Clocktower V1

An on-chain payment and subscription service

Hugo Marx and George Atkinson

June 16, 2023

Background

As web-based services proliferate, subscription payment systems have become an important source of recurring revenue for digital content providers. Centralized payment services have reduced the friction of payments on the web, and have made traditional forms of payment (credit/debit/bank transfer) common and simple. However, this convenience comes at a price–online content and providers frequently pay more than 3% for this functionality and these costs are passed to the consumer. Furthermore, payment platforms have become a critical beachhead for censorship of people and ideas on the web, inspiring some to leave popular crowdfunding platorms in favor of their own platforms. While the major payment networks have generally not (yet) taken an activist role in online political speech, they remain a potential choke-point for free speech and an open internet.

At the same time, we have witnessed a new type of currency layer evolve over the past decades: the cryptocurrency. These systems exist outside of national borders and live on distributed networks called blockchains. While many groups have experimented with payment systems on these networks, until this point, no one has designed a simple solution for the problem of recurrent future payments and subscriptions. The problem is two-fold. The first part relates to the network fee, which on the Ethereum network is referred to as 'gas' and is paid in the native token. The gas price is always in flux, increasing and decreasing with the use of the network. Thus the most immediate issue is how to account for an unknown future gas price on a future transaction. The closely-related second issue is that a decentralized smart contract cannot act on its own–it must be acted upon by outside users. In a sense, it is unaware of time. This limitation makes it impossible to schedule actions in the future, as with a cron job in normal computing. Without the ability to schedule transactions in the future, common financial services like payroll, subscriptions, regular payments, and many others become impossible or overly complicated.

The Clocktower project seeks to solve this problem by acting as a public service unlocking the potential of the future from the limitations of a system stuck in

the perpetual present.

We seek to accomplish this by creating EVM-compliant smart contracts that are polled at regularly timed intervals by other economically incentivized actors. Users will be able to schedule transactions at a future time of their choosing. By incorporating such features as subscriptions, future payments, batch transactions, reversible transactions and ERC20 compatibility, we hope to unlock the potential of other fintech and defi projects seeking a way to expand what is possible while staying true to the principles of privacy, simplicity and decentralization.

Timing System

The Ethereum blockchain is like a giant decentralized clock, currently creating a block every twelve seconds. For each node, seconds matter, as they go through the task of creating blocks and gossiping them to the network. Its therefore ironic that even with this elaborate timing mechanism smart contracts are unable to know what time it is unless asked, like a person with an expensive watch who can't look down at it unless told to do so.

There are many ways to measure time in a scheduling system, the most common being Unix Epoch time which has been incrementing seconds since Thursday January 1st 1970 0:00. However, polling the contract every second would be too expensive and inefficent in the context of subscriptions and payments.

Another option would be using Ethereum blocks themselves as our time increment. Blocks are sometimes used to represent moments in time on Ethereum based systems. The problem with this approach is that there is no guarantee that a block will be set at twelve seconds in the future and one of the critical goals of Clocktower is to eventually be able to schedule transactions years into the future.

The situation is further complicated when considering that timed transactions need to be set at standard increments. But these increments, or time triggers as we call them, have differing scopes. For instance, a weekly subscription needs to be scheduled for a day of the week while a monthly subscription needs a day of the month. And not every month has the same number of days.

With this in mind we have chosen the following standard time trigger ranges that can represent the most common schedules:

- Future Transactions – Unixtime / 3600 (Unix Hours)
- Weekly Subscriptions – 1 - 7 (Weekdays)
- Monthly Subscriptions – 1 - 28 (Day of Month)
- Quarterly Subscriptions – 1 - 90 (Day of Quarter)
- Yearly Subscription – 1 - 365 (Day of Year (not indcluding leap days))

Protocol Lifecycle

The Clocktower lifecycle begins with a provider configuring basic parameters of a paid web service they would like to provide at a fixed interval (weekly, monthly, yearly, etc). This could be done through direct interaction with the contract or, in most circumstances, through a web front-end. After signing the transaction, the subscription is now available to anyone who would like to become a subscriber. Off-chain, the provider advertises service to potential subscribers and can send a link for sign-up. When a potential subscriber wants to signup, they sign two transactions on the blockchain. The first gives unlimited allowance to the contract to take a preferred ERC20 token from the wallet. The second approves the subscription and pays the first payment of the schedule, in addition to filling the fee balance for the account.

At this point, the clocktower payment system will automate future payments for the subscription until one of three scenarios occur: 1. the subscriber unsubscribes 2. the provider cancels the subscription for one or more subscribers 3. a subscriber that does not meet a payment obligation due to insufficient funds will be auto-unsubcribed from a given subscription

This final scenario is built-in to the protocol and necessary for preventing the build-up of ethereum gas fees on accounts with insufficient funds. As long as this balance has sufficient funding to cover the fees and the subscription price, a given subscriber will continue to be in good standing and current. If there is not enough to cover the subscription, but enough to cover the fee, the fee will be taken until the account can no longer cover the fee. At this point, clocktower will automatically remove this account from the list of active subsciptions.

Example: Bob's favorite crypto market analyst, Alice, is offering a monthly subscription to her newsletter for 50 USD/Month. Bob would like access and so he inputs his email to her website, receives a custom link for sign-up. The link takes him to a page where he is able to put in his details including his ethereum wallet address. After confirming the details of his subscription, the UI walks him through the two on-chain transactions with his metamask browser extension. His first payment of $50 is pulled from his wallet in the form of USDC. This entire process of signing up takes less than 5 minutes.

Behind the scenes, the clocktower protocol distributes part of Bob's first payment to the Provider (see table below) and part to his fee bucket within the contract. Each time his subscription payment is remitted in the future, a 1% fee is taken from this fee bucket and passed to the Caller as a reward for calling Remit–in this case, 0.5 USDC. When his fee bucket is exhausted, a similar split as the first transaction will occur again with the next payment.

From Bob's standpoint, all of this is opaque–all he knows is that he is being charged $50 for access to Alice's newsletter. As long as he wishes to continue the subscription and keeps his wallet topped off with enough USDC to cover the charge, there is nothing more for him to do except to enjoy the content. Alice

gets her payments automatically and pays zero percent in fees except in the case of the “key payments” and over time she saves about 2.5% on clocktower-based subscribers vs those who still use Stripe.

Subscription Interval |

Monthly - 12 payments x 1% = 12$ Weekly -

Fee Market and Incentive Structure

symbols:

subscription triggered by call lower case sigma value of a subscription triggered lower case sigma sub v fee % lower case phi or F gas per subscription lower case gamma or G

total sum of the value of all subscriptions per remit call * fee percentage = Per subscription gas cost * # of subscriptions per call * Σ = * Σ

or

F * Σ = G * Σ

The Caller

Clocktower is decentralized–parameters are fixed in an immutable contract with permissionless access available to any provider or subscriber that would like to use it. The other critical decentralized feature of clocktower is a role called the Caller. The Caller is responsible for checking the protocol for any due payments and, if any are found, sending these payments to the appropriate provider. The complexity of this process is abstracted, as all is managed through a single smart-contract call to the remit function. Each time the Caller calls the remit function, she is required to pay all of the required gas for the on-chain transactions. In exchange, the Caller will receive fees from all of the remitted subscriber fee balances. As long as [total remitted fees] > [total gas of remission call], a bot or manual caller in the ecosystem will call the remit function and collect profit. This economic incentive ensures that all subscriptions are checked regularly.

Those familiar with use of Ethereum mainnet since the advent of NFT releases have no doubt observed that short periods of very heavy network congestion do occur, though typically not longer than a few hours at a time. In this and other high gas price situations, the clocktower economic incentive will transiently break, as [total remitted fees] < [total gas of remission call]. During these times, the pool of Callers would not be expected to call the remit function, since they would owe more in gas fees than they would receive in reward. To account for these high-gas scenarios, clocktower V1 has a minimum period of weekly. Future versions may decrease subscription intervals as we expect L2 scaling solutions to provide a more consistent gas price environment. We also expect that with enough adoption of the clocktower protocol, a professional class of Callers will

come into being, similar to the evolution of the MEV market. These professional entities will make a science of calculating potential remit rewards and monitoring gas prices for opportunities to turn a profit. Professional Callers will further increase the efficiency of the protocol to the benefit of subcribers and providers.

Fee Balance Calculation

The ability to incentivise a Caller to pay the gas on future transactions is critical to the clocktower protocol. As such, a fee structure accompanies subscriptions and relies on certain assumptions.

cost remit: 35000 gwei gas price: 20 eth price: 1500

F * Σ = G* n * 0.02 * 20 = 35000 Gwei * 20 * n (total remitted subs) n = 100 subs fee = 2% avg remit = $120

Example Find equilibrium in Eth for fee total vs gas total

fee x # of subs * avg sub transaction value() $* ETH/$ = [Per subscription gas cost * gas price * # of subscriptions remitted] * 1Eth/ 1x10^9 Gwei

[# of subs cancels on both sides]

fee x avg sub tx value ()$xETH/$ = Per sub gas cost * gas price * 1 Eth / 1x10^9 Gwei

0.02 * 100 *120* 1/$1700 = 35000 Gwei * 20 * 100 $240 * 1 Eth / $1700 = 7.0 x 10 ^7 Gwei / 10 141,176,471 > 70,000,000 profit = ~0.07 Eth

major variables are the gas price and the avg sub tx value (or the sum value in dollars of all subscription payments to be remitted by this call)

The fee will be kept as low as possible while still incentivizing a population of Callers to call the remit function on the clocktower contract.