# ResilientDB Overview

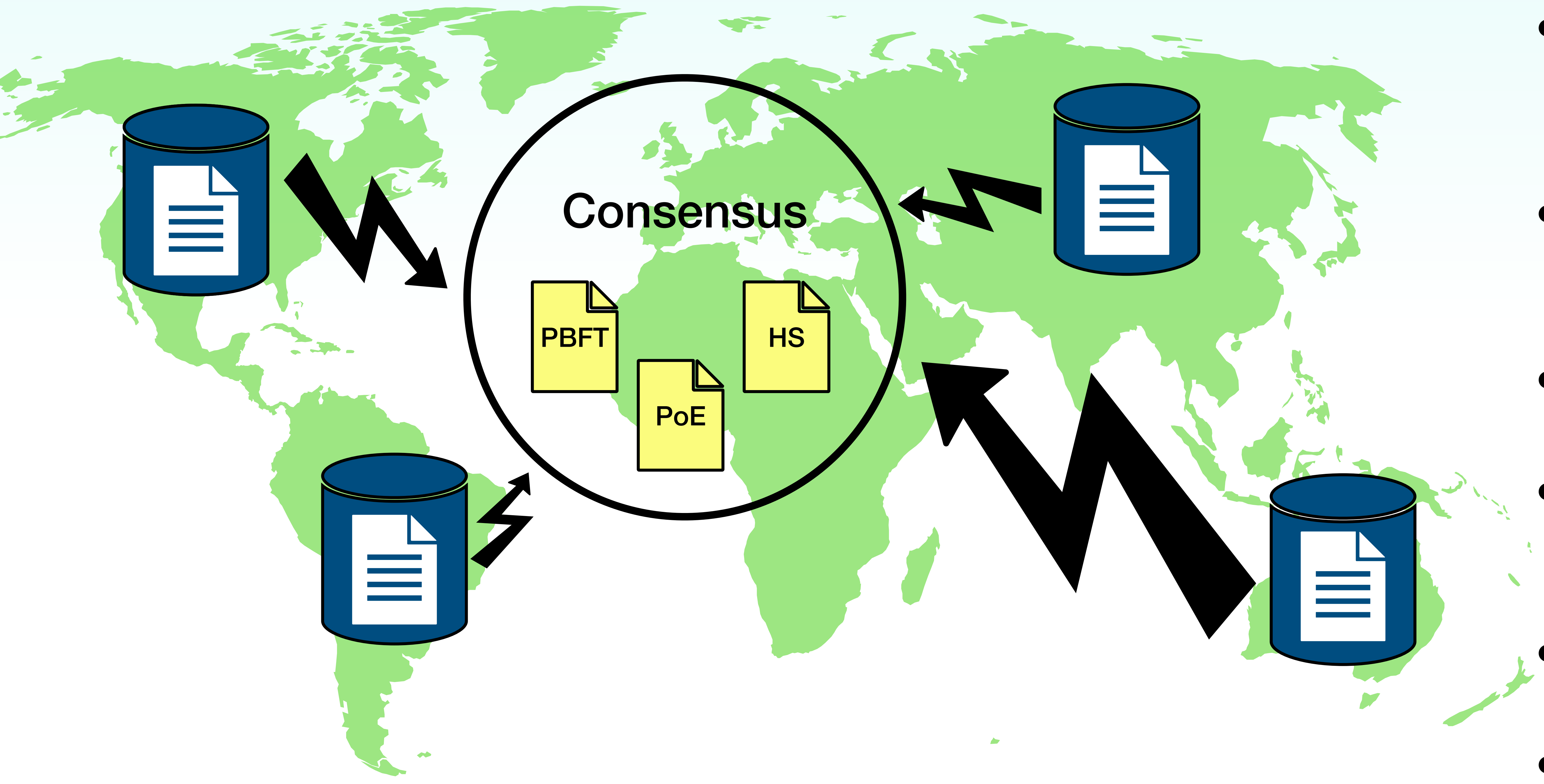
## An Open-Source High-Performance Permissioned Blockchain Platform

**Sep 26, 2025**

UCDAVIS

Apache ResilientDB Incubating

# What is Permissioned Blockchain?



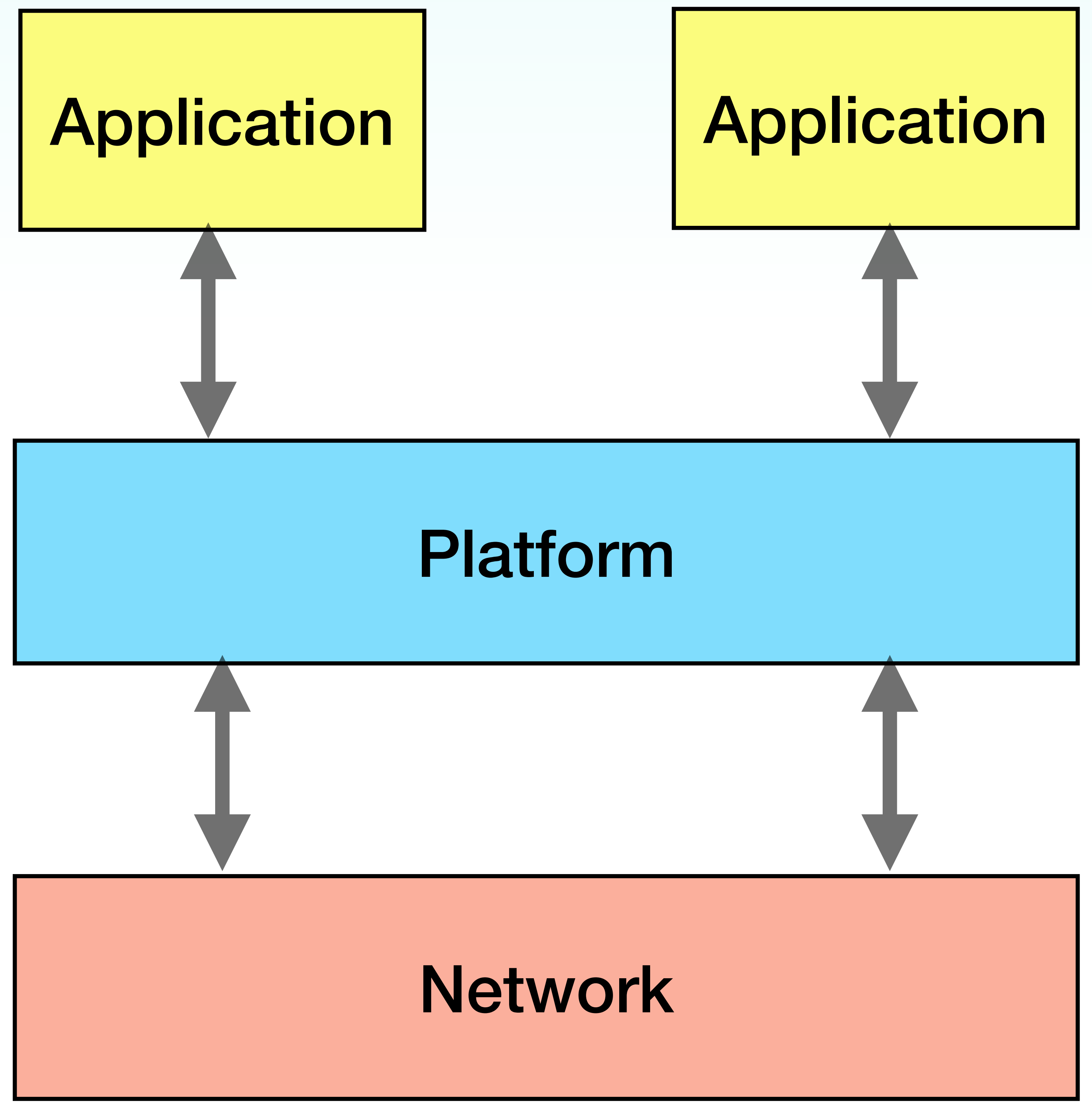
- **Distributed database made up of a fixed set of replicas (participants).**
- **Each replica holds a copy of the ledger, which is a chain of blocks containing transactions.**
- **Consensus Properties: Safety; Liveness.**
- **Fault Model: Byzantine replicas may behave arbitrarily.**
- **Network: Synchronous/Asynchronous/Partial Synchronous**
- **Consensus Protocols: PBFT, PoE, HotStuff, etc**

## ResilientDB

**Application: Submit Transactions**

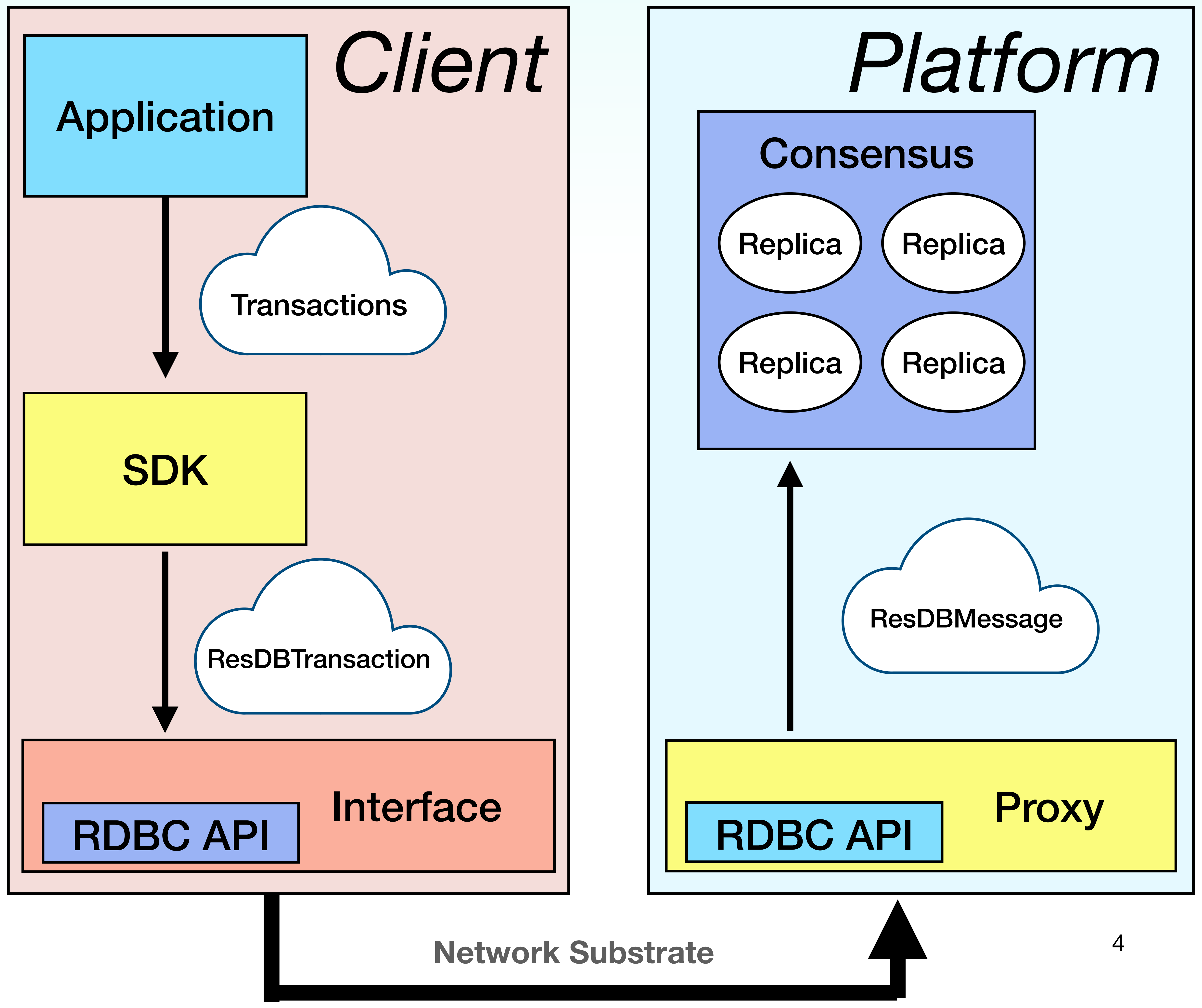
**Platform: Commit Transactions**

**Network: Exchange Messages**



# ResilientDB Transaction Workflow

## Construct and Send a ResDBTransaction to the Platform



1. **Applications** submit **client transactions** to SDK;

2. **SDK** transforms the client transactions into **ResDBTransaction** objects;

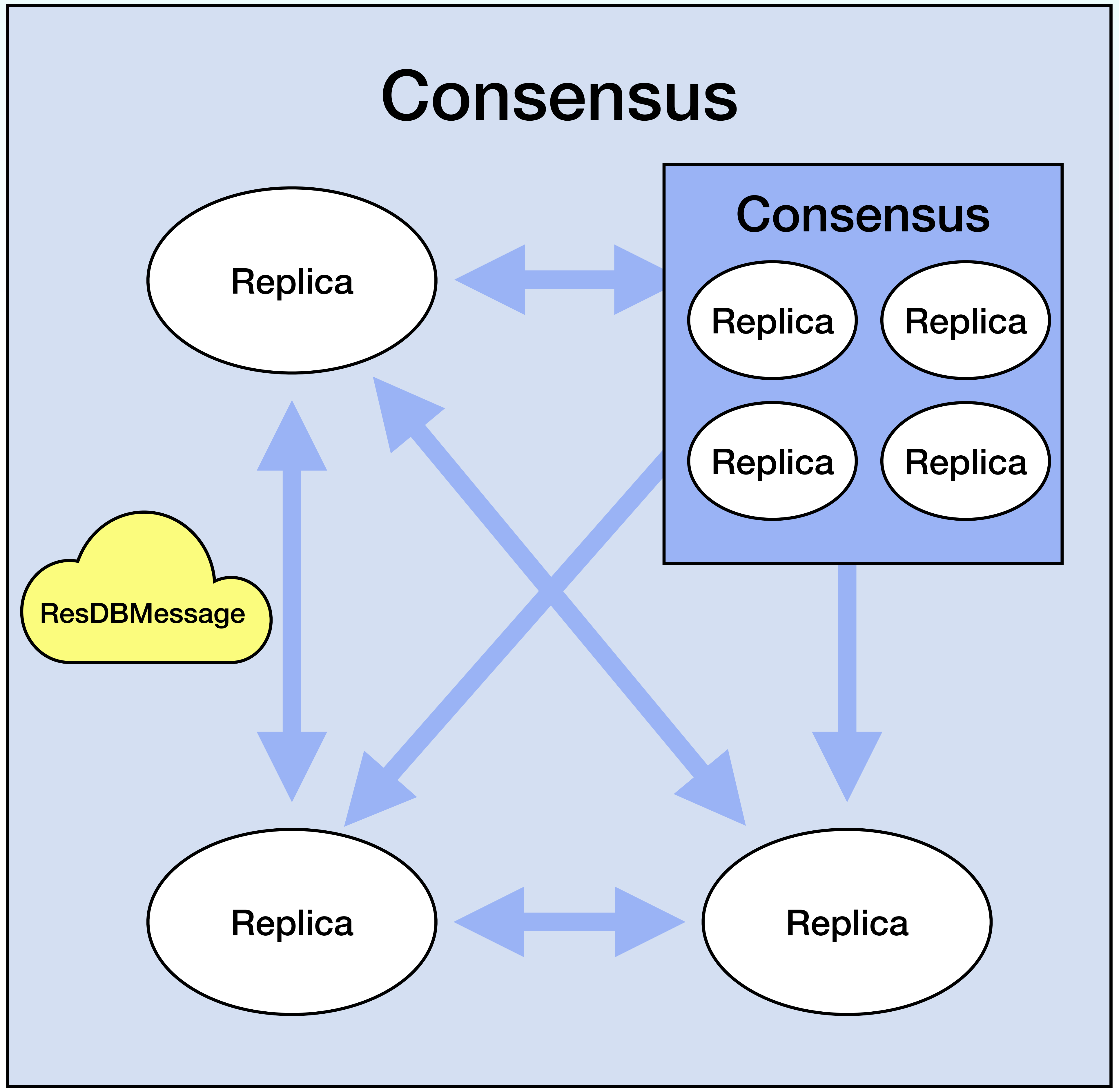
3. Sends the ResDBTransaction to **Proxy** by invoking the **RDBC API**;

4. The **ResDBTransaction** is delivered from the client to the **Proxy** via the **Network Substrate**;

5. The Proxy packs the ResDBTransaction into **ResDBMessage** and forwards it to **Replicas**

# ResilientDB Transaction Workflow

## Reach Agreement on the ResDBTransaction



1. **Applications** send **Operation and Data** to SDK;

2. **SDK** transforms the Operation and Data into a **ResDBTransaction** object;

3. Sends the ResDBTransaction to **Proxy** by **Calling the RDBC API**;

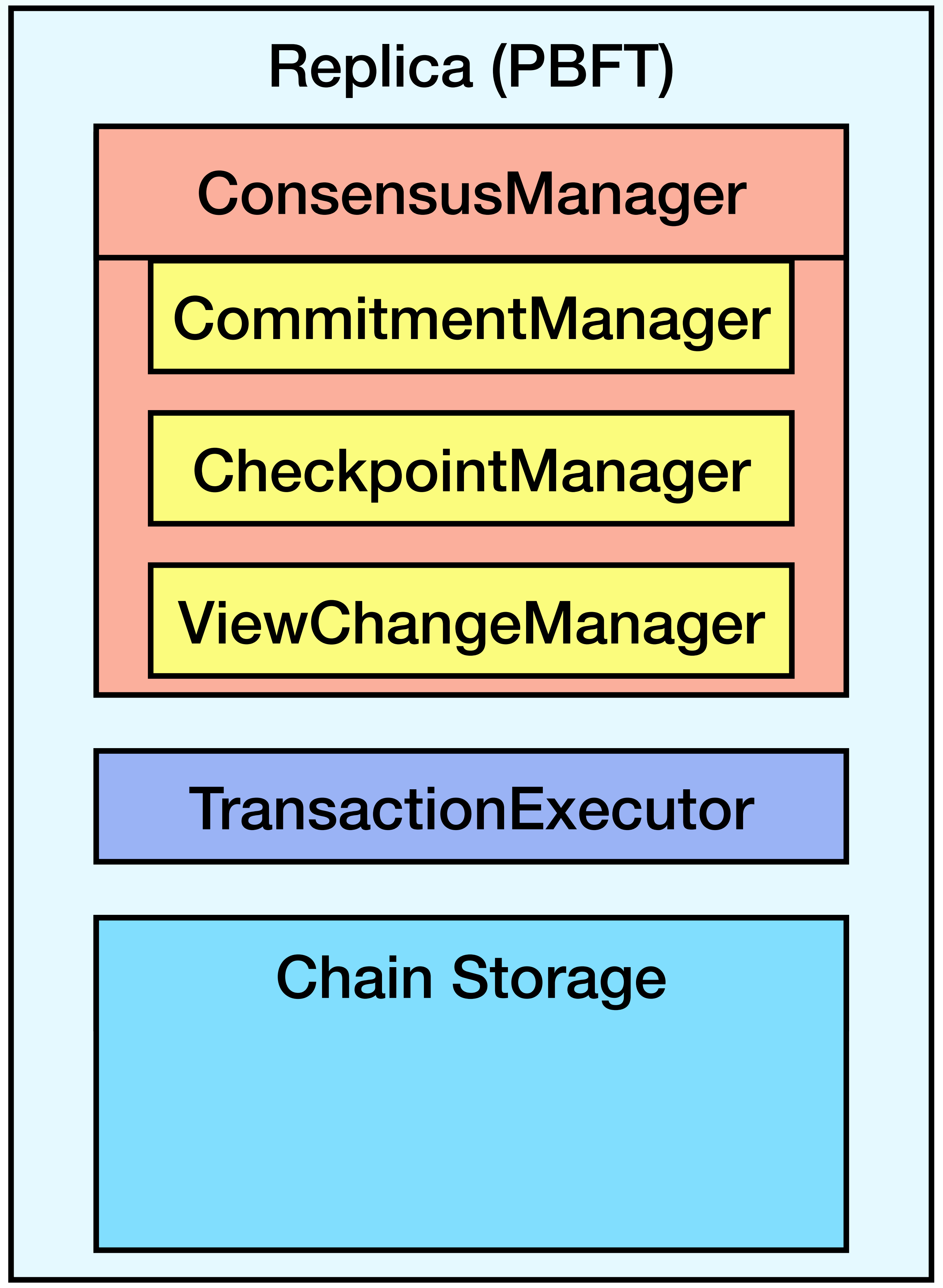
4. The **ResDBTransaction** is delivered from the User to the **Proxy** via the **Network Substrate**;

5. The Proxy packs the ResDBTransaction into **ResDBMessage** and forwards it to **Replicas**

6. Replicas exchange **consensus messages** with each other via the Network Substrate.

# ResilientDB Transaction Workflow

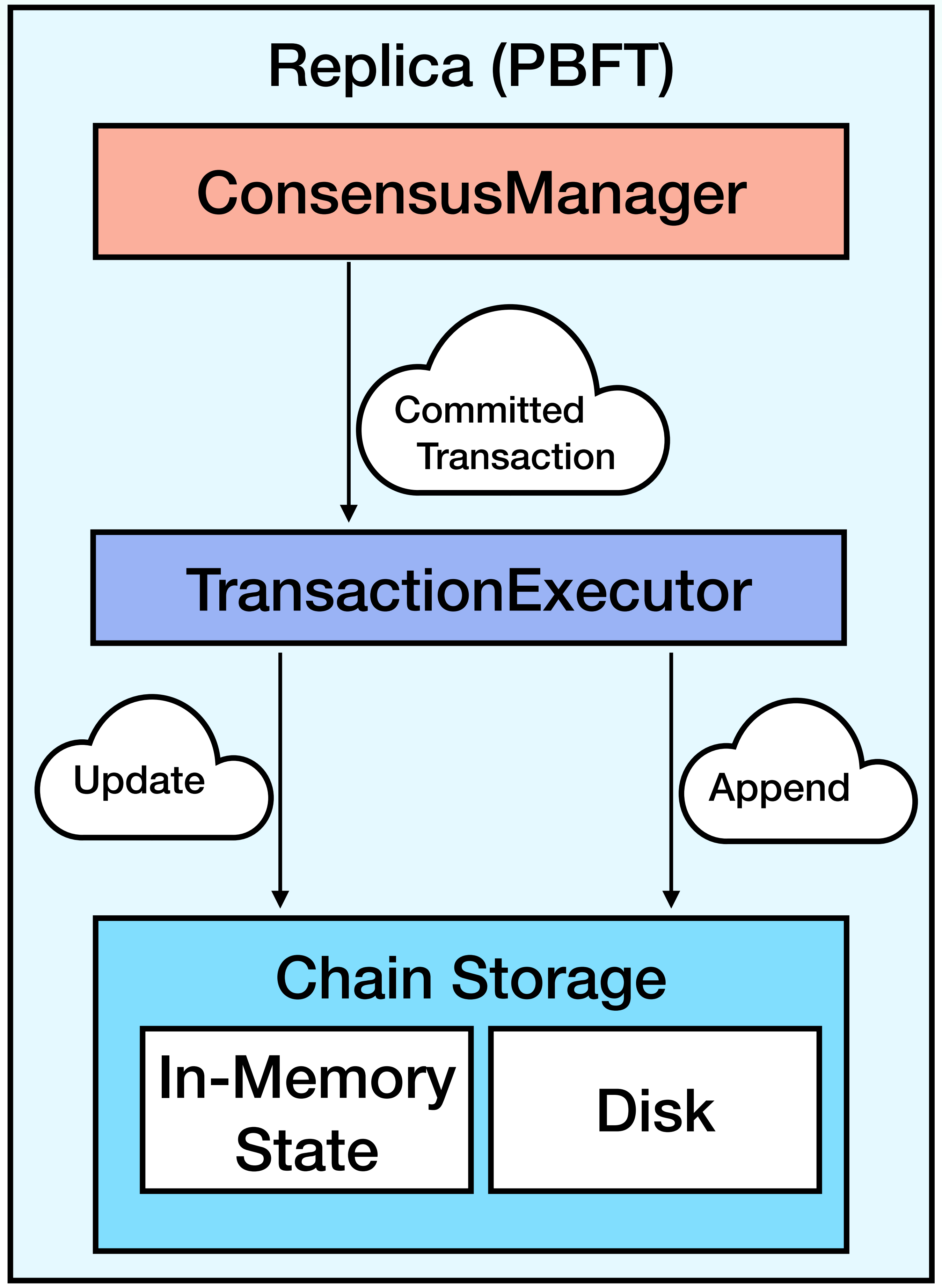
## Internal Structure of a PBFT replica



- **ConsensusManager:** Reaching Consensus on the order of Transactions
  - **CommitmentManager**
  - **CheckpointManager**
  - **ViewChangeManager**
- **TransactionExecutor:** Execute the committed transactions
- **Chain Storage:** In-memory and on disk

# ResilientDB Transaction Workflow

## Internal Structure of a PBFT replica

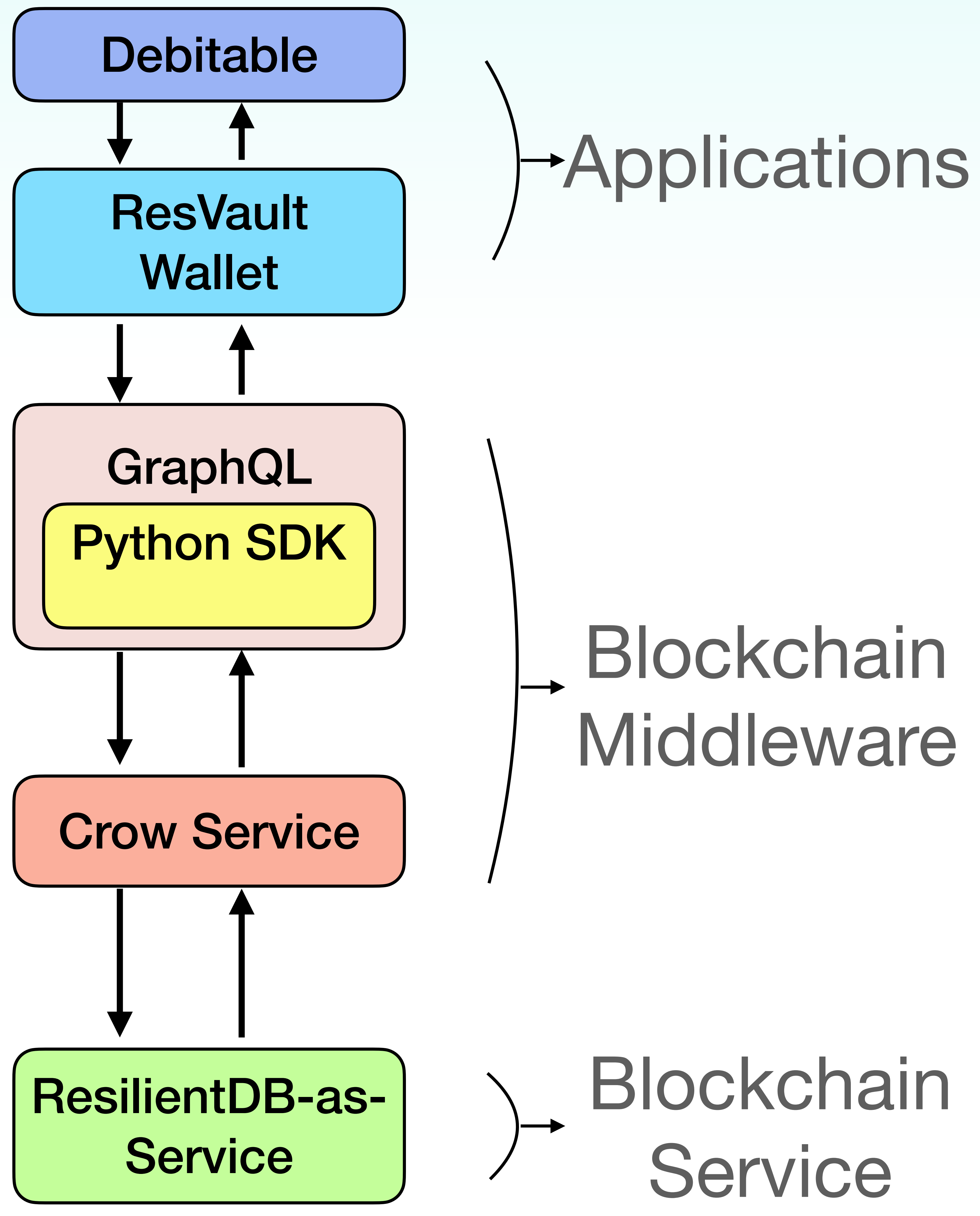


- **ConsensusManager:** Reaching Consensus on the order of Transactions
  - **CommitmentManager**
  - **CheckpointManage**
  - **ViewChangeManager**
- **TransactionExecutor:** Execute the committed transactions
- **Chain Storage:** In-memory and on disk

- Committed transactions are sent to **TransactionExecutor**
- Update the **In-Memory State** based on transaction data
- Append the transaction to ledger stored on **Disk**

# Building DApp on top of ResilientDB

## Debitable: An Example DApp Built on Top of ResilientDB



1. Deploy a ResilientDB Blockchain Service
2. Start Crow HTTP Service which provides HTTP Interface to ResilientDB Service
3. Using Python SDK to send HTTP requests, submitting transactions and fetching results
4. Deploy a GraphQL Server that wraps the Python SDK, supporting more efficient data retrieval and flexible queries
5. Build and install ResVault Wallet which generates and stores tokens securely on the chain
6. Develop and Deploy DApps using ResVault for token management, e.g., Debitable

## ResilientDB Docs

https://beacon.resilientdb.com/docs

## Thank You

https://resilientdb.incubator.apache.org

**Publications:**