7SENG007C.2
Concurrent and Distributed Systems
Coursework 2

Name: Asiri Mevan Senanayake
IIT ID: 20221919
UOW ID: w1986425

Table of Content

1. The Introduction of the System

To power a globally distributed online retail platform with a large user base, a highly available and scalable distributed inventory system is required. This system should allow users to browse through product catalogues, add items to their virtual shopping carts, and successfully complete purchases.

Key functionalities include updating inventory quantities, managing shopping carts, handling simultaneous checkouts for limited-quantity items, and ensuring accurate order processing.

To achieve this, a microservices architecture is recommended, with services dedicated to inventory management, user handling, shopping cart management, checkout processes, payment processing, and order management.

The system should leverage distributed locking mechanisms for item allocation during checkouts and employ containerization, cloud deployment, and data storage solutions for scalability. By focusing on these aspects, the distributed inventory system will enable efficient and reliable online shopping experiences for customers worldwide.

The system can handle multiple order requests by serving orders based on a first come, first served basis.

2. Requirements

2.1 Online Retail Shop with mini-distributed inventory system

2.1.1 Functional requirements

FR-Id	**Requirement**
Fr-01	The system administrator updates the quantities of the goods when a new shipment arrives
Fr-02	Shoppers from different locations log into the system and place different items in the shopping carts that they wish to purchase
Fr-03	Multiple shoppers try to check out the shopping carts which has the same item with limited quantity, and the system should allocate/block items based on the first-come-first-serve basis
Fr-04	The system should be able to handle multiple checkouts made at the same time requiring the same item. So, the same item should not be assigned for two purchases.

Table 1: Functional Requirements

2.1.2 Non-Functional requirements

NFR-Id	**Requirement**
NFr-01	The system should be **highly available**: The system should consist of more than one process which is ready to accept orders. The traders can place orders to any node in the system.
NFr-02	**Accuracy**: The system must always produce consistent results regardless of which node receives the order. (Hint: Consider having only designated one node as the processing node in your system always)
NFr-03	**Fault Tolerance**: In event of a failure of a node, other nodes must continue to process the orders without losing accuracy

Table 2: Non-Functional Requirements

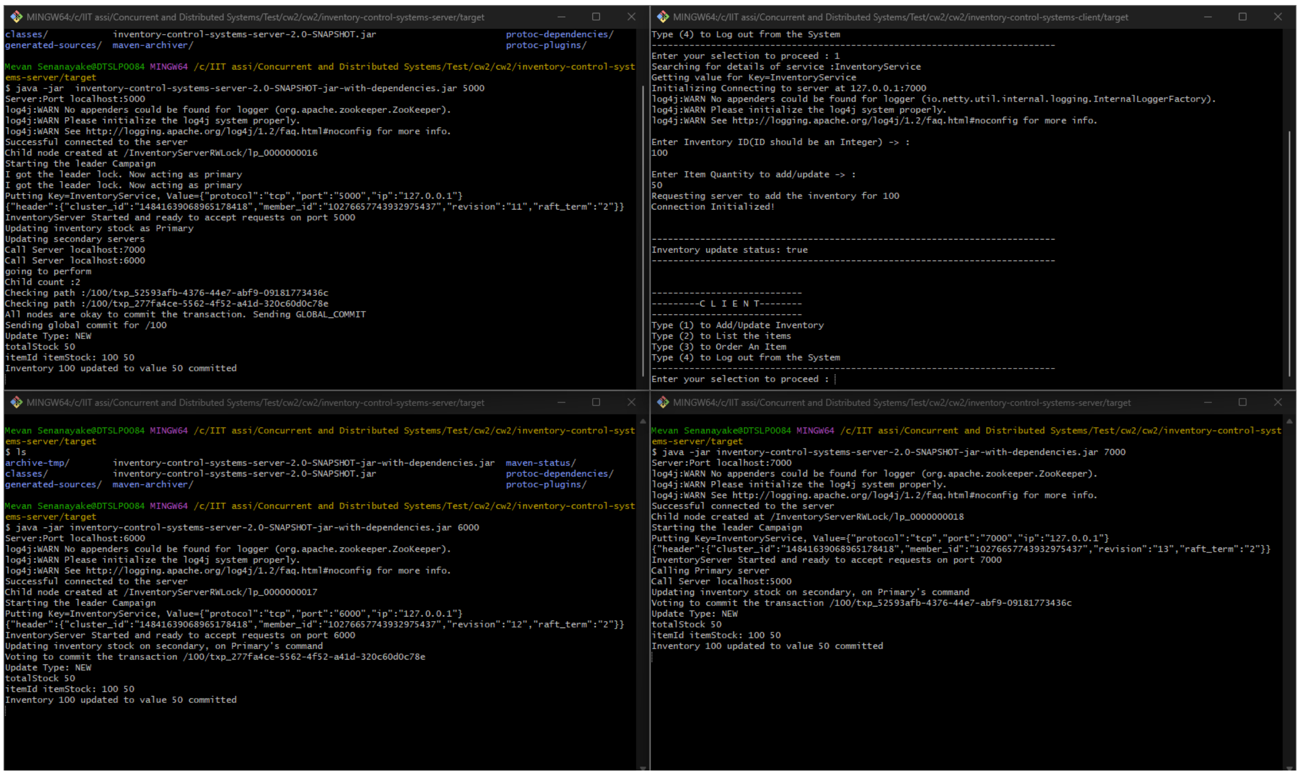
3. System Behaviors

3.1 When a system admin adds items to the system (explain how all nodes are informed about the new arrival of goods)

If the item ID is not in the retail Shop stock, the system will **add** the item along with the amount.

If the item ID is in the retail Shop Stock, the system will **update** the quantity.

If the client is linked to the main, the primary will add or update itself and others, or the backup server will call the primary to ask for an update to the retail Shop Stock.



```
MINGW64:/c/IIT assi/Concurrent and Distributed Systems/Test/cw2/cw2/inventory-control-systems-server/target
classes/           inventory-control-systems-server-2.0-SNAPSHOT.jar                          protoc-dependencies/
generated-sources/  maven-archiver/                                                           protoc-plugins/

Mevan Senanayake@DTSLP0084 MINGW64 /c/IIT assi/Concurrent and Distributed Systems/Test/cw2/cw2/inventory-control-syst
ems-server/target
$ java -jar  inventory-control-systems-server-2.0-SNAPSHOT-jar-with-dependencies.jar 5000
Server:Port localhost:5000
log4j:WARN No appenders could be found for logger (org.apache.zookeeper.ZooKeeper).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Successful connected to the server
Child node created at /InventoryServerRWLock/lp_0000000016
Starting the leader Campaign
I got the leader lock. Now acting as primary
I got the leader lock. Now acting as primary
Putting Key=InventoryService, Value={"protocol":"tcp","port":"5000","ip":"127.0.0.1"}
{"header":{"cluster_id":"14841639068965178418","member_id":"10276657743932975437","revision":"11","raft_term":"2"}}
InventoryServer Started and ready to accept requests on port 5000
Updating inventory stock as Primary
Updating secondary servers
Call Server localhost:7000
Call Server localhost:6000
going to perform
Child count :2
Checking path :/100/txp_52593afb-4376-44e7-abf9-09181773436c
Checking path :/100/txp_277fa4ce-5562-4f52-a41d-320c60d0c78e
All nodes are okay to commit the transaction. Sending GLOBAL_COMMIT
Sending global commit for /100
Update Type: NEW
totalStock 50
itemId itemStock: 100 50
Inventory 100 updated to value 50 committed
```

```
MINGW64:/c/IIT assi/Concurrent and Distributed Systems/Test/cw2/cw2/inventory-control-systems-client/target
Type (4) to Log out from the System
---------------------------------------------------------------------------
Enter your selection to proceed : 1
Searching for details of service :InventoryService
Getting value for Key=InventoryService
Initializing Connecting to server at 127.0.0.1:7000
log4j:WARN No appenders could be found for logger (io.netty.util.internal.logging.InternalLoggerFactory).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.

Enter Inventory ID(ID should be an Integer) -> :
100

Enter Item Quantity to add/update -> :
50
Requesting server to add the inventory for 100
Connection Initialized!


---------------------------------------------------------------------------
Inventory update status: true
---------------------------------------------------------------------------



----------------------------
---------C L I E N T--------
----------------------------
Type (1) to Add/Update Inventory
Type (2) to List the items
Type (3) to Order An Item
Type (4) to Log out from the System
---------------------------------------------------------------------------
Enter your selection to proceed :
```

```
MINGW64:/c/IIT assi/Concurrent and Distributed Systems/Test/cw2/cw2/inventory-control-systems-server/target

Mevan Senanayake@DTSLP0084 MINGW64 /c/IIT assi/Concurrent and Distributed Systems/Test/cw2/cw2/inventory-control-syst
ems-server/target
$ ls
archive-tmp/        inventory-control-systems-server-2.0-SNAPSHOT-jar-with-dependencies.jar  maven-status/
classes/            inventory-control-systems-server-2.0-SNAPSHOT.jar                        protoc-dependencies/
generated-sources/  maven-archiver/                                                          protoc-plugins/

Mevan Senanayake@DTSLP0084 MINGW64 /c/IIT assi/Concurrent and Distributed Systems/Test/cw2/cw2/inventory-control-syst
ems-server/target
$ java -jar inventory-control-systems-server-2.0-SNAPSHOT-jar-with-dependencies.jar 6000
Server:Port localhost:6000
log4j:WARN No appenders could be found for logger (org.apache.zookeeper.ZooKeeper).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Successful connected to the server
Child node created at /InventoryServerRWLock/lp_0000000017
Starting the leader Campaign
Putting Key=InventoryService, Value={"protocol":"tcp","port":"6000","ip":"127.0.0.1"}
{"header":{"cluster_id":"14841639068965178418","member_id":"10276657743932975437","revision":"12","raft_term":"2"}}
InventoryServer Started and ready to accept requests on port 6000
Updating inventory stock on secondary, on Primary's command
Voting to commit the transaction /100/txp_277fa4ce-5562-4f52-a41d-320c60d0c78e
Update Type: NEW
totalStock 50
itemId itemStock: 100 50
Inventory 100 updated to value 50 committed
```

```
MINGW64:/c/IIT assi/Concurrent and Distributed Systems/Test/cw2/cw2/inventory-control-systems-server/target

Mevan Senanayake@DTSLP0084 MINGW64 /c/IIT assi/Concurrent and Distributed Systems/Test/cw2/cw2/inventory-control-syst
ems-server/target
$ java -jar inventory-control-systems-server-2.0-SNAPSHOT-jar-with-dependencies.jar 7000
Server:Port localhost:7000
log4j:WARN No appenders could be found for logger (org.apache.zookeeper.ZooKeeper).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Successful connected to the server
Child node created at /InventoryServerRWLock/lp_0000000018
Starting the leader Campaign
Putting Key=InventoryService, Value={"protocol":"tcp","port":"7000","ip":"127.0.0.1"}
{"header":{"cluster_id":"14841639068965178418","member_id":"10276657743932975437","revision":"13","raft_term":"2"}}
InventoryServer Started and ready to accept requests on port 7000
Calling Primary server
Call Server localhost:5000
Updating inventory stock on secondary, on Primary's command
Voting to commit the transaction /100/txp_52593afb-4376-44e7-abf9-09181773436c
Update Type: NEW
totalStock 50
itemId itemStock: 100 50
Inventory 100 updated to value 50 committed
```

Figure 1: To propagate the modifications to all secondary servers while the server functions as the primary, a call must be sent to them.

3.2 A shopping cart is checked out (explain how the update of the inventory update is done across the system)

If there is retail shop item stock, the system will process the request according to the principle of first come, first served; otherwise, it will refuse the order request.

```
private void updateRetailShop() {

    if (tempDataHolder != null) {
        Integer itemId = tempDataHolder.getKey();
        Integer value = tempDataHolder.getValue();
        Integer currentStock = server.getRetailShopStock(itemId);

        System.out.println("Update Type: " + updateType);
        if (updateType == "NEW") {
            Integer totalStock = (currentStock + value);
            System.out.println("totalStock " + totalStock);
            server.setRetailShopStock(itemId, totalStock);
            transactionStatus = true;
        } else {
            transactionStatus = false;
        }

        if (updateType == "ORDER" && value <= currentStock && currentStock != 0) {
            Integer remainingStock = (currentStock - value);
            System.out.println("remainingStock " + remainingStock);
            server.orderRetailShopStock(itemId, remainingStock);
            transactionStatus = true;
        } else if (updateType == "ORDER") {
            transactionStatus = false;
        }

        System.out.println("RetailShop " + itemId + " updated to value " + value + "
committed");
        tempDataHolder = null;
    }
}
```

Code Snippet 1: RetailShopServiceImpl.updateRetailShop()

3.3 When shoppers try to check out the same item at the same time where the stock of item can only fulfill one order (explain how the system makes sure only one order is processed and the other is rejected)

A primary-based protocol has been used to create the system, which implies that all requests are routed to the primary for processing.

The two previous requests will be forwarded to the primary, and they will be handled according to the principle of first come, first served (who obtains the lock first), as stated in the paragraph mentioned above (2.2).

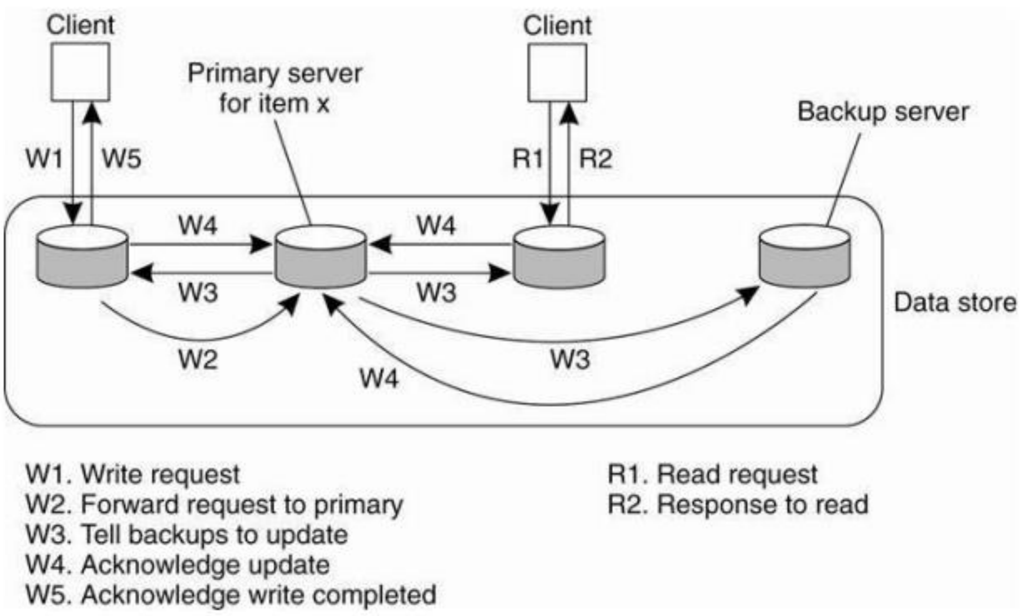
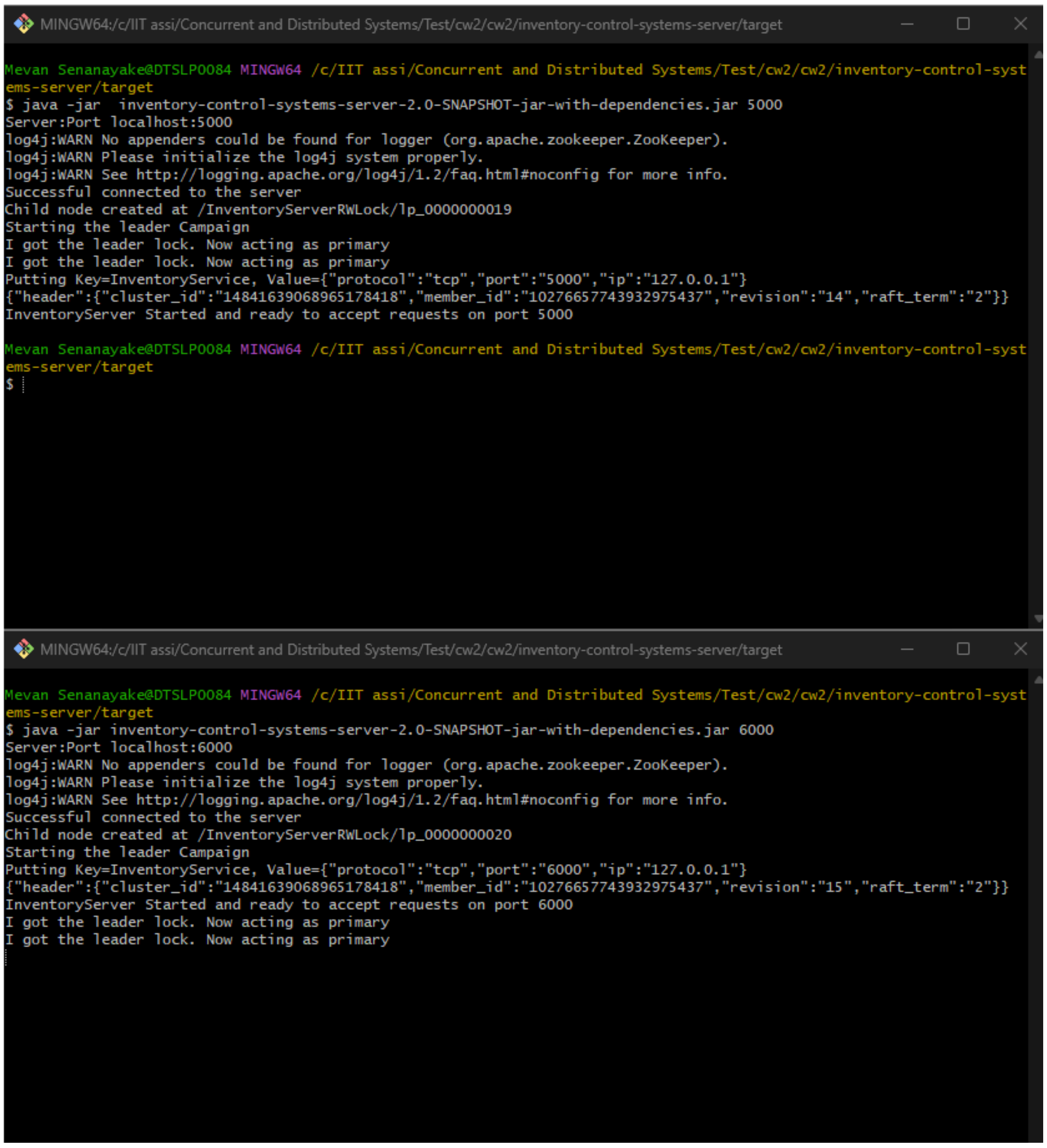


Figure 2: : Primary based remote write protocol

3.4 One node exits the system (explain how the system continues to function without losing its accuracy and availability)

Since the system is built around a primary-based protocol, if the primary server is not available, the system will continue to assign a primary and connect the client to the next server that becomes available. Since all nodes have updated data, there won't be any data corruption as a result. With the newly appointed primary, the system will be operational.



MINGW64:/c/IIT assi/Concurrent and Distributed Systems/Test/cw2/cw2/inventory-control-systems-server/target

```
Mevan Senanayake@DTSLP0084 MINGW64 /c/IIT assi/Concurrent and Distributed Systems/Test/cw2/cw2/inventory-control-syst
ems-server/target
$ java -jar  inventory-control-systems-server-2.0-SNAPSHOT-jar-with-dependencies.jar 5000
Server:Port localhost:5000
log4j:WARN No appenders could be found for logger (org.apache.zookeeper.ZooKeeper).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Successful connected to the server
Child node created at /InventoryServerRWLock/lp_0000000019
Starting the leader Campaign
I got the leader lock. Now acting as primary
I got the leader lock. Now acting as primary
Putting Key=InventoryService, Value={"protocol":"tcp","port":"5000","ip":"127.0.0.1"}
{"header":{"cluster_id":"14841639068965178418","member_id":"10276657743932975437","revision":"14","raft_term":"2"}}
InventoryServer Started and ready to accept requests on port 5000

Mevan Senanayake@DTSLP0084 MINGW64 /c/IIT assi/Concurrent and Distributed Systems/Test/cw2/cw2/inventory-control-syst
ems-server/target
$
```

MINGW64:/c/IIT assi/Concurrent and Distributed Systems/Test/cw2/cw2/inventory-control-systems-server/target

```
Mevan Senanayake@DTSLP0084 MINGW64 /c/IIT assi/Concurrent and Distributed Systems/Test/cw2/cw2/inventory-control-syst
ems-server/target
$ java -jar inventory-control-systems-server-2.0-SNAPSHOT-jar-with-dependencies.jar 6000
Server:Port localhost:6000
log4j:WARN No appenders could be found for logger (org.apache.zookeeper.ZooKeeper).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Successful connected to the server
Child node created at /InventoryServerRWLock/lp_0000000020
Starting the leader Campaign
Putting Key=InventoryService, Value={"protocol":"tcp","port":"6000","ip":"127.0.0.1"}
{"header":{"cluster_id":"14841639068965178418","member_id":"10276657743932975437","revision":"15","raft_term":"2"}}
InventoryServer Started and ready to accept requests on port 6000
I got the leader lock. Now acting as primary
I got the leader lock. Now acting as primary
```

Figure 3: One node exits the system

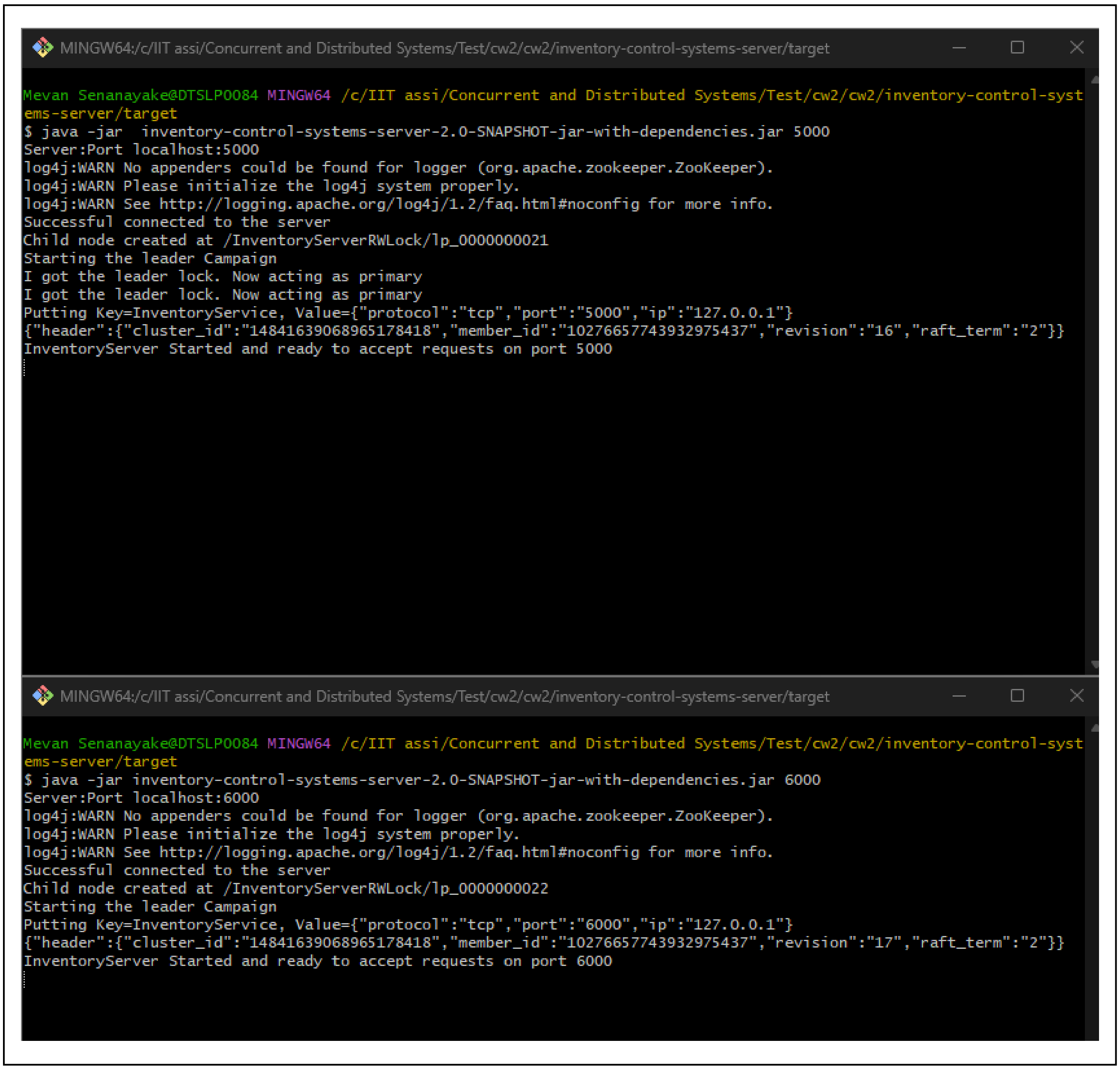
```java
class LeaderCampaignThread implements Runnable {
    private byte[] currentLeaderData = null;

    @Override
    public void run() {
        System.out.println("Starting the leader Campaign");
        try {
            boolean leader = leaderLock.tryAcquireLock();
            while (!leader) {
                byte[] leaderData = leaderLock.getLockHolderData();
                if (currentLeaderData != leaderData) {
                    currentLeaderData = leaderData;
                    setCurrentLeaderData(currentLeaderData);
                }
                Thread.sleep(10000);
                leader = leaderLock.tryAcquireLock();
            }
            System.out.println("I got the leader lock. Now acting as primary");
            currentLeaderData = null;
            beTheLeader();
        } catch (Exception e) {
        }
    }
}
```

Code Snippet 2: Leader Campaign Thread

3.5 One node joins the system or restarts (explain how the new node catches up with the system state and starts providing the service)

Users may get updated data through the client by connecting to the leader node, which will continuously updating all new joins.



```
MINGW64:/c/IIT assi/Concurrent and Distributed Systems/Test/cw2/cw2/inventory-control-systems-server/target

Mevan Senanayake@DTSLP0084 MINGW64 /c/IIT assi/Concurrent and Distributed Systems/Test/cw2/cw2/inventory-control-syst
ems-server/target
$ java -jar  inventory-control-systems-server-2.0-SNAPSHOT-jar-with-dependencies.jar 5000
Server:Port localhost:5000
log4j:WARN No appenders could be found for logger (org.apache.zookeeper.ZooKeeper).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Successful connected to the server
Child node created at /InventoryServerRWLock/lp_0000000021
Starting the leader Campaign
I got the leader lock. Now acting as primary
I got the leader lock. Now acting as primary
Putting Key=InventoryService, Value={"protocol":"tcp","port":"5000","ip":"127.0.0.1"}
{"header":{"cluster_id":"14841639068965178418","member_id":"10276657743932975437","revision":"16","raft_term":"2"}}
InventoryServer Started and ready to accept requests on port 5000
```

```
MINGW64:/c/IIT assi/Concurrent and Distributed Systems/Test/cw2/cw2/inventory-control-systems-server/target

Mevan Senanayake@DTSLP0084 MINGW64 /c/IIT assi/Concurrent and Distributed Systems/Test/cw2/cw2/inventory-control-syst
ems-server/target
$ java -jar inventory-control-systems-server-2.0-SNAPSHOT-jar-with-dependencies.jar 6000
Server:Port localhost:6000
log4j:WARN No appenders could be found for logger (org.apache.zookeeper.ZooKeeper).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Successful connected to the server
Child node created at /InventoryServerRWLock/lp_0000000022
Starting the leader Campaign
Putting Key=InventoryService, Value={"protocol":"tcp","port":"6000","ip":"127.0.0.1"}
{"header":{"cluster_id":"14841639068965178418","member_id":"10276657743932975437","revision":"17","raft_term":"2"}}
InventoryServer Started and ready to accept requests on port 6000
```

Figure 4: One node joins the system or restarts