History Expiry

Problem Statement

Ouroboros - the snake perpetually devouring its own tail. But that tail is growing ever larger, and it's taking longer and longer for the snake to catch up to its own head - not to mention the amount of tail that must be devoured along the way. It's enough to give a snake indigestion!

The current cardano node is genuinely unique in its approach to upgrades. While other networks need to effectively stop and restart in order to deal with upgrades to the core logic, the hard fork combinator and polymorphic ledger have allowed Cardano to seamlessly progress from Byron through to the current Conway era, changing consensus protocols along the way. The current node may just as easily act as a Byron era node, and can validate the full chain from Genesis. This is a valuable property of Cardano that needs to be preserved, as established by community interactions.

However, this benefit has a downside - the Cardano chain now stretches back over 7 years, and this full history must be processed by any node joining the network. Each node is expected to be able to serve the historical chain should it be requested. This has a number of costs:

1. Time to sync a node from scratch is now over a day and growing.
2. Storage requirements for each node are significant, many tens of gigabytes on disk.
3. Network traffic requirements on syncing nodes and the upstream peers serving them are also high.

Requiring honest nodes to be able to serve the historic chain is also a potential risk for new node implementations, most of which do not intend to implement the logic of long-past eras. As an example, Amaru uses Mithril snapshots to bootstrap the node and does not validate, nor even store the blocks before its bootstrap point. Since Amaru nodes cannot serve the full chain, syncing nodes may waste network round-trips requesting blocks without knowing in advance that their request cannot be honored.

In summary, there is some tension between two requirements:

1. Preserving Cardano's unique ability to sync and validate the entire Cardano chain.
2. Allowing nodes to forget part of the history to reduce costs.

Note that once Leios is implemented, increased transaction volume will greatly exacerbate the motivation for reducing storage requirements. This proposal aims to address this tension. It builds upon a previously funded component, the Genesis Sync Accelerator.

Proposal Benefit

With the advent of tools such as Mithril to allow nodes to bootstrap from snapshots, we have the possibility to now allow history to be truncated. This would allow honest cardano nodes to compress and forget all history past a certain point. There would be a number of benefits:

- Significantly faster syncing speed (even Mithril snapshots would be smaller, since there would be no need to distribute the entire historical chain)
- Lower hardware requirements for node storage.
- Lower network costs to synchronize historic state.
- Alternate node implementations would not risk being “technically malicious” due to refusing to serve historical data or else serving historical data that they have not personally validated!

Supplementary Endorsement

- The discussion on https://github.com/cardano-foundation/CIPs/pull/974 endorses this plan. Note that this plan would probably render that CIP redundant.
- Amaru is not planning to support old eras

Proposal Details

Proposal Description

We propose the following work:

1. A reformulation of honest behaviour to support nodes which do not offer the historical chain.
2. Updates to chain sync and block fetch protocols to explicitly support communication about “pre-historic” chains and their availability.
3. Updates to the node to support explicitly starting from a snapshot with no historical chain.
4. Logic in the Cardano node to support moving the historical genesis anchor (e.g. further truncating the chain).

In addition, once regular nodes are running without historical state, we envision the need for specific archive nodes serving the historical chain. Thus we also plan to deliver:

1. A design for the incentives to support running archival nodes.
2. A bare bones archival node which will serve the historical chain in a format suitable for downstream tools.

Dependencies

- Canonical ledger state would be helpful for a few things:
    - Makes the Mithril bootstrap sequence much safer
    - Allows potential use in communicating ledger states rather than blocks
- Canonical Block and Transaction Diffusion Codecs would need to modify the archival node as well, but that would reduce the amount of reference node implementation details in the messages the archival node serves to peers which want to sync the pre-historical chain.
- Both this proposal and Canonical Block and Transaction Diffusion Codecs will eventually require a schema for publicly archiving data: at least EBBs and the source code/binaries/etc of older versions of the node which can validate the pre-historic chain.

Maintenance

The main thrust of this work will actually be to reduce the maintenance burden on the Cardano node. Large chunks of legacy code may be removed, reducing the size of the codebase to maintain. Old versions of the node may be maintained in order to process and verify the historical chain.

Some maintenance may be needed for the archival node, but we envision that this should be designed in such a way as to have minimal logic that will need updating for future eras.

Key Proposal Deliverables

- CPS and CIP setting out the full scope of work
- Update network protocols with support for pre-historic chains.
- Updated Cardano node with support for pre-historic chains and starting from a snapshot.
- Incentive design for archival nodes.
- Simple archival node.

Milestones

Milestone	Deliverables	Acceptance criteria	Weeks
T3.1 Project Kickoff	Public repository and documentation hub. Problem statement and scope definition. Initial design outline for partial-history nodes. Recorded public kickoff/demo.	Repository and documentation are publicly accessible. Demo recording is publicly available on the project website.	2

T3.2 Protocol Specification for Partial-History Nodes (CIP)	Establish a clear consensus on the problem and constraints with the community. Specify what it means for a node not to have the full history: - Mini-protocols: changes to ChainSync to let nodes communicate to partial nodes - Define a minimum history window (e.g. 5 epochs). - Re-affirm the header chain requirement: nodes should have the full header chain to validate blocks.	CIP draft is published and submitted, reaching 'Proposed'. Specification fully and unambiguously defines node behavior. Feedback from consensus/network teams is incorporated. No critical open design questions remain.	4
T3.3 Node Implementation with Bounded History Storage	Implementation of: - sliding window storage model. - protocol changes from CIP. - updated validation and replay logic. Integration with existing node components.	Node runs using bounded history (no full chain required). Functional tests pass under normal operation. Crash/restart recovery strategy is implemented. Implementation reviewed and accepted by relevant teams.	8
T3.4 Ecosystem integration	Align on the integration with Mithril and GSA decide on and implement requirements. Prepare documentation resources for SPOs (configuration, storage requirements, tradeoffs). Deployment guide.	Documentation is complete, and accessible for reference. Documentation is published on Cardano community resources. A third party (external operator) can run a node using only the documentation. No missing critical steps.	8