In our previous two parts of the cross-domain thesis saga, we've primarily focused on the issues that arise technologically when you split up the stack and what changes are required for a cross-domain, modular world to exist. We've covered many developments in the works to solve problems that come naturally in a cross-domain setup. However, in the series's final (and third) part (more like a series of pieces constituting the part), we'd like to focus more on the user experience. We'd like to examine how modularity, customisation and specialisation can help create better applications. This final chapter of the series will look at the exciting and unique creations and possibilities that loom in a modular, cross-domain world for developers to create web2 UX with web3 verifiability.

The reasoning behind building modular shouldn't just be to fit into a narrative, nor just for the sake of it — but rather because it enables us to build better, more efficient or customisable applications. Many incredibly unique capabilities open up when building modular and specialised systems. Some of these are more obvious than others to the naked eye, while others are less obvious unless told. As such, we aim to provide a general overview of modular systems' powers beyond those you already know, such as scalability.

We feel that one of the powers that modularity provides a developer is the ability to build highly customisable and specialised applications that bring a better experience to the end-user. An example of this, which we've previously discussed on a high level, is the power of setting rules or reordering the sequence of which transactions are executed.

Verifiable Sequencing Rules

(hereby VSR) are one of the interesting opportunities that controlling ordering provides, especially for a developer interested in building a "fairer" trading system in terms of execution. Obviously, the relationship of Loss-Versus-Rebalancing (LVR) for liquidity providers is slightly outside the scope of this article, so we will refrain from touching upon this too much. Keep in mind that the setup we're going to explain is primarily for an AMM and not an order book model — although the specialisation, low cost and customisation of opcodes/gas limits (or even alt-VMs) that modularity provides could be used to build very efficient on-chain order books that derive scalability, deterministic DA and more from an efficient underlying DA layer. Furthermore, CLOBs (or even CEXs) would also benefit greatly (from a trust perspective) from utilising verifiable ordering rules catered towards their particular setup. In off-chain setups, some notion of either zero-knowledge or optimistic execution backed by crypto-economic security would obviously be needed.

VSRs are especially interesting when we consider the fact that most non-informed flow (retail users) haven't (or are unlikely to) adopt protection methods. Neither have most wallets/DEXs implemented private mempools, RPC or analogous methods. Most trades are submitted directly through front-ends (whether that is an aggregator or a DEX's frontend, although the rise of intents helps somewhat here). As a result, unless applications directly interfere with how their flow and orders are handled, the end-users are likely to be provided with less-than-ideal execution.

The power of VSRs is evident when we think about wherein the transactional supply chain sequencing lies. It lies where transactions get ordered (or included) by a specialised actor, generally based on some auction or base fee. This ordering is incredibly important; it determines what transactions get executed and when - essentially, the one who holds the power of ordering holds the ability to extract or be paid to extract MEV, often in the form of priority gas fees (or tips).

Because of this, it can be interesting to write rules on how sequencing should be handled to provide fairer execution of trades (in a DEX setup) to the end-user. However, you should largely refrain from such rules if you're building a generalised network (you'll end up hurting some).

Additionally, because some MEV is important; arbitrage, liquidations, etc. - one idea is to have a "highway" lane at the top of the block (ToB) specifically for, perhaps, whitelisted arbitrageurs and liquidators who pay a higher fee and share some of their revenue with the protocol.

Highway (ToB) and Rest of Block

In the paper Credible Decentralized Exchange Design via Verifiable Sequencing Rules

, Matheus V., X. Ferreira and David C. Parkes present a model where the sequencer of a block is constrained to a set

execution ordering rule (and those constraints are verifiable). In the case of non-compliance with the set rules, a watcher could generate a fault-proof (or since the constraints would be mathematically verifiable, you could also imagine a ZK circuit with those constraints with a ZKP acting as a validity proof). The main idea is essentially to provide the end-user (trader)

with an execution price guarantee. This guarantee ensures that the execution price of a trade is as good as if it were the only trade in a block (obviously, some degree of latency is involved here if we assume a Buy/Sell/Buy/Sell ordering based on first-come-first-serve). The base idea of the proposal in the paper is that these sequencing rules will limit the builder (in a PBS scenario) or the sequencer only

to include trades in the same direction (e.g. Sell/Sell) if they're executed at a better price than what was available at the top of the block (or if there are simply no more swaps in the other direction following them). Furthermore, if there's a situation where you have a sell at the end of a long range of buys, then this sell would not be executed (e.g. buy, buy, buy, sell), which could indicate a searcher (or builder/sequencer) using those buys to move the price in their favour). This essentially ensures that a user is guaranteed by the protocol's rules that they won't be used to give a better price to someone else (i.e. a lot of MEV) or given a poor price slippage as a result of priority fees. Obviously, the downside of the rules here (in a situation where there's either a large number more sells than buys or vice versa) is that you're likely to be given a relatively poor price as long tail.

Having these rules as a pure on-chain construct is nigh on impossible with a general smart contract platform, as you're not controlling execution and ordering. At the same time, you're also competing with many others, so trying to enforce those at the top of the block with priority fees would be needlessly expensive. One of the powers of a modular setup, where you control the aforementioned powers, is that it allows an application developer to customise how their execution environment should function. Whether that is sequencing rules, using a different virtual machine or making custom changes to existing ones (such as adding new opcodes or changing gas limits) is really up to the end developer, depending on their product.

In the case of a rollup utilising some data availability/consensus layer, as well as a settlement layer for liquidity, a likely setup would look something like this:

Architecture for a rollup w/ settlement and DA layer

Another possible idea is that of optimal swap splitting; take the case of where you have a single pool (or maybe more); how do we swap large orders (that would cause large slippage)? Is it fair for the end-user if this trade is executed across sequential blocks (or later in the block if it fits within the VSRs)?

Swap Splitting by Sequencer

If the end-user cares about latency (could be alpha-related), then this particular user would likely not want to have his order split up into an order of sequential blocks. However, this is unlikely to be a common occurrence, and optimising for swap splitting on larger orders could lead to more efficient execution for the vast majority of users. Irrespectively, one worry is that MEV searchers could possibly be aware of these sequential trades and try to position themselves to be included before/after the aforementioned trader. However, the total value of extraction will likely be much smaller due to the bite-sized split trades over a sequence of blocks.

Another interesting idea we've mentioned previously in posts is using Frequent Batch Auctions (FBA), championed by the legendary Eric Budish and co here, to process transactions in a batch auction instead of serially. This is to help find coincidences of wants (CoW) and build arbitrage opportunities into the market mechanism design. This also helps to "combat" part of the latency game in continuous block building (or priority fee wars in serial blocks). Thanks to Michael Jordan (DBA) for bringing this paper to our attention and for his excellent work in his preparation for moderating the Latency Roast (highly recommended reading material). Implementing this as part of the fork-choice and sequencing rules for a rollup is also an interesting setup that developers can use, and we've seen it gain significant traction over the past year, especially by Penumbra and CoWSwap but also by many others (such as this paper by Andrea Canidio and Robin Fritsch). A possible setup for this could look something like this:

In this setup, there are no first-come-first-serve nor priority gas fee wars, but rather an end-of-block batch auction out of the accumulative orders during the time between each block.

Generally, in a world where much of the trading has moved to non-custodial "on-chain" venues, an FBA venue is likely one of the more efficient ways for "true" price discovery, depending on block times. Utilising FBA also means that since all block orders are batched up and aren't revealed until the auction has concluded (assuming some encrypted setup), front-running gets reduced considerably. The uniform settlement price is key here (in terms of providing "fair" execution) since there's no point in reordering trades (if you follow the rules set by the protocols, i.e. bond/stake is important).

Something also interesting to point out is that even back in 2018, a setup akin to what we just presented was talked about on the Ethresear.ch forums (see here). Again, Gnosis was before their time (and have generally been very involved with most of the active research in the Ethereum community). In the post, they refer to two papers; these present a batch auction mechanism on Plasma (sort of the prequels to modern-day rollups) where each batch accepts orders to buy ERC20 tokens for other ERC20s at some maximum limit price. These orders are collected over a time interval (as pointed out earlier) and provide a uniform settlement price (USP) on all token pairs. The general idea behind the model is that it would help eliminate much of the front-running evident in popular AMMs.

Another important thing to note is that in these setups, the sequencer(s) will likely need some incentivisation to execute (and enforce) these rules mentioned above. This is often overlooked and handwaved away, but much of the infrastructure of blockchain networks are run by professional companies with completely different cost-basis than a regular home-staker. Incentivisation is, in general, an essential part of secure infrastructure implementations. In the case of incentives aligning with the rules enforced, they (sequencers/builders) are also much more likely to make a greater effort (e.g. specialise more). This means these setups should also have an active market (i.e. be decentralised). Obviously, these sorts of markets are centralising in the sense that the cost of capital to specialise is likely to be high. As such, the smartest (and richest) will likely integrate and specialise to extract as much value as possible. Here, the possibility of exclusive order flow to some actors (as alluded to earlier in the 2nd diagram) is also a possible arrow in the knee, causing an increase in centralisation if block bids are based on the highest fee/best solution, which equals winning the right to propose. A general baseline fee may be enough, but it doesn't really push ordering actors toward specialisation or improvements. As such, you might want to introduce some notion of satisfaction of a trader's outcome with an incentive mechanism that fits with your particular case.

This is clear to most, but it is still relevant to mention when talking about ordering on a rollup level. If you're able to control ordering (as alluded to earlier), it allows for much easier "extraction" (monetisation arguably a better word) of value as a protocol. This is because you control the power to reorder transactions, usually based on priority fees on most L1s (MEV-boost-esque setups). This provides you with a priority fee paid by sophisticated actors looking to extract value somewhere on the chain. These actors are usually willing to pay quite a hefty sum (up until it doesn't provide value anymore) to be able to be the first to extract the said value. However, most current rollups are primarily first-come-first-serve. Most MEV extraction happens via latency wars (and spamming), which puts severe strain on the rollup infrastructure. It also leaves out significant value to be captured. Because of the aforementioned reasons, we're likely to see more and more rollups start to implement ordering structures that have some notion of priority fees (such as Arbitrum's Time Boost).

Another example we like to use is the Uniswap one. Currently, Uniswap as a protocol "creates" a large amount of inefficiencies. These inefficiencies get exploited by actors looking to extract MEV (arbitrage, at the expense of liquidity providers). At the same time, these actors pay large sums for the right to extract this value, but none of it lands in the hands of the Uniswap protocol, nor does it land in the hands of its token holders. Rather, a large position of this extracted value is paid as priority fees to Ethereum proposers (validators) via MEV-Boost for the right to be included at some point in a block that allows for the value to be captured. So, while a large amount of MEV Uniswap flows and ebbs through Uniswap contracts, none of it gets captured by it.

In a world where Uniswap has the control of ordering within the protocol (and the ability to extract priority fees from searchers), it can monetise on these inefficiencies - perhaps even pay out some of these profits to token holders, liquidity providers or others. This world looks more and more likely with a move towards off-chain execution (and Ethereum as a settlement layer) with changes to Uniswap, such as UniswapX and more coming to life.

If we assume some PBS setup for a rollup, the flow and monetisation areas look somewhat like this:

From this, we can extrapolate sequencer/proposer on a rollup monetisation as a function of:

Sequencer Yield = Issuance(PoS) + SumOfFees(+priority) - cost of DA, state pub & storage

A great way to see how much value (especially arbitrage) is extracted currently on Ethereum can be found on Mevboost.pics, and it gives a great overview of how much value is actually extractable from the inefficiencies created.

Furthermore, the separation of priority fee gas wars to off-chain constructs allows for siloing MEV extraction to execution environments - helping to contain supply chain disruption. However, considering the fact that if leader election happens on the rollups (and sequencers control inclusion), most (if not all) MEV will be extracted on the rollup/off-chain system - leaving little to the underlying constructs unless priority fees for DA layer inclusion, settlement layer economies of scale from liquidity

consolidation or other.

Something that might also be worth clarifying is that many of these constructions could function as just pure off-chain constructs, without any verifying bridge or as powerful security guarantees. However, you're obviously making some trade-offs there. We're starting to see more of these pop up, both existing and in stealth. I guess a point to make is that a modular setup doesn't necessarily mean being a rollup.

The ordering rules described above represent one example where fine-tuning infrastructure can greatly benefit an application built on top. In the next sections of part 3, we aim to provide more examples of these fine-tunings. And as always, If you're a builder working on applications that utilise, in this particular case, ordering to your advantage - we would love to talk to you.

If you're interested in reading more about the relationship of MEV in modular setups, we recommend our Modular MEV series - p1 & p2.

Thanks to Quintus (Flashbots), Mike Neuder (EF), the CoWSwap team and many others who've been pivotal in shaping our views.

Thanks to Mathijs van Esch (M11), Dougie de Luca (Figment Cap) and Alex Beckett for discussion or review leading to the release of this article.

Writer:

.css-3uyxxe{max-width:100%;width:100%;}

.css-11j27fm{display:block;background:linear-gradient(to right, rgba(var(--groupBackground), var(--shadeGroupBackground)), rgba(var(--groupBackground), var(--shadeGroupBackground))),linear-gradient(to right, rgb(var(--background)), rgb(var(--background)));color:rgba(var(--foreground), var(--shadeTextSecondary));font-size:1rem;position:relative;border-radius:1.5rem;margin-left:auto;margin-right:auto;width:100%;--relativeBackground:var(--groupBackground);pointer-events:none;}.css-11j27fm::after{content:"";border-radius:1.5rem;position:absolute;top:0px;left:0px;right:0px;bottom:0px;pointer-events:none;border:1px solid rgba(var(--foreground), var(--shadeForegroundSecondary));-webkit-transition:125ms ease-in-out border;transition:125ms ease-in-out border;}.css-11j27fm::before{content:"";border-radius:1.5rem;position:absolute;top:0px;left:0px;right:0px;bottom:0px;pointer-events:none;-webkit-transition:125ms ease-in-out box-shadow,125ms ease-in-out background;transition:125ms ease-in-out box-shadow,125ms ease-in-out background;z-index:-1;box-shadow:0 0 0.5rem rgba(var(--groupBorder), var(--shadeGroupBorder));}

.css-16no7tu{display:-webkit-box;display:-webkit-flex;display:-ms-flexbox;display:flex;-webkit-box-pack:center;-ms-flex-pack:center;-webkit-justify-content:center;justify-content:center;padding-left:2rem;padding-right:2rem;padding-top:4rem;padding-bottom:4rem;width:100%;}

.css-1wpjqbx{height:24px;width:24px;display:block;}

.css-11j27fm{display:block;background:linear-gradient(to right, rgba(var(--groupBackground), var(--shadeGroupBackground)), rgba(var(--groupBackground), var(--shadeGroupBackground))),linear-gradient(to right, rgb(var(--background)), rgb(var(--background)));color:rgba(var(--foreground), var(--shadeTextSecondary));font-size:1rem;position:relative;border-radius:1.5rem;margin-left:auto;margin-right:auto;width:100%;--relativeBackground:var(--groupBackground);pointer-events:none;}.css-11j27fm::after{content:"";border-radius:1.5rem;position:absolute;top:0px;left:0px;right:0px;bottom:0px;pointer-events:none;border:1px solid rgba(var(--foreground), var(--shadeForegroundSecondary));-webkit-transition:125ms ease-in-out border;transition:125ms ease-in-out border;}.css-11j27fm::before{content:"";border-radius:1.5rem;position:absolute;top:0px;left:0px;right:0px;bottom:0px;pointer-events:none;-webkit-transition:125ms ease-in-out box-shadow,125ms ease-in-out background;transition:125ms ease-in-out box-shadow,125ms ease-in-out background;z-index:-1;box-shadow:0 0 0.5rem rgba(var(--groupBorder), var(--shadeGroupBorder));}

.css-16no7tu{display:-webkit-box;display:-webkit-flex;display:-ms-flexbox;display:flex;-webkit-box-pack:center;-ms-flex-pack:center;-webkit-justify-content:center;justify-content:center;padding-left:2rem;padding-right:2rem;padding-top:4rem;padding-bottom:4rem;width:100%;}

.css-1wpjqbx{height:24px;width:24px;display:block;}

.css-16no7tu{display:-webkit-box;display:-webkit-flex;display:-ms-flexbox;display:flex;-webkit-box-pack:center;-ms-flex-pack:center;-webkit-justify-content:center;justify-content:center;padding-left:2rem;padding-right:2rem;padding-top:4rem;padding-bottom:4rem;width:100%;}

.css-1wpjqbx{height:24px;width:24px;display:block;}

.css-1wpjqbx{height:24px;width:24px;display:block;}

Maven11: