# **DISTRIBUTED SYSTEMS**

## **Assignment-2**

Submitted by:-

Abhishek Srivastav - CS21M001

Shikher Chhawchharia – CS21M014

## INTRODUCTION:-

In this assignment, we have implemented a peer-to-peer messaging system which follows a sequential consistency model. This has been achieved using the "**Primary based Remote-Write protocol"** for update propagation and "**Total ordering"** for maintaining a consistent view of the data store. We will present the "Implementation Details" and a "Demonstration" of the same in the subsequent sections.

## IMPLEMENTATION:-

**Setup**

1. Developed on a Linux-based system.
2. Using Python programming language.
3. Mininet has been used to set up a virtual network of 'n' nodes.

**Design**

1. To facilitate communication between the nodes, socket programming has been used as it gives control over the flow of messages.
2. Servers will use multi-threading to carry out several tasks in parallel.
3. The server with id '0' is always assigned as the leader or the primary server.
4. Two separate code files were written; one for the primary server and one for the rest of the replica servers. This was done for ease of development as doing this separates the functionalities of the two types of servers.
5. Types of messages being exchanged:-
    a. UPDATE:- the update that the client wants to initiate. Sent from client to primary.
       Format: <node id|data item|new value>
    b. EXECUTE_UPDATE:- the update that the primary wants all the replicas to execute on their local copies. Sent from the primary server to all the replica servers.
       Format <node id|data item|new value|sequence number>
    c. ACK:- acknowledgments will be sent by the replica servers to the primary after they have executed an update in the format <" update ack"|replica id|sequence number>. The primary will send an acknowledgment to the client, after all the replicas have acknowledged the update, in the format <" update ack">.

    d. Terminate:- when a server wants to stop, it will send a terminate request to the primary and wait for the termination signal. Format: <”Terminate”|node id>.
    e. Terminate_Signal:- when the primary receives a terminate request from all the servers it will send a terminate signal to all the servers. Format: <” Terminate signal”>.
6. The sending and receiving of all the updates will be done on port no. 8080 and that of all the acknowledgments will be done on port no. 8081
7. Threads used at primary server:-
    a. A dedicated thread “recv_thread” which will listen for incoming update messages.
    b. A dedicated thread “ack_thread”which will listen for incoming acknowledgments.
    c. An “update_thread” which will take the update from the “recv_thread”. It will then follow the primary based remote write protocol to make the update to the datastore and send an ack. back to the client.
8. Threads used at replica servers:-
    a. A dedicated thread “recv_thread” which will listen for incoming update messages.
    b. An “update_thread” which will take the update from the “recv_thread”. It will then check the order of the update based on total ordering and then either execute the update, if it follows the ordering, or buffer it.
9. Since there will be multiple threads running in parallel, mutex locks have been used to access shared resources. Different types of resources are accessed through different mutex locks to reduce contention.

## DEMONSTRATION:-

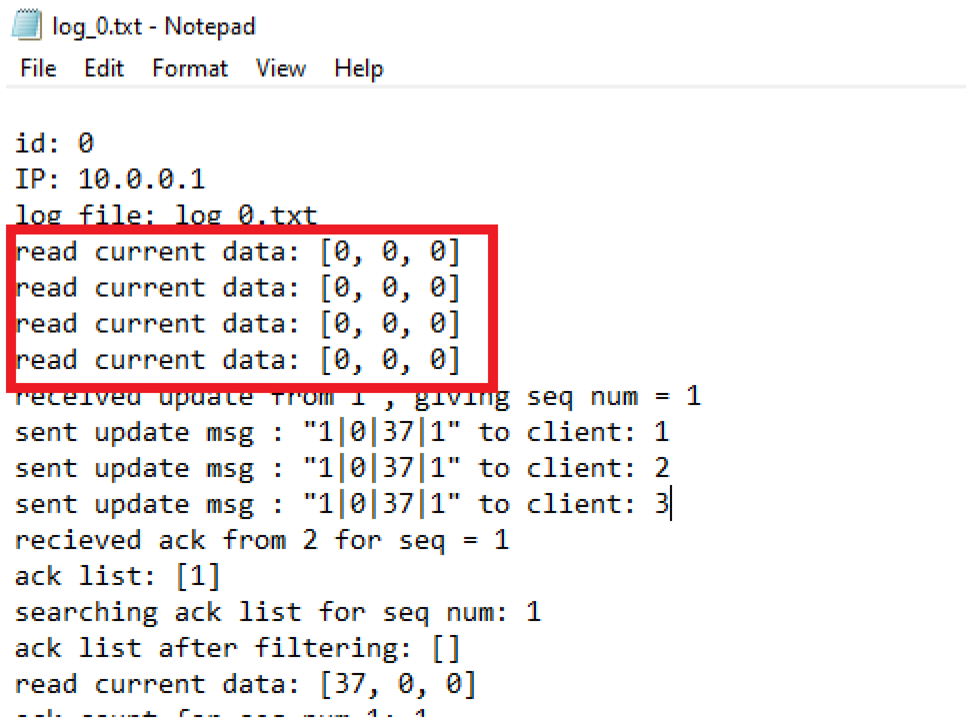
### Running the program

1. on the terminal run -> sudo mn --topo=tree,1,< value of 'n', no. of processes>
2. to the mininet prompt give -> xterm h1 h2 h3 (if no. of processes = 3)
3. to run the primary server -> python3 primary.py 0 <value of 'n'>
4. to run the replica servers -> python3 server.py <server_id starting from 1 to n-1> 0 <value of 'n'> <update prob>

**Example**

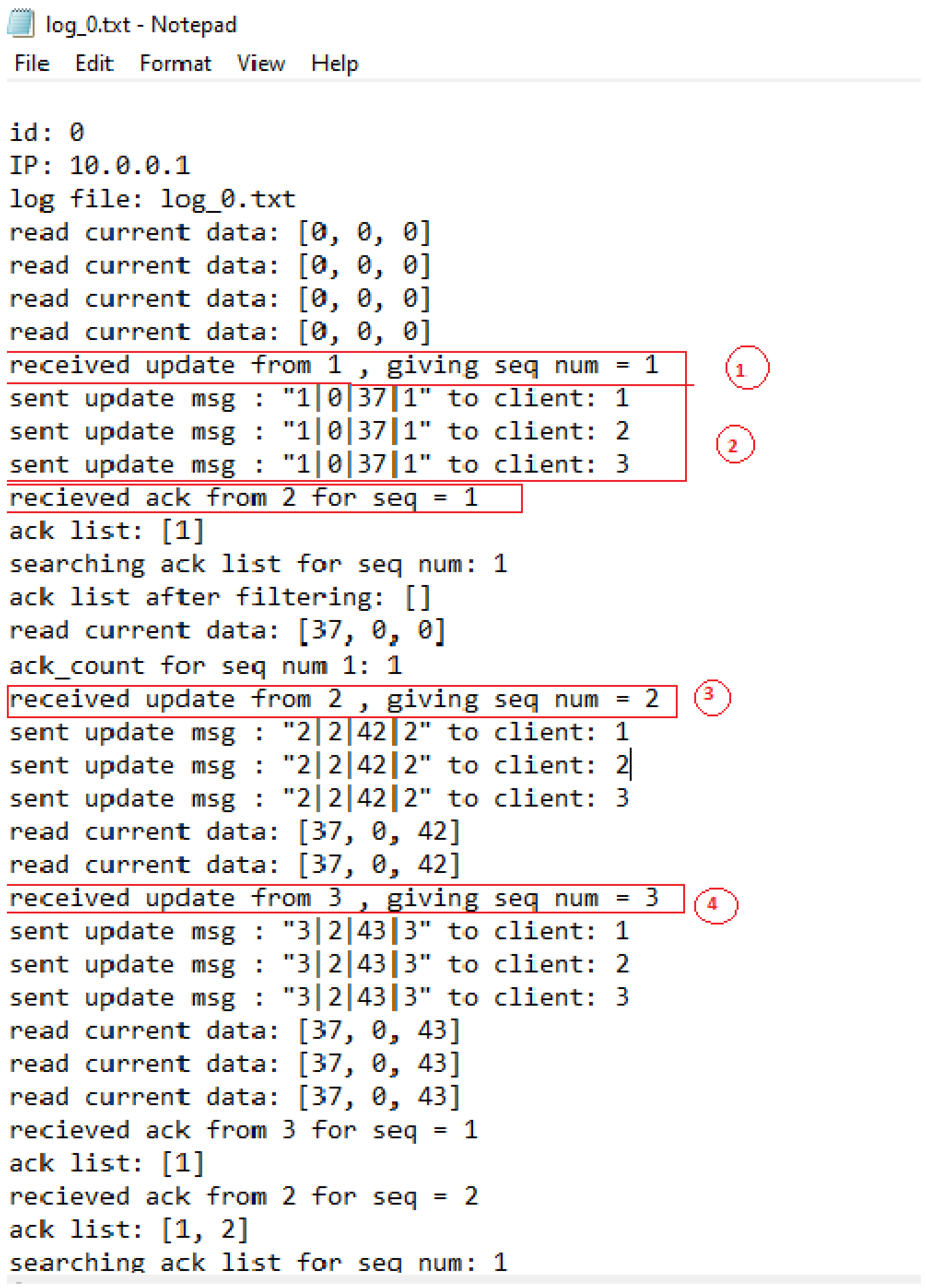
We have assigned P0 as the primary server.

All processes read the data automatically.



```
id: 0
IP: 10.0.0.1
log file: log 0.txt
read current data: [0, 0, 0]
read current data: [0, 0, 0]
read current data: [0, 0, 0]
read current data: [0, 0, 0]
recieved update from 1 , giving seq num = 1
sent update msg : "1|0|37|1" to client: 1
sent update msg : "1|0|37|1" to client: 2
sent update msg : "1|0|37|1" to client: 3
recieved ack from 2 for seq = 1
ack list: [1]
searching ack list for seq num: 1
ack list after filtering: []
read current data: [37, 0, 0]
ack count for seq num 1: 1
```

1. The order in which the primary received updates.

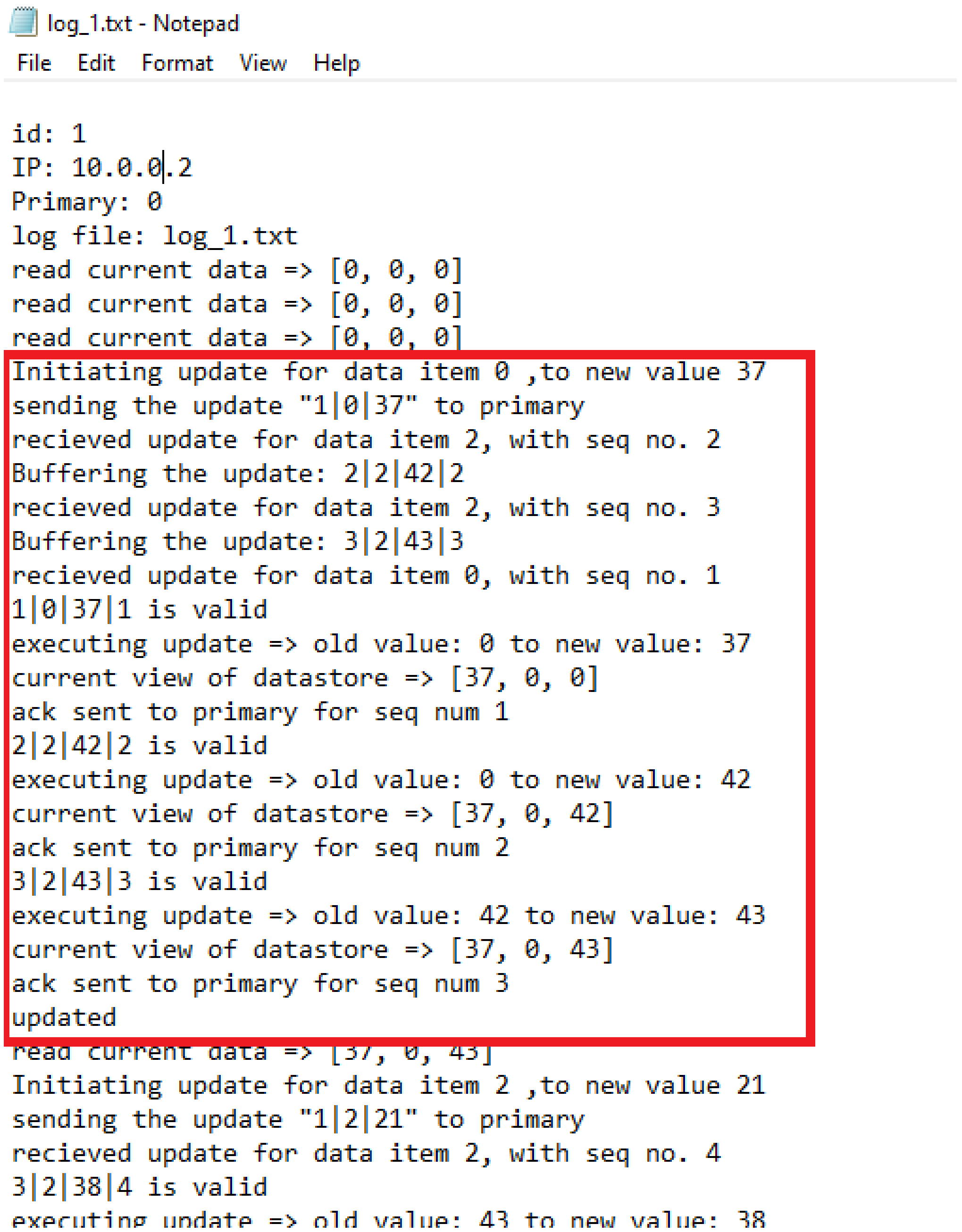


```
id: 0
IP: 10.0.0.1
log file: log_0.txt
read current data: [0, 0, 0]
read current data: [0, 0, 0]
read current data: [0, 0, 0]
read current data: [0, 0, 0]
received update from 1 , giving seq num = 1
sent update msg : "1|0|37|1" to client: 1
sent update msg : "1|0|37|1" to client: 2
sent update msg : "1|0|37|1" to client: 3
recieved ack from 2 for seq = 1
ack list: [1]
searching ack list for seq num: 1
ack list after filtering: []
read current data: [37, 0, 0]
ack_count for seq num 1: 1
received update from 2 , giving seq num = 2
sent update msg : "2|2|42|2" to client: 1
sent update msg : "2|2|42|2" to client: 2
sent update msg : "2|2|42|2" to client: 3
read current data: [37, 0, 42]
read current data: [37, 0, 42]
received update from 3 , giving seq num = 3
sent update msg : "3|2|43|3" to client: 1
sent update msg : "3|2|43|3" to client: 2
sent update msg : "3|2|43|3" to client: 3
read current data: [37, 0, 43]
read current data: [37, 0, 43]
read current data: [37, 0, 43]
recieved ack from 3 for seq = 1
ack list: [1]
recieved ack from 2 for seq = 2
ack list: [1, 2]
searching ack list for seq num: 1
```

Block 1, 3, and block 4 are showing that the primary has received their updates in sequence order.

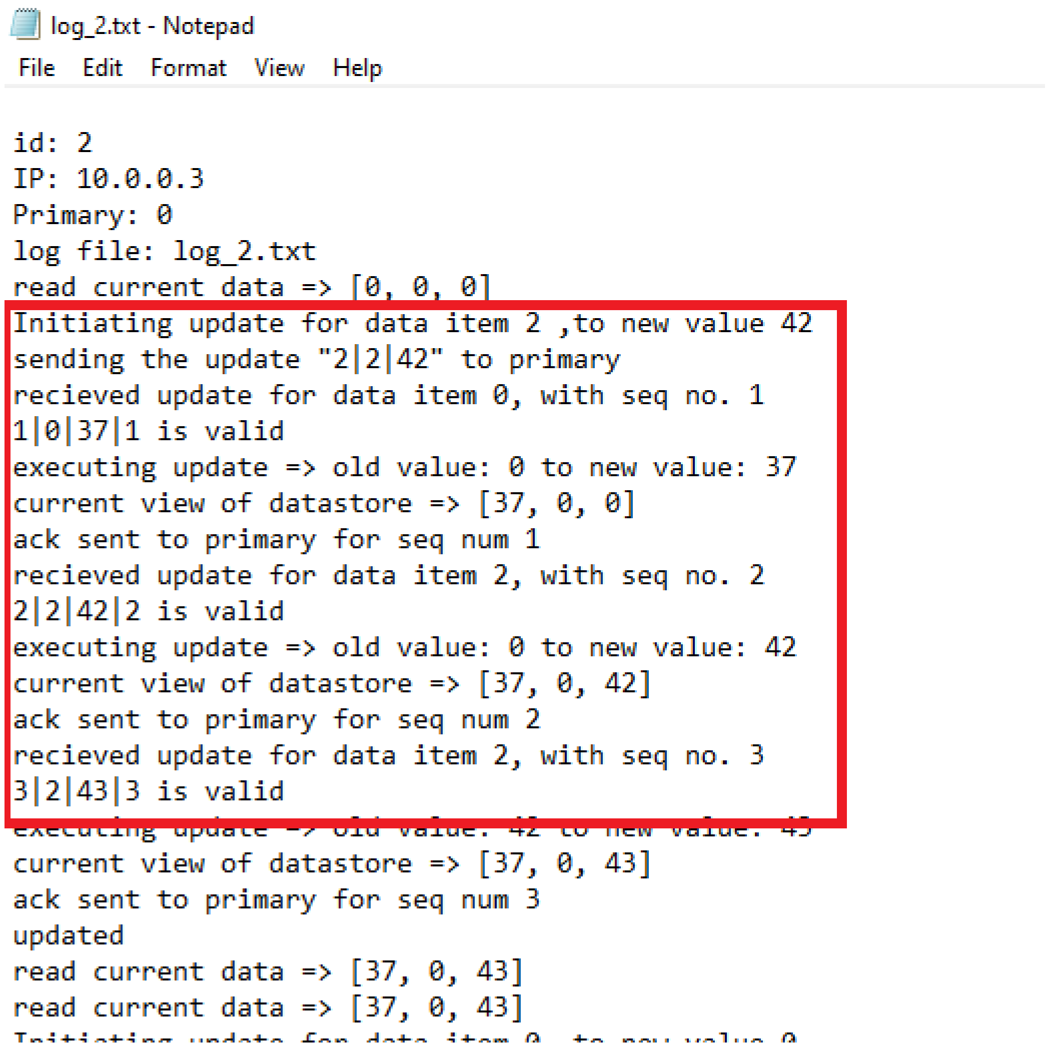
and block 2 is showing that how the update message is sent to all clients.

2. Replica servers are receiving updates in a different order because of network delays.



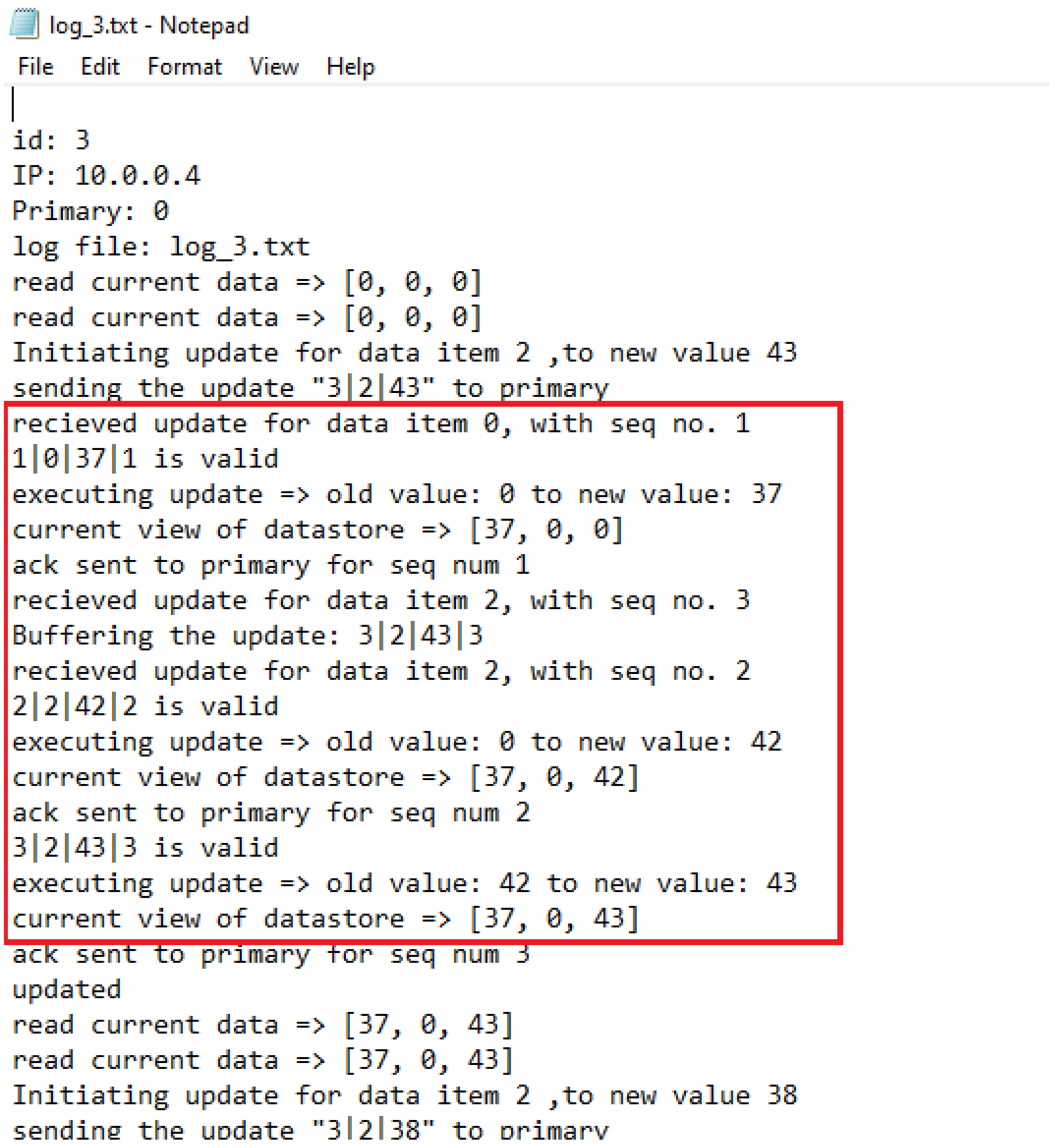
```
id: 1
IP: 10.0.0.2
Primary: 0
log file: log_1.txt
read current data => [0, 0, 0]
read current data => [0, 0, 0]
read current data => [0, 0, 0]
Initiating update for data item 0 ,to new value 37
sending the update "1|0|37" to primary
recieved update for data item 2, with seq no. 2
Buffering the update: 2|2|42|2
recieved update for data item 2, with seq no. 3
Buffering the update: 3|2|43|3
recieved update for data item 0, with seq no. 1
1|0|37|1 is valid
executing update => old value: 0 to new value: 37
current view of datastore => [37, 0, 0]
ack sent to primary for seq num 1
2|2|42|2 is valid
executing update => old value: 0 to new value: 42
current view of datastore => [37, 0, 42]
ack sent to primary for seq num 2
3|2|43|3 is valid
executing update => old value: 42 to new value: 43
current view of datastore => [37, 0, 43]
ack sent to primary for seq num 3
updated
read current data => [37, 0, 43]
Initiating update for data item 2 ,to new value 21
sending the update "1|2|21" to primary
recieved update for data item 2, with seq no. 4
3|2|38|4 is valid
executing update => old value: 43 to new value: 38
```

Figure 1:- log_1.txt



```
id: 2
IP: 10.0.0.3
Primary: 0
log file: log_2.txt
read current data => [0, 0, 0]
Initiating update for data item 2 ,to new value 42
sending the update "2|2|42" to primary
recieved update for data item 0, with seq no. 1
1|0|37|1 is valid
executing update => old value: 0 to new value: 37
current view of datastore => [37, 0, 0]
ack sent to primary for seq num 1
recieved update for data item 2, with seq no. 2
2|2|42|2 is valid
executing update => old value: 0 to new value: 42
current view of datastore => [37, 0, 42]
ack sent to primary for seq num 2
recieved update for data item 2, with seq no. 3
3|2|43|3 is valid
executing update => old value: 42 to new value: 43
current view of datastore => [37, 0, 43]
ack sent to primary for seq num 3
updated
read current data => [37, 0, 43]
read current data => [37, 0, 43]
Initiating update for data item 0 ,to new value 0
```

Figure 2:- log_2.txt



```
id: 3
IP: 10.0.0.4
Primary: 0
log file: log_3.txt
read current data => [0, 0, 0]
read current data => [0, 0, 0]
Initiating update for data item 2 ,to new value 43
sending the update "3|2|43" to primary
recieved update for data item 0, with seq no. 1
1|0|37|1 is valid
executing update => old value: 0 to new value: 37
current view of datastore => [37, 0, 0]
ack sent to primary for seq num 1
recieved update for data item 2, with seq no. 3
Buffering the update: 3|2|43|3
recieved update for data item 2, with seq no. 2
2|2|42|2 is valid
executing update => old value: 0 to new value: 42
current view of datastore => [37, 0, 42]
ack sent to primary for seq num 2
3|2|43|3 is valid
executing update => old value: 42 to new value: 43
current view of datastore => [37, 0, 43]
ack sent to primary for seq num 3
updated
read current data => [37, 0, 43]
read current data => [37, 0, 43]
Initiating update for data item 2 ,to new value 38
sending the update "3|2|38" to primary
```

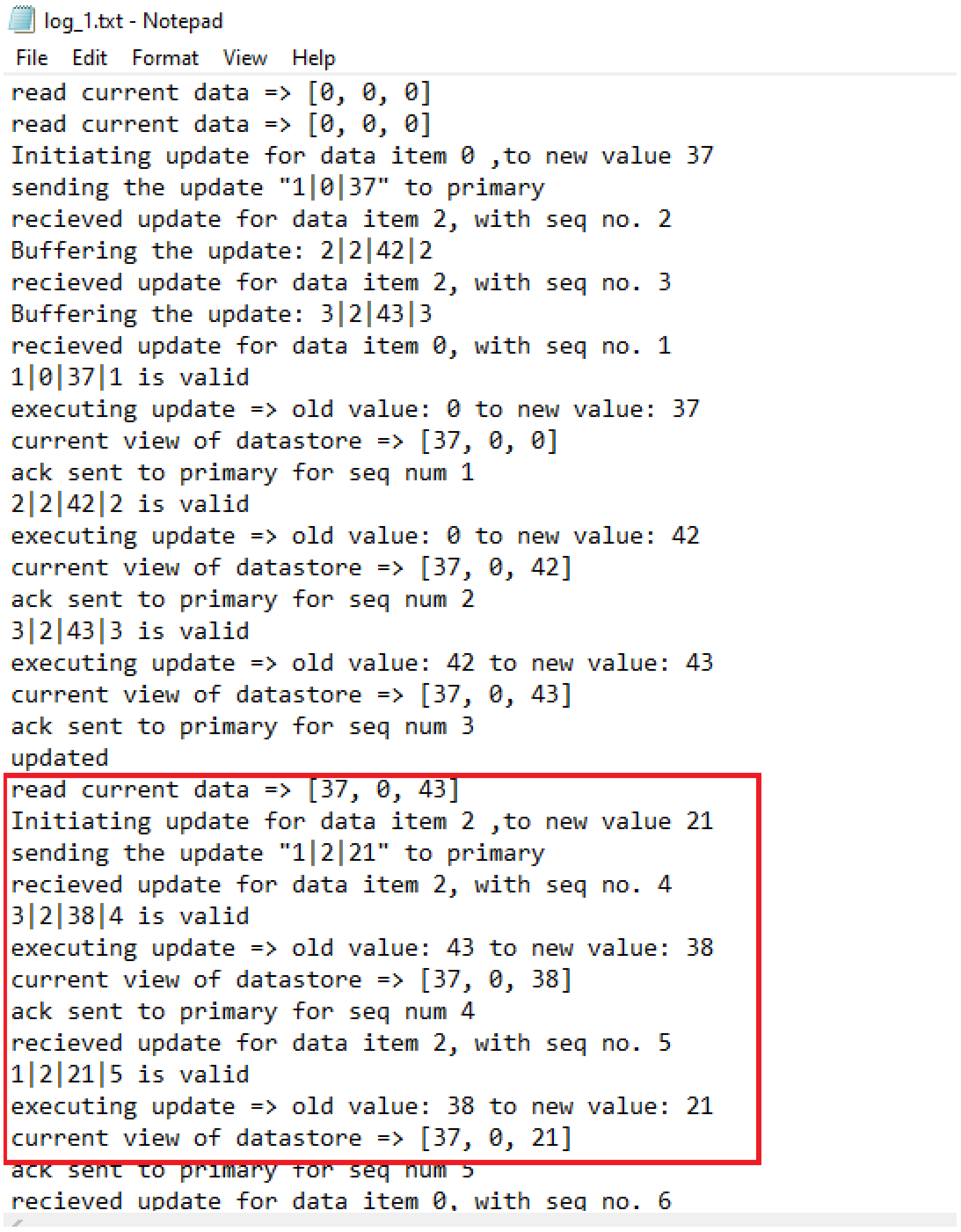
Figure 3:- log_3.txt

Replica servers have received the updated data in a different order as we can see in the figures.

In figure 1, Initiating update for data item 0 to new value 37. Due to network delay, data item 2 with sequence no. 2 has reached at server P1, which is out of order so we store it in buffer, then again server receives the data item 3 with sequence no. 3 which is also not in order then we store it in the buffer. Now server receives the data item 0 with sequence no. 1, it is a valid sequence, so we store the data and update it, and sent the ack to primary for sequence no. 1. Sequence 2 is already stored in the buffer so sequence 2 is also valid and update the current view of the datastore and the same for seq. no. 3 is valid and updates the data store.

In figure 2, we already received the data item valid sequence so no need to store it in a buffer, we can directly update the datastore. We can see, that we received the data item in different orders but store the data item in the data store in sequence order and out-of-order data items are buffered.

3. All of them update their local copies in the same order. and this order is same as the order in which the primary received the updates.

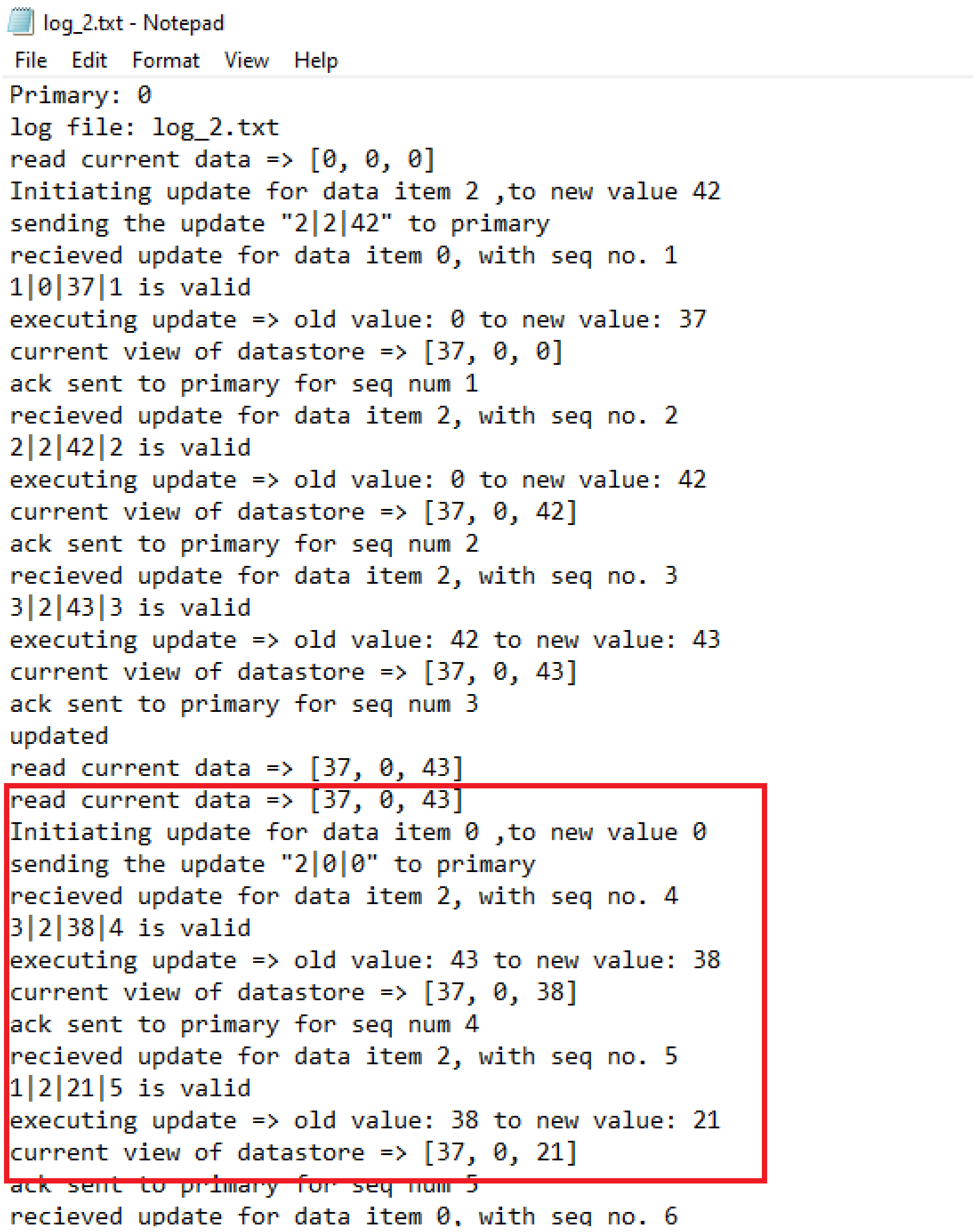


log_1.txt - Notepad

File Edit Format View Help

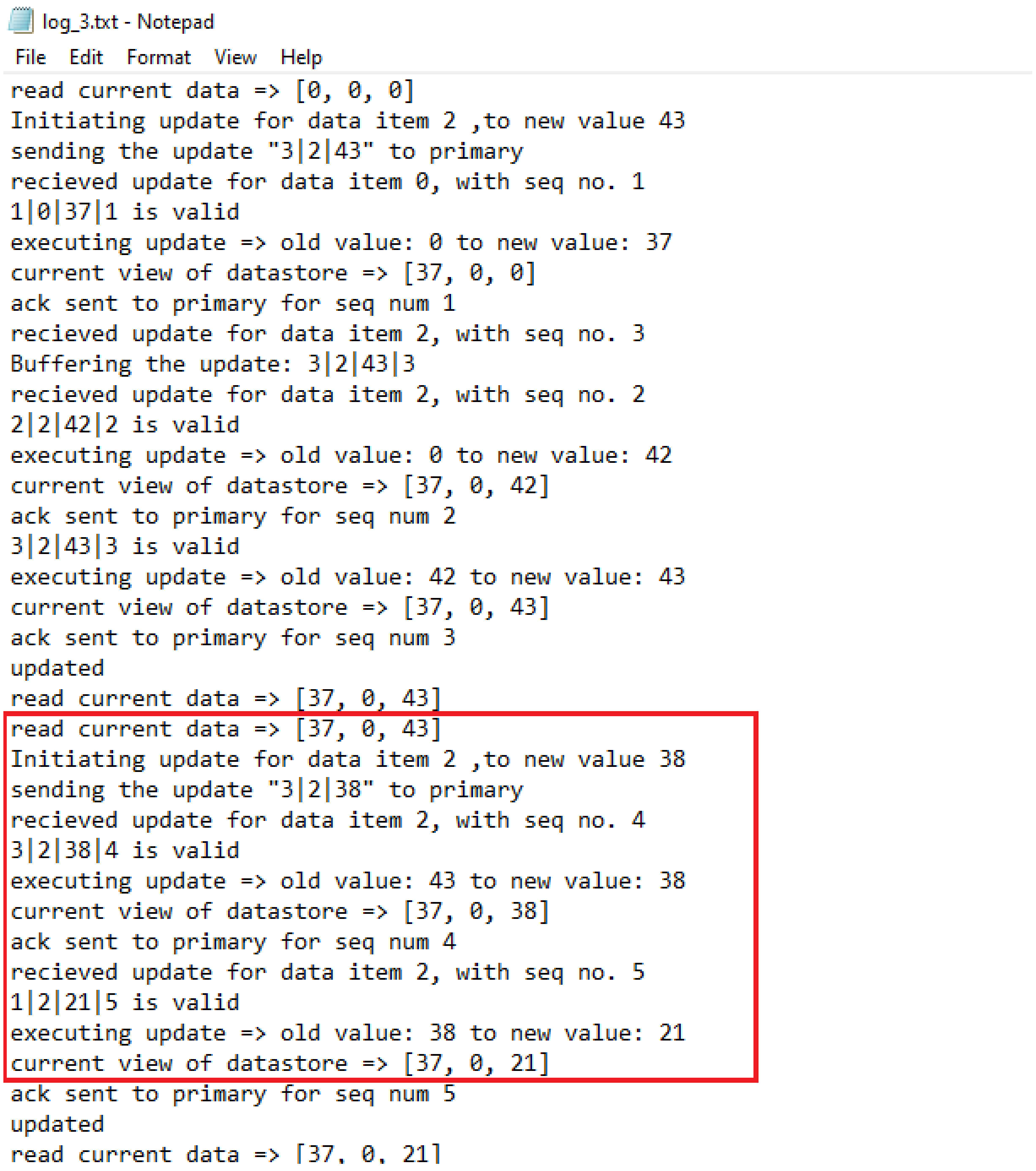
```
read current data => [0, 0, 0]
read current data => [0, 0, 0]
Initiating update for data item 0 ,to new value 37
sending the update "1|0|37" to primary
recieved update for data item 2, with seq no. 2
Buffering the update: 2|2|42|2
recieved update for data item 2, with seq no. 3
Buffering the update: 3|2|43|3
recieved update for data item 0, with seq no. 1
1|0|37|1 is valid
executing update => old value: 0 to new value: 37
current view of datastore => [37, 0, 0]
ack sent to primary for seq num 1
2|2|42|2 is valid
executing update => old value: 0 to new value: 42
current view of datastore => [37, 0, 42]
ack sent to primary for seq num 2
3|2|43|3 is valid
executing update => old value: 42 to new value: 43
current view of datastore => [37, 0, 43]
ack sent to primary for seq num 3
updated
read current data => [37, 0, 43]
Initiating update for data item 2 ,to new value 21
sending the update "1|2|21" to primary
recieved update for data item 2, with seq no. 4
3|2|38|4 is valid
executing update => old value: 43 to new value: 38
current view of datastore => [37, 0, 38]
ack sent to primary for seq num 4
recieved update for data item 2, with seq no. 5
1|2|21|5 is valid
executing update => old value: 38 to new value: 21
current view of datastore => [37, 0, 21]
ack sent to primary for seq num 5
recieved update for data item 0, with seq no. 6
```

Figure 4:- log_1.txt



```
Primary: 0
log file: log_2.txt
read current data => [0, 0, 0]
Initiating update for data item 2 ,to new value 42
sending the update "2|2|42" to primary
recieved update for data item 0, with seq no. 1
1|0|37|1 is valid
executing update => old value: 0 to new value: 37
current view of datastore => [37, 0, 0]
ack sent to primary for seq num 1
recieved update for data item 2, with seq no. 2
2|2|42|2 is valid
executing update => old value: 0 to new value: 42
current view of datastore => [37, 0, 42]
ack sent to primary for seq num 2
recieved update for data item 2, with seq no. 3
3|2|43|3 is valid
executing update => old value: 42 to new value: 43
current view of datastore => [37, 0, 43]
ack sent to primary for seq num 3
updated
read current data => [37, 0, 43]
read current data => [37, 0, 43]
Initiating update for data item 0 ,to new value 0
sending the update "2|0|0" to primary
recieved update for data item 2, with seq no. 4
3|2|38|4 is valid
executing update => old value: 43 to new value: 38
current view of datastore => [37, 0, 38]
ack sent to primary for seq num 4
recieved update for data item 2, with seq no. 5
1|2|21|5 is valid
executing update => old value: 38 to new value: 21
current view of datastore => [37, 0, 21]
ack sent to primary for seq num 5
recieved update for data item 0, with seq no. 6
```

Figure 5:- log_2.txt

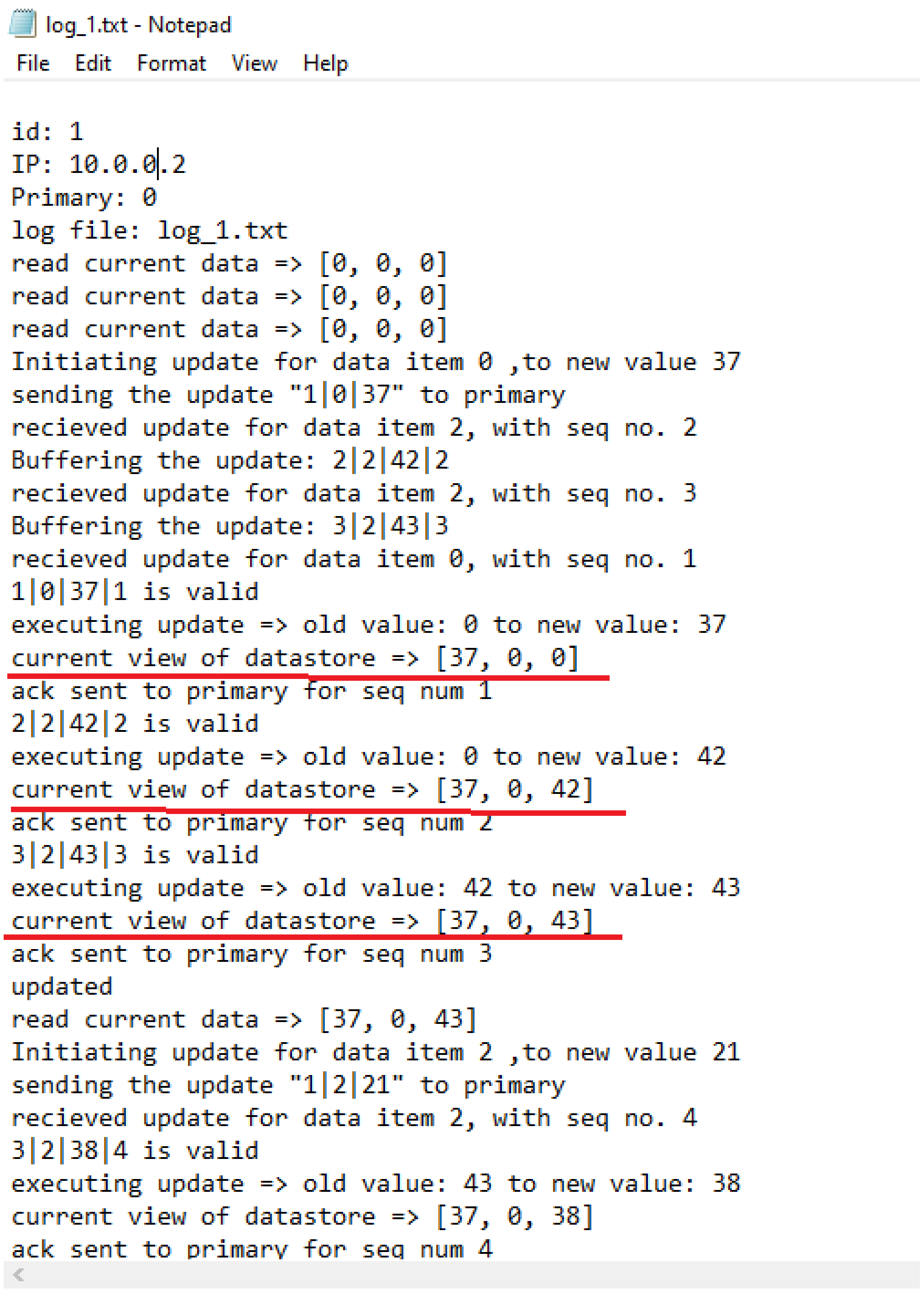


```
read current data => [0, 0, 0]
Initiating update for data item 2 ,to new value 43
sending the update "3|2|43" to primary
recieved update for data item 0, with seq no. 1
1|0|37|1 is valid
executing update => old value: 0 to new value: 37
current view of datastore => [37, 0, 0]
ack sent to primary for seq num 1
recieved update for data item 2, with seq no. 3
Buffering the update: 3|2|43|3
recieved update for data item 2, with seq no. 2
2|2|42|2 is valid
executing update => old value: 0 to new value: 42
current view of datastore => [37, 0, 42]
ack sent to primary for seq num 2
3|2|43|3 is valid
executing update => old value: 42 to new value: 43
current view of datastore => [37, 0, 43]
ack sent to primary for seq num 3
updated
read current data => [37, 0, 43]
read current data => [37, 0, 43]
Initiating update for data item 2 ,to new value 38
sending the update "3|2|38" to primary
recieved update for data item 2, with seq no. 4
3|2|38|4 is valid
executing update => old value: 43 to new value: 38
current view of datastore => [37, 0, 38]
ack sent to primary for seq num 4
recieved update for data item 2, with seq no. 5
1|2|21|5 is valid
executing update => old value: 38 to new value: 21
current view of datastore => [37, 0, 21]
ack sent to primary for seq num 5
updated
read current data => [37, 0, 21]
```

Figure 6:- log_3.txt

We can see in the figures(blocks) 4,5, and 6, update their local copies in the same order as [37, 0, 43] then [37, 0, 38] and then [38, 0, 21].

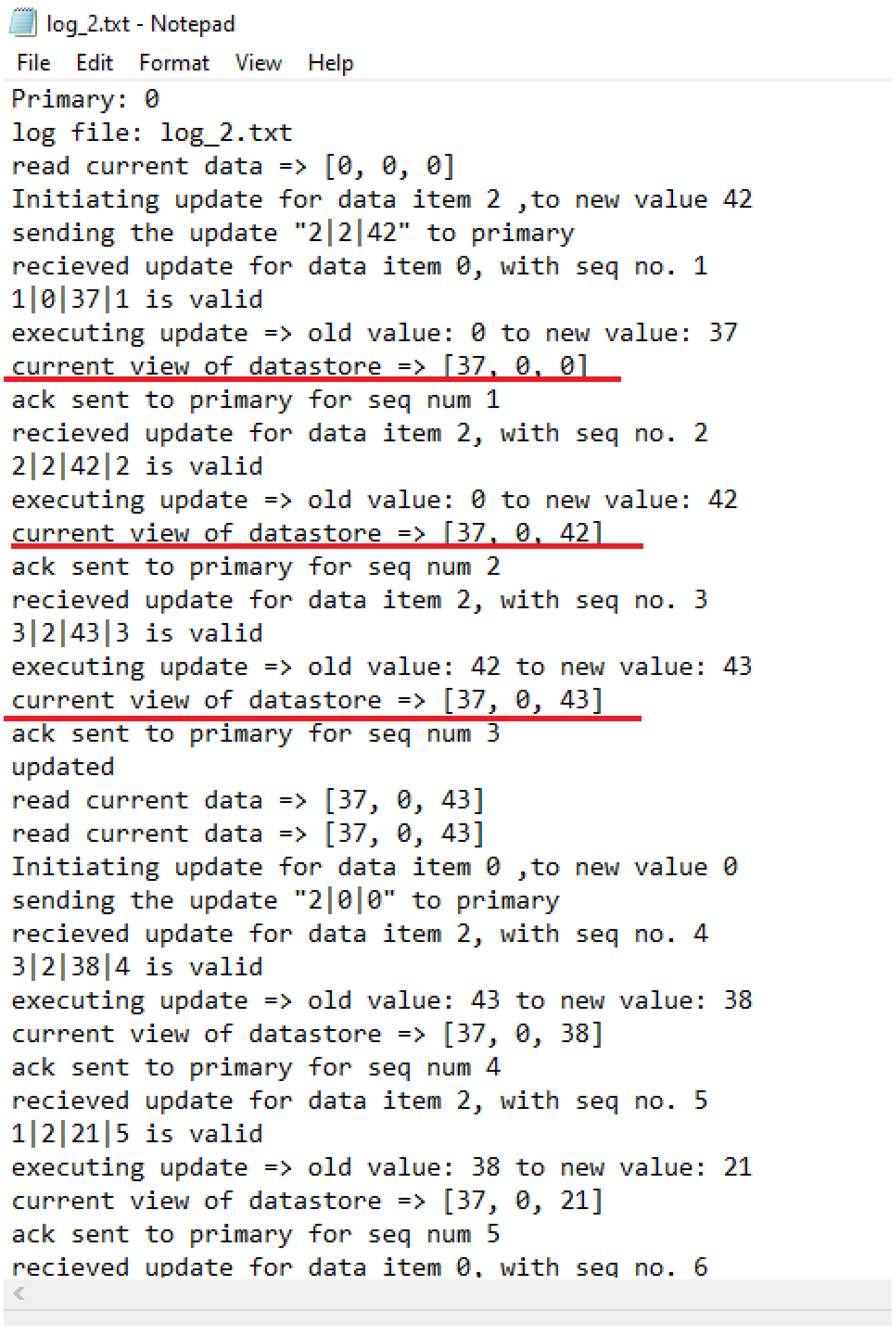
4. at every instant, the local copies at each server have the same values.


log_1.txt - Notepad

File Edit Format View Help

```
id: 1
IP: 10.0.0.2
Primary: 0
log file: log_1.txt
read current data => [0, 0, 0]
read current data => [0, 0, 0]
read current data => [0, 0, 0]
Initiating update for data item 0 ,to new value 37
sending the update "1|0|37" to primary
recieved update for data item 2, with seq no. 2
Buffering the update: 2|2|42|2
recieved update for data item 2, with seq no. 3
Buffering the update: 3|2|43|3
recieved update for data item 0, with seq no. 1
1|0|37|1 is valid
executing update => old value: 0 to new value: 37
current view of datastore => [37, 0, 0]
ack sent to primary for seq num 1
2|2|42|2 is valid
executing update => old value: 0 to new value: 42
current view of datastore => [37, 0, 42]
ack sent to primary for seq num 2
3|2|43|3 is valid
executing update => old value: 42 to new value: 43
current view of datastore => [37, 0, 43]
ack sent to primary for seq num 3
updated
read current data => [37, 0, 43]
Initiating update for data item 2 ,to new value 21
sending the update "1|2|21" to primary
recieved update for data item 2, with seq no. 4
3|2|38|4 is valid
executing update => old value: 43 to new value: 38
current view of datastore => [37, 0, 38]
ack sent to primary for seq num 4
```

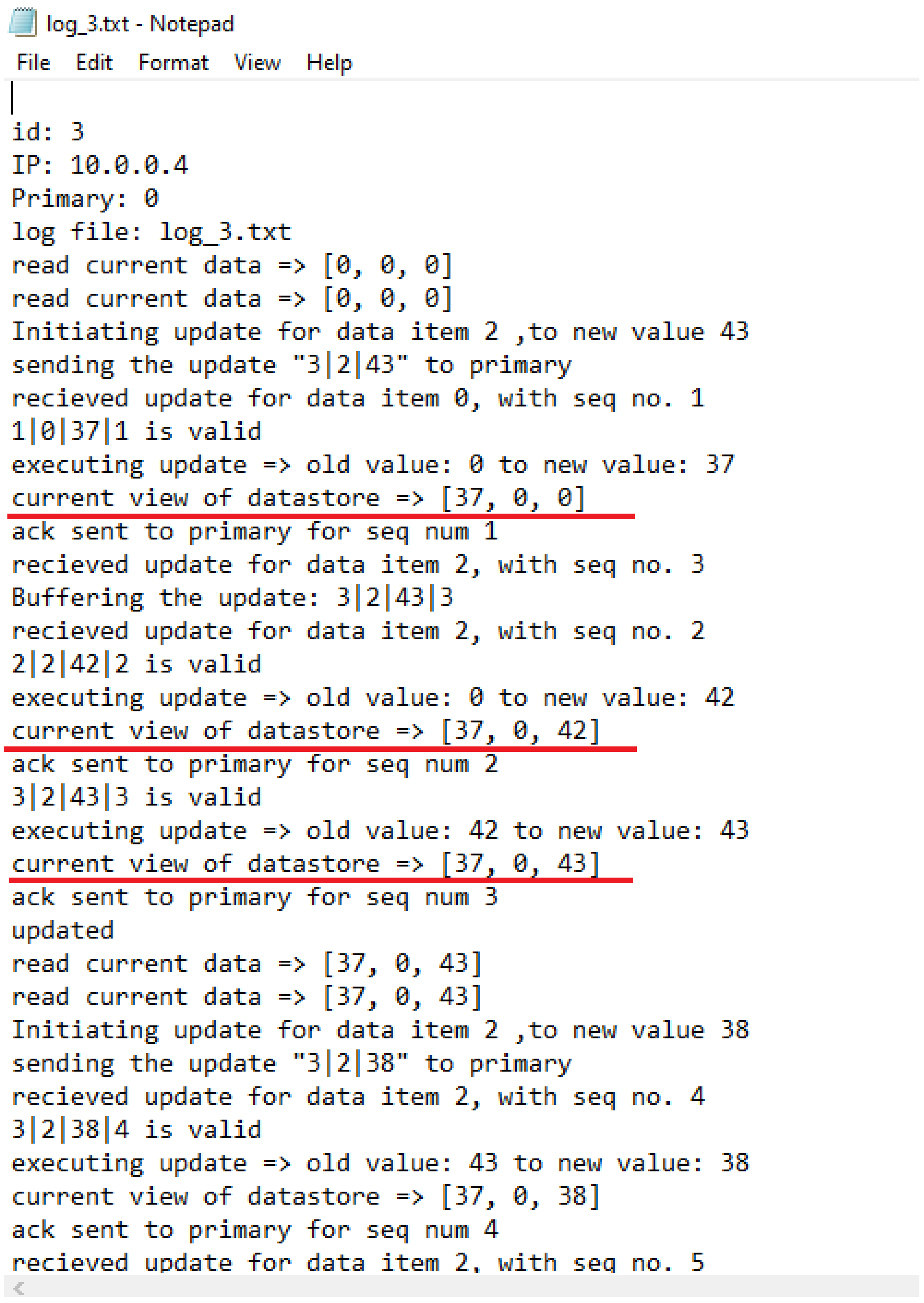
Figure :- log_1.txt


log_2.txt - Notepad

File Edit Format View Help

```
Primary: 0
log file: log_2.txt
read current data => [0, 0, 0]
Initiating update for data item 2 ,to new value 42
sending the update "2|2|42" to primary
recieved update for data item 0, with seq no. 1
1|0|37|1 is valid
executing update => old value: 0 to new value: 37
current view of datastore => [37, 0, 0]
ack sent to primary for seq num 1
recieved update for data item 2, with seq no. 2
2|2|42|2 is valid
executing update => old value: 0 to new value: 42
current view of datastore => [37, 0, 42]
ack sent to primary for seq num 2
recieved update for data item 2, with seq no. 3
3|2|43|3 is valid
executing update => old value: 42 to new value: 43
current view of datastore => [37, 0, 43]
ack sent to primary for seq num 3
updated
read current data => [37, 0, 43]
read current data => [37, 0, 43]
Initiating update for data item 0 ,to new value 0
sending the update "2|0|0" to primary
recieved update for data item 2, with seq no. 4
3|2|38|4 is valid
executing update => old value: 43 to new value: 38
current view of datastore => [37, 0, 38]
ack sent to primary for seq num 4
recieved update for data item 2, with seq no. 5
1|2|21|5 is valid
executing update => old value: 38 to new value: 21
current view of datastore => [37, 0, 21]
ack sent to primary for seq num 5
recieved update for data item 0, with seq no. 6
```

figure:- log_2.txt



```
id: 3
IP: 10.0.0.4
Primary: 0
log file: log_3.txt
read current data => [0, 0, 0]
read current data => [0, 0, 0]
Initiating update for data item 2 ,to new value 43
sending the update "3|2|43" to primary
recieved update for data item 0, with seq no. 1
1|0|37|1 is valid
executing update => old value: 0 to new value: 37
current view of datastore => [37, 0, 0]
ack sent to primary for seq num 1
recieved update for data item 2, with seq no. 3
Buffering the update: 3|2|43|3
recieved update for data item 2, with seq no. 2
2|2|42|2 is valid
executing update => old value: 0 to new value: 42
current view of datastore => [37, 0, 42]
ack sent to primary for seq num 2
3|2|43|3 is valid
executing update => old value: 42 to new value: 43
current view of datastore => [37, 0, 43]
ack sent to primary for seq num 3
updated
read current data => [37, 0, 43]
read current data => [37, 0, 43]
Initiating update for data item 2 ,to new value 38
sending the update "3|2|38" to primary
recieved update for data item 2, with seq no. 4
3|2|38|4 is valid
executing update => old value: 43 to new value: 38
current view of datastore => [37, 0, 38]
ack sent to primary for seq num 4
recieved update for data item 2, with seq no. 5
```

Figure :- log_3.txt

We can see, in above figure of log_1, log_2, log_3.txt have same value at every instances.

All the conditions of sequential consistency have been satisfied, hence the system is sequentially consistent.