One of the problems to solve when charging rent is “how do you charge rent for common data ?” as there are lot of contracts which are very useful to the community but no single entity pays for the contract.

When thinking about charging rent for common data in the current account model, there are 2 types of common data we are dealing with. They are code and storage. They have different properties from a rent perspective. In case of code, rent has to paid for the entire code and it is smallest denominator for which rent has to be paid. But in case of storage, a single storage slot is the smallest denominator for which rent has to be paid. Because of this significant difference, we should analyze code rent and storage rent separately.

Code Rent

In case of code, as the entire code has to be paid for at once. We can use the concept of Royalty to achieve this. Thanks @JustinDrake for introducing to the concept of Royalty.

Royalty is a fee that is paid for every interaction with a contract code. When a user A

deploys the contract that will cost x

as code storage rent for t

time, every user who interacts with the contract code will pay R

royalty. R

is a variable to control the royalty paid based on the usage of the contract. The royalty collected will be used towards paying back the deployment costs to A

and paying for the contract code rent. The user A

that deployed the contract will receive a refund of significant fraction of deployment costs x

.

Royalty has the benefit of organically paying the rent which can be helpful to keep the contract running, without the need for a single body to pay the rent for everything.

I have explore the code rent in more detail inanother post.

Storage Rent

As the smallest unit to pay in the storage is a storage slot, the rent cannot be designed the same way as the code rent as it is too much overhead to maintain the expiry dates of all the storage slots. We can solve this by paying for rent in epochs because we have a common expiry for all the storage slots which makes it possible to revive/initialize individual storage slots without storing the expiry date of each slot.

Storage rent can be paid in terms of epochs which are specified during contract deployment. Any user storing data on contract will store it till the end of current epoch. It will be possible to extend the duration of the current epoch but that needs paying for not only their data but also the entire contract data. I have presented a complete description of this in a previous post.

Each storage slot in the contract either belongs to an individual or common data. The developer of the contract can specify the variables which represent the common data in the contract using a common

keyword. Developer should make sure that it is not easy to bloat the common data. As we know the common data, initialization and revival costs of the common data can be paid by the contract rather than any specific user. A similar concept of Royalty that was used for code rent can also be used for common data in storage rent as well. Anyone trying to use the common data will pay a small fee towards a royalty fund which will be used to initialize or revive the common data specified by the common

keyword.