Current Problems:

1. ETH1.x chain contracts are expected to last forever
2. Future improvements can't break anything "important" and that includes most old contracts.
3. State size keeps growing. Which means client implementations need to eek out performance for larger and larger data sets.
4. Future development will have to get slower and more conservative to avoid breaking things on upgrade of the chain.

So far the proposals I've seen help with some of these but not all.

Solution: What if we introduced ETH2+ rolling chains with End Of Life dates.

1. Each chain's rules could be re-written from the ground up with (almost) NO COMPATIBILITY concerns. (Changes would probably still not be radical, to allow client code reuse and faster development)
2. Each chain would have bridges between it and ETH1 and past/future ETH2+ chains.
3. Each chain has a set expire time at which point the chain and all data on it goes away. (Data people care about would be migrated to the new chain through the bridge. If they don't, it goes away.)
4. Testing should be easier to validate the chain rules and client behavior.
5. Assumptions can be made about max state size that client code needs to handle. (Should be smaller data sets).

A new ETH2+ chain could be scheduled for every 2-4 years perhaps.

The rules could be radically different between each chain deployment as long as there was a way to create bridges between them.

This effectively gives you "state rent" without breaking anything on-chain because all chain state is kept for the life of the chain (you could add other state rent mechanisms on top).

Ideally we'd want to find away to get rid of ETH1 but I doubt that's possible.

Perhaps there would be some auto-migrate features when a chain expires, like migrating all users' ETH balances forward to the next chain.

There would perhaps be a 2 year overlap between old and new chains. (So you'd have at most 2-3 chains running at any point in time).

Development would not happen on older chains (except for critical fixes).