Uncelestial layer-2 consensus chain, and the Celestial layer-1 chain.

contract on L1 to exit the contract with their funds. Thus, the Uncelestial chain uses the SpaceComputer L1 as a backstop for dispute resolution.

The third case of misbehavior is that Uncelestial confirms two different transactions at the same sequence position, i.e., the Uncelestial chain is forked. However, remember that the extra attestation step makes detecting who equivocates and punishes them easily. Anyone can post to the rollup contract a proof that two transactions have been sequenced at the same position and detect stakers that need to be punished.

Finally, the L1 is also used when users want to fund or withdraw from their accounts. To post a fund/withdraw operation, a user must post the latest state of L2 in the bridge contract. The ledger state should reflect committed funding/withdrawal transactions so that the bridge contract can enact them on L1. Another event causing a L1 post is simply creating a checkpoint of L2 without entailing a fund/withdraw action.

Three grades of finality. Integrating fast pre-finality on Uncelestial with Celestial presents a continuum of trust, and therefore, we provide users with three *grades* of finality: pre-finality confirmation, pre-finality with backup, and finality. (We call it pre-finality, not soft-finality, as is the case for rollups, as the Uncelestial provides more security guarantees than the current rollup solutions.)

For **the first grade**, a consensus decision is reached on the Uncelestial chain to include the transaction in the ordered sequence. This is the shortest path to a confirmation (or pre-finality). For users to gain trust in pre-finality, it is accompanied by a punishment mechanism. Figure 8 depicts the end-to-end flow among a user, the Uncelestial L2 consensus chain, and the Celestial L1 chain.

The second grade, depicted in Figure 8b, provides another level of pre-finality confirmation, in addition to the consensus decision reached on the Uncelestial chain. It checks a copy of the decision on a celestial node.

The third grade is when a transaction settles on the SpaceComputer L1. It is depicted in Figure 8c. This is the most secure solution but comes at the cost of the longer finality of the L1. The user considers it finalized after reaching the L1 consensus. This might be relevant for a high-stake transaction.

6 Summary & Next steps

SpaceComputer is a tamper-proof, satellite-based blockchain that aims to solve many of the vulnerabilities faced by Earth-based systems, including side-channel attacks, administrative tampering, and network takeovers. We host nodes on satellites and leverage the Iridium network for communication. Among fitting use cases, we envision secure key management, random number generation, and confidential computations, all while ensuring finality through fitting consensus mechanisms to the constrained environment.

In the next steps, we want to narrow down the selection of individual building blocks for the L1 and L2 by proper dimensioning of the network capabilities. With that in mind, we will select a suitable design that enables fast finality for the users of the system.

References

[1] EU, *Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation)*, May 2016.

[2] Y. Michalevsky and Y. Winetraub, *Spacetee: Secure and tamper-proof computing in space using cubesats*, 2017. arXiv: `1710.01430 [cs.CR]`. [Online]. Available: `https://arxiv.org/abs/1710.01430`.

[3] V. Buterin, *A rollup-centric ethereum roadmap - fellowship of ethereum magicians*, `https://ethereum-magicians.org/t/a-rollup-centric-ethereum-roadmap/4698`, Oct. 2020.

[4] J. Poon and T. Dryja, *Lightning-network-paper.pdf*, `https://lightning.network/lightning-network-paper.pdf`, (Accessed on 10/14/2024).

[5] J. Poon and V. Buterin, *Plasma.pdf*, `https://plasma.io/plasma.pdf`, (Accessed on 10/14/2024).

[6] T. Close, *Nitro protocol*, Cryptology ePrint Archive, Paper 2019/219, 2019. [Online]. Available: `https://eprint.iacr.org/2019/219`.

[7] D. Tortola, A. Lisi, P. Mori, and L. Ricci, "Tethering layer 2 solutions to the blockchain: A survey on proving schemes," *Computer Communications*, vol. 225, pp. 289–310, 2024. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S014036642400255X`.

[8] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, *Hotstuff: Bft consensus in the lens of blockchain*, 2019. arXiv: `1803.05069 [cs.DC]`. [Online]. Available: `https://arxiv.org/abs/1803.05069`.

[9] D. Malkhi and K. Nayak, *Extended abstract: HotStuff-2: Optimal two-phase responsive BFT*, Cryptology ePrint Archive, Paper 2023/397, 2023. [Online]. Available: `https://eprint.iacr.org/2023/397`.

[10] D. Kang, S. Gupta, D. Malkhi, and M. Sadoghi, *Hotstuff-1: Linear consensus with one-phase speculation*, 2024. arXiv: `2408.04728 [cs.DB]`. [Online]. Available: `https://arxiv.org/abs/2408.04728`.

[11] E. Systems, *The espresso sequener: Hotshot consensus and tiramisu data availability*, 2023. [Online]. Available: `https://github.com/EspressoSystems/HotShot/blob/main/docs/espresso-sequencer-paper.pdf`.

[12] J. Neu, E. N. Tas, and D. Tse, *Ebb-and-flow protocols: A resolution of the availability-finality dilemma*, 2021. arXiv: `2009.04987 [cs.CR]`. [Online]. Available: `https://arxiv.org/abs/2009.04987`.