QUASAR FAST WITHDRAWALS
INSTANT TRUSTLESS EXITS ON PLASMA

BY

SOURADEEP DAS

RESEARCH PRODUCED AT

OMISEGO

Prerequisite Reading

1. MoreVP Plasma Framework
   https://github.com/omgnetwork/elixir-omg/blob/master/docs/morevp.md

Contents

Section - I

Quasar Fast Withdrawals: Instant Trustless Exits on Plasma

Research Implementation:
https://github.com/omgnetwork/plasma-contracts/tree/master/plasma_framework/contracts/quasar

Plasma is one of the earliest and amongst the most popular L2 scaling solutions in Ethereum. It scales by moving transactions to a UTXO based “child-chain” (chch) on top of Ethereum. All transactions on the chch are secured as per the consensus of the underlying layer, and a unique Exit Game design ensures funds are always protected from any third party intervention or manipulation by the chch operator.

Part of this exit game design is the crucial component - “withdrawal period”. When users wish to withdraw funds from the Plasma chch, they initiate an exit process, wherein the process includes a time-based challenge period, set to 7 days (up to 14 days for new deposits). During this period, other participants of the network can challenge the validity of the exit by submitting a proof.

If no challenge is raised within the timeframe, the exit is considered valid, allowing funds to be safely transferred back to the Ethereum mainnet. Hence, the withdrawal period of 7 days enables to protect the integrity of the L2 by disallowing any fraudulent withdrawals, with the assumption that 7 days is enough time for a fellow participant to react and submit a proof. While the 7 day withdrawal period is of the utmost necessity, the same can be perceived as a potential limitation for users who want quick access to funds. That a user would wait 7 days for accessing their funds when they move from L2>L1 is an unrealistic expectation, and so, a solution in this regard is of the highest order to make Layer2s synonymous with the ecosystem.

This is an attempt to reduce the time needed to move funds from a Layer 2 to Layer 1 to <= 24 hrs. (The L2 in context is Plasma or an equivalent L2 without smart contracts) This construction is a trustless way to exchange funds using a Layer 1 smart contract, which would enable almost anyone (not only limited to token-issuers or partners) to provide the service for incentives. The approach also pushes for bringing liquidity providers to the exiters, hence eliminating the requirement of a token market.

1. Overview

The core aspect revolves around Exchanging assets with the help of transaction inclusion proofs.

In brief, the experience for Alice, who has PETH (ETH on Plasma Layer 2) and wants Layer 1 ETH is:

- Alice confirms the UTXO which she wants to exit (obtains a ticket from the Layer 1 contract)
- Alice spends the UTXO in a Layer 2 transaction to the address of the Layer 1 contract owner
- Then she waits for the transaction confirmation, provides inclusion proof to the contract and claims the funds on Layer 1

Alice would pay a fee to the Layer 1 contract owner instead of bearing the gas cost for exiting the UTXO

2. The Process in Detail

The Layer 1 contract is the main component which enables the trustless asset exchange.

The contract has to be set up by the owner using certain parameters to tweak its behavior. We will be addressing this setup as a 'Quasar'. Ideally, there will be multiple Quasars in the network, each of which provide a method for fast withdrawals with differences in their properties such as waiting period or fee.

2.1 Setting Up

Anyone can set up a Quasar, for any definite period of time and with whatever amount of liquidity. The steps-

1. Deploy the contract, setting the following attributes
   - The Owner - the account which will receive the funds on Plasma; should be a fresh account with no history of incoming transactions.
   - Safe blocknum (upper-bound) - This is the latest decided safe block by the owner; to acknowledge tickets for UTXOs from this or older blocks; see reason
   - Waiting period - the time to wait from the contract receiving inclusion proofs to providing liquid funds; to protect against operator + user collusion only; see reason
2. Load the contract with Layer 1 funds for liquidity.

2.2 Using and Withdrawing

Alice has some PETH, this is the process she has to go through in order to get liquid funds instantly:

1. Alice takes a note of the UTXO she wants to exit. She has options to select any Quasar out there, comparing the waiting period and the safe blocknum upper-bounds for each of them.
2. Alice obtains a ticket from one of these Quasars, specifying her UTXO. The intention of the ticket is to reserve the amount from the contract. The ticket is valid for a certain period of time and can only be obtained for UTXOs which are within limits of the specified blocknum upper-bound; see reason
3. After successfully obtaining the ticket, Alice spends the UTXO in a transaction to the Owner. The transaction should only include inputs owned by Alice; see reason
4. After the transaction is confirmed, Alice submits a claim to the Quasar with the inclusion proof of the transaction.
5. The claim gets added to the Quasar's queue with a very short waiting period. The waiting period gives the Quasar Owner an opportunity to challenge in case the Operator colludes with Alice
6. After the waiting period passes, upon executing the queue the contract sends rootchain funds to Alice

2.3 Maintenance

The Owner needs to maintain the Quasar contract from time-to-time in the following ways-

- The Owner has to update the safe blocknum upper-bound in intervals. This is totally dependent on the owner and in an ideal situation of multiple quasar options, even newer Utxos should be able to find one.
- The waiting period can be freely set by the Owner. Since the waiting period is only useful in the rare case of the operator submitting invalid blocks(containing a double spent transaction), the period can be set to even lower than 24hrs, or can even be removed completely if Owner trusts the operator.
- The Owner can periodically exit funds from plasma and use them to refill the Quasar's liquidity pool.
- The Owner can choose to empty the liquidity pool. To retract funds, the owner should first freeze the contract which would stop giving away new tickets. The active tickets are still valid, and the remaining portion of liquid funds not covered by the tickets can be removed.

Mandatory Actions

The Owner needs to necessarily -

- Challenge Claims - Challenge usual fast exit claim within the waiting period from the time of the claim
- Challenge IFE-Claim - Challenge IFE-Claims within 7+x days (IFE finalization + buffer) from the time of the claim

3. Byzantine State

The childchain is considered 'byzantine' if the operator performs an invalid action, such as submitting an invalid block. Normally this results in a mass exit i.e. everyone should exit their funds back to Layer 1.

3.1 Freeze during Byzantine State

If the current state is determined to be Byzantine, the Owner should

1. Avoid updating the Safe Blocknum upper-bound.
2. Freeze the contract to stop giving away new tickets.

The existing active tickets however are valid and IFE-Claims can be used to recover these funds.

Note that freezing the contract has to be done within the Required Exit Period (unless automatic safe block is set), hence time-limited to one-week, but could be done sooner by the Owner to reduce their efforts.

3.2 IFE-Claims

In the case that the operator doesn't include Alice's transaction in a block or is even withholding entire blocks, Alice won't have the transaction inclusion proof to claim funds in the usual way from the Quasar. Similar to the MoreVP in-flight exit game, an IFE-Claim is the way to recover funds:

1. Alice starts an IFE on the transaction to the Quasar. Note that an IFE has to be started before starting an IFE-claim, since starting an IFE on the transaction also proves it to be exitable.

2. Alice calls the IFEClaim method on the Quasar contract
3. Alice should not try to double spend the input that was spent with the transaction to ensure the transaction is always canonical
4. The Quasar owner has to Piggyback the output of this transaction
5. The IFE-claim on the Quasar has a higher waiting period here (buffer + IFE finalization time) in order to have enough time to challenge if it is an invalid claim
6. The Quasar Owner gets the output after the IFE is finalized, and Alice receives the liquid funds from the Quasar contract after the waiting period

Note that if Alice *does* double spend the input

1. The double spend transaction is used to challenge the IFE and is determined to be non-canonical
2. The same transaction is used by the Quasar Owner to invalidate Alice's IFE-claim on the Quasar contract
3. The Quasar owner is safe, but Alice may lose her funds.

4. Possible Incentives for the Owner

4.1 Fees for normal fast exits

The fee should take into account:

1. The cost of challenging in case the Operator allows a double-spend transaction of Alice
2. The fee for providing the service - since the waiting period of every Quasar is customizable, the fee can be dependent on the waiting period, opening multiple options of usability. The fee can also alternatively be a cut from the amount.

Another thing to note here, is since Alice doesn't actually exit any funds from the child chain, she doesn't have to pay for the cost of processing an exit (the bounty). Also, the Quasar Owner has the option to merge multiple UTXOs on the child chain before exiting, which reduces gas. Altogether, less gas is spent on processing exits. Hence the Quasar Owner doesn't need to include the cost of exiting the output in the fee. The design of the fees should take this fact into consideration since, Alice would normally have to pay the bounty for a Standard Exit, but as an alternative here, she has to pay only a fee to the Quasar, which could ideally be of similar size.

4.2 Fees for IFE-Claim

The IFE-Claim's should be disincentivized by having a higher fee and should only be used when necessary. The IFE-Claim process requires the Owner to Piggyback. This cost should be borne by Alice and can be included with the fee to the Quasar.

The IFE-Claim is already disincentivized to some extent by having a higher waiting period than a normal IFE. So the fee for an IFE-Claim could simply include

1. Fee of normal fast exit
2. Cost of piggybacking the output
3. Cost of potentially challenging the IFE-Claim

5. Potential Attacks and their Mitigations

5.1 Attacks by Alice

DoS tickets

Obtaining a ticket from a Quasar blocks the amount from the Quasar's real limit. So obtaining several tickets without actually using them will block the capacity of the Quasar. The solution is to have a bond, which can be taken for providing the ticket, and can be returned on claiming the tickets. This can be looked at as to be similar to the exit bond. There can be other approaches too, where the service fee is taken while obtaining tickets.

Ticket for a fake UTXO

A ticket can be obtained for a fake UTXO, but while claiming funds the inclusion proof of the transaction between Alice and Owner is required.

Alice attempts to exit from the UTXO after submitting the claim

Alice needs the inclusion proof in order to claim liquid funds. So assuming the transaction to the Quasar owner was confirmed, if Alice tries to exit, it will be challenged with that same inclusion proof.

If Alice starts an IFE-Claim, the same double spending transaction can be used to challenge both the IFE and the claim.

Alice uses the inclusion proofs of same UTXO twice

Alice can get the ticket for any given UTXO, so it is possible for Alice to try claiming funds twice with the same inclusion proof. The solution is to have a utxo map maintained by the Quasar which can eradicate this. Also in the case of Alice trying to use a different Quasar with an older proof will not succeed since the recipient of the UTXO will not be the Quasar owner.

5.2 Attacks by the Quasar owner

Owner races to empty liquid funds

For situations where The Owner immediately generates a ticket and tries to empty the liquidity funds before Alice claims, the owner still cannot generate a ticket with a value higher than the real(updated) capacity.

Owner retracts liquid funds from contract

The Owner cannot try to take away liquid funds and empty the contract between the time Alice takes to claim after transferring. Since the ticket reserves funds for Alice, Alice can safely extract funds as long as the ticket is still valid.

5.3 Attacks by the Operator

Operator publishes invalid block

If Alice's transaction to the Owner is included *before* the invalid transaction, the Owner can ‘standard exit’ to get the funds

IF Alice's transaction to the Owner is included *after* the invalid transaction, the Owner can in-flight exit to get the funds

After Operator goes Byzantine

If the operator goes Byzantine but still submits blocks, the users can continue to make successive transactions and get proofs for them. This however, will not allow them to use the Quasars. For the older UTXOs (within the limit of the blocknum upper bound specified by the owner) that are transferred, the Owners could still exit, but for successive transfers after the operator goes byzantine, tickets will not be obtainable as it would be a UTXO from a blocknum that was after the safe upper-bound

Operator doesn't include Alice's transaction to Owner

If the operator withholds the transaction, Alice can start an IFE-Claim to recover her funds

Operator allows a double-spend transaction of Alice

This is the only rare chance which brings in the usage of the waiting period.

The issue arises with: Alice spends UTXO1 in TX1 to Malorie, and then Alice colludes with the Operator to include a transaction TX2, this time to the Quasar Owner spending the UTXO1 again. Though the Owner cannot exit this, the contract would still allow Alice to claim liquid funds through UTXO1 (given UTXO1 was within the safe blocknum-upper bound)

The waiting period set for the Quasar protects against this. The Owner should validate their queue, and challenge Alice's claim by revealing an older spend of UTXO1 within the waiting period. Since, this is rare, happening only when the Operator colludes with Alice, if someone trusts the Operator, or the Operator themselves could run a Quasar without the need for validation, with a zero waiting period.

6. Alternative Modifications to the approach

6.1 Single Liquidity Pool

An alternative is to have a single liquidity pool instead of having separate contracts with their own liquidity.

- fee can be distributed among liquidity providers
- removes the requirement of running separate contracts for each liquidity provider

- However, this could potentially prevent from keeping the waiting period low. Since every swap is from the same pool, Liquidity providers are no more accountable for the safety of their liquid funds.
- can be an option when you want to delegate the task of challenging to a small set of other trusted user(s)

6.2 Automating maintenance of the Safe Blocknum upper-bound

In the current design, there is a requirement for the Quasar owner to constantly keep updating the Safe Blocknum upper-bound. An alternative could be - Provide tickets for a UTXO if either one of these satisfy:

- UTXOs is older than (Latest_Block_timestamp - Buffer), ideally the buffer could take the value of the minimum validation period. or
- UTXO is from a safe block (determined by safe blocknum upper-bound) set by the Owner

6.3 Output funds in different form

Since the fast withdrawal approach is essentially a swap of assets between users on different layers. The swap can be extended to provide fund outputs in any form. Instead of providing Liquid funds, a deposit transaction to another Layer 2 could be created (enforcing a swap instead of an exit).

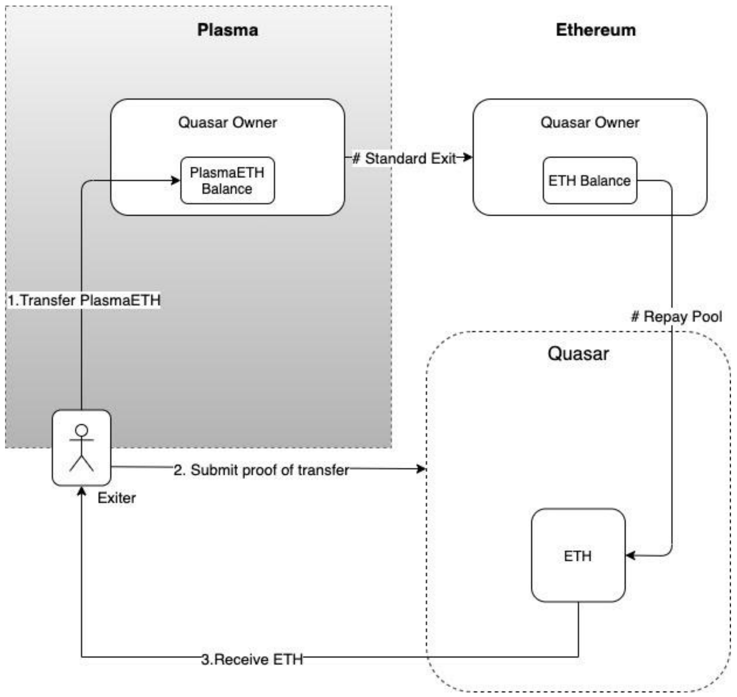
Section - II

Quasar Pool Design

Note: the Quasar pool mentioned here is a plug-in to pool-in liquidity from several lenders and make operations easy for someone running a Quasar contract. It is not an inherent requirement and the most ideal trustless Quasar should have funds pooled in only by the Quasar Owner.

This is a continuation that expands on the idea of using a single Liquidity Pool. This also outlines the trade-offs made between trustless operation and usability for the first implementation.

1. The inception of a Quasar pool



A trustless Quasar with respect to the flow of funds has been illustrated (on the right) (this does not display tickets, challenges, IFEClaims)
Here, the Quasar operator would pool in funds on the ETH pool. (steps marked # are optional)

Why should an ideal trustless Quasar have funds pooled in only by the Quasar operator?

A trustless Quasar shouldn't need to trust anybody including the plasma operator. If the Quasar owner does not trust the plasma operator, they may or may not accept the validity of the tx inclusion proof (from step 2 of the above diagram). In which case a challenge period is utilized to prove any false claim wrong (after step 2). So ideally, the funds at stake on the Quasar should be the owner's own funds and they should be liable to challenge whenever there is a need to protect the pool.

Now also, the need for a challenge would only come up in case there is an invalid inclusion proof (in other words when the child chain is byzantine - a mass exit scenario)

In an implementation where the plasma operator is also running the Quasar - they trust themselves and hence have an opportunity to remove the challenge period.

This brings us to:

Change 1. Quasar with no challenge period

So we have - a plasma operator that trusts themselves that they won't go byzantine and hence, pools in funds on the Quasar. The Quasar's liquidity pool can be further extended to allow external users to provide liquidity, enabling shared profits and avoiding the requirement for the owner to put up a large sum on the pool themselves.

So the trust model gradually changes from -

Initially: Nobody trusts anybody.

With no challenge period: plasma operator trusts -> plasma operator

With multiple suppliers pool: There are 2 choices:

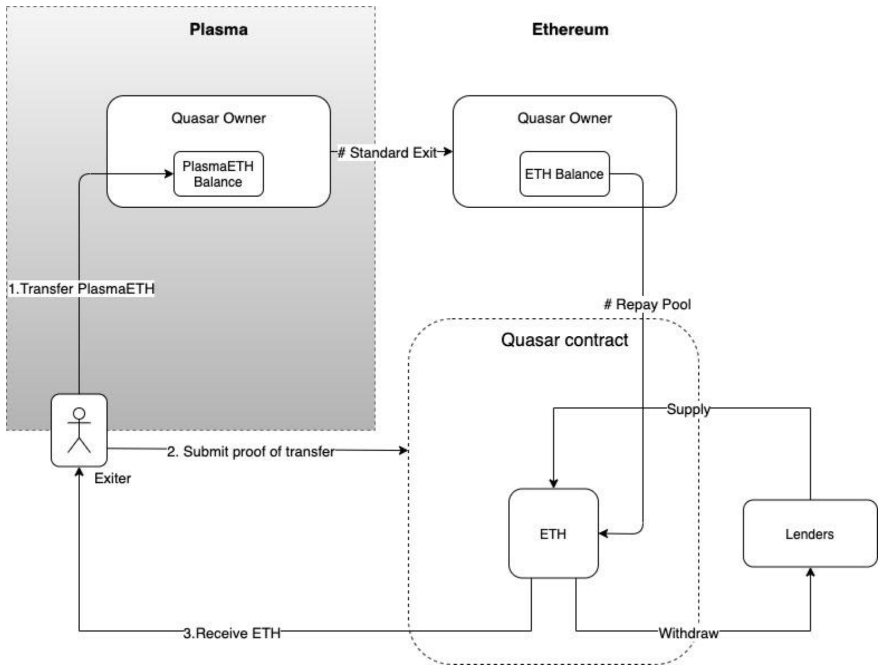
- *A) With collateral by operator:* plasma operator trusts -> plasma operator
- *B) Without collateral by operator:* Lenders trust -> plasma operator

For the first implementation, we selected option B)

This brings us to:

Change 2. Quasar pool with multiple suppliers

The integrated structure is now illustrated (on the right)



This pool operation can be viewed to be similar to Compound (where the Quasar contract is the compound supply pool). The borrowers are the exiters, but the one who repays is the trusted plasma operator.

While Compound generates returns on the basis of block time, here we do it on the basis of the number of exits.

2. Returns calculation

Every output sent to the Quasar owner on Plasma, will allow withdrawals of output_val - quasar_fee from the Quasar contract on Ethereum.

This quasar_fee margin will be distributed among the pool of lenders in the ratio of each lender's stake. The calculations are done in a similar way to a defi lending protocol, using qTokens.

Each token supplied to the pool has its exchangeRatio (k).

When liquidity is supplied to the pool, qOMG = OMG / k is minted for the supplier.

On withdrawal, qOMG is burnt and OMG = qOMG * k is withdrawn.

The exchangeRatio (k) determines the returns for suppliers.

Fees accumulate in the pool every time a fast exit happens, and k updates to k' = f / Tq + k

Where

- f = fixed quasar_fee from every exit
- Tq = total qTokens supply

2.1 Proof of Equation

Return rates are dependent on the qtokens and the exchange rate k, for that specific token, at that point of time.

Assume, initially the pool is filled by n suppliers,
The total supplied assets

$$T_a = a_1 + a_2 + a_3 + .. + a_n$$

where, amount supplied by 'm' is a_m
Since, this is the initial point and no exit has taken place k has been same for every supplier
Equivalently the total amount of qtokens it translates to,

$$T_q = q_1 + q_2 + q_3 + .. + q_n$$

where,

$$q_m = \frac{a_m}{k} \qquad \text{--------------------- (1)}$$

so,

$$T_q = \frac{a_1 + a_2 + a_3 + .. + a_n}{k}$$

$$T_q * k = a_1 + a_2 + a_3 + .. + a_n \qquad \text{--------------------- (2)}$$

<u>After first Exit</u>

amount(fee): f to be distributed among T_q tokens

$$\text{fee per token} = \frac{f}{T_q} \qquad \text{--------------------- (3)}$$

And,

$$k' = \frac{f}{T_q} + k$$

Maximum amount that can be withdrawn from pool now $= T_q * k'$ ------ (4)

$$= T_q(\frac{f}{T_q} + k)$$

$$= f + T_q . \mathrm{k}$$

= fee from first exit + total initial amount

[from (2)]

Withdrawal amount for 'm' = $q_m * k'$ ------ (5)

$$= q_m * \frac{f}{T_q} + q_m . k$$

$= \text{fee for } q_m \text{ tokens} + a_m$ *[from (3) and (1)]*

<u>Case 1</u>

After first exit, withdrawals amounting *W* happen and now

Total $= T'_q$,where $T'_q < T_q$

Now, after next exit

$$k'' = \frac{f}{T'_q} + k'$$

Maximum amount that can be withdrawn from pool now $= T'_q * k''$

$$= T'_q(\frac{f}{T'_q} + k')$$

$$= f + (T_q - W).k'$$

$$= f + T_q . \ k' - W.k'$$

= fee + max amount first round - already withdrawn

[from (4) and (5)]

<u>Case 2</u>

After first exit, additional W assets are supplied and now

Total $= T'_q$,where $T'_q > T_q$

Now, after next exit

$$k'' = \frac{f}{T'_q} + k'$$

Maximum amount that can be withdrawn from pool now $= T'_q * k''$

$$= T'_q\left(\frac{f}{T'_q} + k'\right)$$

$$= f + (T_q + W).k'$$

$$= f + T_q.\ k' + W.k'$$

= fee + max amount first round + new suppliers total

[from (4)]

Sample run, return calculations https://github.com/omgnetwork/brainstorm/tree/main/quasarPool_returns

2.2 Returns Per Exit

Note, the fee has to be distributed for each qToken in the supply of the pool. (the qTokens denote the total supply of the pool)

So a suppliers return per exit = no of qTokens they have * fee per qToken

If there are many lenders to the pool, the fee per qToken would be less, If there are small no of lenders, the fee per qToken is more. We want the number of qToken in the pool to adjust automatically according to their return requirements. When people supply, more qTokens are in supply. When people withdraw, less qTokens are in supply

2.3 Returns Per Week

Calculating returns per week would give a clearer idea of the movement in the pool as this brings the number of exits into the equation. The returns are compounded.

Returns per week = (quasar_fee per exit * no of exits in the week * no of your qTokens) / qToken total supply

This implies the returns will increase/decrease with the increase/decrease in the number of exits

Scenario 1

When the number of exits increases, pool supply is idle:

Returns per lender increases, which may attract more lenders to supply, which in turn could again normalize the return to the average rate,

If new lenders aren't attracted, the existing suppliers enjoy higher rates.

Scenario 2

When the number of exits decrease, pool supply is idle:

Returns per lenders decrease, which could mean some lenders withdraw from the pool. The ones who stay in the pool enjoy higher returns. This normalizes the return rate to average again

If lenders don't leave, all of them are content with the lower return rate

Scenario 3

When no one supplies to the pool:

The first person to supply owns all of the qTokenSupply, and gets the whole return to themselves.

This makes it a self-regulating pool, in terms of the need or use of fast exits in the system. A higher number of fast exits would need a higher liquidity pool size, and scenario 1 shows that this will incentivize people to supply more. Similarly if there aren't enough fast exit happening, we are better off with a small pool size(since that would be enough liquidity for a smaller number of exits)

Section - III

Quasar for cross-L2 fast exits

With regards to enhancing the experience of using Plasma, entry and exit are two things that often come up to be a barrier for usability. Ideally, both entry and exit should be time and cost effective. While, entries (deposits) are fast, but not as cheap, exits are neither. The Quasar attempts to solve this by allowing fast exits off Plasma, which still isn't very cheap because of the Ethereum transactions and the gas costs associated with them.

This is an attempt to enforce a method where the Quasar could allow Fast Exits onto a different L2, potentially without touching Ethereum, and removing the gas costs hence, from the equation.

1. Direct exits to a L2 (Optimism)

1.1 The Process

To do this, the Quasar construction needs to be split into two parts:

1. L2Quasar - rests on Optimism - the primary contract that someone interacts with, holds the storage and the quasar pool
2. L1QuasarScope - rests on Ethereum - the contract that is used only when required, whose role is to verify and call a method from the L2Quasar

The Quasar gives out liquid funds after verifying transaction inclusion proofs, which directly needs the block roots committed to the Plasma contract on L1. A contract on L2, which is meant to be the Quasar, cannot read these roots, hence is the requirement of the L1QuasarScope contract on L1 which verifies before triggering the appropriate method on L2Quasar.

The UX is still maintained-

1. Quasar owner loads up the L2Quasar contract with ETH
2. Alice, who wants to fast exit onto Optimism, obtains a ticket on L2Quasar, providing the UtxoPos and the amount to reserve.

point of difference: Additional details like Tx and inclusion proof are not to be provided in this step. The first iteration of quasar used the tx and proof to extract value of output and to verify Alice owned this UTXO, but doesn't confirm if she hasn't spent it. Though this was a good way to stop people from bothering others UTXOs, we can't do this since we don't have access to Plasma block roots on L2. So we go without the need to specify it, asking Alice to specify both the utxoPos and reserved amount (increasing the bond size a bit more)

3. Alice spends the Utxo in a tx to the Quasar owner on Plasma
4. Now, the Quasar owner verifies off-chain that Alice has spent the utxo with the ticket correctly in a tx to the quasar owner and takes the authority to call the L2Claim method on L2Quasar which would redeem the ticket and transfer equivalent amount to Alice from the L2Quasar Pool.

5. However, if the Quasar owner doesn't send the funds to Alice, in other words, doesn't call the L2Claim method for Alice's Utxo, the L2Quasar would still have the ticket active and unclaimed. And, only in this case, Alice would have to use the L1Claim from L1QuasarScope which verifies and triggers the L2Quasar contract to send her funds. Alice would provide the Utxo, spending Tx and the inclusion proof to L1Claim. The L1QuasarScope verifies the Tx was correctly formed, valid and calls the L2Quasar through a bridge contract to send Alice her funds and marks her ticket as claimed.

1.2 Additional Vectors

- Ideally Alice wouldn't have to use the L1QuasarScope contract. The UX is step 2-4. Step 5 is required only when the Quasar owner acts malicious. Since there is no advantage for acting malicious here, step 5 is unused.
- The utxoPos and the amount reserved should be correctly specified by Alice while obtaining a ticket. If Alice provides a false reserved amount, the ticket will not be correctly claimable by Alice(if needed) and the quasar owner will have no obligation to send Alice her funds.
- The IFEClaims and ClaimChallenges can be used in the same way, calling the L1QuasarScope where the proofs are verified and calls L2Quasar that updates the state.
- ProcessClaims, withdrawing unclaimed bonds can all be done through the L2Quasar. The L1QuasarScope comes in whenever there is a conflict and a proof has to be verified.
- Optimism's cross-domain message is used to call L2Quasar from L1QuasarScope
- Quasar owner/maintainer can be used interchangeably here, the maintainer can be given the tasks and the owner a sole address for receiving.

Since the transactions happen over L2Quasar on the happy path, this allows avoiding gas fees on Ethereum, and makes these exits cheaper.

1.3 Claim methods

The L2Claim method-

- to be called only by the Quasar owner
- transfers funds to Alice
- mark ticket as claimed

(or)

The L1Claim method-

- to be called by Alice
- provided with Utxo to spend, Tx, proofs
- after verification on L1, calls L2Quasar through bridge
- L2Quasar transfers funds to Alice
- marks ticket as claimed

1.4 Potential optimizations

- The automatic safe blocknum logic does not work here, since the L2Quasar doesn't have access to the Plasma blocks. The manual safe blocknum update by the maintainer needs to be considered,

or a time based update. A positive point is that the regular safe blocknum updates are not transactions on ethereum (hence, saving some cost).
- Someone who wants to fast exit must have eth on optimism to pay for calling the obtainTicket method. Since, the obtainTicket method can be called by any one (not limited to someone owning the UTXO), there could alternatively exist relayers who could call obtainTicket on behalf.

2. Exits via hopping through Ethereum

L2s on Ethereum function by taking asset deposits on L1 which then provides the equivalent assets on L2 (a deposit). The prospect of this approach is to use the deposit method with a quasar to provide a choice of endpoints for the users to receive their liquid funds. Instead of sending the funds over Eth, the quasar would call the deposit methods to these other L2s on claiming. This specifically can be a way to Fast Exit on a L2 which doesn't have a L1 -> L2 bridge. (for eg. xDAI, Loopring) This, however requires Ethereum Transactions and hence gas fees, only saving the costs of one transaction if a user wants to move assets between L2s.

changing the current process from,
Process Exit on Plasma -> funds on Eth -> Deposit on other L2

into
Claim/Process Fast Exit -> Deposit on other L2

2.1 The Process

- Requires the deposit method of the (receiving) Layer2's contracts on Ethereum to allow specifying - the recipient/from account. i.e. Depositing to the Layer 2 on someone's behalf should be possible.
- The Quasar is to remain exactly similar to the original quasar design, the only difference being in the way liquid funds are given out. Contrary to the original quasar design, the way to retrieve the liquid funds depends on the user calling claim() where they specify the form/chain/denominations in which they want to receive their liquid funds.