Testing Scenarios And Results

Getting Started

Start Central Server and affiliated servers:

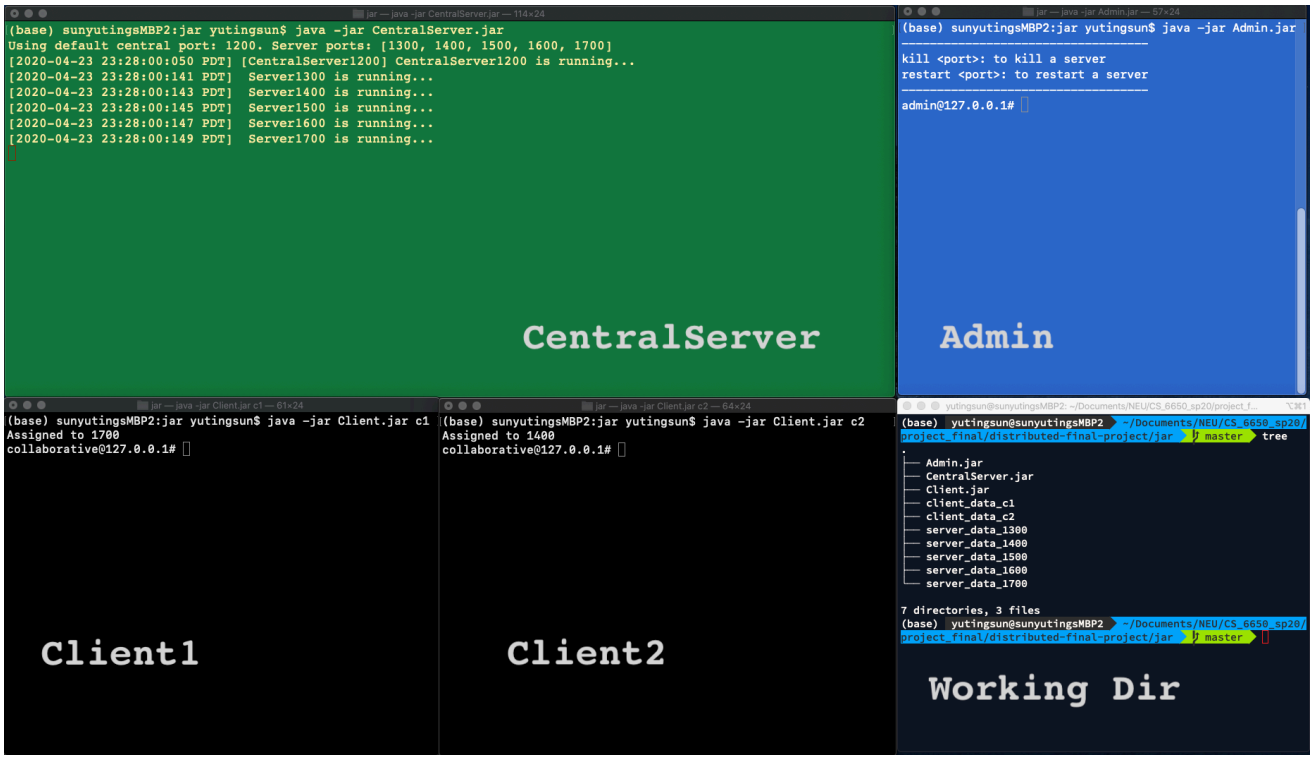
```
# (default central port: 1200, default server ports: 1300, 1400, 1500,
1600, 1700):
$java -jar CentralServer.jar

#Start Clients c1 and c2:
$java -jar Client.jar c1
$java -jar Client.jar c2

#Start Admin:
$java -jar Admin.jar
```

- We can see 5 servers running.
- After starting central server: corresponding server directories are created

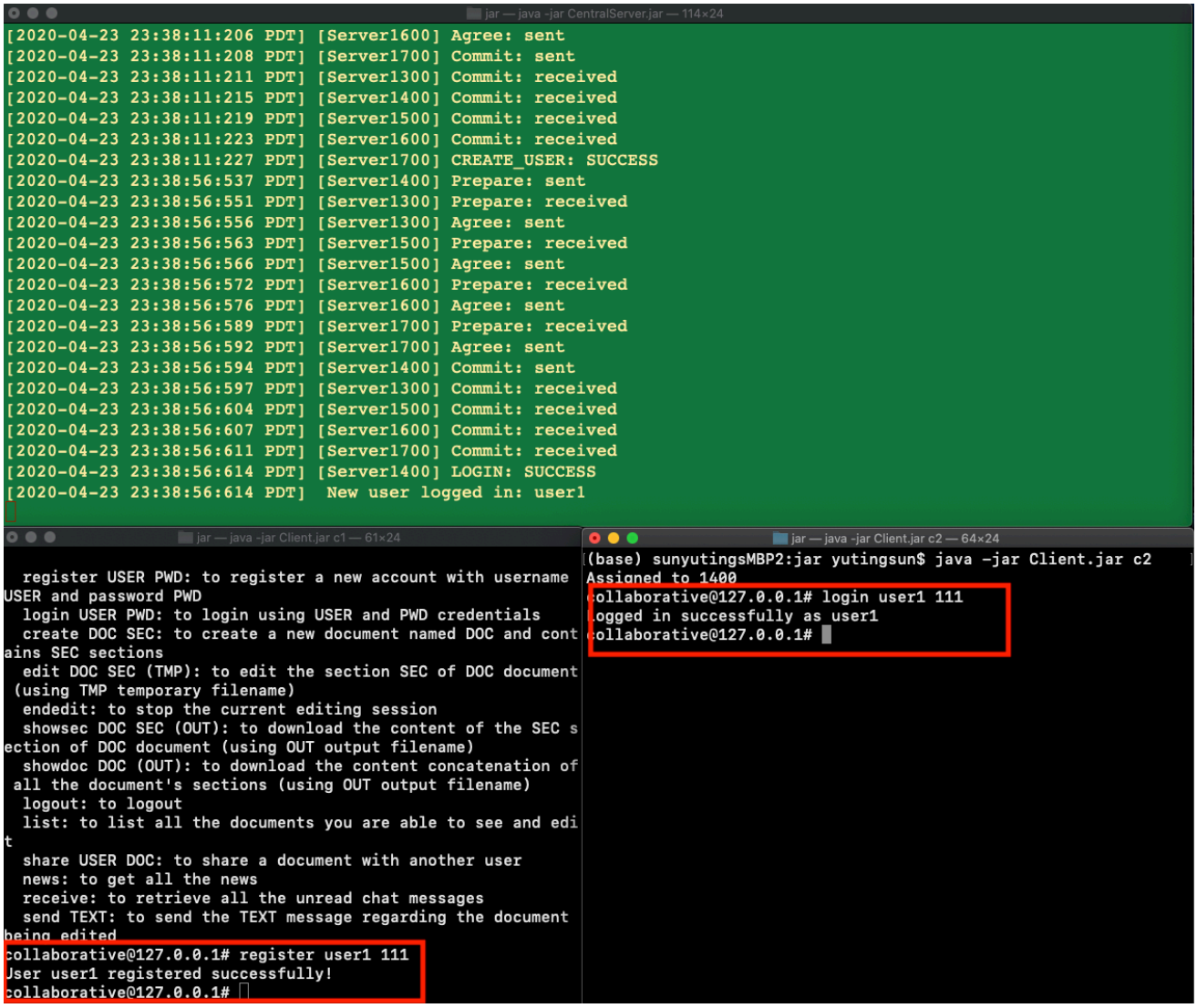
  After starting client <clientname>: corresponding client directory is created



Test1 User Management and File Editing

1. Create a new user `user1` in client1

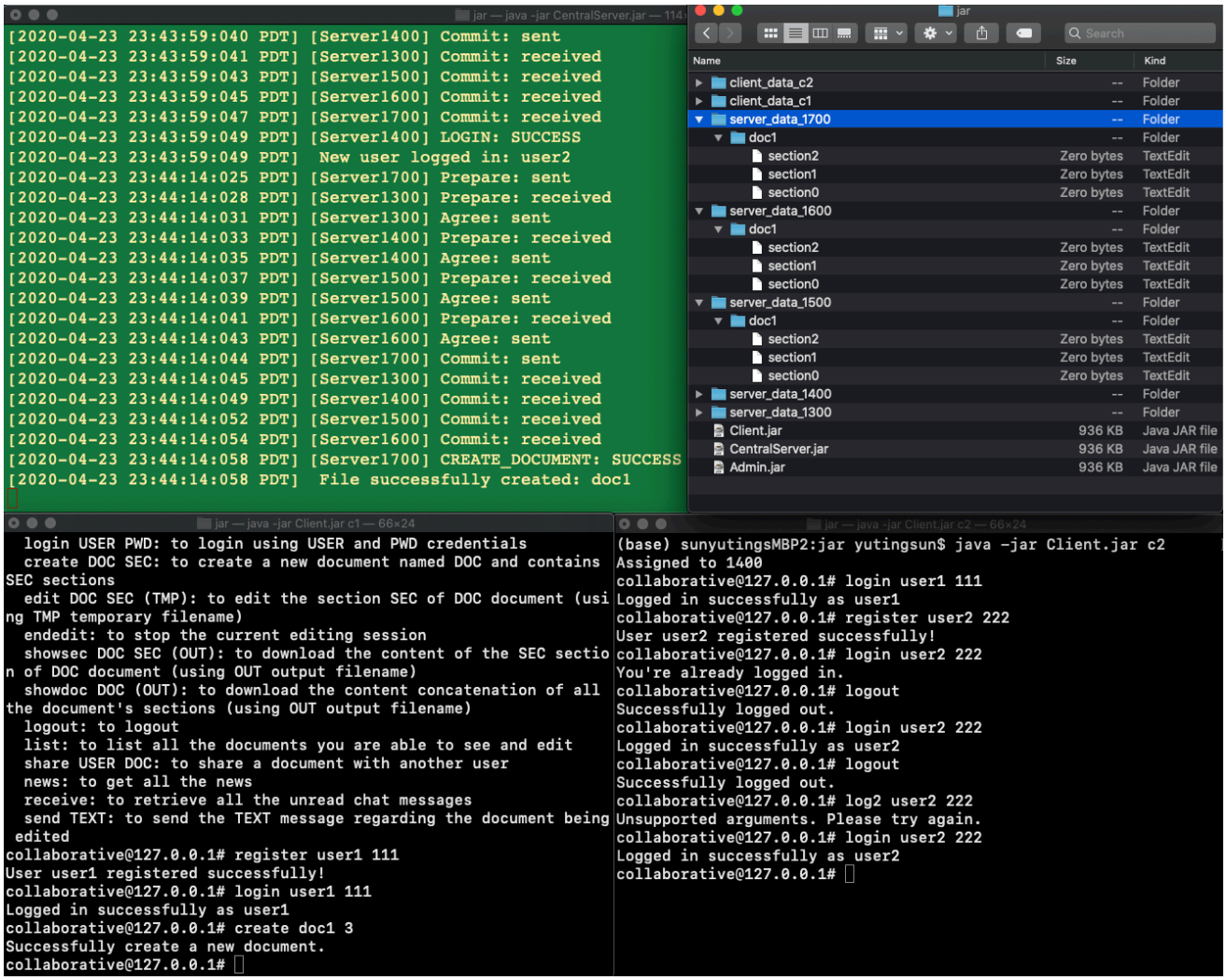
- We can see the 2pc transaction steps in central server
- Check consistency by loggin in `user1` at client2 --> success!



2. Create doc

Login user1 in client1, Create a new document with 3 sections with command `create doc1 3`

- we can see the 2PC log in central server
- now each server directory has a new directory `doc1` with 3 section files.



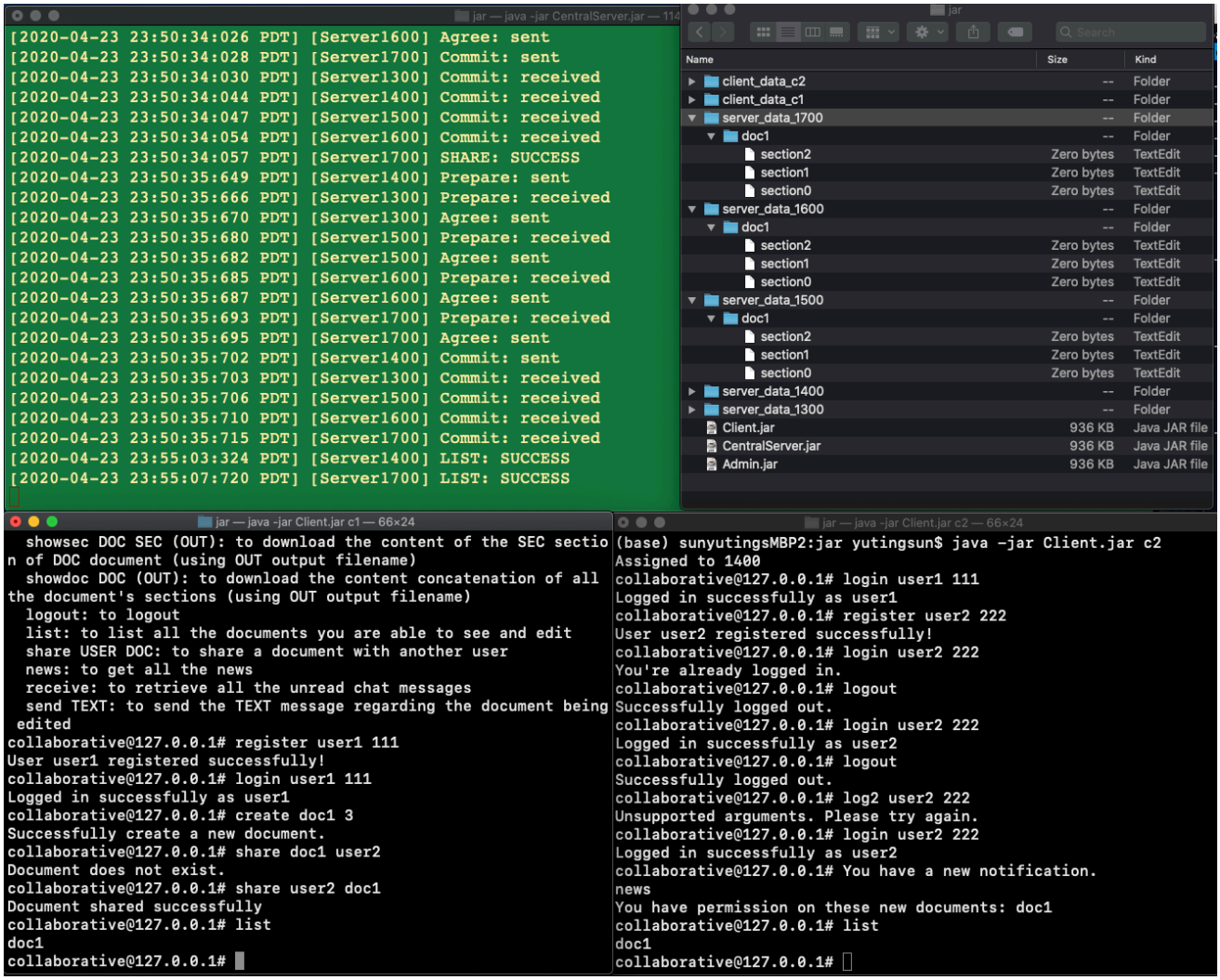
3. Share doc

Share the doc1 to user2 by command: `share user2 doc1`

- We can see share success in central server log
- User2 will receive a notification of new file sharing. We can also see the notification 2PC log in central server
- Then we type `news` command in client2. we can get the new notification:

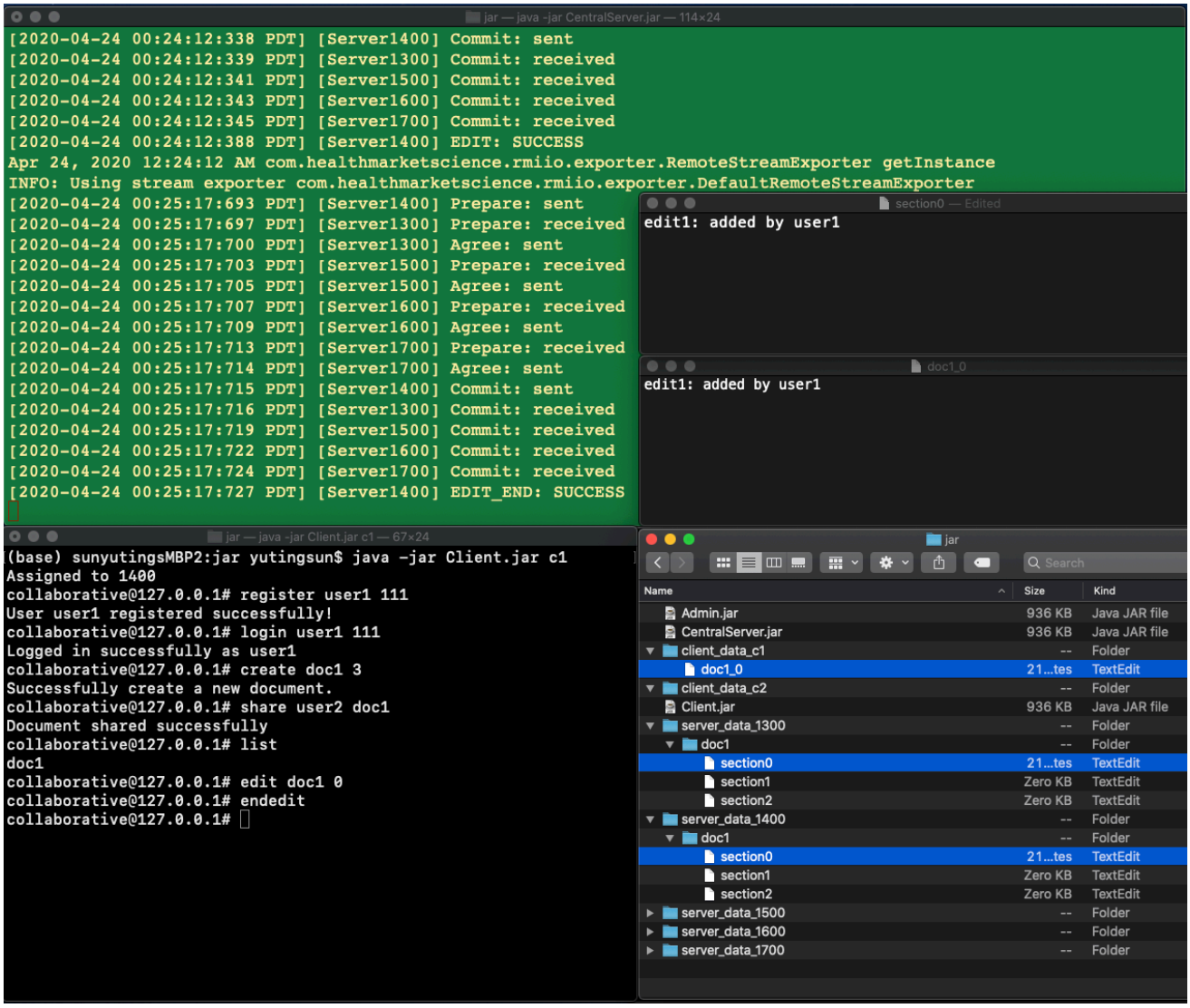
```
You have permission on these new documents: doc1
```

- we can check the sharing result by `list` command: both clients showed `doc1`, indicating that they have access to edit doc1.



4. Edit doc by 1 user

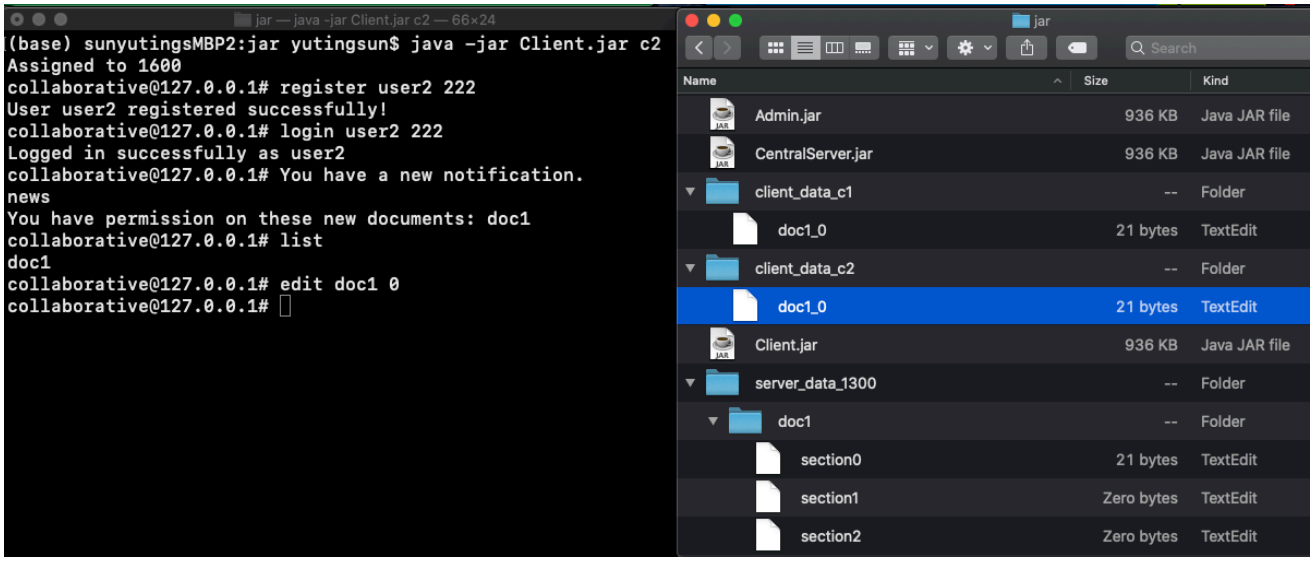
- Client can start editing section 0 of doc1 by command `edit doc1 0`. This would create a new file in the client directory `client_data_c1`.
- Add text in doc1_0 in a text editor.
- When finish editing, user1 enters `endedit`:
    - server 1400 starts 2PC, when commited, we can see the text in file `doc1/section0` in every server directory. We can also notice the file size of `section0` is changed.



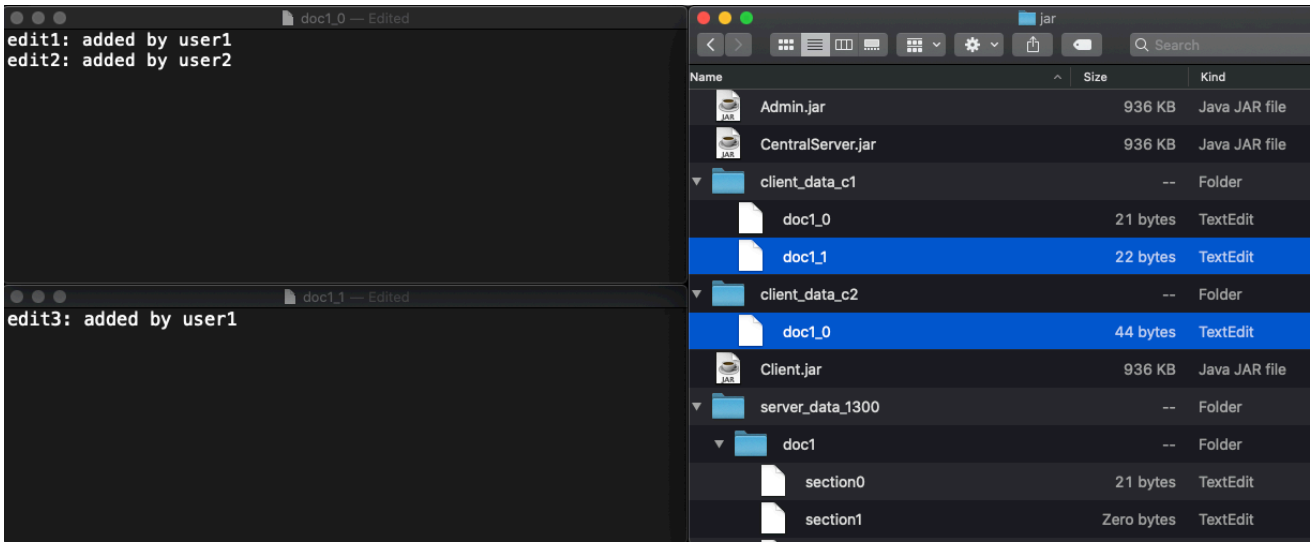
5. Edit the same doc by 2 users

- user2 starts editing `doc1` section 0 by command `edit doc1 0`

  this downloads the latest tmp section file `doc1_0` from server dir to client2 dir



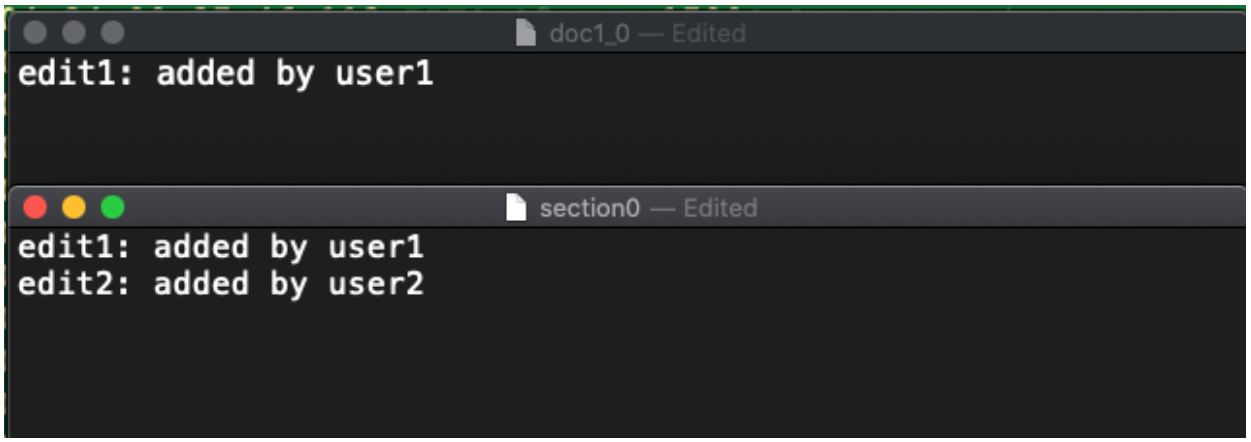
- user2 can edit doc1_0 in a text editor
- In the mean time, user1 can edit doc1 section1 (but not section 0)



- Each user can submitt their edit by command `endedit`. Then the server would start 2PC to sync the latest section file among all servers.

  We can see the section files in server directory are updated.

- At this time the temp `doc1_0` file in client1 folder is not synced.



When user1 start to edit doc1 section0 again, the `doc1_0` file will be updated.

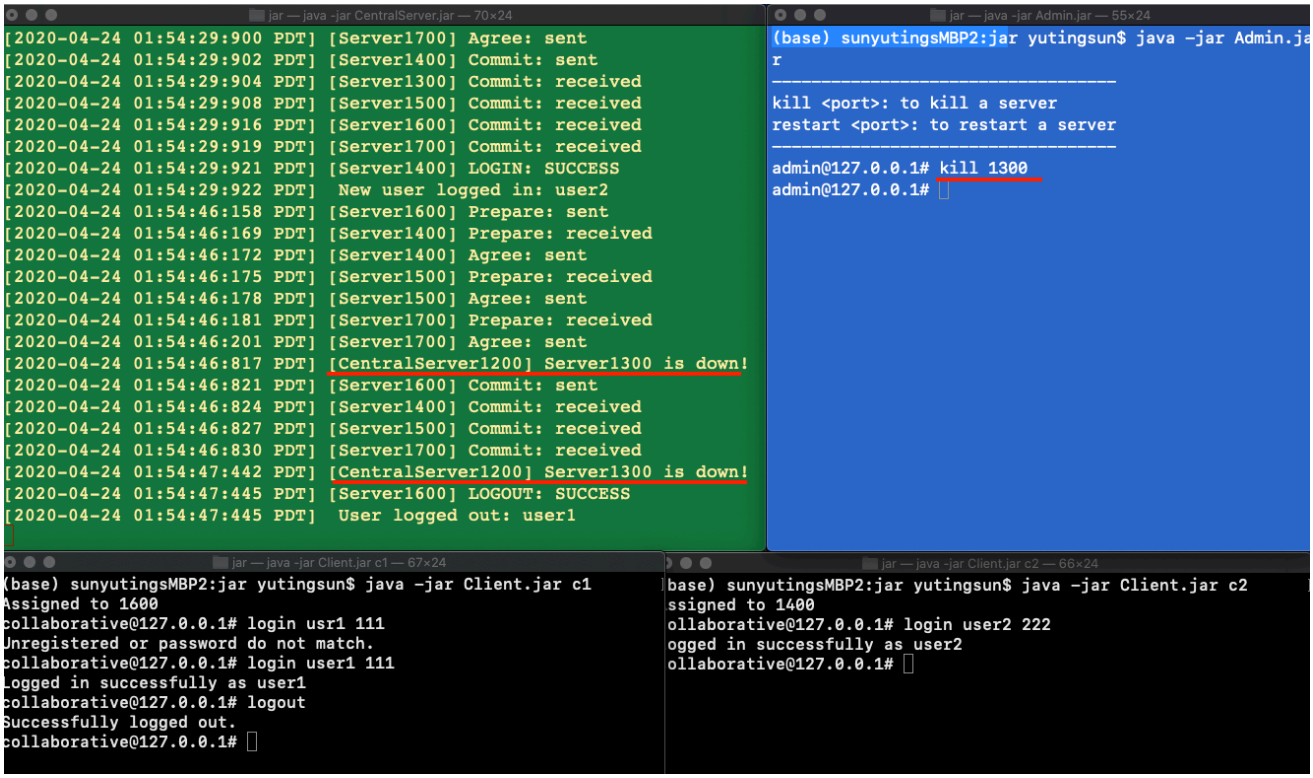
Test2. 2-Phase Commit with Server Failure

- We can kill a server 1300 with command `kill 1300` in Admin.

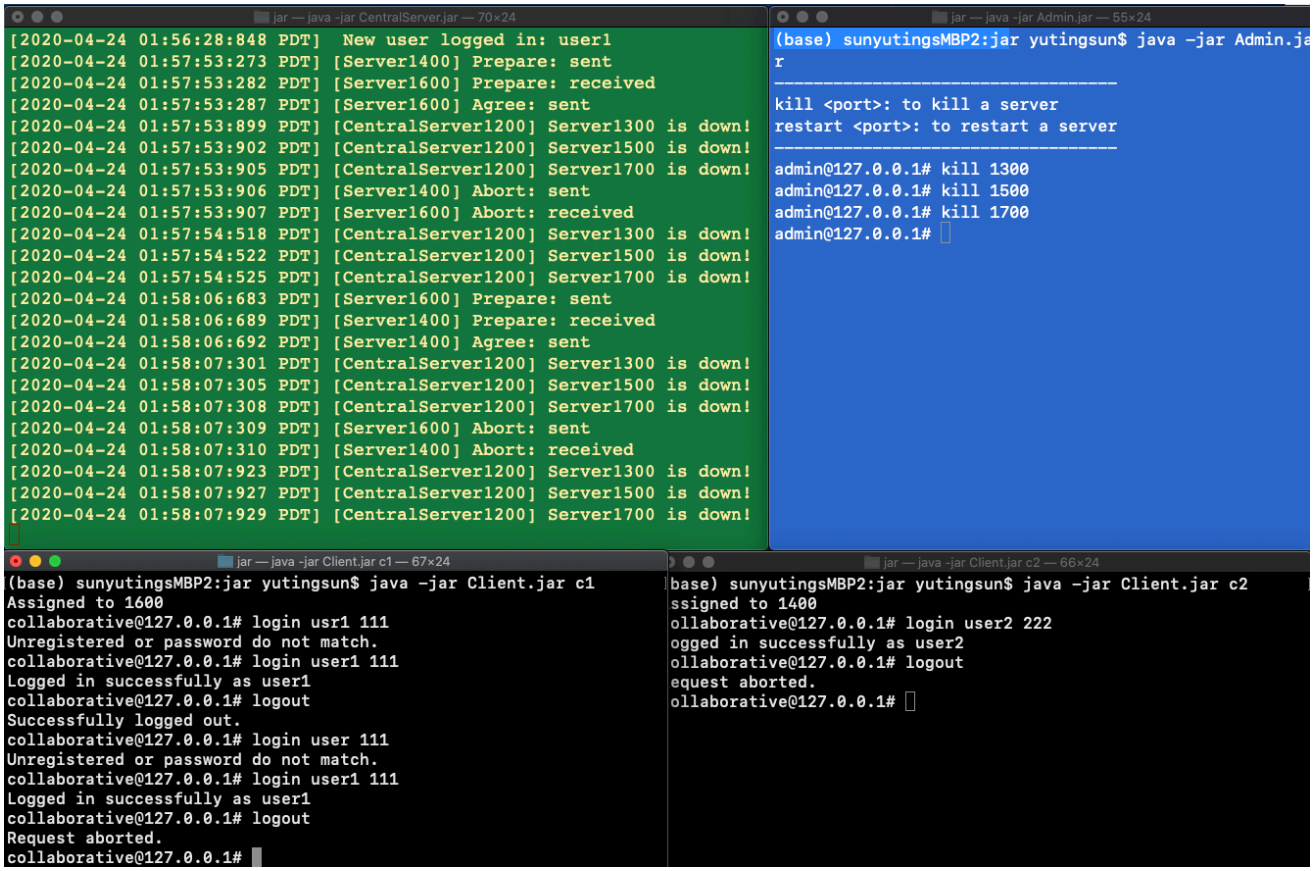
  Central Server and Client would not detect the server failure at this time.

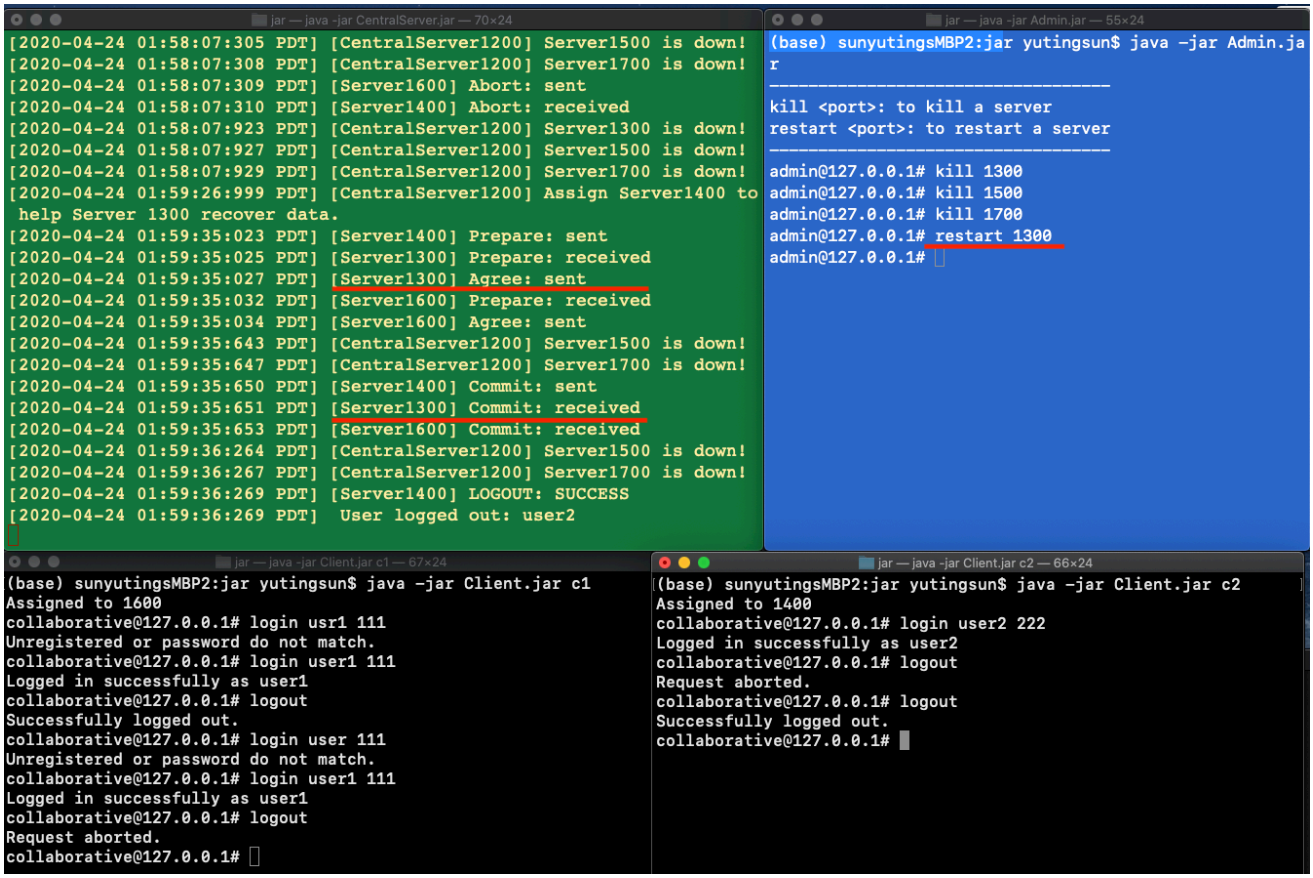
- When user1 login/logout, this needs 2PC to synce among all servers. The coordinator would detect the failure server in this process and report to central server.

  Since the majority servers are still alive (4/5), the transaction is commited.



- When 3 servers alive --> COMMIT
- When only 2 servers alive -> ABORT
- When server is restarted , we can see it's alive from central server log

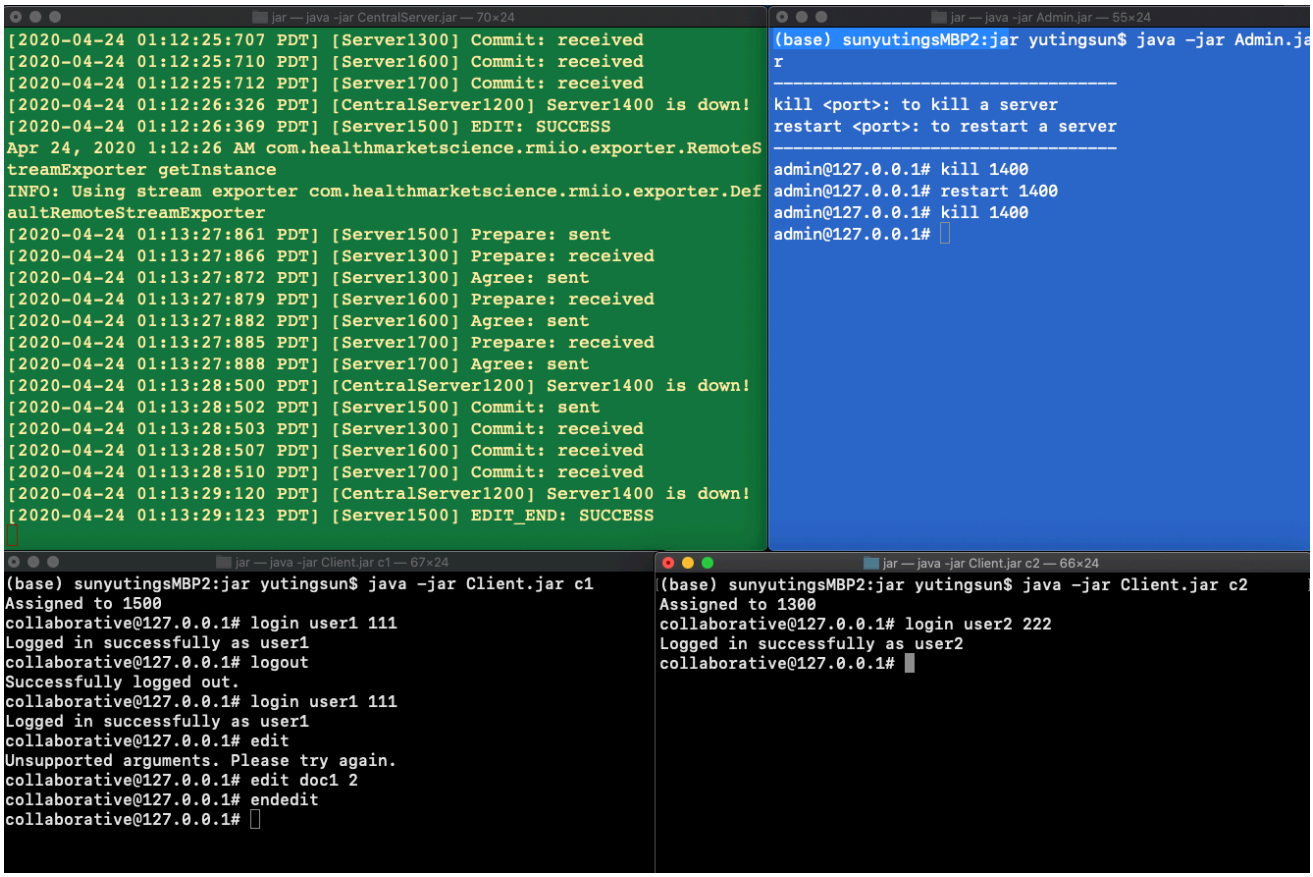

jar — java -jar CentralServer.jar — 70×24
[2020-04-24 01:56:28:848 PDT] New user logged in: user1
[2020-04-24 01:57:53:273 PDT] [Server1400] Prepare: sent
[2020-04-24 01:57:53:282 PDT] [Server1600] Prepare: received
[2020-04-24 01:57:53:287 PDT] [Server1600] Agree: sent
[2020-04-24 01:57:53:899 PDT] [CentralServer1200] Server1300 is down!
[2020-04-24 01:57:53:902 PDT] [CentralServer1200] Server1500 is down!
[2020-04-24 01:57:53:905 PDT] [CentralServer1200] Server1700 is down!
[2020-04-24 01:57:53:906 PDT] [Server1400] Abort: sent
[2020-04-24 01:57:53:907 PDT] [Server1600] Abort: received
[2020-04-24 01:57:54:518 PDT] [CentralServer1200] Server1300 is down!
[2020-04-24 01:57:54:522 PDT] [CentralServer1200] Server1500 is down!
[2020-04-24 01:57:54:525 PDT] [CentralServer1200] Server1700 is down!
[2020-04-24 01:58:06:683 PDT] [Server1600] Prepare: sent
[2020-04-24 01:58:06:689 PDT] [Server1400] Prepare: received
[2020-04-24 01:58:06:692 PDT] [Server1400] Agree: sent
[2020-04-24 01:58:07:301 PDT] [CentralServer1200] Server1300 is down!
[2020-04-24 01:58:07:305 PDT] [CentralServer1200] Server1500 is down!
[2020-04-24 01:58:07:308 PDT] [CentralServer1200] Server1700 is down!
[2020-04-24 01:58:07:309 PDT] [Server1600] Abort: sent
[2020-04-24 01:58:07:310 PDT] [Server1400] Abort: received
[2020-04-24 01:58:07:923 PDT] [CentralServer1200] Server1300 is down!
[2020-04-24 01:58:07:927 PDT] [CentralServer1200] Server1500 is down!
[2020-04-24 01:58:07:929 PDT] [CentralServer1200] Server1700 is down!
jar — java -jar Admin.jar — 55×24
(base) sunyutingsMBP2:jar yutingsun$ java -jar Admin.jar
kill <port>: to kill a server
restart <port>: to restart a server
admin@127.0.0.1# kill 1300
admin@127.0.0.1# kill 1500
admin@127.0.0.1# kill 1700
admin@127.0.0.1#
jar — java -jar Client.jar c1 — 67×24
(base) sunyutingsMBP2:jar yutingsun$ java -jar Client.jar c1
Assigned to 1600
collaborative@127.0.0.1# login usr1 111
Unregistered or password do not match.
collaborative@127.0.0.1# login user1 111
Logged in successfully as user1
collaborative@127.0.0.1# logout
Successfully logged out.
collaborative@127.0.0.1# login user 111
Unregistered or password do not match.
collaborative@127.0.0.1# login user1 111
Logged in successfully as user1
collaborative@127.0.0.1# logout
Request aborted.
collaborative@127.0.0.1#
jar — java -jar Client.jar c2 — 66×24
base) sunyutingsMBP2:jar yutingsun$ java -jar Client.jar c2
ssigned to 1400
ollaborative@127.0.0.1# login user2 222
ogged in successfully as user2
ollaborative@127.0.0.1# logout
equest aborted.
ollaborative@127.0.0.1#

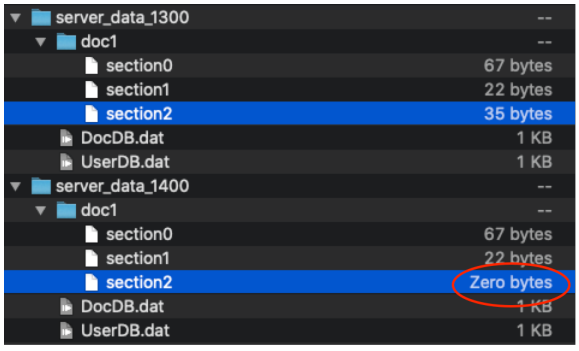

jar — java -jar CentralServer.jar — 70×24
[2020-04-24 01:58:07:305 PDT] [CentralServer1200] Server1500 is down!
[2020-04-24 01:58:07:308 PDT] [CentralServer1200] Server1700 is down!
[2020-04-24 01:58:07:309 PDT] [Server1600] Abort: sent
[2020-04-24 01:58:07:310 PDT] [Server1400] Abort: received
[2020-04-24 01:58:07:923 PDT] [CentralServer1200] Server1300 is down!
[2020-04-24 01:58:07:927 PDT] [CentralServer1200] Server1500 is down!
[2020-04-24 01:58:07:929 PDT] [CentralServer1200] Server1700 is down!
[2020-04-24 01:59:26:999 PDT] [CentralServer1200] Assign Server1400 to help Server 1300 recover data.
[2020-04-24 01:59:35:023 PDT] [Server1400] Prepare: sent
[2020-04-24 01:59:35:025 PDT] [Server1300] Prepare: received
[2020-04-24 01:59:35:027 PDT] [Server1300] Agree: sent
[2020-04-24 01:59:35:032 PDT] [Server1600] Prepare: received
[2020-04-24 01:59:35:034 PDT] [Server1600] Agree: sent
[2020-04-24 01:59:35:643 PDT] [CentralServer1200] Server1500 is down!
[2020-04-24 01:59:35:647 PDT] [CentralServer1200] Server1700 is down!
[2020-04-24 01:59:35:650 PDT] [Server1400] Commit: sent
[2020-04-24 01:59:35:651 PDT] [Server1300] Commit: received
[2020-04-24 01:59:35:653 PDT] [Server1600] Commit: received
[2020-04-24 01:59:36:264 PDT] [CentralServer1200] Server1500 is down!
[2020-04-24 01:59:36:267 PDT] [CentralServer1200] Server1700 is down!
[2020-04-24 01:59:36:269 PDT] [Server1400] LOGOUT: SUCCESS
[2020-04-24 01:59:36:269 PDT] User logged out: user2
jar — java -jar Admin.jar — 55×24
(base) sunyutingsMBP2:jar yutingsun$ java -jar Admin.jar
kill <port>: to kill a server
restart <port>: to restart a server
admin@127.0.0.1# kill 1300
admin@127.0.0.1# kill 1500
admin@127.0.0.1# kill 1700
admin@127.0.0.1# restart 1300
admin@127.0.0.1#
jar — java -jar Client.jar c1 — 67×24
(base) sunyutingsMBP2:jar yutingsun$ java -jar Client.jar c1
Assigned to 1600
collaborative@127.0.0.1# login usr1 111
Unregistered or password do not match.
collaborative@127.0.0.1# login user1 111
Logged in successfully as user1
collaborative@127.0.0.1# logout
Successfully logged out.
collaborative@127.0.0.1# login user 111
Unregistered or password do not match.
collaborative@127.0.0.1# login user1 111
Logged in successfully as user1
collaborative@127.0.0.1# logout
Request aborted.
collaborative@127.0.0.1#
jar — java -jar Client.jar c2 — 66×24
(base) sunyutingsMBP2:jar yutingsun$ java -jar Client.jar c2
Assigned to 1400
collaborative@127.0.0.1# login user2 222
Logged in successfully as user2
collaborative@127.0.0.1# logout
Request aborted.
collaborative@127.0.0.1# logout
Successfully logged out.
collaborative@127.0.0.1#

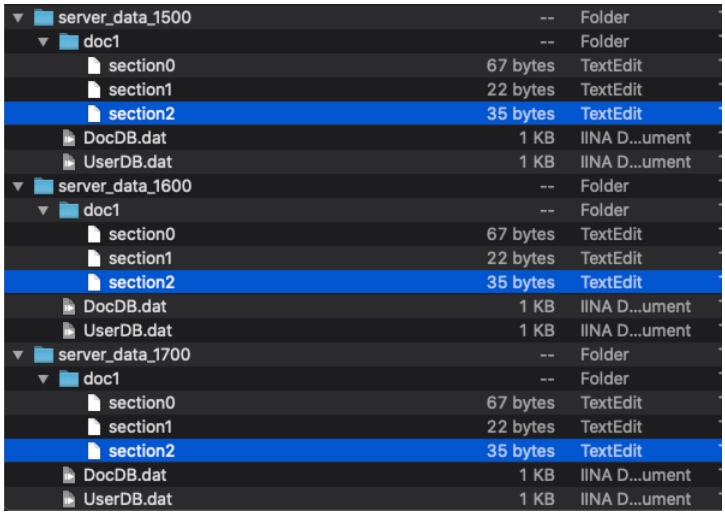
Test3 Data Recovery

1. Kill server 1400 and edit doc1 section2.

The file is synced among all servers except for server1400.

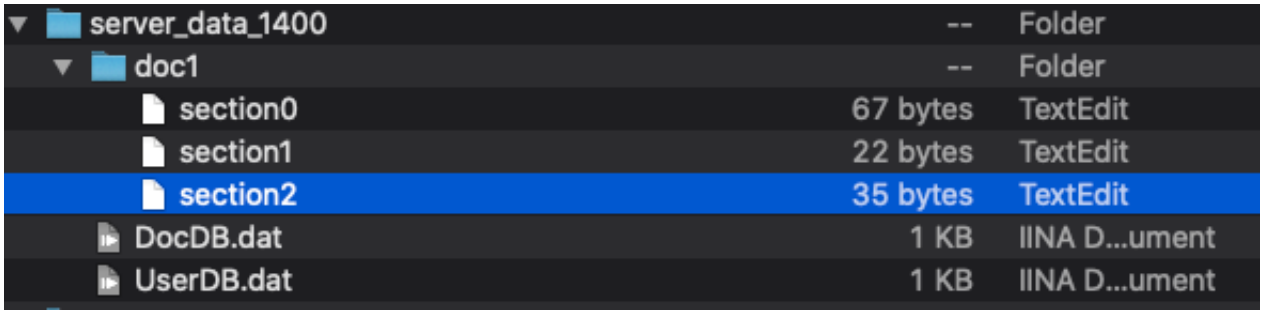






2. **Restart server 1400 by command `restart 1400` in Admin**

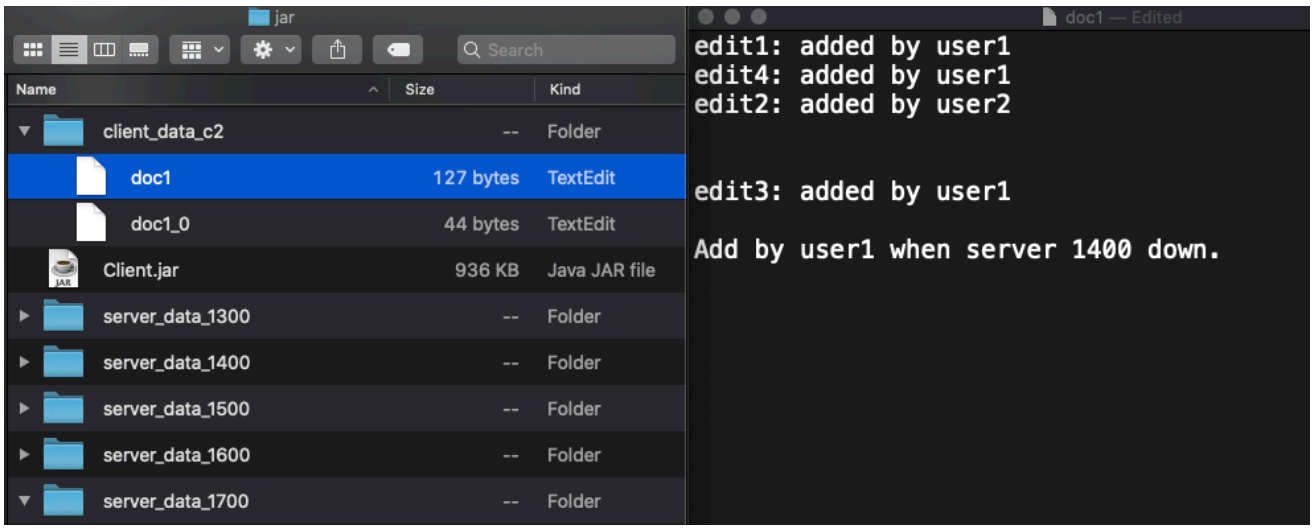
We can see the data is recovered



3. **Test whether client can fetch correct data from the restarted server**

- reconnect client2 to server 1400. This may take several trials, since the server port is assigned randomly.
- login as user2 and download latest `doc1` file from server1400 by command `showdoc doc1`.

  We can see the complete file of doc1 in client directory. It contains all sections including the recovered one!



Test4. Chatting

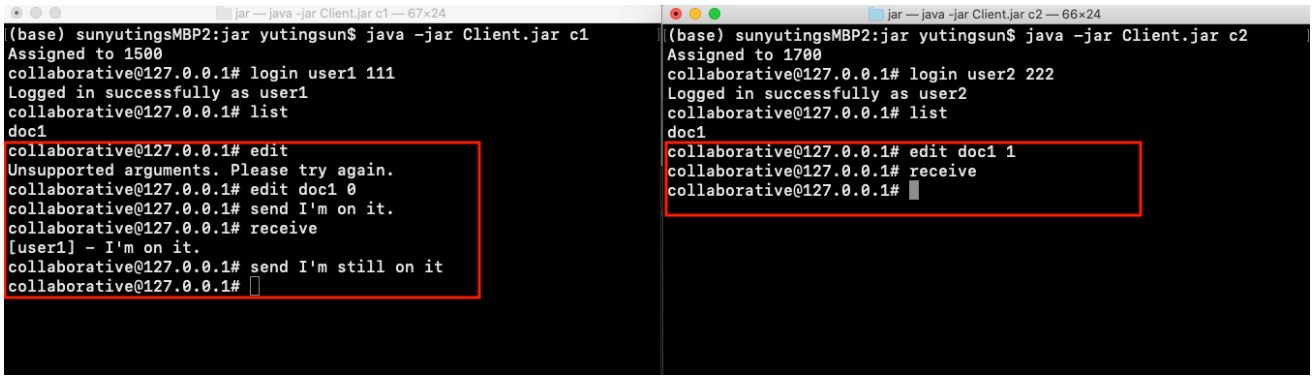
Start:

User1 logged in client1, user2 logged in client2, they both have access to edit doc1.

Step1:

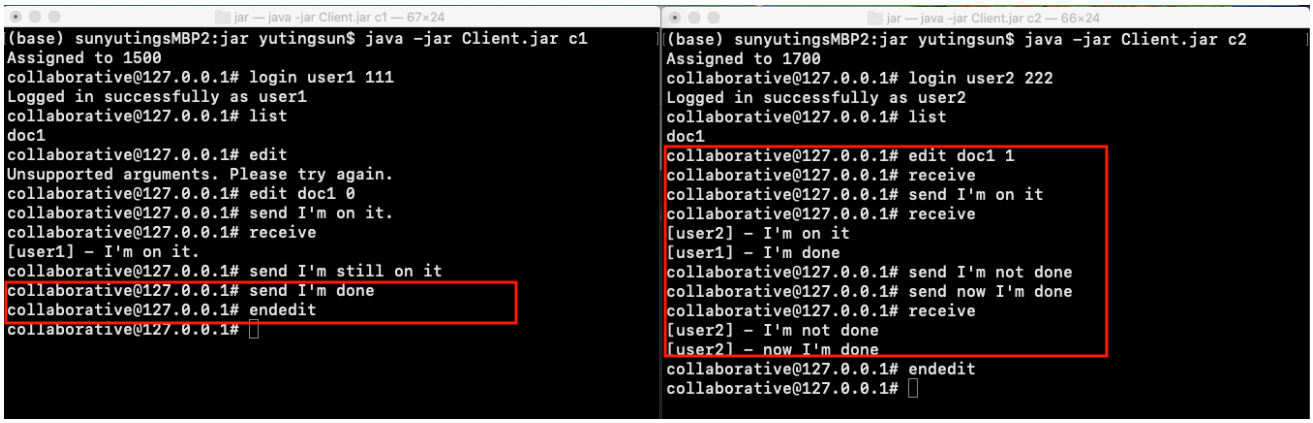
- User1 starts to edit doc1
- User1 sends message to the chatroom by command `send`
- User1 can receive the message history by command `receive`

- User2 starts to edit doc1. User2 cannot receive chatroom messages before.



Step2:

- User2 sends message to the chatroom: I'm on it
- User1 sends message: I'm done, and stops editing by command `endedit`
- user2 can receive the message history from the point s/he enters the chatroom by command `receive`
- user2 sends more messages. When enters `receive` again, s/he can only receive unread messages.



Step3:

- User1 comes back to the chatroom by editing doc1 again (can be a different section)
- User1 can only receive message history when s/he was IN the chatroom. User1 cannot receive messages sent by user2 while user1 was not in the chatroom.