Clocktower V1

A Protocol for Recurrent Payments

Hugo Marx and George Atkinson

July 2023

Abstract

Clocktower is a decentralized protocol for recurrent payments. The system allows Providers and Subscribers to collaborate off-chain for the initial set-up and third-party agents (known as Callers) are financially incentivized to check the protocol contract regularly new due transactions. Clocktower allows for scheduled payments to be processed in the future in a way that is current unavailable in smart contract blockchains. This document will explore the mechanisms of the protocol.

1. Introduction

As web-based services proliferate, recurrent payment systems have become an important source of 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[1] and these costs are inevitably 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.[2] While the major payment networks have generally remained neutral politically, 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.[3] 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, the problem of recurrent future payments and subscriptions has not yet been adequately addressed.

The challenges of executing recurrent payments on blockchains is actually 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 demand for blockspace on 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 triggered to take action. 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 are impossible in these decentralized systems.

The Clocktower protocol solves these issues by creating a specialized smart contract that is polled at regularly timed intervals by economically incentivized actors. This allows users 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, Clocktower will unlock the

potential of fintech and defi projects seeking recurrent payments while staying true to the principle of decentralization.

2. Goals

To further elaborate on the purposes and constraints of the protocol, we have developed the following goals:

Decentralization
While centralized payment facilitators are the norm, they bring with them a myriad of problems, from high fees to censorship. However, with a decentralized blockchain contract there is no point of failure, no censorship and no arbitraty gatekeeping.

Immutability
A system that cannot be changed is a system that cannot be censored.

Easy of Use
In the past setting up your own subscription service has been too difficult for normal users. But with Clocktower, if you can use a decentralized app, you can create or join a subscription.

Inexpensive
Traditional payment networks have had the advantage of being able to charge high fees due to sizable fixed costs of network build-out. By building on the existing backbone of EVM compliant blockchains, we believe we can eventually undercut the existing networks.

No Oracles
An oracle is a third party data source. Unfortunately, oracles have been manipulated to steal funds[45]. This protocol will not use these data sources as they subject users to unnecessary risk.

Minimum tokens in contract
Hackers hack where the money is kept. Traditional contracts have become targets largely because they rely on the "vault" model where all value is stored within the contract. We seek to turn this model on its head by seeking to hold as little value as possible in the contract. This makes the contract less of a target and allows users to keep their own funds secure in their own wallets.

No protocol token
We believe a protocol should never need its own token to work. A token needed for functionality creates friction for the user when they have to convert it and can lead to inflationary tokenomics. If Clocktower ever issues its own token, it will be used soley for governance purposes.

3. Protocol Lifecycle

At its core, Clocktower is a series of functions that allow two parties, a Subscriber and a Provider, to orchestrate recurrent payments for a service or good. 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 (Figure 1).

At this point, the clocktower payment system will automate future payments with the help of a third party, the Caller. The Caller is responsible for checking the protocol for any payments due and, for those that are found, sending these payments to the appropriate provider. The complexity of this

process is abstracted, as all is managed through a single function called remit(). Each time the Caller uses remit(), 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 (see Figure 2). As long as [total remitted fees] > [total gas of remission call], a bot or manual caller in the ecosystem will call the remit(), sending the appropriate payments, and collecting a profit. This economic incentive ensures that all subscriptions are checked regularly.

For any given agreement between a Subscriber and Provider, the above system will continue until one of four scenarios occur: 1. the subscriber unsubscribes 2. the provider cancels the subscription for one or more subscribers 3. the provider cancels the service 4. a subscriber is unable to meet a payment obligation due to insufficient funds and is auto-unsubcribed from a given subscription

This final scenario is necessary for preventing the accumulation 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 subscriptions.

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 subscribes through her website, and receives a custom link for sign-up via email. The link takes him to a page where he is able to provide his details including his ethereum wallet address. After confirming the details of his subscription, the interface walks him through the two on-chain transactions with his browser wallet extension. His first payment of $50 is then pulled from his wallet in the form of USDC. This entire sign-up process 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 reserve balance within the contract. Each time his subscription payment is remitted in the future, a 1% fee is taken from his reserve and passed to the Caller as a reward for calling remit()–in this case, 0.5 USDC. When his reserve balance drops below the caller fee amount, a similar split as this first transaction will occur with the next payment, serving to refill the reserve. These special transactions that include a filling of the Subscriber's reserve balance are known as Key Payments and have differential splits depending on the subscription interval.

Table 1: The amount of Reserve filling during a key payment depends on the subscription interval

Frequency	Percent of Reserve Filled by Key Payment
Weekly	100
Monthly	100
Quarterly	33
Yearly	8.3

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 subscribers who still use Stripe.

Subscription Interval	Number of Payments	%Fee	Total
Weekly	52	1	12
Monthly	12	1	12

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

4. Time Trigger Ranges

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. Unfortunately, polling a smart contract every second on a public blockchain would be too expensive and inefficent in the context of subscriptions and payments. Furthermore, the EVM currently creates blocks every 12 seconds, so measuring times less than block size are not really possible.

What about using Ethereum blocks themselves as our time increment? In addition to only being slightly more efficient than the one second interval, there is no guarantee that a block will be set at twelve seconds in the future. One of the goals of Clocktower is to allow transactions to be scheduled years into the future, and a change in the block production interval could cause significant problems around future transaction timing. In order to keep the system as resilient as possible

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 will be available for use on Clocktower V1:

- 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))

5. Mitigating Gas Costs

As Clocktower relies on [total caller fees received] > [total gas], the protocol as been designed with a number of mechanisms for minimizing gas.

Code

The first and most obvious is optimization of the code, which is ongoing–the more efficient the core functions can be, the less it costs to perform those functions. Future versions will integrate any changes to ethereum and relevant layer 2 protocols that allow for more efficient transactions, including account abstraction.

Layer 2

The most significant gas mitigation strategy is the roll-out on the so-called layer 2 of Ethereum. These networks rely on optimistic or zkproofs to consolidate a group of many transactions into a single mainnet transaction, allowing for significantly cheaper transactions. As we write this paper, the L2 ecosystem is growing significantly with a number of highly scalable, low-cost options such as Optimism,

Arbitrum, Zksync and others. Clocktower will be released on one or more L2's, not on L1 Ethereum. This strategy assures a much lower gas cost to ensure that the protocol is scalable and reliable.

Data Accessibility

Clocktower has been built with data accessibility in mind. On-chain logs will be emit pertinent information about the number of transactions in queue, estimated gas costs of those transactions, total value of the queue'd transactions, etc, so that Callers can make informed decisions about using remit() at the most effecient time.

Gas Cost Estimations

Given the importance of allowing Clocktower to scale over time, we have considered a variety of scenarios for the gas price. We expect mainnet gas prices will not be an option for this system, as demonstrated in the figure below. However, L2 options have a much greater scalability as gas prices are expected to be much more reasonable. [insert graphs and discussion of the interpretations]

Protocol Lifecycle

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.

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.

5.

Discussion and Use Cases

In this paper, we have presented the Clocktower protocol and examined the structure and players. We have outlined the three participants in the system (Subscriber, Provider, Caller) and walked through a basic example of an online subscription.

As with any system, there will be trade-offs in regards to it's use. Table 2 outlines the primary costs and benefits. The primary benefits that, Clocktower provides a relatively easily accessible, censorship-resistant method of receiving recurrent payments through use of a variety of ERC20 tokens.

Table 3: Trade-offs of the Clocktower Payments Protocol

Benefits	Costs
- censorship resistance	- unfamiliar to non-crypto users
- low fees (L2)	- initial centralization
- permissionless	- limited to daily interval times
- system improves with more users	
- at scale, will be fully self-sustaining	

Snippets

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.

pandoc snippet.md –from=markdown –to=pdf –filter=pandoc-citeproc –bibliography=whitepaper.bib –output=whitepaper.pdf

pandoc snippet.md –from=markdown+multiline_tables –to=pdf –filter=pandoc-citeproc –bibliography=whitepaper.bib –output=whitepaper.pdf

The Ethereum blockchain is a decentralized clock 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.

The arrangement of price and the good/service to be exchanged is determined off-chain through any standard path–typically this would involve a website but could also be arranged at an in-person meeting, etc. After agreeing on the exchange details, subscription information is written to a smart contract, including an interval (daily, weekly, monthly, quarterly, annually) and an amount to be collected from the subscriber. Once established, subscriber and provider roles are complete until one of the parties wishes to make a change the in the exchange agreement.

It is at this point the subscription lifecycle, we introduce the third player in the system–the Caller. The job of the Caller is very simple: run a function (called 'Remit')that checks to see if any of the subscriber payments are due to their respective providers, and if so, moves the appropriate value (remittance) to the appropriate provider. When a caller makes a call to the contract, he is charged a fee by the network, and so the Caller is incentivized with compensatory fees designed to offset this cost and provide profit. This incentive to call the contract is at the heart of Clocktower. At the most basic level, the Caller is making a decision based on the following scenarios:

A. total gas cost < total fees –> call contract B. total gas cost >= total fees –> do not call contract

In times of low gas prices (scenario A), running the remit function will be profitable and thus will be called; human callers (or as the ecosystem matures, bots) will call remit() and funds will move appropriately. In times of high gas costs (scenario B), the total cost of gas will exceed the gas and potential callers will wait until those prices decrease to call remit().

1. Helcim.com Visa interchange rates usa. https://www.helcim.com/visa-usa-interchange-rates/.

2. Goggin, B. (2018). A top patreon creator deleted his account, accusing the crowdfunding membership platform of 'political bias' after it purged conservative accounts it said were associated with hate groups. https://www.businessinsider.com/sam-harris-deletes-patreon-account-after-platform-boots-conservatives-2018-12.

3. Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf.

4. Chainalysis (2023). Oracle manipulation attacks are rising, creating a unique concern for defi. https://blog.chainalysis.com/reports/oracle-manipulation-attacks-rising/.

5. Afser, Z. (2021). https://hackernoon.com/how-dollar100m-got-stolen-from-defi-in-2021-price-oracle-manipulation-and-flash-loan-attacks-explained-3n6q33r1.