Ride

A DAO-as-a-Service platform for ride-sharing and delivery

Benjamin Ho
junhuangho27@gmail.com

1. Introduction

Problem Statement

Large centralized companies in the business of ride-sharing or delivery service often face several hardships during operation. Examples include costs of local regulation compliance for each region, battles with the taxi industry such as driver protest on pay and well-being, non-transparent pricing, and surge in pricing which are unfavorable to customers. These are just some of the recurring issues that plague the ride-sharing/delivery industry [1].

A Fair Future

What do all these issues have in common? The drivers. No one knows the local region better, and no one feels the pain of when oil prices increase more, than the drivers. So why can't the power be with the drivers to fairly decide with each other what is right for themselves? If only there was a way where drivers could team up to better serve their local community, and be rewarded fairly for their efforts.

A team allows its member drivers to effectively determine when is the best time to provide its service and how best to serve, tailoring it to its community's needs. Regulations are easily handled as every team only needs to consider it for their local region. Teams are rewarded for their efforts by setting their own rates to what they feel is a fair price. Healthy competition among teams that overlap in region leads to lower prices that benefit the customer.

Decentralized Solution

The Ride protocol is a set of smart contracts implemented on the Ethereum Virtual Machine. The core mechanism of it facilitates the transport of a package from one point to another in a flexible manner that allows different levels of verifiability to be enforced throughout the transportation process. This brings only the essential building blocks required to form a ride-sharing/delivery service platform onto the blockchain.

Hives, a form of Decentralized Autonomous Organization (DAO) - backed by the Ride protocol, can be formed by a group of drivers who are interested in starting their own local ride-sharing/delivery service in their own local region. Hives allow voting between members on the fair price and other important aspects of a ride-sharing/delivery service business.

The Ride protocol acts as an infrastructure layer, and decentralized applications (DApp) such as ride-sharing or food deliveries are built on top of it. These DApps would be in the form of a marketplace where customers can choose which Hive service to use.

2. Main Components

Hives

Hives are a form of DAOs backed by the Ride protocol. A Hive is formed when a group of drivers form their own ride-sharing/delivery service in their local area and register their own DAO on the Ride protocol. The protocol provides factory functions to easily create a Hive. Once a Hive is formed with its initial members, new members can join through governance voting. Every Hive member can register on the protocol to become a runner (the person that delivers), and together, these Hive serve their local communities and can expand as they see fit.

The main benefit of operating a ride-sharing/delivery business through a Hive is that members can set their own fees, which ultimately contributes to the value of delivery payment. The fees are set through governance voting of their own Hive to ensure the fair and optimum fees are set. Hive members can also vote to temporarily ban a requester if needed - refer *penalty* in section 3. With runners having full control of their rates, they can better take care of themselves as they see fit without needing to rely on centralized sources to dictate their living expenses, as it is often observed with existing ride-sharing services.

In addition, this concept of self governance allows runners to more easily handle any runner-requester (customer) relation as runners can work together to provide a better service. Hive members are incentivised to work together to provide a better customer experience as any ratings provided by the requesters are recorded on-chain and aggregated to reflect onto the Hive as a whole. It is difficult and costly for a single company to handle every complaint from every region especially if the company does not know anything about the local community. Self-governing Hives can significantly ease this pain as it is built with the “by the community, for the community” concept in mind, where Hives can tailor their experience to the community.

Moreover, multiple Hives can exist in the overlapping regions which encourages healthy competition and reduces cost for requesters. If a Hive member wishes to switch to another Hive, they are required to leave the existing Hive first before joining another. Hive members can also contribute funds to the Hive honey pot where funds are collected and proposals are voted on by the members to determine where the funds should be put to use.

Note that Hives would have their own token for voting, which is basically the Ride protocol’s native token wrapped in the Hive specific wrapper token - refer section 5.

Interaction Flow

At its core, Ride protocol validates the delivery of a package from point A to point B. This package could be an item or a person. The user who makes the request for the delivery is called the requester. When a runner matches a requester, an active job is initiated with a unique job ID and only the minimal delivery information required to secure and validate the transaction

is stored on-chain. The following is a general description of how things could take place when a job is initiated, refer to Figure 1 for reference.

It begins by a user requesting for a runner. Information such as package details, destination, transaction currency and estimated delivery distance and time (where some of this information is encrypted for privacy purposes) are sent to the protocol. The distance and time along with other factors are then used to compute the value of the delivery service - refer to *Fees* in section 3. This value is converted into the desired on-chain currency - refer to *Currency Registry & Conversion* in section 3, and locked in the protocol's escrow-like vault. A unique job ID is generated, and the relevant information is stored in a job ticket. This ticket is emitted into a pool of tickets off-chain, waiting to be accepted by runners. Before the ticket is accepted by any runners, the requester may cancel, where no cancellation fee is charged and all the ticket information is cleared from the protocol's storage.

When a runner accepts the ticket request, information such as the runner's accepted transactional currency is verified with the requestor's locked currency value through the job ticket to ensure the accepted job is a valid one. Once the check is complete, the equivalent amount of value from the runner side is also locked in the protocol's vault to ensure the runner's commitment. The job ticket is now updated with the runner's information and the job becomes active. At this point, both requester and runner may cancel if needed, but the user which cancels would incur a cancellation fee where this fee would be transferred from the user that canceled to the other user.

Once the runner reaches the location of the package, the runner can collect the package and optionally verify the package address. This package address would be the supplier's address for a delivery item or requestor's address for ride-sharing. If the package is verified (using package address) at this point, then on-chain requester and runner stats would be updated accordingly. Package verification has other benefits and functionality in general (such as rewards) and further down the delivery process. Verification is encouraged on every job. The reason this is optional is to ease the user experience on the requester/supplier side where it might not always be convenient to get verified. In such situations, the delivery job can continue without interruption.

From here, both parties may still cancel if required. If the requester cancels without any verification, the requester only pays a cancellation fee, with verification, the entire value of delivery. If the runner cancels, the full value of the delivery is paid to the requester. This is to encourage the runner to make things work out instead of abandoning the job if anything goes wrong. One reason the requester might cancel is if there is a dispute from the requester side. A dispute may be used by the requester if there is a disagreement on the collection of the package. This is to prevent the runner from abusing the protocol where the runner may not even be at the package location and perform the collect, deliver and complete action in series to unlock the value without actually completing the job. The deliver package and complete job on-chain actions have built-in timelock delay to give the requester some time to trigger the dispute if necessary.

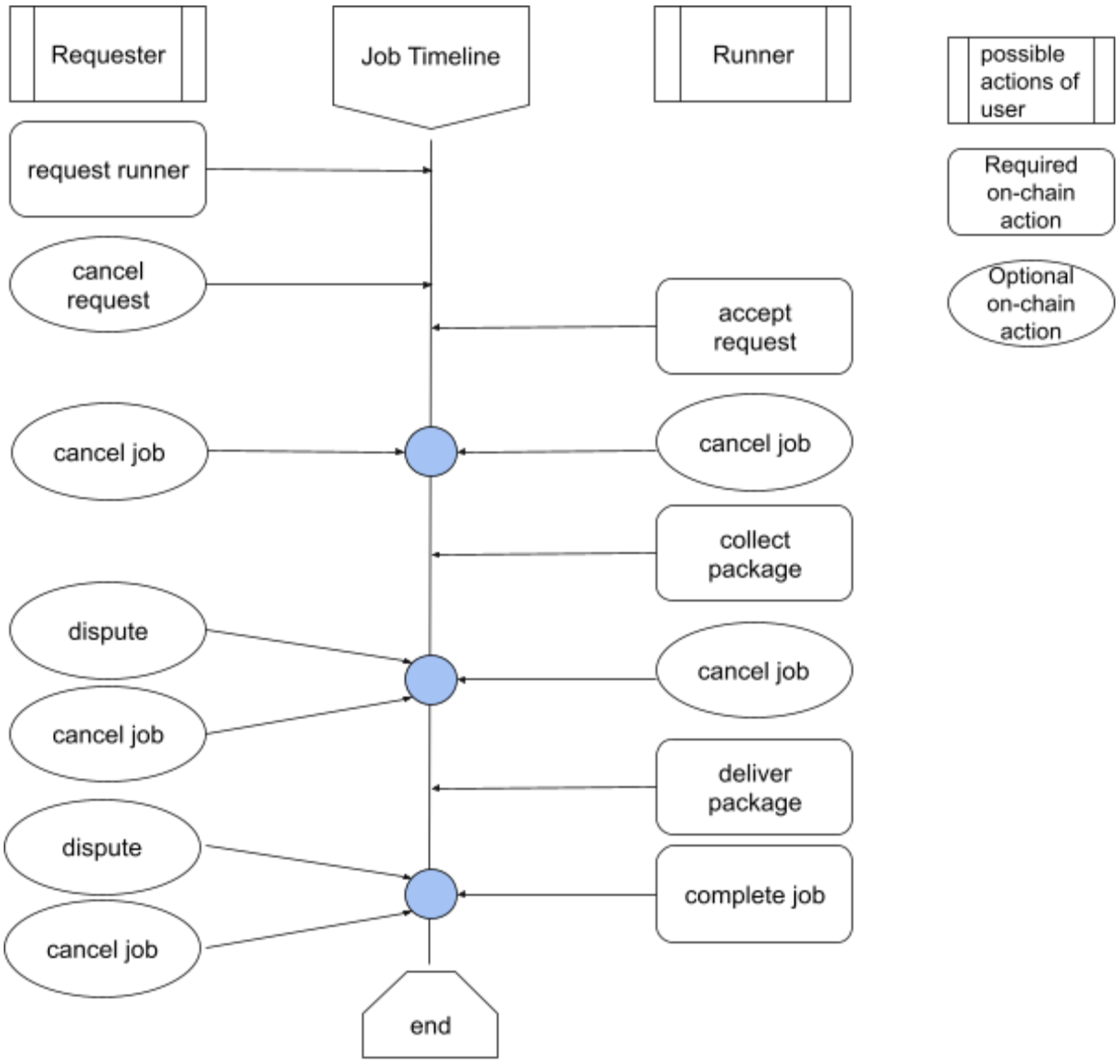

Figure 1: Interaction between requester and runner during an active job cycle.

Once at the intended destination, the runner runs the deliver package action to state the package has been delivered. The requester can optionally again dispute or cancel if needed, such as if the package was not actually delivered. If there were no disputes, then the runner could complete the job and the value of delivery would be unlocked from the vault and transferred to the runner. Otherwise, the runner needs to accept any dispute and only a cancellation fee is paid from requester to runner. The job is now complete and a transaction has been made through the Ride protocol. The completed or canceled jobs along with all the details are periodically cleared. It is not immediately cleared on job completion or cancellation to allow post job actions such as requester rating - refer *Rating* in section 3 or runners imposing penalties if needed - refer *Penalty* in section 3.

In general, most of the requester-runner interaction cases would only require the requester to perform one action “request runner” and the runner to perform four actions, “accept request”, “collect package”, “deliver package”, and “complete job”, with an additional step of verification which is encouraged. The core mechanism layout is designed to give the requester the best user experience while allowing the runner to securely get paid for delivering the service.

3. Other Components

User On-chain Details

Minimum requester and runner job details are recorded on-chain such as number of jobs started/completed, total distance traveled by a runner, and runner’s rating. These details could be used for rewards and could be used as a reputation system for Hives. Overall, it encourages all users to keep to a single address when using Ride protocol and promotes good behavior as to lose accumulation of these on-chain details, such as requesters getting banned by runners - refer *Penalty*.

Currency Registry & Conversion

The transfer of funds in return for a delivery service is at its core of Ride protocol, and this fund is to be paid primarily in the form of ERC20 tokens. To determine how much to pay, the value of the service is first calculated in the country’s native currency of which the service is being carried out.

The Ride protocol only supports tokens and local currencies that were registered, which are stored internally as currency key notation for protocol references. For example, the local currency USD and tokens BTC and USDT are registered with Ride protocol. This means that all delivery services can only be paid using BTC or USDT, with the value of the service first calculated using USD - refer *Fees*. After calculating the value in USD, a decentralized oracle is used to convert the value in USD to either BTC or USDT. The calculation and conversion is done dynamically per job.

Price conversions from token to non-USD are typically not supported by decentralized oracles, while the USD to other local currency conversion is. To get the token to non-USD conversion, two conversions take place, for example if the BTC/AUD price is required, then the division of BTC/USD by AUD/USD is used to compute it, where both of these base pairs needs to be already registered with the protocol.

$$\frac{BTC/USD}{AUD/USD} = BTC/AUD$$

Fees

The value of a delivery service can be calculated using base fee and several rates:

$$value = base\ fee + (cost\ per\ minute * estimated\ minutes) + (cost\ per\ metre * estimated\ metres)$$

where the base fee, cost per minute/metre along with cancellation fee are set by the Hives themselves, and each Hive has their own unique rates.

Holding

Before any users can request or accept a job, their holding in the protocol must have sufficient funds with the amount at least being the value of the job and cancellation fee, otherwise the job request or acceptance will not register in the protocol. For example, the value and cancellation fee for a job totals to 10 ABC token, both requester and runner must have 10 ABC in their protocol holding in order to be registered with the job. This amount will be locked in the protocol's vault during the job's active lifespan. It is possible for users to have multiple active jobs as long as there is sufficient holding to be locked in the vault.

Penalty

Penalties are required to encourage good behavior. Any form of penalty implemented through Ride protocol requires passing through a governance voting. Currently there is only one form of penalty, that is, runners are allowed to vote on users to temporarily ban from their Hive service, and it is possible to ban both requesters and runners to encourage good behavior from all sides.

Rating

After each job completion or cancellation, the requesters are allowed to rate the runner of the job on a 1 to 5 scale (bad to good). The job details will persist for a certain period after the job is over before being cleared by the system periodically. After the job is cleared, the requester will not be able to rate the runner anymore.

4. Protocol Fee

The concept of Hives allow many of the traditional costs associated with large centralized ride-sharing/delivery services to be off-loaded to the Hives. Some of the costs include legal fees for related regulatory compliances and runner (driver) relations/benefits. Ride becomes more of a connecting tool / decentralized marketplace to connect requesters to runners for ride-sharing/delivery services - refer to section 6.

Nevertheless, there are still costs associated with the maintenance and upgrades of the protocol, DApp, Hive support, and marketing. Hence, a protocol fee can be applied to every job executed, where the value of this fee is yet to be determined. This fee should be low enough to not impede on Hives operations and allow them to be competitive with existing centralized services.

5. Tokenomics

An ERC20 token would be linked to the Ride protocol as a native token. The purpose of this token is to act as a utility token for the protocol. The protocol would support a selected range of ERC20 tokens such as wrapped BTC and stablecoins. However, to encourage the use of the native token, discounts could be given to requesters, and a reduction in protocol fee could be given to runners who use the native token. Further rewards such as discounts could be given to users who successfully complete a certain number of jobs.

The native token is also used for Ride protocol's governance voting. Additionally, each Hive has their own token which is used for setting fees/rates and any other decision making related to the Ride protocol. The Hive tokens are wrapper tokens that wrap the Ride protocol's native token. This acts as a natural staking mechanism that reduces the supply of the Ride native token.

As Ride protocol requires a physical presence of runners fulfilling jobs and bringing value to local requesters, the native token should also be distributed with the region in mind. A limited amount is supplied per region/Hive to ensure a relatively even distribution of tokens, where sufficient amounts of the token is held by the requesters and runners in the region who would use them, and not large amounts held by a small number of users. This ensures the token's utility, where it is used for its intended purpose (ride-sharing/delivery services) of being used through the Ride protocol. Once a homogeneous supply of the native token is reached amongst its user base, a cap should be set in terms of large token supply to prevent further inflation of the token's price.

Overall, the general trend of the supply of native tokens would be similar to an increasing stepped line plot which slowly tapers off after a homogeneous supply has been reached, similar to Figure 2. Each step corresponds to the fresh supply of region/Hive.

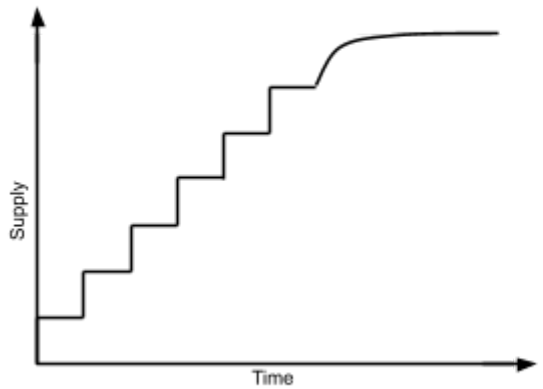

Figure 2: Stepped line plot of token supply against time (for illustration purposes only)

6. Mobile DApp

Ride-sharing/delivery services are typically more practical on mobile. This is why any application that utilizes the Ride protocol's core mechanisms should be mobile first. One major issue that currently exists with mobile DApps is the interaction with the user's wallet. For a web wallet, the user is required to constantly confirm any on-chain transaction via a web interface. For mobile wallets, the user is required to go back and forth between apps for each on-chain transaction. Both cases are not practical and in turn, Ride protocol would not be utilized.

In order to solve this, a mobile DApp with integrated wallet is created to seamlessly allow the users to interact with the Ride protocol. For the most part, the wallet is kept separate from the DApp, except for the transaction confirmations actions where the user is able to confirm transactions on the DApp interface without leaving the page, and the DApp automatically connects to the wallet to confirm the transaction, all done in the background.

The mobile DApp can act as a marketplace where requesters can select the Hive service in their region suitable for them, where each Hive might have different rates or features unique to the Hive.

7. Discussion

The Ride protocol is a general protocol that can be applied to many delivery categories such as ride-sharing, food delivery, grocery delivery, or any general delivery services. It is even possible to apply it to the supply chain industry in certain cases. This allows many applications to sit on top of the Ride protocol layer, such as the mobile DApp discussed in section 6. Hives with sufficient resources that do not want to use the marketplace could also launch their own app that utilizes Ride protocol. It is also possible for existing centralized companies who want to pivot into decentralization to utilize the protocol.

While the Ride protocol and Hives alleviates many of the issues faced by large centralized ride-sharing/delivery companies as mentioned above, it also brings additional benefits with it. For example, in the long term, there will be multiple Hives that may overlap over the same region that increases the competition not only with the centralized companies but also between Hives, which in turn brings the benefits of a competitive industry with it such as lower prices and better quality of service. Another benefit is the transparency of the protocol on the blockchain which benefits requesters as they know exactly what to expect. One example is how the value of delivery is calculated which can be seen on-chain.

Oracles and built-in currency conversion mechanism in the protocol makes it easy to support existing Hives in terms of accurate pricing, and Hive governance keeps things fair for the Hive members and in turn fair for requesters.

In summary, the Ride protocol facilitates the transport of a package from one point to another in a flexible manner with each job, and Hives allows runners to deliver on these jobs in a novel

way. This system brings new utility to the blockchain space, bringing us one step closer to a blockchain based future, while empowering millions of drivers around the world.

References

[1] ZHAO Xingyue, & SU Qin. (2020). Existing Issues of Ride Sharing Company Operation and Sharing Economy in China: Uber Case Analysis. Management Studies, 8(1). https://doi.org/10.17265/2328-2185/2020.01.005