LayerCake: Zero-Collateral L1 Bridges

Sean Rowan[1] Hugo Philion[1] Naïri Usher[1]

[1]Flare Research
{sean,hugo,nairi}@flare.network

March 16, 2022

Abstract

Bridges between L1 blockchains enable: 1) the transport of an asset from one L1 to another for use in applications on the travelled-to L1 and 2) the return of that asset to the original L1 after potentially acquiring new ownership based on the state transition rules on the travelled-to L1. A safety risk of existing designs for such bridges is that they lose safety if either 33% of the validators of a connected proof-of-stake L1, or 51% of the validators of a connected Nakamoto consensus L1 decide to attack the bridge. Due to this risk, L1 bridges today have an implicit limit to the amount of value that can cross them of below 33% of the staked value on the less-valuable connected L1. In this paper we present *LayerCake*, a novel L1 bridging protocol that has only a short-duration collateral requirement while crossing the bridge, as opposed to a full-duration collateral requirement that other protocols require for the entirety of time that an asset is represented on the travelled-to L1.

1 Introduction

Consider the fundamental security guarantee of how an L1 like Bitcoin operates: imagine that all of the hash-power in the world today mining Bitcoin suddenly decided to move someone's Bitcoin on the network without that person signing a valid transaction to do so. If this happened, then the entire world watching the network would independently discern that they should ignore all of these miners, user funds would not be at risk, and Bitcoin could continue making progress with new honest miners.

Conversely, bridges across L1s today do not share the same fundamental security guarantees of an L1. Instead, these bridges typically involve relying on a globally trusted set of entities to custody user funds. If these custodians become faulty, then all user funds on the bridge are at risk of being lost. This is the model adopted by Chainlink's CCIP protocol [5], the AVAX Bridge [1], LayerZero [13], Axelar [2] and Wormhole Bridge [12].

Another more technical but less popular approach for creating a bridge across L1s involves the use of light-client relays [3] to prove the state of an L1 directly to a contract on another L1. There are a number of security flaws with light-client relays, and intuitively this is because blockchain consensus protocols are not optimally constructed as a smart contract. Flattening a blockchain into a light-client relay necessarily removes the richness of properties and security guarantees that are available in their full original implementation. In practice when used in this setup, they also operate very slowly and with a large code surface due to the use of data availability sampling to improve the security of light client relays to a degree [3]. The fundamental limit to the security of light client relays used for bridging across L1s is that they cannot protect against theft of user funds under a 33% attack on an L1 they bridge, and so they must rely on slashing faulty validators of that L1 in order to protect value that crossed the L1 bridge [9, 8]. This model is leveraged by NEAR's Rainbow Bridge [9], Harmony Horizon Bridge [6] and Poly Network [11].

Cumulatively in the past year, $1Bn in user-value has been hacked from bridges connecting two L1s [10]. Although these hacks were on account of implementation errors, they highlight the demand that the world is placing on cross-L1 bridges, and we must be able to do more with less collateral while operating with safety guarantees. Otherwise, cross-L1 bridges will have severe limitations on the amount of value that can cross them due to the so-called anti-network effect [4]: the more users the system has, the less useful it becomes due to the increased risk of hacking.

In this paper, we propose a solution to the anti-network effect problem of cross-L1 bridges using a novel state relay mechanism called the *state connector*, which delivers information from connected blockchains without degrading their full security properties. The state connector can directly validate state correctness without any need for data availability sampling or challenge periods, and it can also implicitly withstand safety and liveness attacks on connected L1s. A resulting cross-L1 bridge construction called *LayerCake*, which leverages the state connector, requires only a temporary collateral lock-up while crossing the bridge that is proportional to the amount of value being sent across the bridge. LayerCake makes no assumption about requiring slashing of validators on connected L1s in its security model, and so the system has a zero long-term collateral requirement once value has crossed the bridge. Compare this to other designs that are constrained by requiring the total value deposited on their bridge across all users to be below 33% of the staking value on the less-valuable connected L1. If the total value deposited surpasses this threshold, then it becomes favorable to attack such designs.

2 The State Connector

The purpose of the state connector is to relay the state of connected blockchains in a manner that: proactively validates state correctness, withstands safety attacks, and recovers from liveness failures. The ability to proactively validate state correctness speeds up state relay due to not requiring data availability sampling or challenge periods, as is required in light-client relay setups. These properties are achieved using the two core protocols of the state connector: the RCR and branching protocols.

The RCR protocol enables users to request information about connected blockchains to be proven, e.g. the existence or non-existence of a transaction. These requests are interleaved with committed and revealed responses from *attestation providers*, which can be operated by anyone without any staking requirement.

The branching protocol enables proactive local oversight on the RCR protocol by enabling each node operator in a state connector enabled network, e.g. Flare, to locally override the default decision of the state connector with their own set of attestation provider(s) that they wish to rely on, as illustrated in figure 2. The branching protocol always guarantees binary deliberations, so any node that overrides the default decision of the state connector at a particular attestation round yields a consistent state.

2.1 RCR Protocol

There are three phases of the RCR protocol: Request, Commit and Reveal. These occur sequentially, as shown in figure 1.

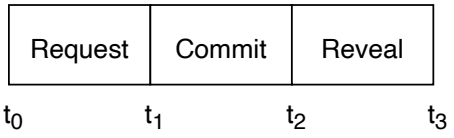

Figure 1: The three phases of the RCR protocol. Note that the protocol can be parallelised, see appendix A.2 for more details.

Request Phase During this phase, any user can submit a request to Flare to have information from another blockchain proven. User requests in the RCR protocol must be paired with a data availability proof. For example, to prove an L1 transaction that requires N block confirmations, the request must be paired with the blockhash of the N^{th} block after the block that contains the transaction being proven. This proves that the user knew the block hash ahead of time, signalling the block's availability.

Commit Phase During the next window of time, attestation providers have the opportunity to commit an obfuscated Merkle root hash that forms the basis to a Merkle tree of the proofs requested in the previous phase. Anyone may operate as an attestation provider without any capital requirement, but a default set is used as the minimal requirement for passing a vote. The default set may be overridden locally on one's own Flare node using the branching protocol described in section 2.2.

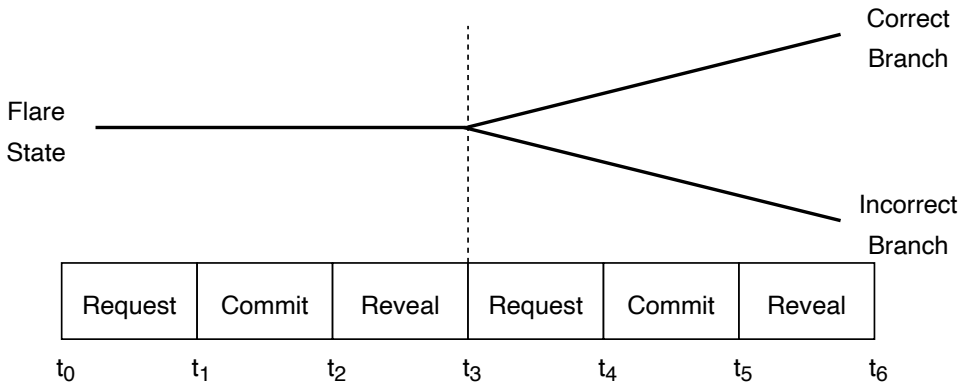

Figure 2: The state connector branching protocol. In the RCR protocol at time t_3 in the above example, a majority of default attestation providers have committed the same incorrect Merkle root hash. This will split the network into a correct branch and an incorrect branch, where the correct branch is formed by ignoring the latest revealed attestation round as if it reached no default attestation provider majority.

Reveal Phase Finally in the next window of time, attestation providers reveal the Merkle root hashes that they committed to in the previous round. Once the reveal phase concludes and the next phase begins, the revealed votes are automatically counted. If a majority has been reached in the default attestation provider set, and one's local attestation provider concurs with the result, then all requested proofs become immediately available to all contracts on the network using the Merkle root hash to prove them.

2.2 Branching Protocol

The branching protocol protects Flare against incorrect interpretation of external blockchain state proactively, such that there are never any rollbacks on the Flare state. Instead of having rollbacks, contention on state correctness is handled via automatic state branching into a correct and incorrect path. The security assumption is that if you as an independent node operator are correctly attesting the requested blockchain state, then you will always end up on the correct branch of the Flare state.

Figure 3 presents an intuitive depiction of the role of the branching protocol. At block N, independent nodes observing the network are in agreement with the relayed state of the network. Next, assume that after a round of the RCR protocol, a majority decision is reached which is not accurate. For instance, a transaction which has not occurred is accepted as being confirmed. At this point, nodes observing the state connector will disagree with this outcome, since it does not match the reality they observe. They will thus fork as they disagree with the default result of the state connector. At block $N+1$, all non-faulty independent nodes will have discarded the default outcome of the state connector and thus once again the state of the network will be consistent with reality.

When an independent node disagrees with the majority decision by the default attestation providers, its state database remains unchanged as if the default attestation providers were unable to reach a majority vote on a state connector round. This means that the independent node takes one step into the state branch that it believes is correct, and it safely halts. This gives a physical signal for example to an exchange, a collateral provider or other users involved in cross-chain use-cases that run their own Flare infrastructure that the default branch of the state connector on Flare may be faulty. Non-faulty core validators on a network leveraging the state connector will ultimately adopt the correct branch of Flare; in the same manner of how a new dominant chain history appearing in a proof-of-work network causes all nodes to automatically switch to that chain history [7].

2.3 Safety

The state connector operates by proving that a transaction either exists, or does not exist on an observed chain. This means that the entire state of a connected chain is not relayed by the state connector, and only specific information requested by users is relayed. Applications building on the

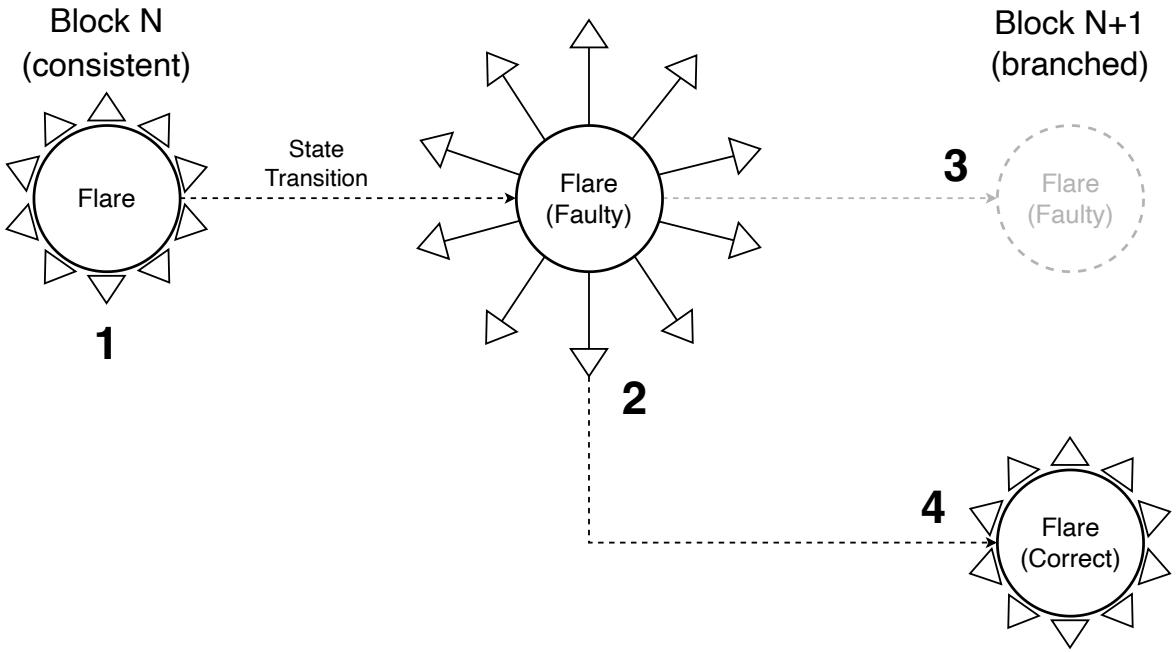

Figure 3: An intuitive depiction of the branching protocol of the state connector on Flare network. Independent nodes, represented by triangles in the figure, can be run by anyone without any stake in the network. 1) Independent nodes agree with the default state connector relayed state at Flare block N. 2) Independent nodes fork when they disagree with the incoming default state connector relayed state. The state connector branching protocol always guarantees a binary decision that causes independent nodes that fork to be consistent. 3) At block N+1, there exists two possible states of the Flare network after forking occurs: one that adopts the default majority decision of the state connector, and another that has the same state as if the default state connector set couldn't reach any majority decision. Anyone in the world watching the network can independently discern that they should ignore and discard the default path if it is faulty as depicted in the diagram above. 4) Without communicating to each other and with open membership, all independent nodes can determine what the correct branch is, and the rule for the network is that this branch must be adopted as the only valid state transition at this block height.

state connector can use this to prove that a reorg occurred on an observed blockchain which affects a transaction they are interested in: a transaction may be proven to exist on an observed blockchain, and then later it may be proven to no longer exist on that same blockchain. A practical example of an application leveraging this pattern in combination with a time-bounded collateral lockup to mitigate a 51% attack on an observed chain is described in section 3.5.

2.4 Liveness

A proof-of-stake blockchain loses liveness if more than 33% of the validators go offline. If these validators remain offline, then the live pool of validators and the world observing the blockchain may engage in 'social consensus' to bench the non-live validators and restart the network. However, this social recovery mechanism would not work in a light-client relay protocol. Consider the scenario where more than 33% of the validator set in control of a light-client relay go offline. The smart contract for the light-client relay would only be permitted to make progress if more than 66% of the validators sign a relay message. This liveness risk has the ability to lock deposit contracts in bridges using light-client relays unless a trusted fall-back mechanism is put in place to mitigate it for such relay protocols.

However, for the state connector, recovering from an L1 network liveness failure requires no trusted fallback mechanism since the state connector regains liveness as soon as the L1 regains liveness regardless of the L1 having benched any number of non-live validators on its network to regain liveness.

3 LayerCake L1 Bridge Construction

Anyone may operate as a *bandwidth provider* for a LayerCake L1 bridge if they lock up sufficient collateral on both the origin L1 and on Flare. Crucially, the sum of all bandwidth collateral across all bandwidth providers does not represent a 1:1 backing of the total value that crosses the LayerCake bridge. Instead, the bandwidth collateral represents the *rate* at which value can cross the bridge. As an analogy to a real-world car bridge, total bandwidth collateral is akin to how many cars the bridge can hold as they are crossing, but not the total number of cars that have crossed.

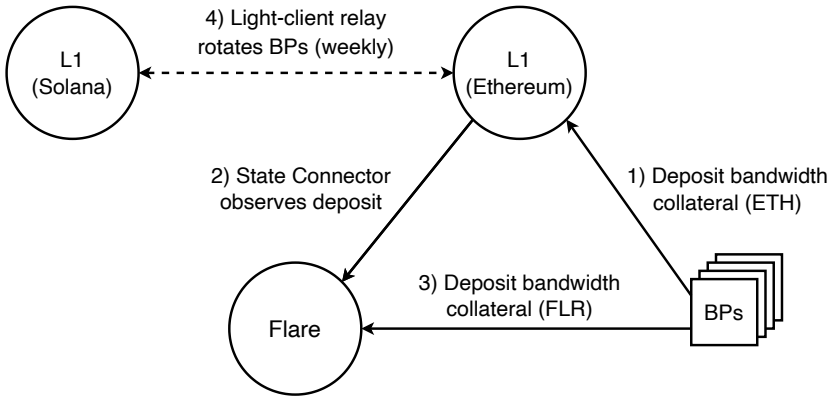

Figure 4: Bandwidth provider (BP) collateral setup with Ethereum as the origin L1.

Users crossing a LayerCake bridge pay a market-priced fee, similar to a gas fee, to compete for reservation of bandwidth provider collateral on both the origin L1 and on Flare. The reservation of bandwidth provider collateral acts as the backing that secures the user's transit across the bridge. LayerCake can handle any combination of smart contract L1s as origin and destination L1s to form a bridge. It is not recommended or accommodated in the protocol to bridge to a destination L1 from an origin L1 via an intermediate L1.

New bandwidth providers are included in a LayerCake bridge through a weekly changeover process that leverages a light-client relay directly between L1s to prove new bandwidth collateral deposits on the origin L1. The usage of a light-client relay in this step is resistant to reorgs due to operating slowly once per week. If the light-client relay encounters a liveness failure, then the LayerCake bridge would still be able to operate given at least one honest bandwidth provider, as they would enable users to safely exit the bridge. The collateral deposited by a bandwidth provider on Flare in its native asset FLR matches the amount of value deposited by the bandwidth provider on an origin L1. The purpose of this additional collateral on Flare on top of the origin L1 collateral is for both mitigating 51% attacks on connected L1s and penalising faulty bandwidth providers.

Figure 4 illustrates a bandwidth provider set up with Ethereum as the origin chain. They first deposit collateral in ETH on Ethereum and collateral in FLR on Flare. The bandwidth provider will then be included at the next light-client relay rotation.

3.1 Mint Flow

Users crossing the LayerCake bridge deposit their asset on the origin chain, along with the bandwidth reservation fee which they can increase to have higher priority in crossing the bridge sooner. Bandwidth collateral is then reserved on both the origin L1 and on Flare by proving the deposit using the state connector.

After bandwidth providers receive the signal that the user's deposit is sufficiently collateralised for crossing the bridge, the bandwidth providers then mint the corresponding representation of the user's asset on the destination L1, and they earn the user's fee which is paid in the origin L1 asset directly.

The bandwidth collateral used for this bridge crossing on the origin L1 has a short cooldown time before it is unlocked for use by another user crossing the bridge. Multiple users can cross the bridge at the same time given sufficient supply of bandwidth collateral. The bandwidth collateral used on Flare however has a longer cooldown time due to being used for mitigating 51% attacks where the user's deposit on the origin L1 may disappear if the origin L1 encounters a reorg.

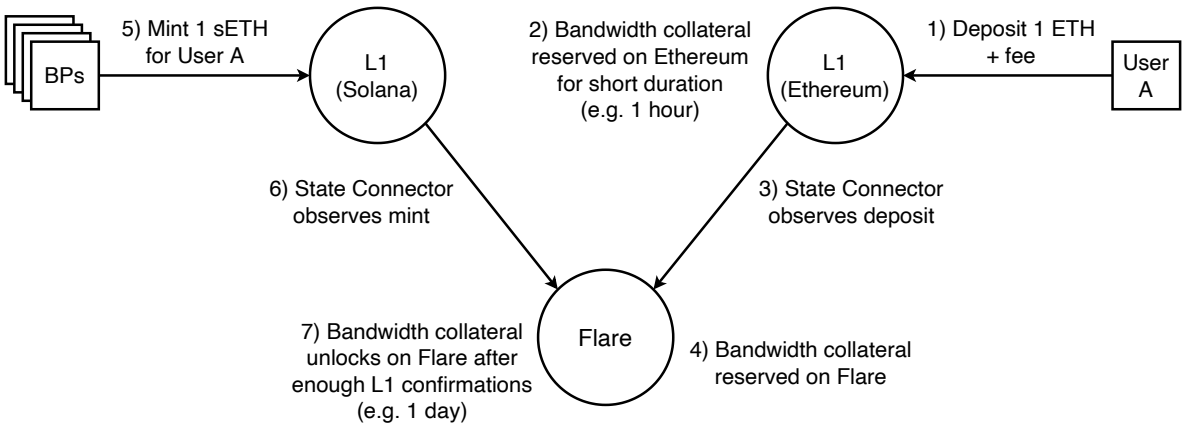

Figure 5: LayerCake minting for a bridge between Ethereum and Solana. User A deposits 1 ETH on Ethereum, as well as a fee, and receives 1 sETH in return on Solana.

In figure 5, a mint flow is illustrated for a user crossing from Ethereum to Solana. User A deposits 1 ETH as well as a fee on Ethereum. This immediately reserves bandwidth collateral on Ethereum. Once the state connector has observed this deposit, bandwidth collateral is then reserved on Flare and 1 sETH is minted by bandwidth providers on Solana for user A. The state connector then observes this mint, which then unlocks the collateral on Flare after enough L1 confirmations are observed such that it's assumed reorgs cannot occur; e.g. after 1 day of L1 confirmations.

3.2 Redeem Flow

A different user may now own the minted asset on the destination chain. The new minted asset owner may redeem the underlying asset by burning the minted asset on the redeem contract on the destination chain. This leverages the same collateral reservation fee mechanism as in the mint flow, where collateral on both the origin L1 and Flare are reserved.

3.3 Faulty Bandwidth Providers

If a bandwidth provider incorrectly mints or unlocks value on either the destination or origin chains, then they are temporarily benched from participation in the bridge by admins whose only authority in the protocol is to bench bandwidth providers after they have executed at least one transfer of user value since being instated as a bandwidth provider. The LayerCake smart contracts only permit bandwidth providers to periodically transfer value up to an amount equal to the bandwidth collateral they have deposited on the origin L1. For example, over the span of an hour, bandwidth providers would be able to transfer up to the amount of value they have deposited as bandwidth collateral.

Figure 6 shows how the risk of a faulty bandwidth provider is mitigated. Bandwidth providers can be benched immediately after minting/unlocking value incorrectly. Then, any bandwidth collateral they had on Flare is immediately seized using the state connector to prove an invalid mint/unlock, and then burned. Inevitably then at the next weekly bandwidth provider rotation time, the invalid mint/unlock will be proven to the origin L1 deposit contract, and the bandwidth provider's origin L1 collateral is seized and used to repair the backing to the user deposit contract back to parity with the minted pool.

3.4 Faulty Admins

Faulty admins attempting to shut down the LayerCake bridge can at most only cause a slowdown of the bridge but never a halt. Admins are only permitted to bench a bandwidth provider after they make at least one value-transfer since being instated as a bandwidth provider each week. Honest bandwidth providers are reinstated automatically if they were incorrectly benched, so admins have the ability to slow down the LayerCake bridge traffic to at worst the total bandwidth collateral moving once per week.

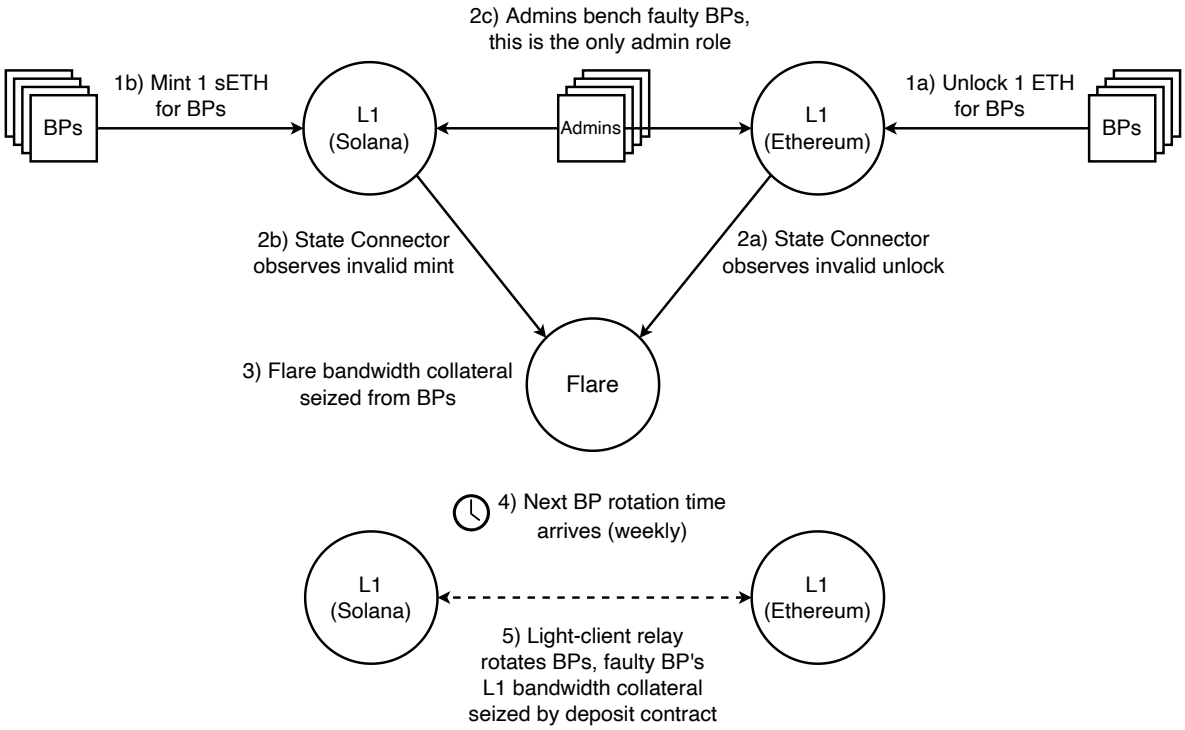

Figure 6: Benching faulty bandwidth providers in response to incorrect minting / unlocking of user value.

Admins are a closed set of entities that do not custody user funds, and only a single admin is required to bench a bandwidth provider. This means that in order to bench a faulty bandwidth provider, the security assumption is that there is at least one honest and live admin.

3.5 Faulty L1s

The only safety assumption in LayerCake about connected L1s is that reorgs on them cannot occur after some period of time, for example 1 day. This is the amount of time that bandwidth provider collateral on Flare is locked after a user crosses the bridge, so that it is available in effect as an insurance payout to the bridge user-deposit pool in the event of a Byzantine attack.

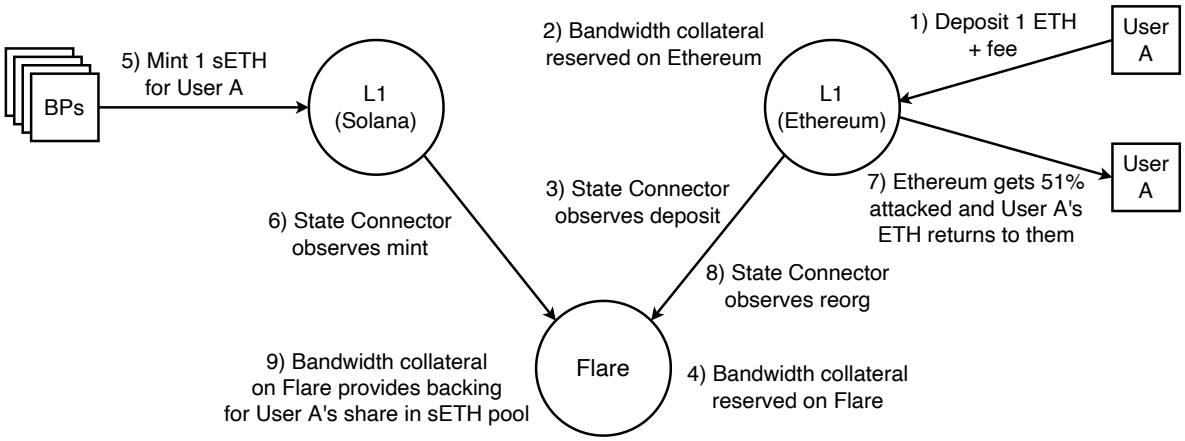

Figure 7: Byzantine attack mitigation against a connected L1.

This is illustrated in figure 7, in the context of a bridge connecting Ethereum and Solana. User A mints 1 sETH as before. If Ethereum then suffers from a 51% attack, resulting in user A's deposited

ETH being returned to them, then this will in turn be observed by the state connector. A portion of the bandwidth collateral on Flare now provides backing for user A's share in the sETH pool.

4 Conclusion

We have proposed a solution to the anti-network effect problem of cross-L1 bridges. The protocol only requires collateral for value in transit across the bridge, and therefore has no requirement for collateralising the total value that has crossed the bridge. The protocol is designed to withstand safety and liveness attacks on connected L1s, as well as faulty bandwidth providers that incorrectly move funds.

References

[1] Avalanche. *Avalanche Bridge (AB) FAQ*. Accessed: 2022-03-06. URL: https://docs.avax.network/learn/avalanche-bridge-faq/.

[2] Axelar Network. "Axelar Network: Connecting Applications with Blockchain Ecosystems". In: *https://axelar.network/wp-content/uploads/2021/07/axelar_whitepaper.pdf* (2021).

[3] Mustafa Al-Bassam, Alberto Sonnino, and Vitalik Buterin. "Fraud and data availability proofs: Maximising light client security and scaling blockchains with dishonest majorities". In: *arXiv preprint arXiv:1809.09044* (2018).

[4] Vitalik Buterin. *Fundamental limits to the security of bridges that hop across multiple "zones of sovereignty"*. Ed. by Reddit. [Online; last accessed 05-March-2022]. 2022. URL: https://old.reddit.com/r/ethereum/comments/rwojtk/ama_we_are_the_efs_research_team_pt_7_07_january/hrngyk8/.

[5] Chainlink. *Introducing the Cross-Chain Interoperability Protocol (CCIP) for Decentralised Inter-Chain Messaging and Token Movements*. Accessed: 2022-03-06. URL: https://blog.chain.link/introducing-the-cross-chain-interoperability-protocol-ccip/.

[6] Rongjian Lan et al. "Horizon: A Gas-Efficient, Trustless Bridge for Cross-Chain Transactions". In: *arXiv preprint arXiv:2101.06000* (2021).

[7] Satoshi Nakamoto. "Bitcoin: A peer-to-peer electronic cash system". In: *Decentralized Business Review* (2008), p. 21260.

[8] NEAR Network. *Rainbow Bridge CLI*. Ed. by Github. 2022. URL: https://github.com/aurora-is-near/rainbow-bridge#security.

[9] NEAR Network. *ETH-NEAR Rainbow Bridge*. Accessed: 2022-03-06. URL: https://near.org/blog/eth-near-rainbow-bridge/.

[10] Rekt News. *Leaderboard*. Accessed: 2022-03-06. URL: https://rekt.news/leaderboard/.

[11] Poly Network. *An Interoperability Protocol for Heterogeneous Blockchains*. 2020. URL: https://poly.network/PolyNetwork-whitepaper.pdf.

[12] Wormhole. *Token Bridge*. Accessed: 2022-03-06. URL: https://docs.wormholenetwork.com/wormhole/existing-applications/token-bridge.

[13] Ryan Zarick, Bryan Pellegrino, and Caleb Banister. "LayerZero: Trustless Omnichain Interoperability Protocol". In: *arXiv preprint arXiv:2110.13871* (2021).

A Appendix

A.1 Faulty State Connector Example

If the state connector suffers from a false decision, then this decision will effectively be ignored by the network. Thus, if the state connector incorrectly observes a deposit, then the branching protocol ensures that eventually each node will ignore the incorrect observation. Furthermore, the state connector

will eventually observe the correct deposit, and thus the LayerCake bridge crossing can continue as normal. This is illustrated in figure 8.

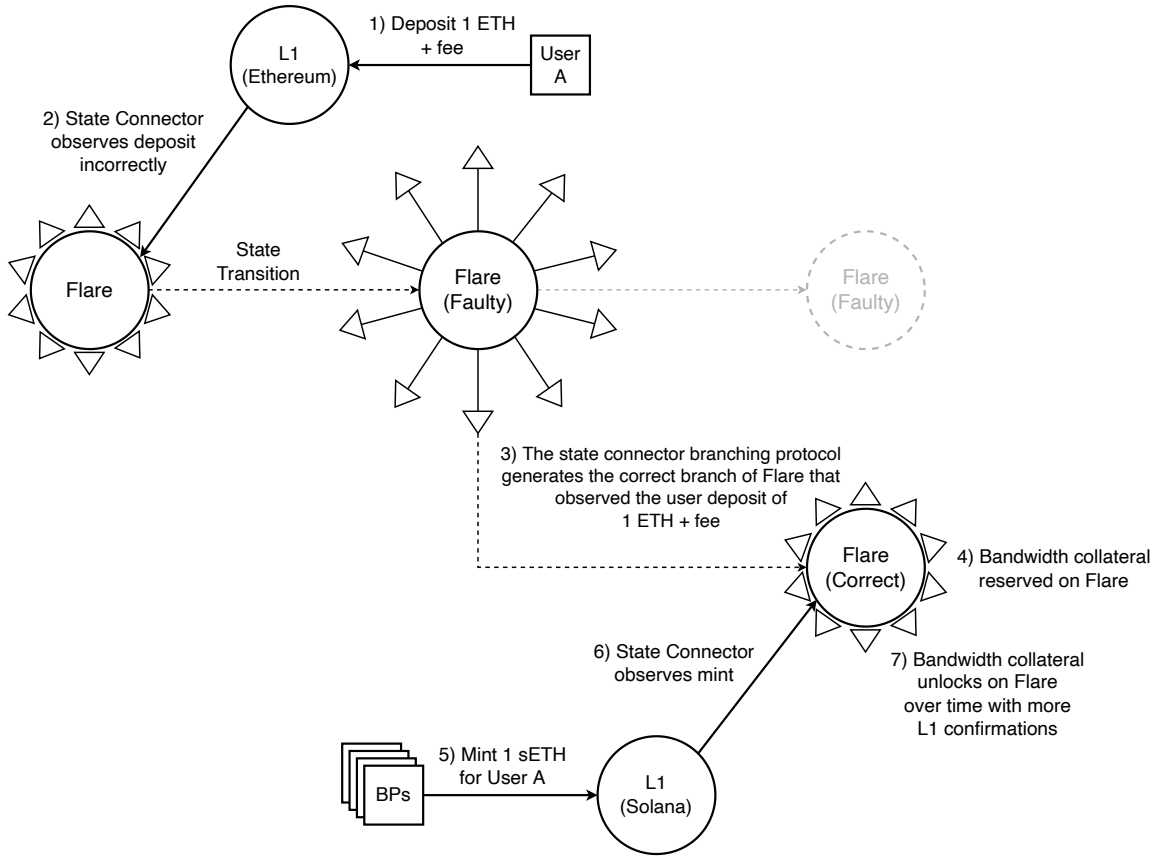

Figure 8: Faulty default state connector operation is repaired implicitly by the branching protocol of the state connector, such that collateral held on Flare always ultimately matches reality on L1s.

A.2 RCR Protocol Optimisations

The RCR protocol is optimised as follows. First, every window of time during the RCR protocol is an opportunity to request L1 proofs, meaning that while a new proof is being requested, prior proofs can be voted on in both the commit and reveal phase. This multiplies the throughput of the RCR protocol by a factor of three, and is illustrated in figure 9.

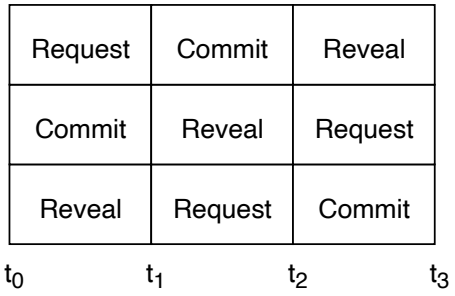

Figure 9: Optimisation of the RCR protocol by overlapping voting rounds.

Second, when requests for new proofs are submitted to the state connector, storage is not invoked. Instead, a Solidity event is emitted. This enables the total cost of the event request transaction to be near 2k gas, i.e. 10% of the cost of a simple payment.

Finally, the gas usage of attestation providers is always constant, despite the number of proof requests they handle, because they construct the proofs into a Merkle tree and simply vote on the Merkle tree root hash. The Merkle tree algorithm can also be swapped out over time to more efficient algorithms without impacting the core RCR protocol which always just votes on the root hash.

A.3 Bandwidth Provider Economics

Suppose there is a LayerCake bridge from Ethereum to Solana that can only move $1/day. The safety assumption is that neither of these L1s can encounter a reorg after 1 week.

On Monday, a user moves $1 from Ethereum to Solana. In order to do this, $1 of collateral is reserved both on Ethereum and on Flare. The collateral on Ethereum unlocks after 1 day, however the collateral on Flare unlocks after 7 days in order to mitigate a reorg during that time. Suppose that the user continues doing this every day for one week and then they stop. This would result in an increasing amount of collateral locked on Flare: on day 7, only $1 would be locked on Ethereum, and $7 would be locked on Flare, see table 1.

Day	Ethereum collateral locked	Flare collateral locked
1	$1	$1
2	$1	$2
3	$1	$3
4	$1	$4
5	$1	$5
6	$1	$6
7	$1	$7
8	$0	$6
9	$0	$5
10	$0	$4
11	$0	$3
12	$0	$2
13	$0	$1
14	$0	$0

Table 1: An example usage of bandwidth collateral in LayerCake, showing how collateral is locked for longer on Flare than on the origin L1. The example demonstrates how the system can always be made whole, i.e. fully backed 1:1 sETH to ETH, using Flare collateral in response to a reorg on a connected L1 that occurs within 1 week.

Bandwidth providers are like insurance companies that steadily earn fees, and then occasionally have to pay out a claim in the event of a reorg attack. The bandwidth providers can control their risk by lowering the bandwidth of the bridge such that it doesn't operate too quickly with too much value at risk if a reorg occurs. If the security assumption is that reorgs on a connected L1 cannot occur after 1 week, then bandwidth providers can move a maximum of 52x their collateral on Flare through the LayerCake bridge in one year. At a fee model then of 10 basis points (bps), they would yield 5.2% maximum on their Flare collateral per year. However, if the security assumption is relaxed to reorgs not occurring after 1 day, then bandwidth providers can move a maximum of 365x their collateral on Flare through a LayerCake bridge in one year. At 10 bps fees, they yield 36.5% maximum per year. Collateral held on Flare can be required to be over-collateralised, such that the value locked on Flare is a multiple of the value on the bridged L1, in order to stabilise the Flare collateral backing in times of high price volatility. At an over-collateralisation ratio of 1.5x, the maximum yield per year is divided by this figure to return 24.3%. Fees may also spike to for example 50 bps in response to user demand in crossing the bridge when it is at capacity. Also, while collateral is being used in LayerCake on Flare, the Flare Time Series Oracle (FTSO) reward can be simultaneously earned by passively delegating to FTSO data providers directly from the LayerCake contracts on Flare.