

Manual

Name: Ning Zhang

CWID: A20336916

Department of Computer Science, Illinois Institute of Technology

Step1. Firstly, you should enter the SourceCode directory. And entering the command 'ant compile' in terminal to use ant script which is called as 'build.xml' in the SourceCode directory to compile the code.

```
zns-MacBook-Pro:SourceCode zn$ ant compile
Buildfile: /Users/zn/Desktop/DistributedHashTable/SourceCode/build.xml

compile:
[javac] /Users/zn/Desktop/DistributedHashTable/SourceCode/build.xml:21: warning: 'includeantrun'
[javac] Compiling 7 source files to /Users/zn/Desktop/DistributedHashTable/SourceCode/target/classes
[javac] Note: /Users/zn/Desktop/DistributedHashTable/SourceCode/src/Parser.java uses unchecked or
[javac] Note: Recompile with -Xlint:unchecked for details.

BUILD SUCCESSFUL
Total time: 0 seconds
```

Step 2. Then entering the command 'ant makejar' in terminal to generate the .jar file.

```
zns-MacBook-Pro:SourceCode zn$ ant makejar
Buildfile: /Users/zn/Desktop/DistributedHashTable/SourceCode/build.xml

compile:
[javac] /Users/zn/Desktop/DistributedHashTable/SourceCode/build.xml:21: warning: 'includeantrun'
[javac] Compiling 7 source files to /Users/zn/Desktop/DistributedHashTable/SourceCode/target/classes
[javac] Note: /Users/zn/Desktop/DistributedHashTable/SourceCode/src/Parser.java uses unchecked or
[javac] Note: Recompile with -Xlint:unchecked for details.

makejar:
[jar] Building jar: /Users/zn/Desktop/DistributedHashTable/SourceCode/target/jar/napsterP2P.jar

BUILD SUCCESSFUL
Total time: 0 seconds
```

Step3. Then entering the command 'ant runserver' in terminal to run the DistributedHashTable class which contains servers and clients. We can see the outputs in the following screenshot.

```

zns-MacBook-Pro:SourceCode zn$ ant runserver
Buildfile: /Users/zn/Desktop/DistributedHashTable/SourceCode/build.xml

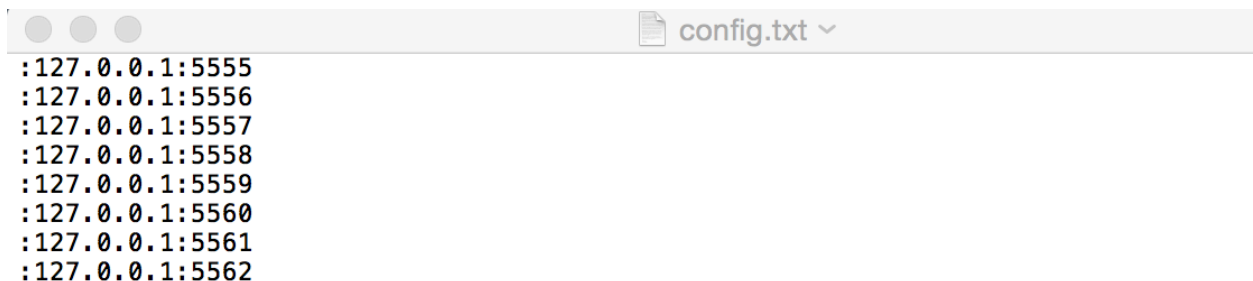
compile:
[javac] /Users/zn/Desktop/DistributedHashTable/SourceCode/build.xml:21: warning: 'inclu
[javac] Compiling 7 source files to /Users/zn/Desktop/DistributedHashTable/SourceCode/t
[javac] Note: /Users/zn/Desktop/DistributedHashTable/SourceCode/src/Parser.java uses un
[javac] Note: Recompile with -Xlint:unchecked for details.

makejar:
[jar] Building jar: /Users/zn/Desktop/DistributedHashTable/SourceCode/target/jar/naps

runserver:
[java] *****
[java] *          Peer Operation Command          *
[java] *
[java] * 1.PUT      (upload the key and value)      *
[java] * 2.GET      (download the value)           *
[java] * 3.DELETE   (delete the key)               *
[java] * 4.TEST     (test the system performance)  *
[java] * 5.EXIT     (exit the peer)                *
[java] *
[java] *****
[java] Port:

```

Step5.we must open the configure file named as “config.txt” in the directory “src” .



```

config.txt
:127.0.0.1:5555
:127.0.0.1:5556
:127.0.0.1:5557
:127.0.0.1:5558
:127.0.0.1:5559
:127.0.0.1:5560
:127.0.0.1:5561
:127.0.0.1:5562

```

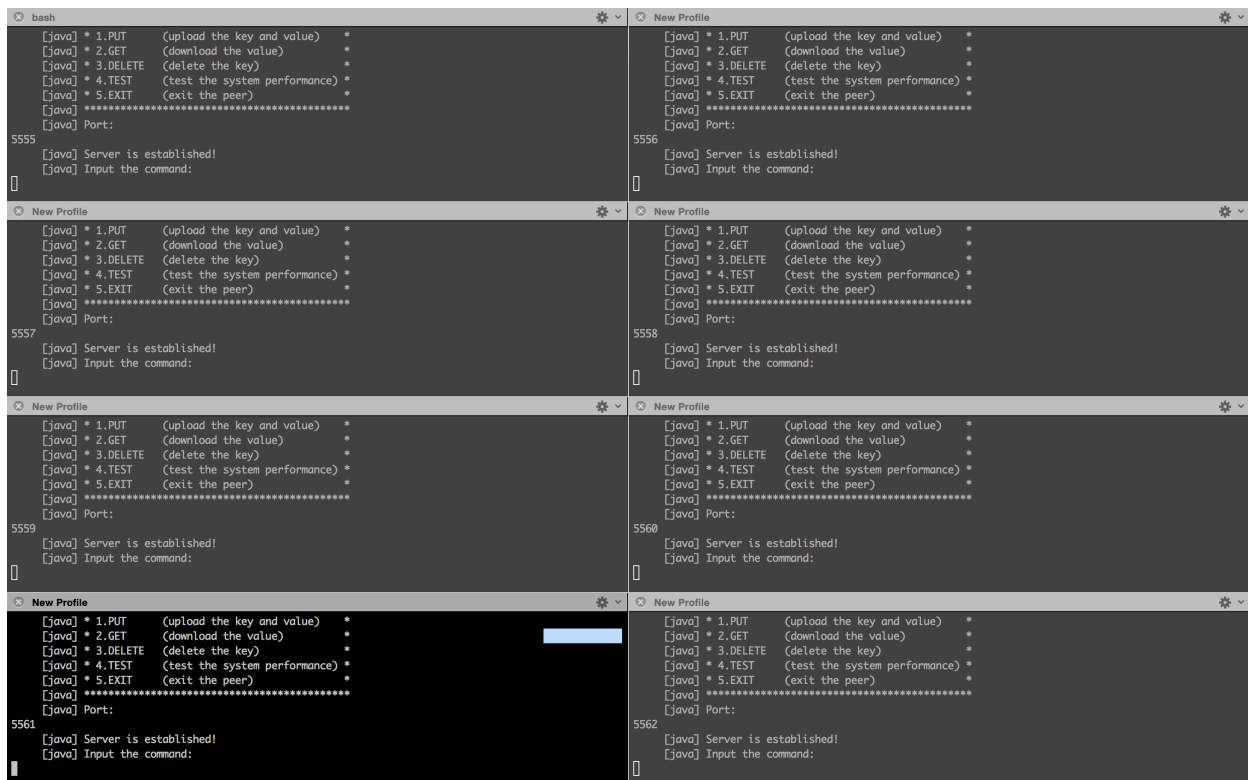
we can see that there are 8 sets including ip address and port number, which are the server ip and port. We must input the server port in the terminal.

Step6.run 8 servers and input the related port numbers.

```

runserver:
[java] *****
[java] *          Peer Operation Command          *
[java] *
[java] * 1.PUT      (upload the key and value)      *
[java] * 2.GET      (download the value)            *
[java] * 3.DELETE   (delete the key)                *
[java] * 4.TEST    (test the system performance)   *
[java] * 5.EXIT    (exit the peer)                 *
[java] *****
[java] Port:
5555
[java] Server is established!
[java] Input the command:

```



Step7.input the PUT command to put key and value into this distributed hash table

```

runserver:
[java] *****
[java] *          Peer Operation Command          *
[java] *
[java] * 1.PUT      (upload the key and value)      *
[java] * 2.GET      (download the value)           *
[java] * 3.DELETE   (delete the key)               *
[java] * 4.TEST    (test the system performance)  *
[java] * 5.EXIT    (exit the peer)                *
[java] *****
[java] Port:
5555
[java] Server is established!
[java] Input the command:
PUT
[java] key:

```

Then input the key.

```

runserver:
[java] *****
[java] *          Peer Operation Command          *
[java] *
[java] * 1.PUT      (upload the key and value)      *
[java] * 2.GET      (download the value)           *
[java] * 3.DELETE   (delete the key)               *
[java] * 4.TEST    (test the system performance)  *
[java] * 5.EXIT    (exit the peer)                *
[java] *****
[java] Port:
5555
[java] Server is established!
[java] Input the command:
PUT
[java] key:
file1
[java] value:

```

Then input the value.

```

[java] *          Peer Operation Command          *
[java] *
[java] * 1.PUT      (upload the key and value)      *
[java] * 2.GET      (download the value)           *
[java] * 3.DELETE   (delete the key)               *
[java] * 4.TEST     (test the system performance)  *
[java] * 5.EXIT     (exit the peer)                *
[java] *****
[java] Port:
5555
[java] Server is established!
[java] Input the command:
PUT
[java] key:
file1
[java] value:
CS550PA2
[java] Input the command:

```

In the end, complete the PUT operation.

Step8.input the GET command to get value from this distributed hash table

```

[java] * 1.PUT      (upload the key and value)      *
[java] * 2.GET      (download the value)           *
[java] * 3.DELETE   (delete the key)               *
[java] * 4.TEST     (test the system performance)  *
[java] * 5.EXIT     (exit the peer)                *
[java] *****
[java] Port:
5555
[java] Server is established!
[java] Input the command:
PUT
[java] key:
file1
[java] value:
CS550PA2
[java] Input the command:
GET
[java] key:

```

Then input the key.

```

[java] * 4.TEST      (test the system performance) *
[java] * 5.EXIT      (exit the peer)                *
[java] *****
[java] Port:
5555
[java] Server is established!
[java] Input the command:
PUT
[java] key:
file1
[java] value:
CS550PA2
[java] Input the command:
GET
[java] key:
file1
[java] the value is CS550PA2
[java] Input the command:

```

we can get the value of the key “file1”

Step9.input the DELETE command to delete the key from this distributed hash table

```

[java] *****
[java] Port:
5555
[java] Server is established!
[java] Input the command:
PUT
[java] key:
file1
[java] value:
CS550PA2
[java] Input the command:
GET
[java] key:
file1
[java] the value is CS550PA2
[java] Input the command:
DELETE
[java] key:

```

Then input the key you want to delete.

```

5555
    [java] Server is established!
    [java] Input the command:
PUT
    [java] key:
file1
    [java] value:
CS550PA2
    [java] Input the command:
GET
    [java] key:
file1
    [java] the value is CS550PA2
    [java] Input the command:
DELETE
    [java] key:
file1
    [java] Input the command:

```

The key is deleted. We must test if we delete this key. So we input the GET command.

```

file1
    [java] value:
CS550PA2
    [java] Input the command:
GET
    [java] key:
file1
    [java] the value is CS550PA2
    [java] Input the command:
DELETE
    [java] key:
file1
    [java] Input the command:
GET
    [java] key:
file1
    [java] the key doesn't exist!
    [java] Input the command:

```

We can see the key “file1” is deleted from this distributed hash table.

Step10.we input the TEST command to test the performance of the distributed hash table. The test includes 100K PUT commands, 100K GET commands and 100K DELETE commands.

```
file1
  [java] the value is CS550PA2
  [java] Input the command:
DELETE
  [java] key:
file1
  [java] Input the command:
GET
  [java] key:
file1
  [java] the key doesn't exist!
  [java] Input the command:
TEST
  [java] Start performance test, please be patient ...
  [java] put test takes: 63.96 us
  [java] get test takes: 55.739999999999995 us
  [java] delete test takes: 53.05 us
  [java] Input the command:
```

we can see the results of the test, the time is the average time of every 100K commands. One put operation can cost 63.96 us, one get operation can cost about 55.74 us and one delete operation can cost 53.05 us.