Arch Network Rollback Mechanism

1 Introduction

In this document, we provide an explanation of the rollback and reapply mechanisms implemented in the arch network. This system ensures that transactions are correctly processed, either reverting their effects when necessary (rollback) or reinstating them when conditions permit (reapply). We will delve into how the transaction graph is built, how transactions are validated, and the intricate logic governing rollback and reapplication.

2 Transaction Graph Overview

A transaction graph models dependencies between transactions, capturing parent-child relationships. Each node in the graph represents a transaction, while directed edges indicate dependencies.

2.1 Graph Representation

- **Nodes:** Represent individual transactions.
- **Edges:** Directed links that show transaction dependencies (e.g., transaction T_2 depends on T_1).
- **Adjacency List:** The internal data structure for storing the graph, using a HashMap.
- **Source Nodes:** Transactions with no dependencies (i.e., no incoming edges).

3 Rollback Mechanism

Rollback is the process of undoing transactions, ensuring that invalid or orphaned transactions do not remain in the system.

3.1 Conditions for Rollback

1. If a transaction has been anchored in Bitcoin but its corresponding Bitcoin transaction is missing, it should be rolled back.
2. If a transaction is already marked as rolled back, it should be excluded from further rollback processing.
3. State-only transactions (transactions that do not have an associated Bitcoin transaction) are only included for rollback if their parent transactions are also rolled back.

3.2 Rollback Execution

3.2.1 Step 1: Graph Construction

- Start at the target transaction ($arch\_txid$) to be rolled back.
- Expand **backwards** to identify dependencies (parent transactions).
  - If a parent transaction is anchored but not found in Bitcoin, it is included.
  - If a parent transaction is state-only, it is marked for further validation.
- Expand **forwards** to ensure all affected child transactions are properly processed.
  - Include transactions that have not been previously rolled back.

3.2.2 Step 2: Validation

Each transaction is validated using:

- $check\_if\_arch\_tx\_is\_anchored\_in\_btc(txid)$: Determines if the transaction is associated with a confirmed Bitcoin transaction.
- $check\_if\_arch\_tx\_has\_been\_rolled\_back(txid)$: Checks if the transaction has already been rolled back.

3.2.3 Step 3: Recursive Expansion

- If a transaction is included in rollback, its dependencies are explored recursively.
- Transactions marked as *ExcludeButCheckNexts* are not added but their dependencies are still examined.
- Finally, transactions marked for rollback are processed accordingly.

3.3 Example: Rollback in a Simple Chain

Given a transaction chain $T_1 \rightarrow T_2 \rightarrow T_3$, where:

- T_1 has a confirmed Bitcoin transaction.
- T_2 is a state-only transaction.
- T_3 has a missing Bitcoin transaction.

Rollback Process:

1. Identify that T_3 is missing in Bitcoin.
2. Move **backwards**:
   - T_2 is state-only $\rightarrow$ Exclude but check T_1.
   - T_1 is found $\rightarrow$ Exclude (as it is already confirmed).
3. Move **forwards**:
   - If T_2 was included in rollback, all its child transactions would be examined.

4 Reapply Mechanism

Reapplication is the process of reinstating transactions that were previously rolled back. This is necessary when conditions for valid inclusion are restored (e.g., a Bitcoin transaction is later confirmed).

4.1 Conditions for Reapply

1. If a transaction was previously rolled back but is now confirmed in Bitcoin, it should be reapplied.
2. Transactions that depend on a re-applied transaction should also be re-considered.
3. State-only transactions are included if their required parent transactions exist.

4.2 Reapply Execution

4.2.1 Step 1: Graph Construction

- Start from the target transaction ($arch\_txid$) being reapplied.
- Expand **backwards**:
  - Include transactions that were previously rolled back.

  - Ensure all dependencies exist.

- Expand **forwards**:
  - Check if the transaction is anchored.
  - Include state-only transactions to maintain execution consistency.

4.3 Example: Reapply in a Simple Chain

Using the previous rollback example:

- $T_1 \to T_2 \to T_3$, where T_3 was rolled back because its Bitcoin transaction was missing.
- Later, T_3's Bitcoin transaction is found, requiring it to be reapplied.

Reapply Process:

1. Identify T_3 as a reapply candidate.
2. Move **backwards**:
   - T_2 was previously rolled back $\to$ Include it.
   - T_1 is already confirmed $\to$ Exclude.
3. Move **forwards**:
   - Include child transactions that were also rolled back.

5 Conclusion

The rollback and reapply mechanisms ensure the integrity of transaction execution in a dependency graph. The system carefully tracks dependencies, validates conditions, and applies recursive logic to maintain correctness. By implementing thorough checks and recursive expansion, the transaction graph builder efficiently processes rollback and reapplication scenarios, even in complex transactional structures.