**CS 544 – Requirements Document and DFA validation**
**Group – 4**
**Members – Ankush Israney, Anthony Emma, Edwin Dauber, Francis Obiagwu**

## Language:

All of our code has been written using the java SE 7 platform and its corresponding network libraries. The initial basic socket programming skeleton (files RTCEClient.java and RTCEServer.java) has been a direct intercept from forouzan top down approach – socket programming using java in which we used the concurrent server model. The rest of the code in its entirety has been done independently by the group.

## Platform (in Read me as well) :

We have tested all our code on the Windows 10 platform. Since Java is platform independent, there should be no reason for the code not to work on other platforms which support Java. Although testing this code on Windows would be recommended. Instructions to run are in the Readme file although mentioning it here, we can simply run it using the following commands from the directory of the jar package.

Run the jar file RTCEServer.jar (the command is java -jar RTCEServer.jar).

Run the jar file RTCEClient.jar (the command is java -jar RTCEClient.jar)

## Protocol Requirements

- ***Stateful*:**

The DFA has been modelled completely as per the updated design document. The RTCEServer.java (server package) along with the RTCEClient.java & the RTCEClientUIInput.java (client package) model the statefulness of the protocol respectively. The RTCEServer.java keeps track of each client thread state using a thread model and a driver function for the thread. An approach of nesting and flags has been used to achieve this statefulness. The DFA has been tested and verified for each transition and a demo in the video explains the statefulness implemented by the Server.

The client is restrictive and adheres to all the states and transitions in the DFA as well although a point to make here is that to accommodate for a malicious client, direct typing of PDU messages on the command line has been purposely kept as an option to try to test some basic form of fuzzing. It can be easily removed or commented out in RTCEClientUIInput.java but can be executed for the purpose of testing while grading.

- ***Concurrency :***

The concurrent thread model of the java programming language using the runnable interface and the thread start and run functions have been used at each stage to model the concurrent nature of both the client and the server.

The server uses concurrency to handle multiple client threads on port number 50000. A discovery thread *(Extra Credit portion)* is first started which receives client discovery requests on port number 4446. This logic is implemented in the RTCEDiscoveryServer class.

The server receives a datagram on port 4446 from a client. It then responds to the client by sending a 'HI' on port number 4447 of the datagram. If the client receives the 'HI' message from the server then it requests for a connection on port 50000.

It then starts a thread for the client in the run function of the RTCEServer.java class. Each thread (for each client) runs its own driver function with its local variables and global shared resources which are the document, the ServerLog.java - connection_list and the ServerRecordMgmt - clientRecord lists, section lists, and token lists.

The control to these shared resources has not been modelled using a critical section approach but rather a naïve use of static types and controlling has been implemented to model the application side of access control in the protocol.

As far as the front end is concerned, UI threads run for each client input and output which are started in the RTCEClient.java class and belong to the RTCEClientUIInput.java and RTCEClientUIOutput.java respectively. They are used to refresh the UI screen which is just a command line UI in our case.

***The service:***

The client binds to the reserved TCP port number 50000 in the case of our RTCE protocol which is passed as an argument to the constructor of the RTCEClient.java class for the RTCEClient object. Within the constructor, a new socket is created and the socket is used to initialize the getInputStream() and getOutputStream() methods of the java.net.*; package of the tcp connection.

- ***Client:***

The client is capable of either discovering the server using the discoverServer() function in the RTCEClient.java class of the client package of the code. This extra credit portion is defined well in the heading under Extra Credit below. The result of this function is passed as an argument to the socket of the client in the constructor of the RTCEClient.java class.

Although if the extra the credit portion fails, we have a backup. If the client cannot find the server until a certain number of attempts to receive (10) then the client allows the user to enter the ip address of the server manually on the command line.

- ***User Interface:***

The UI is a command line UI in which we can enter commands in the order of the DFA and receive responses if our requests are granted or denied. The commands are mentioned ahead. The UI is modelled in the clientUIInput.java and clientUIOutput.java files of the client package.

**Commands on the UI:**

We have created only 2 different logins (no spaces in between commas):

login,cs544,cs544,cs544,example1

login,group4,password,group4,example1

These 2 commands have 2 be entered when the client starts and has found the Server or a connection has been established.

The first portion says its login, the second after the comma is username, the third is password and the fourth is document name.

The next is that we can get a request after we have received a document by entering the request for a section:

request,2

Here, request says that the command is of type request and section number says that we are requesting for section 2.

The next is if we get a successful response and get access we can commit our changes as follows.

commit,1,2

Here 2 denotes the section for which we are commenting and 1 denotes the section previous to 2. It just is needed for us to locate section 2 in our commit pdu. When we enter the text for the commit after enter new text prompt, it should give us the updated document with a success.

Then if we are a super user or the owner (or the first client to form a connection to the server in our implementation) then we can block a user using the block command.

block,group4 (example)

This command will block the user from executing his statements.

If we want to logoff, we have type LOGOFF (in caps).

LOGOFF

If we want to abort without an acknowledgment then we must type ABORT (in caps)

ABORT

If we want to test whether the server is active after forming a connection (connected state) then we have to type

ECHO and we should hopefully get an echo back.

***Extra Credit Portion:***

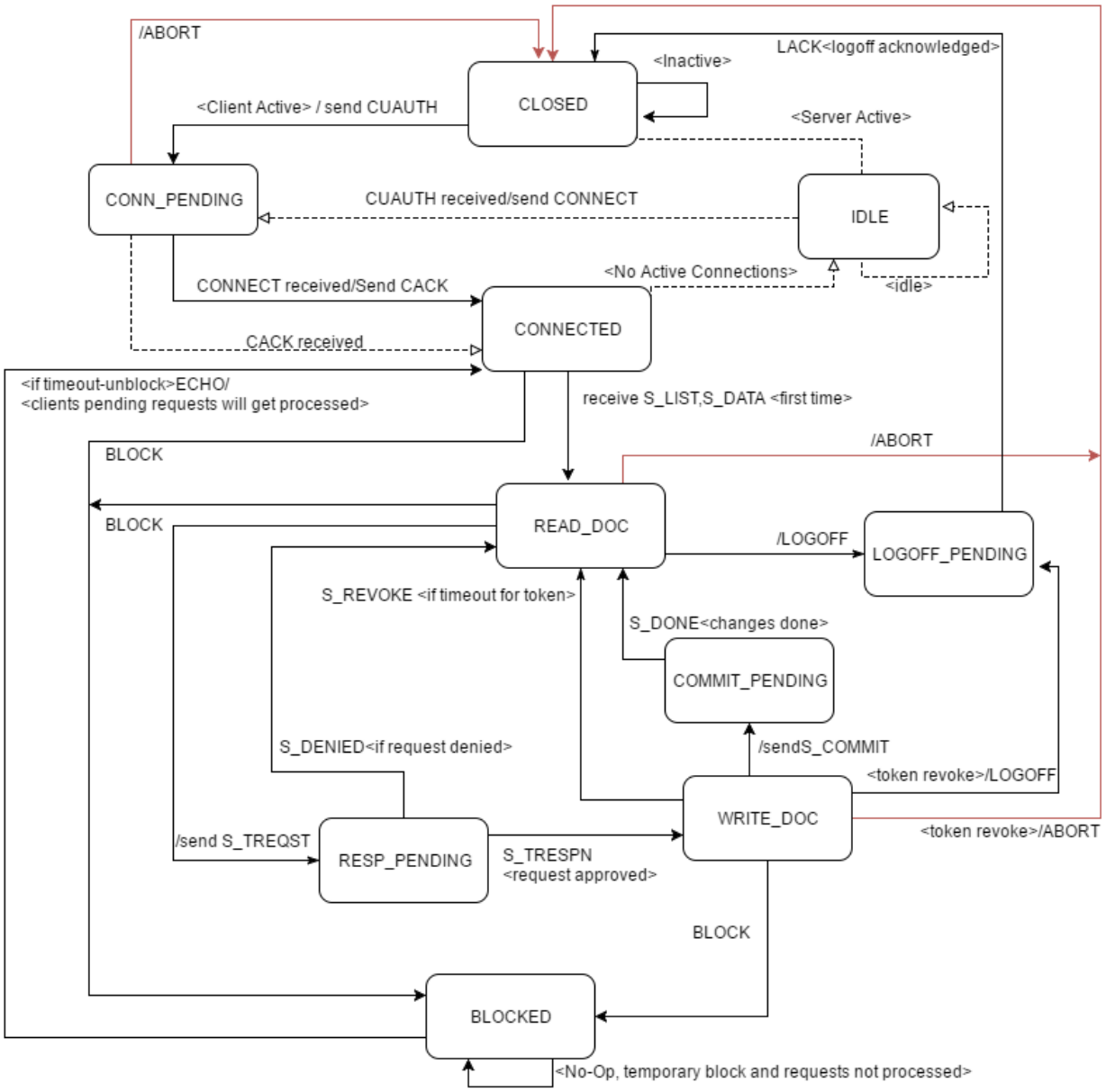
The server as mentioned starts a discovery thread on port number 4446 of the RTCEDiscoveryServer.java class. The server responds clients on this port with a datagram saying ‘HI’ if the udp packet length is 4. When the client receives this ‘HI’ on port 4447 in the RTCEClient.java class constructor which gives a call to the discoverServer() function, the client knows the ip address of the server and supplies this ip address to the socket else the ip address can be supplied manually.

*Where does the client send the udp packet :*

The server discovery is implemented by having the server create a separate thread to respond with a "HI" to any client who sends a message. When a client starts he will make a call to Java's routine to get the client's IP address, and then attempt to change the fourth octet of the IP address and see if any server responds back with a "HI". So for example if the client's IP address is 192.168.1.9, it will try 192.168.1.* where * is 1 - 255 to see if any server says "HI". This limits discovery to the network, and will not discover server's that exist beyond the network (through routing tables and gateways for example)

## *(PTO for client in Action…)*

***Verification of DFA states and Protocol in Action:***




---

Client 192.168.33.1 is searching for a server... **(EXTRA CREDIT)**

Found Server: /192.168.33.1

---

Document: null

---

Enter Command>>

**login,cs544,cs544,cs544,example1** **(LOGIN)**

Logging in as cs544 to edit document cs544/example1

---

Document: null

---

1:Once upon a midnight dreary, while I pondered, weak and weary, Over many a quaint and curious volume of forgotten lore

2:While I nodded, nearly napping, suddenly there came a tapping, As of some one gently rapping, rapping at my chamber door

3:

4:

5:

6:

Enter Command>>

---

Document: null

---

1:Once upon a midnight dreary, while I pondered, weak and weary, Over many a quaint and curious volume of forgotten lore

2:While I nodded, nearly napping, suddenly there came a tapping, As of some one gently rapping, rapping at my chamber door

3:'Tis some visitor,' I muttered, 'tapping at my chamber door-  Only this and nothing more.'

4:Ah, distinctly I remember it was in the bleak December; And each separate dying ember wrought its ghost upon the floor.

5:

6:

Enter Command>>

---

Document: null

---

1:Once upon a midnight dreary, while I pondered, weak and weary, Over many a quaint and curious volume of forgotten lore

2:While I nodded, nearly napping, suddenly there came a tapping, As of some one gently rapping, rapping at my chamber door

3:'Tis some visitor,' I muttered, 'tapping at my chamber door- Only this and nothing more.'

4:Ah, distinctly I remember it was in the bleak December; And each separate dying ember wrought its ghost upon the floor.

5: Eagerly I wished the morrow;-vainly I had sought to borrow From my books surcease of sorrow-sorrow for the lost Lenore

6:For the rare and radiant maiden whom the angels name Lenore- Nameless here for evermore

## Moving from CLOSED to CONN_PENDING to CONNECTED to READ_DOC.

- ➔ As we can see here, the client sends a CUAUTH message to the server while logging in the RTCEServerMessage.java class of the client.
- ➔ The login in the RTCEClientUI.java gives a call to the setCUAUTH in the RTCEClientMessage.java class of the client and sets the cuauth flag as true.
- ➔ The Server responds with a CONNECT in the driver function of the client thread in its RTCE.java file by again calling its RTCEServerMessage.java to setCONNECT() function and then send the response across
- ➔ The client receives this response in the RTCEClient.java getResponse() and responds with a CACK as sets the connect flag as true.
- ➔ If RTCEServer has received cack then it sets the cack flag as true.
- ➔ If CACK flag is true and CUAUTH flag is true, then the server starts a session driver for the client and inserts the active client connection in the the RTCEServerLog and inserts a client record in the RTCERecordMgmt in the back end.
- ➔ Also, if the Log is empty and a new connection comes in means this client is regarded as the owner (administrative authority) by our implementation and the owner flag is set to true in the RTCEServer.java file of the driver function.

## Moving from READ_DOC to RESP_PENDING to WRITE_DOC using S_TREQST and receiving S_TRESPN

Enter Command>> **(REQUEST)**
**request,2**
Requested for: 2
Got Access to update the requested section!

---

Document: null

---

1:Once upon a midnight dreary, while I pondered, weak and weary, Over many a quaint and curious volume of forgotten lore

**2:While I nodded, nearly napping, suddenly there came a tapping, As of some one gently rapping, rapping at my chamber door**

3:'Tis some visitor,' I muttered, 'tapping at my chamber door- Only this and nothing more.'

4:Ah, distinctly I remember it was in the bleak December; And each separate dying ember wrought its ghost upon the floor.

5: Eagerly I wished the morrow;-vainly I had sought to borrow From my books surcease of sorrow-sorrow for the lost Lenore

6:For the rare and radiant maiden whom the angels name Lenore- Nameless here for evermore

Enter Command>>

- ➔ As we can see here, the client requests for a section number 2 using the command request,2. When this happens, the RTCEClientUIInput() class checks if the client is connected using the connect flag. If the client is connected and the client does not have a token, then the client can request for a token.
- ➔ Here, we see that the server responds with a S_TRESPN using the sessionDriver() and giving a call tio its setS_TRESPN only if the section requested is free right now by checking the *record.checkFreeSection(section) && record.clientHasToken(client)!=true* i.e. only if the section is free and the client does not already have a previous token then the request is processed in the RTCEServerMessage.java class and the getS_TREQST() function

**Lets check if the client can make another token request!**

Document: null

---

1:Once upon a midnight dreary, while I pondered, weak and weary, Over many a quaint and curious volume of forgotten lore

2:While I nodded, nearly napping, suddenly there came a tapping, As of some one gently rapping, rapping at my chamber door

3:'Tis some visitor,' I muttered, 'tapping at my chamber door- Only this and nothing more.'

4:Ah, distinctly I remember it was in the bleak December; And each separate dying ember wrought its ghost upon the floor.

5: Eagerly I wished the morrow;-vainly I had sought to borrow From my books surcease of sorrow-sorrow for the lost Lenore

6:For the rare and radiant maiden whom the angels name Lenore- Nameless here for evermore

Enter Command>>

request,3

**Cannot request another token.  Client already has token for Section 2**

- ➔ As we can see here, the client cannot request another token since it has a previous token and has to commit data **(before a timeout!)**
- ➔ Even if the client was malicious the server has a check in the backend in the RTCEServerMessage.java class to process the S_TREQST only if *record.checkFreeSection(section) && record.clientHasToken(client)!=true* of the ServerRecordMgmt.java class to not process this request for a second request.

**From WRITE_DOC to COMMIT_PENDING to READ_DOC by sending S_COMMIT and receiving S_DONE.**

Enter Command>>

**commit,1,2** **(COMMIT)**

**Enter new text**

**abc**

**document updated..**

---

Document: null

---

1:Once upon a midnight dreary, while I pondered, weak and weary, Over many a quaint and curious volume of forgotten lore

**2:While I nodded, nearly napping, suddenly there came a tapping, As of some one gently rapping, rapping at my chamber door**

3:'Tis some visitor,' I muttered, 'tapping at my chamber door-  Only this and nothing more.'

4:Ah, distinctly I remember it was in the bleak December; And each separate dying ember wrought its ghost upon the floor.

5: Eagerly I wished the morrow;-vainly I had sought to borrow From my books surcease of sorrow-sorrow for the lost Lenore

6:For the rare and radiant maiden whom the angels name Lenore- Nameless here for evermore

Enter Command>>

---

Document: null

---

1:Once upon a midnight dreary, while I pondered, weak and weary, Over many a quaint and curious volume of forgotten lore

**2:abc**

3:'Tis some visitor,' I muttered, 'tapping at my chamber door-  Only this and nothing more.'

4:Ah, distinctly I remember it was in the bleak December; And each separate dying ember wrought its ghost upon the floor.

5: Eagerly I wished the morrow;-vainly I had sought to borrow From my books surcease of sorrow-sorrow for the lost Lenore

6:For the rare and radiant maiden whom the angels name Lenore- Nameless here for evermore

Enter Command>>

---

Document: null

---

1:Once upon a midnight dreary, while I pondered, weak and weary, Over many a quaint and curious volume of forgotten lore

2:abc

3:'Tis some visitor,' I muttered, 'tapping at my chamber door-  Only this and nothing more.'

4:Ah, distinctly I remember it was in the bleak December; And each separate dying ember wrought its ghost upon the floor.

5: Eagerly I wished the morrow;-vainly I had sought to borrow From my books surcease of sorrow-sorrow for the lost Lenore

6:For the rare and radiant maiden whom the angels name Lenore- Nameless here for evermore

- ➔ As we can see here, that when the client types in commit,1,2 as per the command list in the UI. If the client is connected and he has a token > 0 in the RTCEClientUIInput of the client package then the client sends a S_COMMIT to the server.
- ➔ In the backend, before processing S_COMMIT, the RTCEServerMessage.java class checks in the getS_COMMIT() function whether the client has the required token in his mapping in the clientRecordList(). If the client does that means the token still belongs to the client and the server processes the request. After returning from this function, the RTCEServer.java class sends the document to all active clients in the log.
- ➔ When the server responds with a S_DONE, we get a document updated message in the UI. Not only that but we also get the updated document which is not clearly reflected in the DFA but in our implementation we employ an instantaneous update to all active clients in the ServerLog.

**To check if we can commit without token!**

**Enter Command>>commit,1,2**

**INVALID: No token acquired**

- ➔ As we can see here, the client is not allowed to do a commit without a token.
- ➔ Even though the server has a check in the back end to not let a malicious client commit without a token request, we can see here in RTCEClientUIInput(). That if no token is acquired by RTCEClient class or if the parent.token value is reset to 0 then it will not allow us to make a commit to the server. But even if it does, our server is strong enough to handle illegal requests.

**From WRITE_DOC to READ_DOC (S_REVOKE due to Timeout)**

**request,1**

Requested for: 1

Got Access to update the requested section!

---

Document: null

---

1:Once upon a midnight dreary, while I pondered, weak and weary, Over many a quaint and curious volume of forgotten lore

2:abc

3:'Tis some visitor,' I muttered, 'tapping at my chamber door-  Only this and nothing more.'

4:Ah, distinctly I remember it was in the bleak December; And each separate dying ember wrought its ghost upon the floor.

5: Eagerly I wished the morrow;-vainly I had sought to borrow From my books surcease of sorrow-sorrow for the lost Lenore

6:For the rare and radiant maiden whom the angels name Lenore- Nameless here for evermore

Enter Command>>

**Access to update is revoked due to timeout. You need to make a new request again**

**(REVOKE)**

---

Document: null

---

1:Once upon a midnight dreary, while I pondered, weak and weary, Over many a quaint and curious volume of forgotten lore

2:abc

3:'Tis some visitor,' I muttered, 'tapping at my chamber door-  Only this and nothing more.'

4:Ah, distinctly I remember it was in the bleak December; And each separate dying ember wrought its ghost upon the floor.

5: Eagerly I wished the morrow;-vainly I had sought to borrow From my books surcease of sorrow-sorrow for the lost Lenore

6:For the rare and radiant maiden whom the angels name Lenore- Nameless here for evermore

Enter Command>>

- ➔ As we can see here, a timeout occurred if we didn't do anything with the token. This tells us that our implementation in the RTCEServer.java class started a timer as startTokenTimer(client); if the S_TREQST was matched by the server and a S_TRESPN was given in response when S_TREQST was processed.
- ➔ Currently, we have set a 45 seconds timer so if we don't commit changes to our small document within 45 seconds then our token will expire and we will have to issue a new token.

➔ In RTCEServer.java, The token expired after the timeout and a tokenTimer.cancel() method was called to set the tokenWaitTimeout value to true and send a S_REVOKE to the client after processing a tokenRevoke() method in the class of ServerRecordMgmt() to revoke the token from the specific client by accessing its clientRecord.

**Did Token Actually get Revoked?**

Enter Command>>
Access to update is revoked due to timeout. You need to make a new request again
---
Document: null
---
1:Once upon a midnight dreary, while I pondered, weak and weary, Over many a quaint and curious volume of forgotten lore
2:abc
3:'Tis some visitor,' I muttered, 'tapping at my chamber door- Only this and nothing more.'
4:Ah, distinctly I remember it was in the bleak December; And each separate dying ember wrought its ghost upon the floor.
5: Eagerly I wished the morrow;-vainly I had sought to borrow From my books surcease of sorrow-sorrow for the lost Lenore
6:For the rare and radiant maiden whom the angels name Lenore- Nameless here for evermore

**Enter Command>>**
**commit,0,1**
**INVALID: No token acquired**

As we can see here, the token got revoked when the S_REVOKE message was received by the client by checking in the getResponse() method of the RTCEClient.java class.

**<u>Starting 2 clients and from RESP_PENDING to READ_DOC</u>**

Enter Command>>
login,group4,password,group4,example1 **(DIFFERENT USER LOGIN)**
Logging in as group4 to edit document group4/example1
---
Document: null
---
1:Once upon a midnight dreary, while I pondered, weak and weary, Over many a quaint and curious volume of forgotten lore

2:abc
3:
4:
5:
6:

Enter Command>>
---
Document: null
---
1:Once upon a midnight dreary, while I pondered, weak and weary, Over many a quaint and curious volume of forgotten lore
2:abc
3:'Tis some visitor,' I muttered, 'tapping at my chamber door- Only this and nothing more.'
4:Ah, distinctly I remember it was in the bleak December; And each separate dying ember wrought its ghost upon the floor.
5:
6:

Enter Command>>
---
Document: null
---
1:Once upon a midnight dreary, while I pondered, weak and weary, Over many a quaint and curious volume of forgotten lore
2:abc
3:'Tis some visitor,' I muttered, 'tapping at my chamber door- Only this and nothing more.'
4:Ah, distinctly I remember it was in the bleak December; And each separate dying ember wrought its ghost upon the floor.
5: Eagerly I wished the morrow;-vainly I had sought to borrow From my books surcease of sorrow-sorrow for the lost Lenore
6:For the rare and radiant maiden whom the angels name Lenore- Nameless here for evermore

**We can see here clearly that not only was the user able to login but he was also able to see the latest updated document changed by user 1 (cs544 here)**

The user goes through the same procedure as user 1.
**These 2 users can also run simultaneously on the same machine!**

**FROM READ_DOC to RESPN_PENDING to READ_DOC**

**Client 1:**

Enter Command>>
request,2
Requested for: 2
**Got Access to update the requested section!**
---
Document: null
---
1:Once upon a midnight dreary, while I pondered, weak and weary, Over many a quaint and curious volume of forgotten lore
2:abc
3:'Tis some visitor,' I muttered, 'tapping at my chamber door- Only this and nothing more.'
4:Ah, distinctly I remember it was in the bleak December; And each separate dying ember wrought its ghost upon the floor.
5: Eagerly I wished the morrow;-vainly I had sought to borrow From my books surcease of sorrow-sorrow for the lost Lenore
6:For the rare and radiant maiden whom the angels name Lenore- Nameless here for evermore

**Client 2:**

Enter Command>>
request,2
Requested for: 2
**Sorry the section is not free right now.. (DENIED)**
---
Document: null
---
1:Once upon a midnight dreary, while I pondered, weak and weary, Over many a quaint and curious volume of forgotten lore
2:abc
3:'Tis some visitor,' I muttered, 'tapping at my chamber door- Only this and nothing more.'
4:Ah, distinctly I remember it was in the bleak December; And each separate dying ember wrought its ghost upon the floor.
5: Eagerly I wished the morrow;-vainly I had sought to borrow From my books surcease of sorrow-sorrow for the lost Lenore
6:For the rare and radiant maiden whom the angels name Lenore- Nameless here for evermore

If we see that both the clients try to request at the very same time then if the request for cs544 got processed then group4 client cannot get the token. The group4 gets a new S_DENIED message which says that the section is not free right now.

- ➔ On the server side, when group4 sends S_TREQST the RTCEServer.java thread checks that the requested section is not free right now while processing the request in RTCEServerMessage.This is done using the same record.checkFreeSection(section) && record.clientHasToken(client)!=true condition of the record for the client in RTCEServerRecordMgmt. If the condition fails a S_DENIED message is sent by the RTCEServer
- ➔ On the client side, the message is processed in the getS_DENIED() method of the RTCEClientMessage.java class.

**FROM WRITE_DOC TO BLOCK TO CONNECTED (SIMILAR FOR READ_DOC TO BLOCK OR CONNECTED TO BLOCK)**

This feature shows us how to block a client temporarily from accessing the document.

Enter Command>>

**block,group4** **(BLOCK)**

**request,2**

**Requested for: 2**

**Got Access to update the requested section!**

---

Document: null

---

1:Once upon a midnight dreary, while I pondered, weak and weary, Over many a quaint and curious volume of forgotten lore

2:abc

3:'Tis some visitor,' I muttered, 'tapping at my chamber door- Only this and nothing more.'

4:Ah, distinctly I remember it was in the bleak December; And each separate dying ember wrought its ghost upon the floor.

5: Eagerly I wished the morrow;-vainly I had sought to borrow From my books surcease of sorrow-sorrow for the lost Lenore

6:For the rare and radiant maiden whom the angels name Lenore- Nameless here for evermore

Enter Command>>

Enter Command>>

**request,2**

**Requested for: 2**

**Sorry you are blocked temporarily..**

---

Document: null

---

1:Once upon a midnight dreary, while I pondered, weak and weary, Over many a quaint and curious volume of forgotten lore

2:abc

3:'Tis some visitor,' I muttered, 'tapping at my chamber door- Only this and nothing more.'

4:Ah, distinctly I remember it was in the bleak December; And each separate dying ember wrought its ghost upon the floor.

5: Eagerly I wished the morrow;-vainly I had sought to borrow From my books surcease of sorrow-sorrow for the lost Lenore

6:For the rare and radiant maiden whom the angels name Lenore- Nameless here for evermore

Enter Command>>

- ➔ As we can see here, when the owner of the document (discussed initially) tries to block a client as block,group4 in this case then we see here that when client group4 tries to process a request he is blocked out. He receives the BLOCK message from the server. Although as we shall see his requests are queued in the server.
- ➔ The block time is preconfigured in the readConfigFile() method of the RTCEServerConfig.java file of the server package. Right now, we have set this timeout to 5 minutes.
- ➔ When the client who is a owner sends a BLOCK, the checkOwner() function of the ServerLog is first processed to check whether the client is actually the owner and then blocks the username mentioned in the BLOCK pdu. All this processing happens in the getBLOCK() method of the RTCEServerMessage class after checking the static owner flag in the sessionDriver() of the RTCEServer.java class.

**What happens after 5 minutes?**

Enter Command>>

request,2

Requested for: 2

Sorry you are blocked temporarily..

---

Document: null

---

1:Once upon a midnight dreary, while I pondered, weak and weary, Over many a quaint and curious volume of forgotten lore

2:abc

3:'Tis some visitor,' I muttered, 'tapping at my chamber door-  Only this and nothing more.'

4:Ah, distinctly I remember it was in the bleak December; And each separate dying ember wrought its ghost upon the floor.

5: Eagerly I wished the morrow;-vainly I had sought to borrow From my books surcease of sorrow-sorrow for the lost Lenore

6:For the rare and radiant maiden whom the angels name Lenore- Nameless here for evermore

Enter Command>>

echo received back.. **(ECHO)**

---

Document: null

---

1:Once upon a midnight dreary, while I pondered, weak and weary, Over many a quaint and curious volume of forgotten lore

2:abc

3:'Tis some visitor,' I muttered, 'tapping at my chamber door-  Only this and nothing more.'

4:Ah, distinctly I remember it was in the bleak December; And each separate dying ember wrought its ghost upon the floor.

5: Eagerly I wished the morrow;-vainly I had sought to borrow From my books surcease of sorrow-sorrow for the lost Lenore

6:For the rare and radiant maiden whom the angels name Lenore- Nameless here for evermore

- ➔ As we can see here, after 5 minutes the client connection is restored by the server. In the RTCEServer.java class, the thread which is blocked enters an infinite loop until timeout.
- ➔ Although the client token is revoked before a client is blocked. This feature is important with respect to our DFA and document management where we revoke access from the client so that the owner who has blocked a client can request this token to make an important update now. The method is show below.

```
if(client.block == true)
                    {
                        control.tokenRevoke(client); //revoke any token if client is blocked
                        sendResponse(RTCEMessageType.BLOCK, -1, -1, this.sock); //send block message to client
                        startBlockTimer(client);
                        while(client.block==true){
                            /*if(blockTimeout){
                    client.block = false;
                    cancelBlockTimer();
                }*/

                        try{Thread.sleep(1000);} catch (Exception e){}
                        }
                    sendResponse(RTCEMessageType.ECHO, -1, -1, sock);
                        cancelBlockTimer();
                    }
```

We see that after the timeout occurs, an ECHO is sent by the server and the ECHO message is processed by the client in the RTCEClientMessage.java class.

**WRITE_DOC/ READ_DOC TO LOGOFF_PENDING AND ABORT:**

Enter Command>>

request,2

Requested for: 2

Got Access to update the requested section!

---

Document: null

---

1:Once upon a midnight dreary, while I pondered, weak and weary, Over many a quaint and curious volume of forgotten lore

2:abc

3:'Tis some visitor,' I muttered, 'tapping at my chamber door- Only this and nothing more.'

4:Ah, distinctly I remember it was in the bleak December; And each separate dying ember wrought its ghost upon the floor.

5: Eagerly I wished the morrow;-vainly I had sought to borrow From my books surcease of sorrow-sorrow for the lost Lenore

6:For the rare and radiant maiden whom the angels name Lenore- Nameless here for evermore

Enter Command>>

**LOGOFF**

**Logoff acknowledged by server (LOG OFF ACKNOWLEDGEMENT)**

BUILD SUCCESSFUL (total time: 30 minutes 16 seconds)

*Code:*

```
logoff = true;
                                        control.deleteClientRecord(client);
                                        log.removeActiveConnection(client);
                                        sendResponse(RTCEMessageType.LACK, -1, -1,
this.sock);
                                        flagConn = false;
                                        this.close();
                                        break;
```

- ➔ So we see here that the client types a basic LOGOFF and he is allowed to logoff from the client only after he receives a successful validation of his LOGOFF using a LACK on the server end.
- ➔ On the server side, the server revokes the client tokens, releases all his resources and closes this connection. The RTCEServer.java class also removes the client record in the backend and also removes the client from the ServerLog.
- ➔ So we see here, the threads stop from running by deleting all client records, sending a LACK to the client, setting the flagConn = false which stops the active thread from running and then gives a call to the RTCEServer.java function this.close() which closes the send and recv stream for the TCP connection of the client.
- ➔ The client receives the LACK in the RTCEClientUIInput class and does a similar closure.

**The Notion of ABORT:**

The abort message essentially in our implementation does the same things as the LOGOFF on the server side except sending a LACK back. This implies different semantics to ABORT. The client right now is capable of aborting from the READ_DOC and WRITE_DOC states but a client can also be implemented to ABORT during the Connection establishment phase as shown in the DFA to allow it ABORT on the basis of not being able to agree with the authentication of the server or for other reasons.

Also since our implementation is real time, if we happen to test it with multiple clients, a client might want to quickly ABORT without waiting for an Acknowledgment from the server.

---

Document: null

---

1:Once upon a midnight dreary, while I pondered, weak and weary, Over many a quaint and curious volume of forgotten lore

2:abc

3:'Tis some visitor,' I muttered, 'tapping at my chamber door- Only this and nothing more.'

4:Ah, distinctly I remember it was in the bleak December; And each separate dying ember wrought its ghost upon the floor.

5: Eagerly I wished the morrow;-vainly I had sought to borrow From my books surcease of sorrow-sorrow for the lost Lenore

6:For the rare and radiant maiden whom the angels name Lenore- Nameless here for evermore

Enter Command>>

ABORT **(ABORT)**

BUILD SUCCESSFUL

**********