Project Documentation

CSE 310

Wei Tang

Stephen Egert

**Overview**

As for now, we have implemented both stage one and stage two. Major implementation includes type, done, help, put, get, del, browse and exit. Please notice that the implementation of Stage 2, we have dropped all type fields in command.

Major Function:

*type [t]*

return the port number of the DNS server of t type and establish TCP connection between client and DNS server. Return not found if no DNS with given type.

*done*

close current TCP connection with DNS server and establish connection with manager application

*help*

print help message

*put [name] [IP addr]*

put command will put a DNS record with name field, ip address and the type assigned to this specific DNS server.

*get [name]*

get command will return the IP address of the given name, return not found if no record with given name is founded.

*del [name]*

del command will delete the record in the DNS server with the given name. return not found if no record with given name in database

*browse*

browse all the record in the database within the current DNS server.

*exit*

close TCP connection. Client application exits.

**User documentation**

Our implementation using runtime.exec() method to create multiple server with distinct type and with in a single DNS server, we use java thread class to handle concurrent client requests. Java thread class is also used in manager to serve multi clients at the same time. We never really test the limit for my project. In my test cases it supports 5 server at the same time.

To compile manager application (Server side):

*javac multiserver.java*

*javac manager.java*

(need to compile both java files)

To run manager application:

*java manager.java*

To compile and run client side code

*javac TCPClient.java*

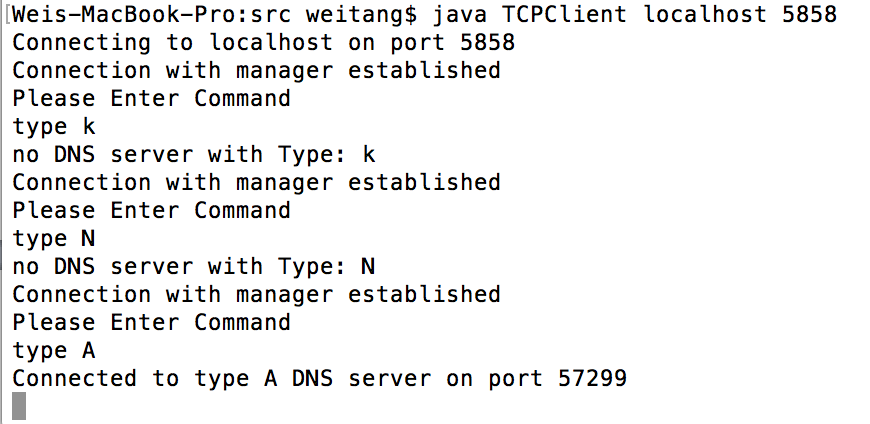
*java TCPClient <hostname> 5858*

(Our implementation of manager application is on port 5858)

Once the connection between manager and client established client will promote user to input command, at this stage only command help, type[type] and exit can be used (because now the client is connected to manager not a DNS server)

Syntax and parameters for my program:

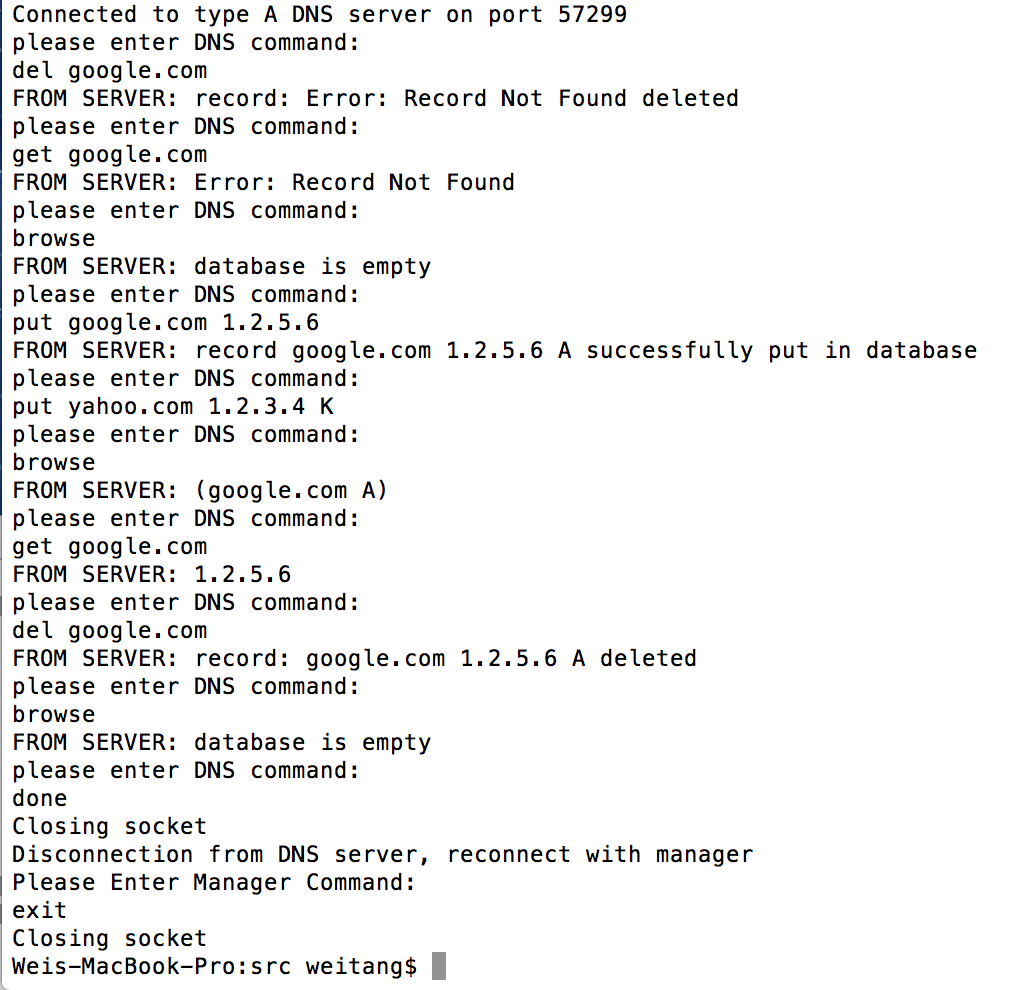
Example for connect to type N DNS server (with error cases):

**

Possible error message here would be “no DNS server with type found” or “unknown manager command” for wrong command

Once connection between a DNS and client is established then all other function is now available to user (browse, get, del, done and help)

Example for a sequence of command:



possible error message:

“Database empty” for empty database

“record not found” for no matching record in database for both get and del

notice that in the sequence put yahoo.com 1.2.3.4 K. this command does not put the record to the data base because we drop the type field in the command which make it an illegal command.

Once user enters done, connection between DNS and client will be closed and establish new connection to manager for using other DNS servers.

**System documentation**

Main Java Classes:

*Manager class*:

Manager application classes that parse the manager.io and launch corresponding DNS servers(in different processes), record their type and port number in an ArrayList and listen to client connection. Once connection is established pass the connection to ManagerServiceThread Class

*ManagerServiceThread Class:*

managerServiceThread class extends thread class. The thread will handle client requests.

*Multiserver class:*

Multiserver class is a DNS server with a specific type and a random assigned port number. It will pipe the port number using getLocalPort() method and pass it to the Manager Class for it to store service record. It will listen to the socket and pass socket connection to a ClientServiceThread class once it establish TCP connection with a client.

*ClientServiceThread class:*

ClientServiceThread class extends thread class. It will listen to clients commands and output the correct respond

Communication Protocol:

The communication between clients and manager or server is through a pipe.

Client can send message using writeUTF()

Server or manager can send message using readUTF()

All responses are print in terminal.

Data Structor:

We use different txt file for different DNS server. if the DNS server is type N then the database file for it would have the name of “N.txt”

The data record format : [name] [IP\_address] [type]

Error Conditions:

Error case most involve unknown command from client and data record not exist.

**Test Documentation:**

Test case 1: DNS server start up in manager class (create process in manager):

Reason: multi process worth 30 percent of the project

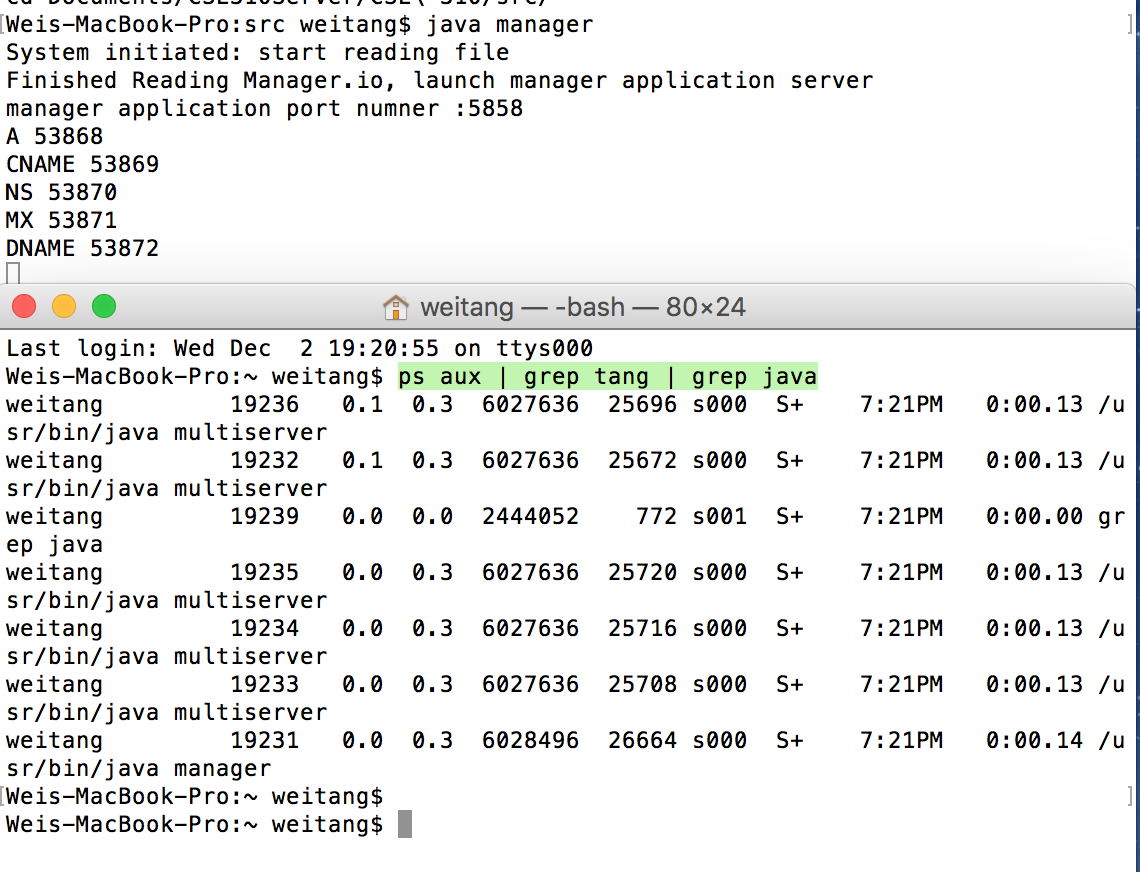
Precondition: all file compiled, manager.in file exists

Step: 1. Run “java manager”

2. ps aux | grep tang | grep java

Expect result: one manager process with several multiserver processes (depend on manager.in)

Actual result: 1 manager process with 5 multiserver processes



Test Case 2: TYPE command

Reason: functionality for manager and implementation of automatically connection to DNS server after client’s inquiry for a specific DNS server

Precondition: all file complied. Manager application launched

Step: 1. “java TCPClient <host address> 5858”

2. type K

3.type A

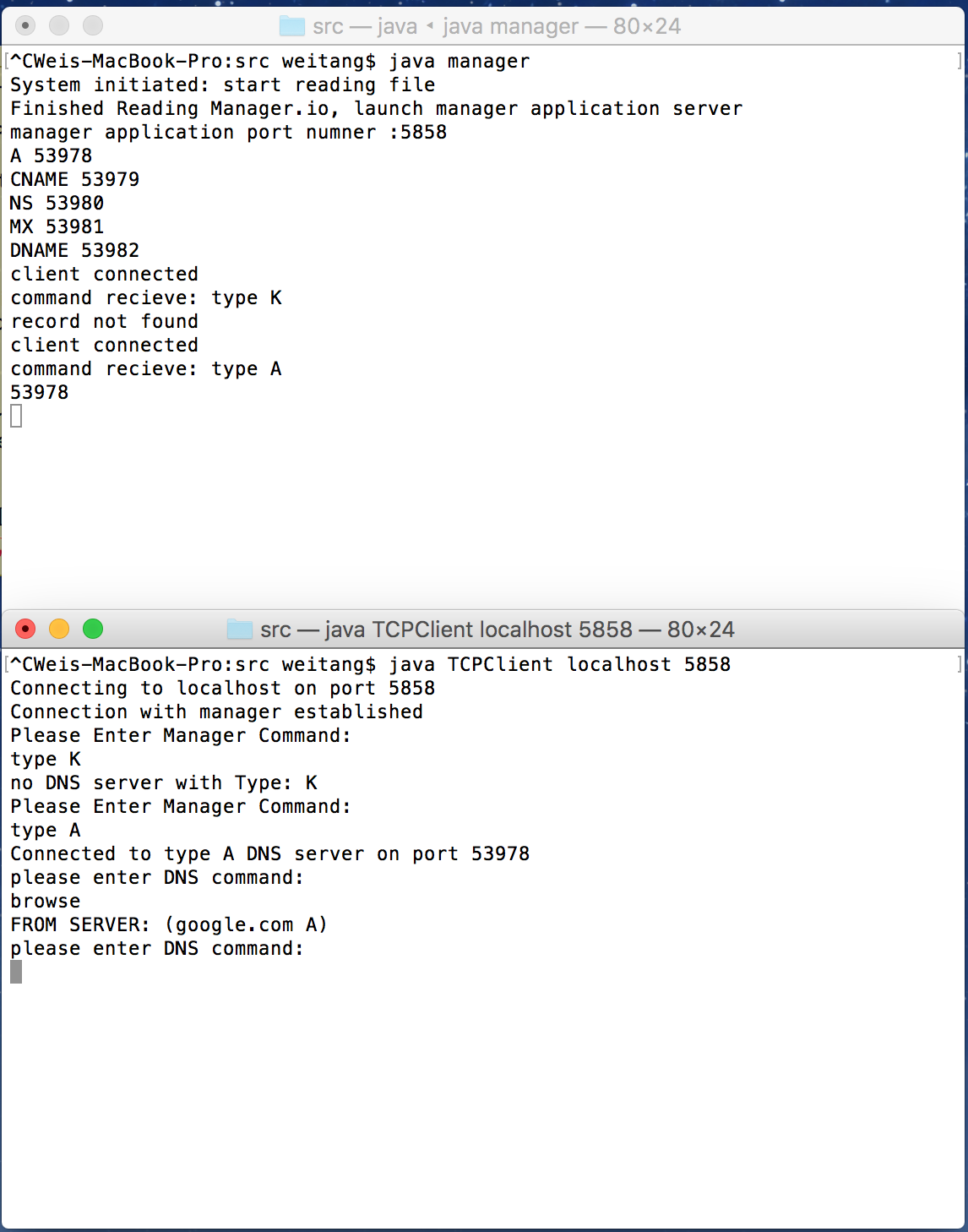
Expected result:

after 2: error message should returned

after 3: client should connect to type A DNS server and be able to use the DNS server

Actual result:

Work perfect see screenshots.



Test case 3: a sequence of put, get, del and browse command with multi clients

Reason: major function test and concurrent client test. It also test concurrency of manager.

Precondition: database empty, 2 clients connected to a DNS server

Step: 1. “browse” to make sure data base is empty

2. “put google.com 1.2.3.4” on client 1

3. “put yahoo.com 4.3.2.1” on client 2

4. “browse” on either client

5. “del yahoo.com” on either client

6. “get google.com” on either client

7. “get yahoo.com” on either client

8. “browse” on either client

Expect result:

After step 4. Terminal should show 2 record with correct information

After step 6. Terminal should show google.com’s IP address

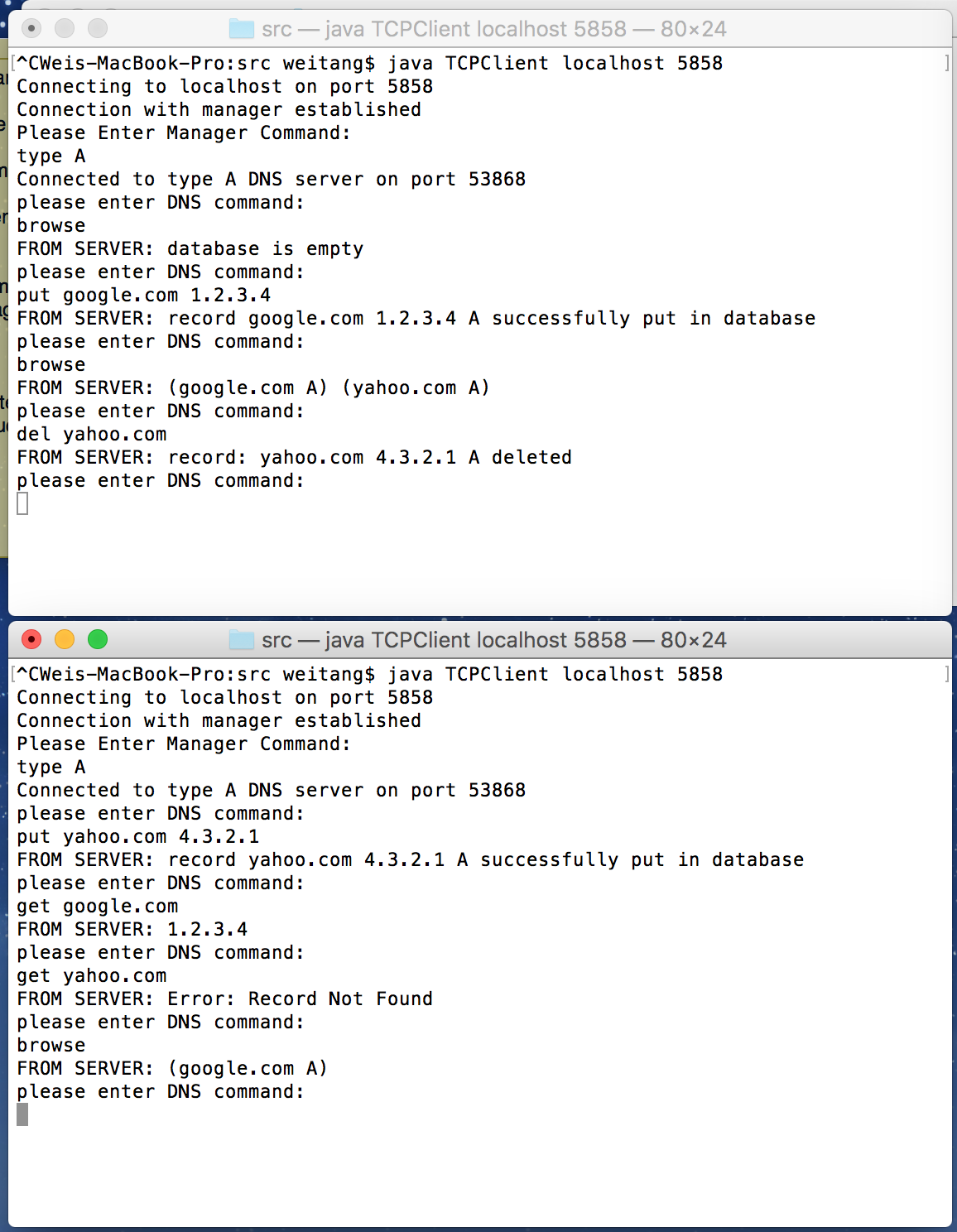
After step 7. Error case should return

After step 8. Only 1 record should shown

Actual result:

(next page screenshot)

work perfect



Test Case 4: done command and exit command

Reason: done should disconnect client from DNS server and reconnect it to manager

Exit should exit client application

Precondition: client is connect to manager.

Step: 1. “type A”

2. “done”

3. “type NS”

4. “exit”

Expected result:

After 2. Client should reconnect to manager

After 5. Client app should terminated.

Actual result:

Work perfect

