# Lab 5: Understanding Replication and Consistency in ZooKeeper

## Objective

This lab aims to provide an understanding of how Apache ZooKeeper ensures replication and consistency in a distributed system. You will also learn how to modify a Java-based key-value store that utilizes ZooKeeper to automatically discover and connect to available servers.

## 1. ZooKeeper: Replication and Consistency

### Replication in ZooKeeper

ZooKeeper follows a leader-follower model for replication:  
- A leader node is elected to manage writes (modifications to data).  
- Multiple follower nodes replicate the data from the leader to ensure redundancy.  
- If the leader fails, a new leader is elected through the ZAB (ZooKeeper Atomic Broadcast) protocol.  
- All write operations go to the leader and are then replicated across all followers.

### Consistency in ZooKeeper

ZooKeeper provides sequential consistency, ensuring that:  
- All clients see updates in the same order.  
- Reads may be stale but will reflect the most recent committed state once they are updated.  
- The ZooKeeper sync() operation helps clients ensure they see the latest committed data.

### Important: Not Strong Consistency

The current implementation follows sequential consistency, meaning that a client may read stale data if it connects to a follower that has not yet received the latest update from the leader.

### Not Eventual Consistency

In Eventual consistency updates may be seen in different orders by different clients.

## 2. Code Overview

### ZookeeperClient.java

Establishes a connection with the ZooKeeper service and provides a method to retrieve the connected instance.

### ZookeeperKeyValueStore.java

Implements a key-value store where keys are stored in ZooKeeper. Maintains a local cache for fast access while ensuring updates are propagated to ZooKeeper.

### ApplicationServer.java

Represents a distributed server that interacts with clients. Registers itself with ZooKeeper for dynamic discovery and handles client requests to store and retrieve key-value pairs.

### ClientApp.java

Connects to ZooKeeper and retrieves the list of available servers. Interacts with the chosen server to store (put key=value) and retrieve (get key) data.

## 3. Running the Code (Eclipse Project)

### Prerequisites

Ensure you have the following installed:  
- Java (JDK 8 or later)  
- Apache ZooKeeper (Running on localhost:2181)  
- Eclipse IDE (with Java support)

### Steps to Run

### Start ZooKeeper

Open a terminal and start ZooKeeper:  
  
zkServer.sh start

### Import the Project into Eclipse

Open Eclipse and import the project ***keyvalrep***. Ensure that all .java files are in the src/ directory.

### Start Multiple Application Servers

Run multiple instances of ApplicationServer.java, each with a different port:  
  
java ApplicationServer 5000  
java ApplicationServer 5001  
java ApplicationServer 5002

### Run the First Client and Store Data

Run ClientApp.java and manually enter a server address (e.g., localhost:5000). Test storing and retrieving key-value pairs:  
  
put name=Alice  
get name

### Run a Second Client to Show Replication Works

Start another instance of ClientApp.java. Enter a different server address (e.g., localhost:5001). Retrieve the previously stored value:  
  
get name  
  
If the value is retrieved correctly, it shows that replication and consistency is working.

## 4. Exercise: Implement Automatic Server Selection

Objective: Modify the client so that it automatically selects an available server instead of requiring manual input.  
  
Tasks:  
1. Modify runClient() in ClientApp.java:  
 - Remove the manual prompt for entering a server address.  
 - Call fetchAvailableServers() to retrieve available servers.  
 - Call getAvailableServer() to select a random server.  
 - Implement retry logic if no servers are available.  
  
2. Test the Updated Client:  
 - Run the modified ClientApp.java.  
 - Ensure it selects an available server from ZooKeeper.  
 - Verify that if a server goes down, the client retries another server.  
  
Expected Outcome:  
- The client automatically connects to an available server.  
- If a server fails, the client retries another available server.  
- The client can store and retrieve data seamlessly.

## Submission

1. Explain how Sequential consistency is different from Eventual and Strong consistencies. Write your answer in a text file.
2. Zip the updated project and the text file. Upload the zip file to the courseweb link. The file name should be your registration number.