

Testing Scenarios And Results

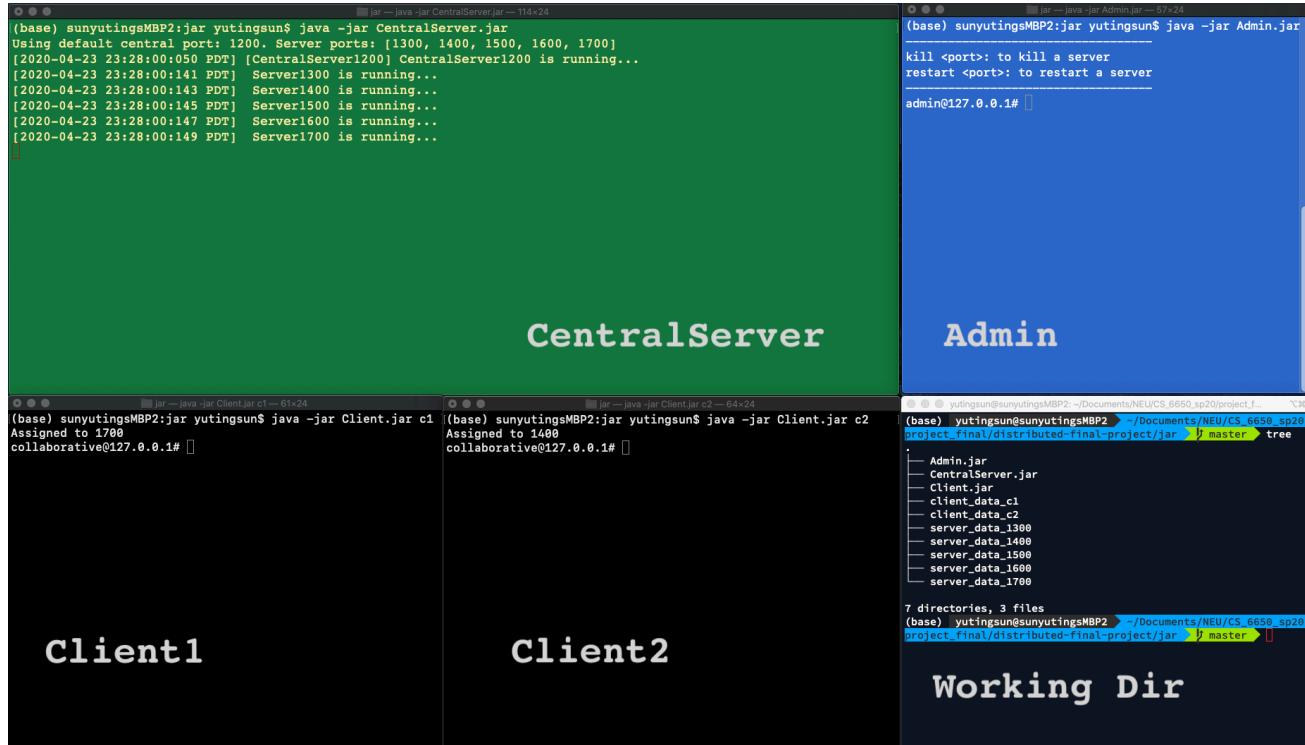
Getting Started

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



The image displays four terminal windows arranged in a 2x2 grid, illustrating the setup process:

- CentralServer**: Shows the command \$java -jar CentralServer.jar being run, outputting logs about servers starting at various ports (1300, 1400, 1500, 1600, 1700).
- Admin**: Shows the command \$java -jar Admin.jar being run, outputting help text for managing servers.
- Client1**: Shows the command \$java -jar Client.jar c1 being run, outputting that it is assigned to port 1700.
- Client2**: Shows the command \$java -jar Client.jar c2 being run, outputting that it is assigned to port 1400.
- Working Dir**: Shows the command tree being run in a directory named 'master', displaying the following directory structure:

```
.  
├── Admin.jar  
├── CentralServer.jar  
├── Client.jar  
├── client_data_c1  
├── client_data_c2  
└── server_data_1300  
    ├── server_data_1400  
    ├── server_data_1500  
    └── server_data_1600  
        └── server_data_1700
```

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!

The image shows three terminal windows side-by-side:

- Central Server Log:** Shows a sequence of 2PC transaction logs between multiple servers (Server1600, Server1700, Server1300, Server1400, Server1500) over several minutes. It includes "Agree: sent", "Commit: sent", and "Commit: received" messages, along with a "CREATE_USER: SUCCESS" message for user1.
- Client c1 Log:** Displays a help menu for the Client application. A red box highlights the command: `register USER PWD: to register a new account with username USER and password PWD`. Below this, a red box highlights the output of the command: `collaborative@127.0.0.1# register user1 111` followed by `User user1 registered successfully!`.
- Client c2 Log:** Shows a login session. It starts with a prompt: `(base) sunyutingsMBP2:jar yutingsun$ java -jar Client.jar c2`. It then shows the user being assigned to port 1400: `Assigned to 1400`. The user logs in with the command: `collaborative@127.0.0.1# login user1 111`, followed by `logged in successfully as user1`.

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.

```
[2020-04-23 23:43:59:040 PDT] [Server1400] Commit: sent
[2020-04-23 23:43:59:041 PDT] [Server1300] Commit: received
[2020-04-23 23:43:59:043 PDT] [Server1500] Commit: received
[2020-04-23 23:43:59:045 PDT] [Server1600] Commit: received
[2020-04-23 23:43:59:047 PDT] [Server1700] Commit: received
[2020-04-23 23:43:59:049 PDT] [Server1400] LOGIN: SUCCESS
[2020-04-23 23:43:59:049 PDT] New user logged in: user2
[2020-04-23 23:44:14:025 PDT] [Server1700] Prepare: sent
[2020-04-23 23:44:14:028 PDT] [Server1300] Prepare: received
[2020-04-23 23:44:14:031 PDT] [Server1300] Agree: sent
[2020-04-23 23:44:14:033 PDT] [Server1400] Prepare: received
[2020-04-23 23:44:14:035 PDT] [Server1400] Agree: sent
[2020-04-23 23:44:14:037 PDT] [Server1500] Prepare: received
[2020-04-23 23:44:14:039 PDT] [Server1500] Agree: sent
[2020-04-23 23:44:14:041 PDT] [Server1600] Prepare: received
[2020-04-23 23:44:14:043 PDT] [Server1600] Agree: sent
[2020-04-23 23:44:14:044 PDT] [Server1700] Commit: sent
[2020-04-23 23:44:14:045 PDT] [Server1300] Commit: received
[2020-04-23 23:44:14:049 PDT] [Server1400] Commit: received
[2020-04-23 23:44:14:052 PDT] [Server1500] Commit: received
[2020-04-23 23:44:14:054 PDT] [Server1600] Commit: received
[2020-04-23 23:44:14:058 PDT] [Server1700] CREATE_DOCUMENT: SUCCESS
[2020-04-23 23:44:14:058 PDT] File successfully created: doc1

login USER PWD: to login using USER and PWD credentials
create DOC SEC: to create a new document named DOC and contains
SEC sections
edit DOC SEC (TMP): to edit the section SEC of DOC document (usi
ng TMP temporary filename)
endedit: to stop the current editing session
showsec DOC SEC (OUT): to download the content of the SEC sectio
n of DOC document (using OUT output filename)
showdoc DOC (OUT): to download the content concatenation of all
the document's sections (using OUT output filename)
logout: to logout
list: to list all the documents you are able to see and edit
share USER DOC: to share a document with another user
news: to get all the news
receive: to retrieve all the unread chat messages
send TEXT: to send the TEXT message regarding the document being
edited
collaborative@127.0.0.1# register user1 111
User user1 registered successfully!
collaborative@127.0.0.1# login user1 111
Logged in successfully as user1
collaborative@127.0.0.1# create doc1 3
Successfully create a new document.
collaborative@127.0.0.1# 

(base) sunyutingsMBP2:jar yutingsun$ java -jar Client.jar c2
Assigned to 1400
collaborative@127.0.0.1# login user1 111
Logged in successfully as user1
collaborative@127.0.0.1# register user2 222
User user2 registered successfully!
collaborative@127.0.0.1# login user2 222
You're already logged in.
collaborative@127.0.0.1# logout
Successfully logged out.
collaborative@127.0.0.1# login user2 222
Logged in successfully as user2
collaborative@127.0.0.1# logout
Successfully logged out.
collaborative@127.0.0.1# log2 user2 222
Unsupported arguments. Please try again.
collaborative@127.0.0.1# login user2 222
Logged in successfully as user2
collaborative@127.0.0.1# 
```

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.

Name	Size	Kind
client_data_c2	--	Folder
client_data_c1	--	Folder
server_data_1700	--	Folder
doc1	--	Folder
section2	Zero bytes	TextEdit
section1	Zero bytes	TextEdit
section0	Zero bytes	TextEdit
server_data_1600	--	Folder
doc1	--	Folder
section2	Zero bytes	TextEdit
section1	Zero bytes	TextEdit
section0	Zero bytes	TextEdit
server_data_1500	--	Folder
doc1	--	Folder
section2	Zero bytes	TextEdit
section1	Zero bytes	TextEdit
section0	Zero bytes	TextEdit
server_data_1400	--	Folder
server_data_1300	--	Folder
Client.jar	936 KB	Java JAR file
CentralServer.jar	936 KB	Java JAR file
Admin.jar	936 KB	Java JAR file

```

jar — java -jar CentralServer.jar — 114
[2020-04-23 23:50:34:026 PDT] [Server1600] Agree: sent
[2020-04-23 23:50:34:028 PDT] [Server1700] Commit: sent
[2020-04-23 23:50:34:030 PDT] [Server1300] Commit: received
[2020-04-23 23:50:34:044 PDT] [Server1400] Commit: received
[2020-04-23 23:50:34:047 PDT] [Server1500] Commit: received
[2020-04-23 23:50:34:054 PDT] [Server1600] Commit: received
[2020-04-23 23:50:34:057 PDT] [Server1700] SHARE: SUCCESS
[2020-04-23 23:50:35:649 PDT] [Server1400] Prepare: sent
[2020-04-23 23:50:35:666 PDT] [Server1300] Prepare: received
[2020-04-23 23:50:35:670 PDT] [Server1300] Agree: sent
[2020-04-23 23:50:35:680 PDT] [Server1500] Prepare: received
[2020-04-23 23:50:35:682 PDT] [Server1500] Agree: sent
[2020-04-23 23:50:35:685 PDT] [Server1600] Prepare: received
[2020-04-23 23:50:35:687 PDT] [Server1600] Agree: sent
[2020-04-23 23:50:35:693 PDT] [Server1700] Prepare: received
[2020-04-23 23:50:35:695 PDT] [Server1700] Agree: sent
[2020-04-23 23:50:35:702 PDT] [Server1400] Commit: sent
[2020-04-23 23:50:35:703 PDT] [Server1300] Commit: received
[2020-04-23 23:50:35:706 PDT] [Server1500] Commit: received
[2020-04-23 23:50:35:710 PDT] [Server1600] Commit: received
[2020-04-23 23:50:35:715 PDT] [Server1700] Commit: received
[2020-04-23 23:55:03:324 PDT] [Server1400] LIST: SUCCESS
[2020-04-23 23:55:07:720 PDT] [Server1700] LIST: SUCCESS

jar — java -jar Client.jar c1 — 66x24
showsec DOC SEC (OUT): to download the content of the SEC section of DOC document (using OUT output filename)
showdoc DOC (OUT): to download the content concatenation of all the document's sections (using OUT output filename)
logout: to logout
list: to list all the documents you are able to see and edit
share USER DOC: to share a document with another user
news: to get all the news
receive: to retrieve all the unread chat messages
send TEXT: to send the TEXT message regarding the document being edited
collaborative@127.0.0.1# register user1 111
User user1 registered successfully!
collaborative@127.0.0.1# login user1 111
Logged in successfully as user1
collaborative@127.0.0.1# create doc1 3
Successfully create a new document.
collaborative@127.0.0.1# share doc1 user2
Document does not exist.
collaborative@127.0.0.1# share user2 doc1
Document shared successfully
collaborative@127.0.0.1# list
doc1
collaborative@127.0.0.1# 

(base) sunyutingsMBP2:jar yutingsun$ java -jar Client.jar c2
Assigned to 1400
collaborative@127.0.0.1# login user1 111
Logged in successfully as user1
collaborative@127.0.0.1# register user2 222
User user2 registered successfully!
collaborative@127.0.0.1# login user2 222
You're already logged in.
collaborative@127.0.0.1# logout
Successfully logged out.
collaborative@127.0.0.1# login user2 222
Logged in successfully as user2
collaborative@127.0.0.1# logout
Successfully logged out.
collaborative@127.0.0.1# log2 user2 222
Unsupported arguments. Please try again.
collaborative@127.0.0.1# login user2 222
Logged in successfully as user2
collaborative@127.0.0.1# You have a new notification.
news
You have permission on these new documents: doc1
collaborative@127.0.0.1# list
doc1
collaborative@127.0.0.1# 

```

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 committed, we can see the text in file `doc1/section0` in every server directory. We can also notice the file size of `section0` is changed.

The screenshot displays three windows illustrating the document editing process:

- CentralServer Log:** Shows the server's activity from April 24, 2020, at 00:24:12. It includes log entries for committing, preparing, and agreeing phases across multiple servers (1400, 1300, 1500, 1600, 1700), followed by an EDIT: SUCCESS message.
- Text Editor:** A window titled "edit1: added by user1" containing the text "section0 — Edited". Below it is another window titled "edit1: added by user1" containing the text "doc1_0".
- File Browser:** A Mac OS X Finder window showing the directory structure. It includes JAR files for Admin, CentralServer, and Client, along with client and server data folders. Under "client_data_c1", there is a "doc1_0" file (21...tes) and a "doc1" folder containing "section0", "section1", and "section2". Under "server_data_1400", there is a "doc1" folder containing "section0", "section1", and "section2".

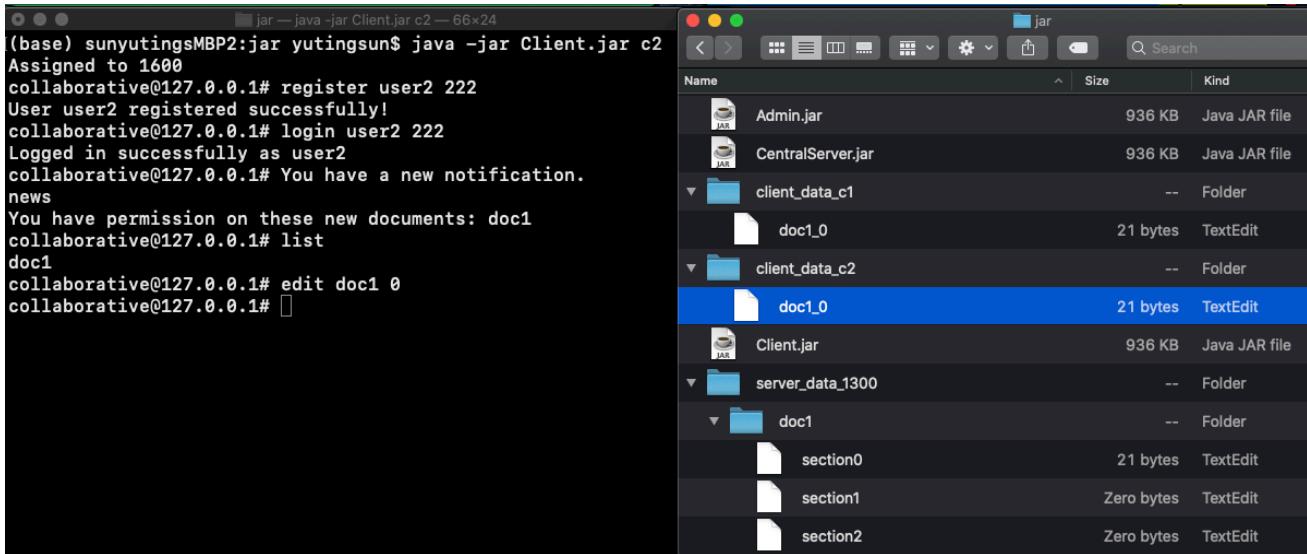
```
[2020-04-24 00:24:12:338 PDT] [Server1400] Commit: sent
[2020-04-24 00:24:12:339 PDT] [Server1300] Commit: received
[2020-04-24 00:24:12:341 PDT] [Server1500] Commit: received
[2020-04-24 00:24:12:343 PDT] [Server1600] Commit: received
[2020-04-24 00:24:12:345 PDT] [Server1700] Commit: received
[2020-04-24 00:24:12:388 PDT] [Server1400] EDIT: SUCCESS
Apr 24, 2020 12:24:12 AM com.healthmarketscience.rmiio.exporter.RemoteStreamExporter getInstance
INFO: Using stream exporter com.healthmarketscience.rmiio.exporter.DefaultRemoteStreamExporter
[2020-04-24 00:25:17:693 PDT] [Server1400] Prepare: sent
[2020-04-24 00:25:17:697 PDT] [Server1300] Prepare: received
[2020-04-24 00:25:17:700 PDT] [Server1300] Agree: sent
[2020-04-24 00:25:17:703 PDT] [Server1500] Prepare: received
[2020-04-24 00:25:17:705 PDT] [Server1500] Agree: sent
[2020-04-24 00:25:17:707 PDT] [Server1600] Prepare: received
[2020-04-24 00:25:17:709 PDT] [Server1600] Agree: sent
[2020-04-24 00:25:17:713 PDT] [Server1700] Prepare: received
[2020-04-24 00:25:17:714 PDT] [Server1700] Agree: sent
[2020-04-24 00:25:17:715 PDT] [Server1400] Commit: sent
[2020-04-24 00:25:17:716 PDT] [Server1300] Commit: received
[2020-04-24 00:25:17:719 PDT] [Server1500] Commit: received
[2020-04-24 00:25:17:722 PDT] [Server1600] Commit: received
[2020-04-24 00:25:17:724 PDT] [Server1700] Commit: received
[2020-04-24 00:25:17:727 PDT] [Server1400] EDIT-END: SUCCESS

```

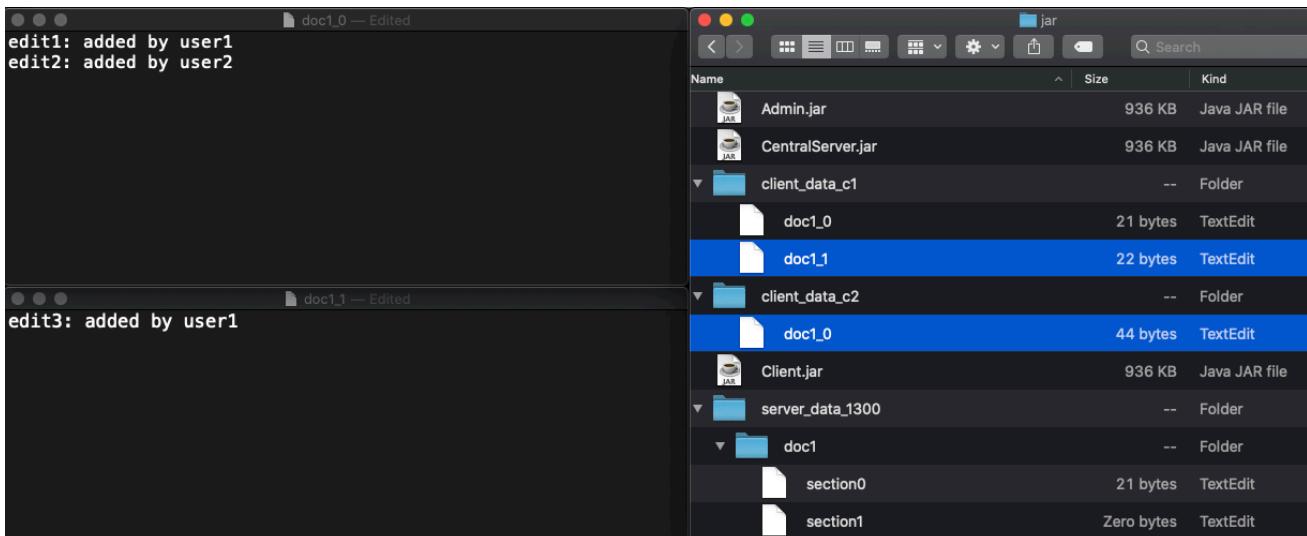
```
(base) sunyutingsMBP2:jar yutingsun$ java -jar Client.jar c1
Assigned to 1400
collaborative@127.0.0.1# register user1 111
User user1 registered successfully!
collaborative@127.0.0.1# login user1 111
Logged in successfully as user1
collaborative@127.0.0.1# create doc1 3
Successfully create a new document.
collaborative@127.0.0.1# share user2 doc1
Document shared successfully
collaborative@127.0.0.1# list
doc1
collaborative@127.0.0.1# edit doc1 0
collaborative@127.0.0.1# endedit
collaborative@127.0.0.1#
```

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.

▼ server_data_1300	--	Folder
▼ doc1	--	Folder
§ section0	44 bytes	TextEdit
§ section1	22 bytes	TextEdit
§ section2	Zero bytes	TextEdit
▼ server_data_1400	--	Folder
▼ doc1	--	Folder
§ section0	44 bytes	TextEdit
§ section1	22 bytes	TextEdit
§ section2	Zero bytes	TextEdit
▼ server_data_1500	--	Folder
▼ doc1	--	Folder
§ section0	44 bytes	TextEdit
§ section1	22 bytes	TextEdit
§ section2	Zero bytes	TextEdit
► server_data_1600	--	Folder
► server_data_1700	--	Folder

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

The screenshot shows two terminal windows side-by-side. The top window has a title bar "doc1_0 — Edited" and contains the text "edit1: added by user1". The bottom window has a title bar "section0 — Edited" and contains the text "edit1: added by user1" followed by "edit2: added by user2". Both windows have standard OS X window controls (red, yellow, green buttons) at the top.

```

doc1_0 — Edited
edit1: added by user1

section0 — Edited
edit1: added by user1
edit2: added by user2

```

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 sync 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 committed.

The image displays four terminal windows arranged in a 2x2 grid, illustrating a 2-phase commit process with a server failure.

- Top Left Terminal:** Shows the log of the CentralServer.jar application. It records various 2PC messages between multiple servers (1700, 1400, 1300, 1500, 1600) and a client. A red box highlights the message "[2020-04-24 01:54:46:181 PDT] [CentralServer1200] Server1300 is down!" indicating the failure of server 1300.
- Top Right Terminal:** Shows the log of the Admin.jar application. It includes help text for commands like `kill <port>` and `restart <port>`, followed by a command to kill port 1300, which is highlighted in red.
- Bottom Left Terminal:** Shows the log of the Client.jar application (client c1). It shows user1 logging in with password 111, and user2 logging in with password 222.
- Bottom Right Terminal:** Shows the log of the Client.jar application (client c2). It shows user2 logging in with password 222.

- 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 — 70x24

```
[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 — 55x24

```
(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 — 67x24

```
(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 — 66x24

```
(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# 
```

jar -- java -jar CentralServer.jar — 70x24

```
[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] [Server1400] 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 — 55x24

```
(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 — 67x24

```
(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 — 66x24

```
(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.

The image displays four terminal windows side-by-side:

- Top Left:** Shows log output from a CentralServer jar. It includes messages like "Commit: received" for various servers (1300, 1600, 1700) and a message "[2020-04-24 01:12:26:326 PDT] [CentralServer1200] Server1400 is down!". There are also INFO logs related to stream exporters.
- Top Right:** Shows a user running Admin.jar. The user enters commands to kill and restart server 1400, which are then echoed back.
- Bottom Left:** Shows a user running Client.jar c1. They log in as user1, edit a document, and then log out.
- Bottom Right:** Shows a user running Client.jar c2. They log in as user2, edit the same document, and then log out.

The image shows a file system browser comparing four servers (1300, 1500, 1600, 1700) for a document named 'doc1'. Each server has a folder 'server_data' containing sub-folders for each server ID. Inside each server folder, there is a 'doc1' folder containing three sections: 'section0', 'section1', and 'section2'. The 'section2' file on server 1300 is highlighted and circled in red, showing a size of 'Zero bytes'. The other sections and their sizes are as follows:

Server	Section	Size	Type
server_data_1300	section0	67 bytes	TextEdit
	section1	22 bytes	TextEdit
	section2	Zero bytes	TextEdit
server_data_1500	section0	67 bytes	TextEdit
	section1	22 bytes	TextEdit
	section2	35 bytes	TextEdit
server_data_1600	section0	67 bytes	TextEdit
	section1	22 bytes	TextEdit
	section2	35 bytes	TextEdit
server_data_1700	section0	67 bytes	TextEdit
	section1	22 bytes	TextEdit
	section2	35 bytes	TextEdit

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

We can see the data is recovered

▼	server_data_1400	--	Folder
▼	doc1	--	Folder
	section0	67 bytes	TextEdit
	section1	22 bytes	TextEdit
	section2	35 bytes	TextEdit
	DocDB.dat	1 KB	IINA Document
	UserDB.dat	1 KB	IINA Document

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!

The screenshot shows two windows. On the left is a file manager window titled 'jar' showing the directory structure:

- client_data_c2 (Folder)
 - doc1 (TextEdit) - 127 bytes
 - doc1_0 (TextEdit) - 44 bytes
 - Client.jar (Java JAR file) - 936 KB
- server_data_1300 (Folder)
- server_data_1400 (Folder)
- server_data_1500 (Folder)
- server_data_1600 (Folder)
- server_data_1700 (Folder)

On the right is a text editor window titled 'doc1 — Edited' containing the recovered document content:

```
edit1: added by user1
edit4: added by user1
edit2: added by user2

edit3: added by user1

Add by user1 when server 1400 down.
```

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.

The screenshot shows two terminal windows side-by-side. The left window (Client.c1) shows User1's session. User1 logs in as user1 (id 111), lists 'doc1', and attempts to edit it but gets an unsupported arguments error. Then User1 sends a message 'I'm on it.' and receives a message from User2 saying '[user2] - I'm on it.' The right window (Client.c2) shows User2's session. User2 logs in as user2 (id 222), lists 'doc1', edits it, and sends a message 'I'm still on it.' User2 also receives the message from User1. Both windows show a list of commands at the bottom: 'edit', 'send', 'receive', and 'endedit'.

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.

The screenshot shows the same two terminal windows. In the left window (Client.c1), User1 sends 'I'm still on it.' and 'endedit'. In the right window (Client.c2), User2 receives '[user1] - I'm still on it.', then sends '[user2] - I'm on it.' and '[user2] - I'm not done'. After User1's 'endedit', User2 receives '[user1] - I'm done' and '[user2] - now I'm done'. Both windows show the full command set at the bottom: 'edit', 'send', 'receive', and 'endedit'.

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.

The image shows two terminal windows side-by-side. Both windows have a title bar indicating they are running 'java -jar Client.jar'.

Terminal 1 (User1):

```
Assigned to 1500
collaborative@127.0.0.1# login user1 111
Logged in successfully as user1
collaborative@127.0.0.1# list
doc1
collaborative@127.0.0.1# edit
Unsupported arguments. Please try again.
collaborative@127.0.0.1# edit doc1 0
collaborative@127.0.0.1# send I'm on it.
collaborative@127.0.0.1# receive
[user1] - I'm on it.
collaborative@127.0.0.1# send I'm still on it
collaborative@127.0.0.1# send I'm done
collaborative@127.0.0.1# endedit
collaborative@127.0.0.1# edit doc1 2
The section is being edited
collaborative@127.0.0.1# receive
You're not editing any document
collaborative@127.0.0.1# edit doc1 1
collaborative@127.0.0.1# receive
[user1] - I'm still on it
[user2] - I'm on it
[user1] - I'm done
collaborative@127.0.0.1#
```

Terminal 2 (User2):

```
(base) sunyutingsMBP2:jar yutingsun$ java -jar Client.jar c2
Assigned to 1700
collaborative@127.0.0.1# login user2 222
Logged in successfully as user2
collaborative@127.0.0.1# list
doc1
collaborative@127.0.0.1# edit doc1 1
collaborative@127.0.0.1# receive
collaborative@127.0.0.1# send I'm on it
collaborative@127.0.0.1# receive
[user2] - I'm on it
[user1] - I'm done
collaborative@127.0.0.1# send I'm not done
collaborative@127.0.0.1# send now I'm done
collaborative@127.0.0.1# receive
[user2] - I'm not done
[user1] - now I'm done
collaborative@127.0.0.1# endedit
collaborative@127.0.0.1#
```

A red rectangular box highlights the portion of the User1 log where they attempt to edit 'doc1' again, receive a message from User2, and then end their edit session. Another red box highlights the portion of the User2 log where they send a message to User1 while User1 is not in the room, and User1 receives it once they re-enter.