Report of A DTN Simulator - THE ONE

Xu Liu *and Yuanzhu Chen

Department of Computer Science
Memorial University of Newfoundland

May 11, 2013

1 Introduction

1.1 Background

THE ONE is one of DTN simulator in the world. It is written by Java. In this simulator, it offers researchers many source codes of famous DTN routing protocol. Based on these implementations of routing protocol, researchers can easily understand main logical of some DTN routing algorithms. However, after I studied THE ONE recently, I found that the documents and tutorial [2] related to THE ONE are not enough for a beginner to catch up. In order to help me easily recall my memory once I need to use THE ONE again in my later life, also, in order to record what I have learn from THE ONE, I start to write this report. I wish in my later life, based on this report, Others and I can easily configure THE ONE simulator and conduct experiments.

1.2 Outline

In this report, I cover three sections and I will cover more sections during my later study period. The purpose of these three sections are as follows:

- Any one who knows basic knowledge of programing or computer science can easily start to use THE ONE by following this report
- Any one can quickly get the main idea and design logical of THE ONE project code by using this report.
- Any one can learn THE ONE by themselves based on some materials and useful information that I given

The structure of this report are very simple. Firstly, I introduce how we install and configure our environment to debug THE ONE simulator. Secondly, I give some useful materials and information for us to self-learning. Thirdly, I briefly go through the running process of THE ONE and give some explanation of THE ONE project source code.

2 Install and Configure

2.1 Install

- Until I write this report, the newest version of THE ONE is version 1.4.1 which was published on 31.January 2011. It can be downloaded from `http://www.netlab.tkk.fi/tutkimus/dtn/theone/down/one_1.4.1.tar.gz`

*liuxu.mun@gmail.com

- At beginning of installing THE ONE, we need to download the source code of THE ONE from the link above in to a folder, Say "MyTheOne".
- After we finish downloading the source code of THE ONE, we go into the folder "MyTheOne" from "Terminal" (Let me suppose you use either Linux or Unix operation system). Then, type command "tar xvf one_1.4.1.tar.gz " to uncompress the package. Then, we can get a folder called "one_1.4.1" in our "MyTheOne" folder.
- After we finish uncompressing the package of THE ONE, we go to "one_1.4.1" folder and type command "./compile.bat" in order to compile the Java codes. If there are some problems happen during compiling. It is better to type command "sudo ./compile.bat" instead of typing command "./compile.bat". "sudo ./compile.bat" command will force our compiling processing under root priority. I use Mac OS to compile it. Therefore, the command is "sudo ./compile.bat". Different operation system may have different command to go to root priority.
- When we finish compiling the codes. We can find that the size of our "one_1.4.1" folder is bigger than we finish downloading it. The reason of this is that compiler of java create many classes based on the source code and automatically put them into the source folders (such as these folders: report, routing, core and so on). In these folders, we can see each .java file has a .class file.
- Now, all the basic steps have done. We go to folder "one_1.4.1" and type comannd "./one.sh". Then we will find that a window with some paths and nodes should popup. That means THE ONE successfully compiled and set up, we can start our personal research of DTN. If these are no window popup, it is more likely that we have no authority to run this file. We need to type command "chmod 777 one.sh" to make this file as a executable, then type command "./one.sh".

2.2 Configure

I can not deny that we can use editor "vi" or "nano" to change the source code and compile THE ONE to reach our personal study. But they are not efficient and it is hard for us to trace our code and middle process result. Therefore, I select an IDE "Eclipse IDE for Java Developer " which can be downloaded from `http://www.eclipse.org/downloads/` to debug and run THE ONE project. Also, we can select "Netbeans" which can be downloaded from `https://netbeans.org/community/releases/73/index.html`. But as my previous experience, "Netbeans" is more UI-friendly in development than "Eclipse", However, compared to "Eclipse", it costs too much computer resources and the running speed of it is slower than "Eclipse". Therefore, "Eclipse" is to be my final choice.

- After we finish downloading "Eclipse" from the link that I mentioned above. Then, we uncompress the "Eclipse" package to a folder. And run "Eclipse" application.
- After that, we click "File" menu of IDE "Eclipse", Then click "New" menu item in order to create a new java project. (Figure 1)

  The UI of the screenshot from Figure 1 may be different from yours and mine. Because I installed Android Plugin. So, that is why there is an Android Application Project shows in the menu. In this menu, we select "Java Project"
- After we finish select "Java Project", we will get a configuration window as Figure 2. Originally, "use default location" will be checked. But, we need to uncheck that option and navigate to our THE ONE project folder "MyTheOne/one_1.4.1" in order to import our THE ONE project. We can define the project name by ourselves. In this report, I define the project name as "the_one_1.4.1"
- When we finish configure our java project. Then, we click "Next" button and go to next page of configuration. In the next page of configuration, we select "libraries" tab. The aim of clicking "libraries" tab is to import two important "jar" for THE ONE to compile. When we are in the "libraries" tab, we click "add external jar" button. Then, there will be a "File Browser" window popup. We go to folder "MyTheOne/one_1.4.1/lib" and import DTNConsoleConnection.jar and ECLA.jar as Figure 3. After I looking into DTNConsoleConnection.jar file. It has two classes, the first one is DTNConsoleConnection.classes and the other one is DTNConsoleConnectionSinDrainer.classes. Because I can not reflect these classes to java source code. Also I can not find relative documents. I guess DTNConsoleConnection.jar offers a way to DTN simulator to use command to communicate with DTN simulated scenarios. Namely, we can manage and get information from console instead of only gaining information from output file. As

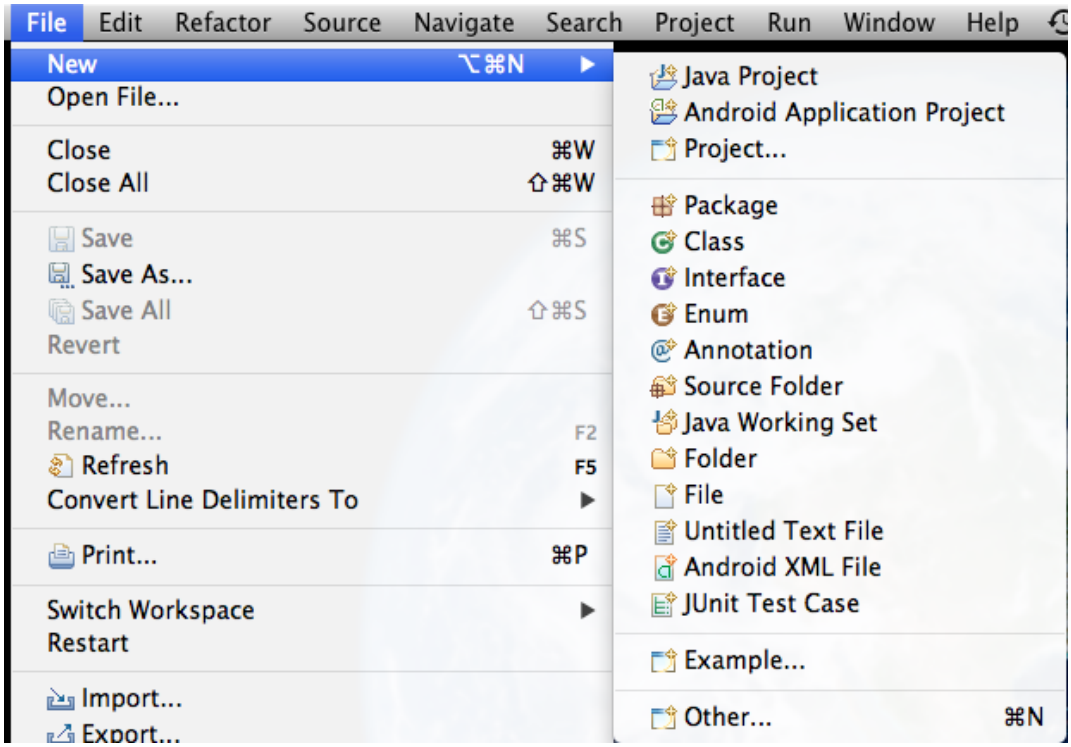

Figure 1: Create a new project from Eclipse

for ECLA.jar file, there are many classes related to bundle, interface and data structure parse. I guess this jar is useful to simulate Network Interface Hardware and it tries to finish the work on Link Layer. So that, THE ONE only need to deal with the situation on Network Layer and above layer. No matter how I guess, after I deeply study THE ONE, I will detail this part.

- Once we finish importing the two jars. Then we click "Finish" button and finish config THE ONE project in eclipse environment. At this time, we can see that our project are listed in Eclipse as Figure 4.

  But due to there is an error in "test" package. If we want to debug our application, we have to delete the "test" package in order avoid error (deleting this "test" package won't affect THE ONE's feature and functions). Then we can follow the process of Figure 5 to run application.

  "test" package is a test case for different functions and classes in THE ONE. Keeping it will help us to understand THE ONE logical and function. Therefore, it is recommended to solve these errors although removing it will not affect THE ONE. In order to solve this problem, we have to go to JUnit website to download its jar. The link is: `http://search.maven.org/remotecontent?filepath=junit/junit/4.11/junit-4.11.jar`. After we finish download this jar. we right click our "the_one_1.4.1" project in "Package Explorer Window". Then click "properties" in the menu. after that, we go to "Java Build Path" tab and then select "Libraries" tab (default situation "Libraries" tab is selected). Then click "add external jar" and import the junit jar that downloaded by us. Once we import this jar. all the errors will disappear. When the errors disappears, we can follow the steps in Figure 5 to run THE ONE simulator.

3 Self-learning for THE ONE

From what I know that the documents and tutorials for THE ONE are not enough for us to self-learning. Therefore, collecting a serials of ways and materials to learn THE ONE seems to be very important. Some useful materials and

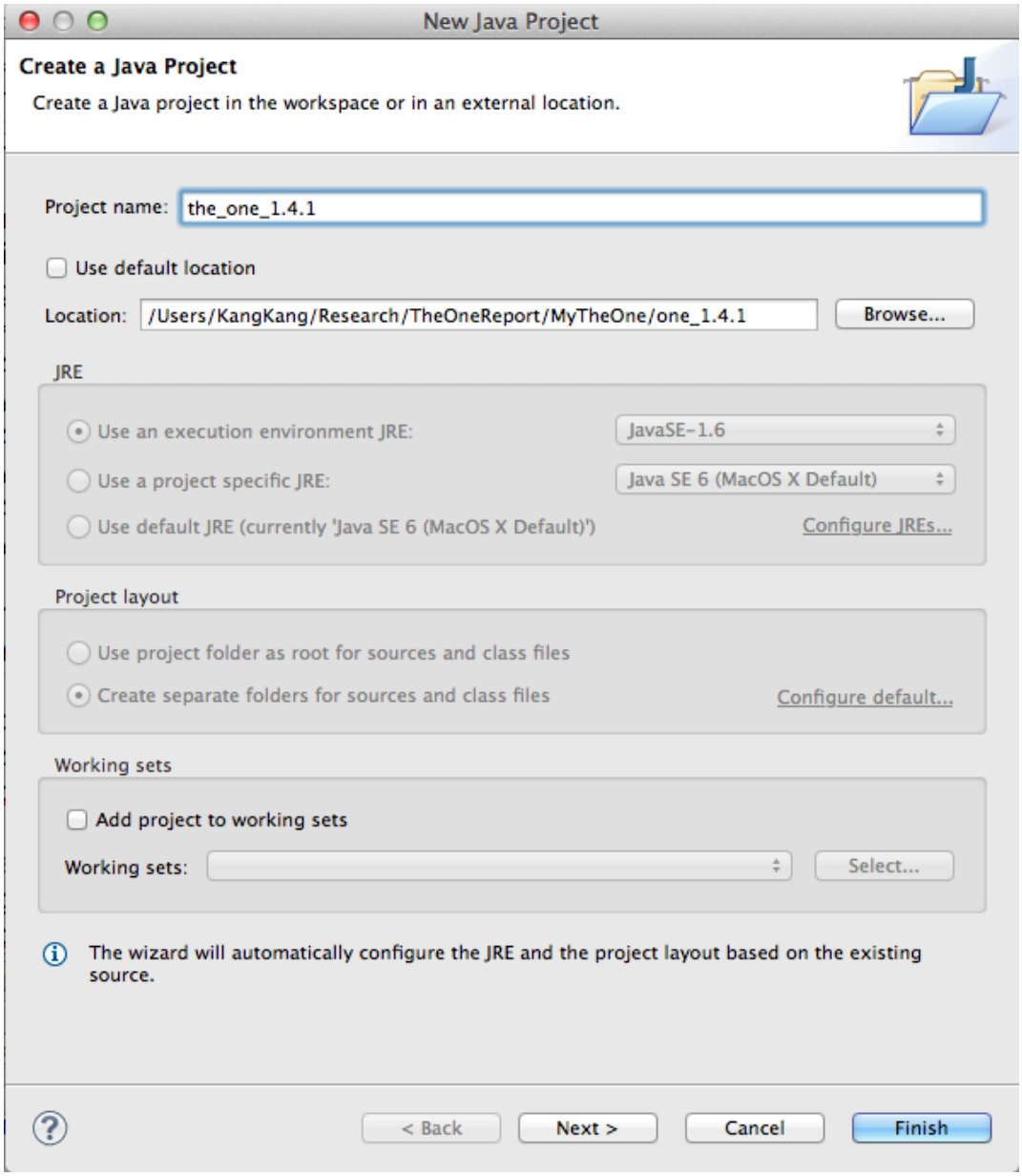

Figure 2: Java Project Configuration

information related to learn THE ONE are listed as below:

- The first website I have to mention is the official THE ONE website. The link is `http://www.netlab.tkk.fi/tutkimus/dtn/theone/`. In this website, we can download the newest version of THE ONE. But from the updating information, It seems that THE ONE was not updated for about 2 years since 2011, January 31st. In this official website, it gives us some introduction of how compile THE ONE project. But it does not detailedly mention how to set up and configure Eclipse for running THE ONE. The person who want to use Eclipse to debug THE ONE require experience of using Eclipse before.
- The most useful website and tutorial for THE ONE is `http://delay-tolerant-networks.blogspot.ca/p/one-tutorial.html`. However as for this website, it only teaches us how to configure a DTN network scenario instead of giving us some example like modifying a routing algorithm or improve a routing algorithm and so on. It also does not

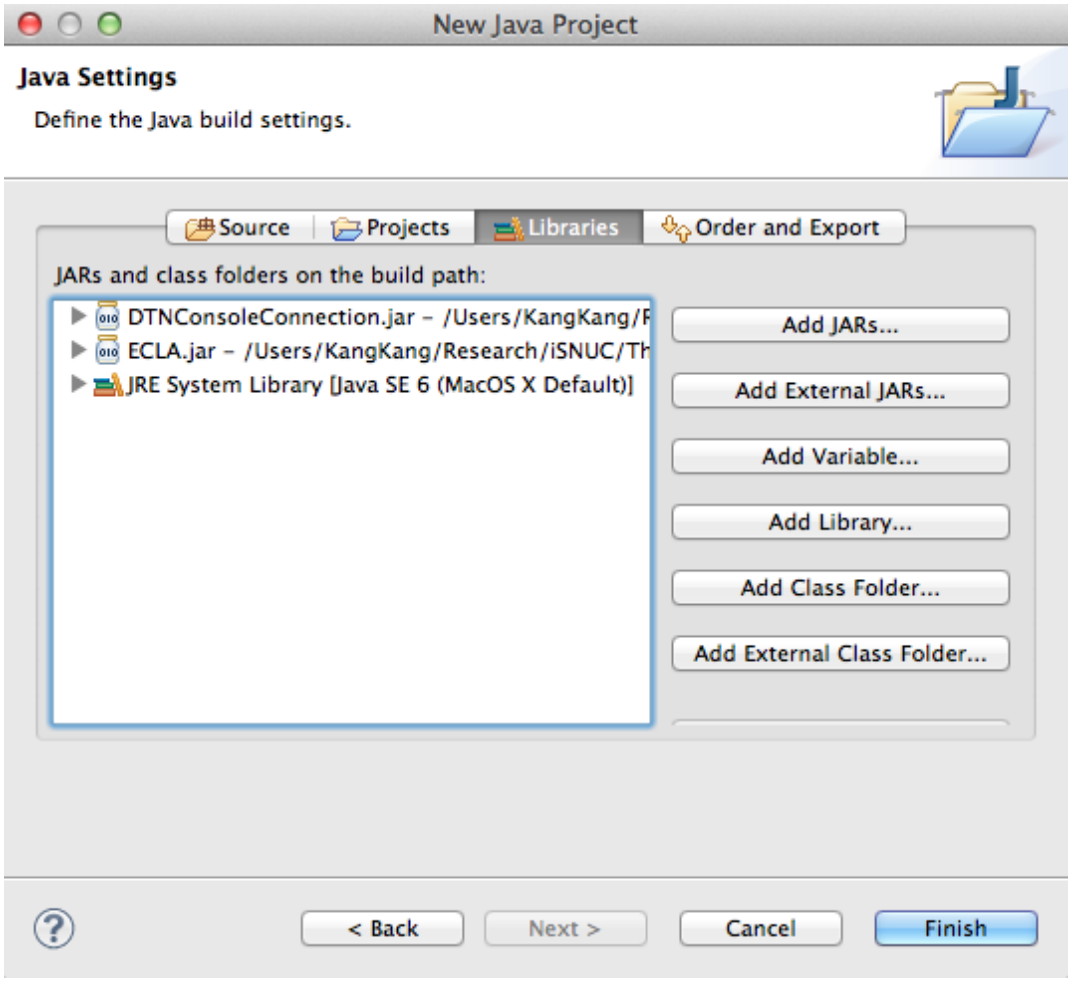

Figure 3: Import Libraries

introduce the whole THE ONE project codes to us. But no matter how it is not detailed. At least, we can find this website and learn some knowledge about THE ONE.

- The other useful materials originally exists in THE ONE project code. We need to go to the folder "MyTheOne/one_1.4.1" and then go to folder "doc". We can see that there is a shell script there. We need to type command "./create_doc.sh" to generate the THE ONE documents from THE ONE project source code. Once we finish "./create_doc.sh" process, we can find that there are a plenty of html files in "doc" folder. Theses files are the API manuals of THE ONE project. Although these files contain many information related to THE ONE. However, they are only APIs. We can not read the whole API and work out our experiments based on THE ONE simulator. Therefore, these documents are not good for us to learn THE ONE. It only can be a manual for us when we want to look for some declaration of functions.

- The most high efficient way is that we can put ourselves into THE ONE questions mail list. If we are in this list, we can receive questions related to THE ONE from other researchers. We can go to the link: `https://www.netlab.tkk.fi/mailman/listinfo/theone` to register our email address and receive daily THE ONE questions. Based on other's questions, it is useful for us to solve our problems. Also, we can give our problems to this email list and ask other professional researchers to help us solve our problem. But sometimes, the answers are not detailed. They are just like a brief guidance. It requires some backgrounds related to THE ONE to understand what they are talking about. Therefore, this way is not good for a beginner of THE ONE learner.

- There are several papers about "THE ONE" which were listed in "THE ONE" official website `http://www.netlab.`

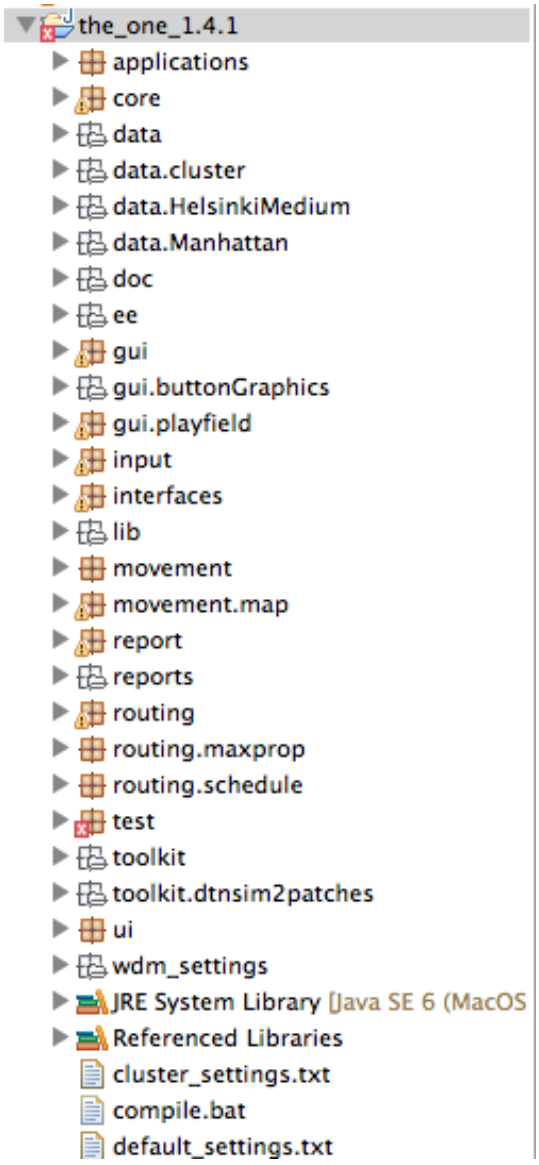

Figure 4: THE ONE files list

`tkk.fi/tutkimus/dtn/theone/`. They introduce some features and logical designs of THE ONE. If we want to know some structures of "THE ONE". It is valuable to read these papers. The paper [1] is a very good one for us to understand THE ONE simulator. However, these papers, to some extend, are a little bit abstract for a beginner to hold the main idea.

- As for tutorial video of THE ONE, until now, I didn't find any useful video from Internet. There are three videos related to THE ONE on `http://www.youtube.com` (By searching keywords: The One, The One DTN, The One DTN Simulator). They are too short. They even have no value for us to watch.

4 Running Process of THE ONE

Due to the deficiency of my knowledge related to THE ONE, more or less, I have some misunderstandings in my report. If there are some problems or vital mistakes, please point it out to me, I will correct it as soon as possible. In order to

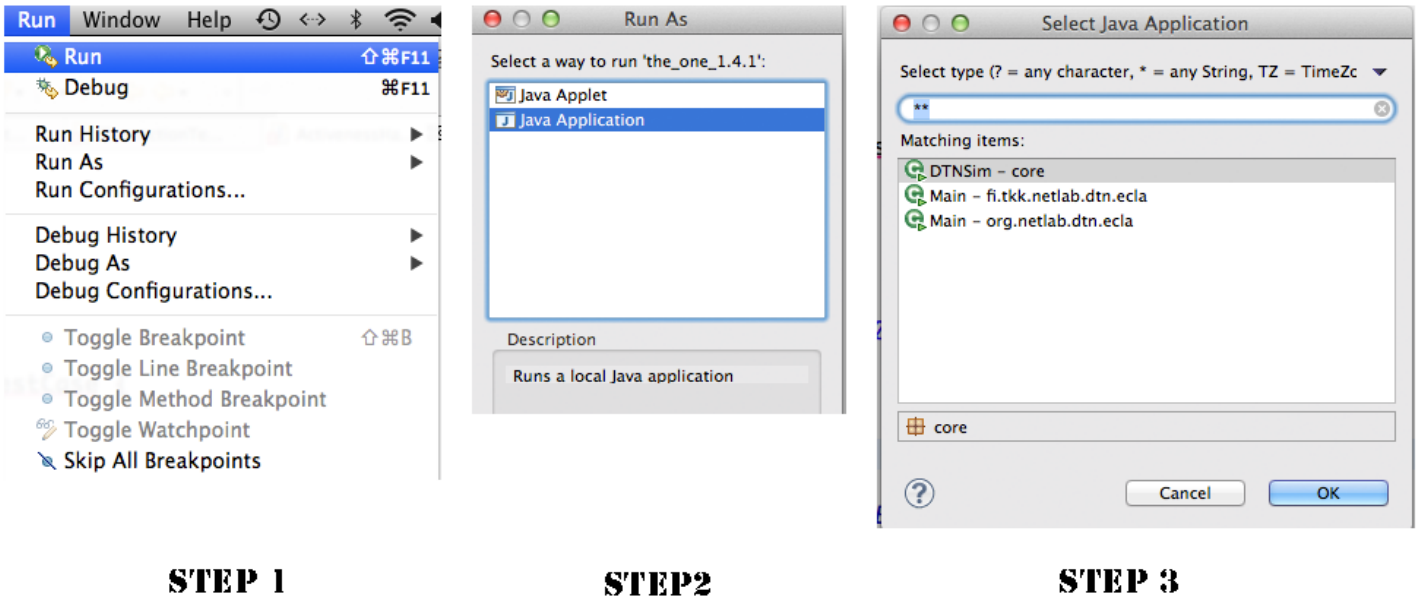

Figure 5: Run THE ONE project

deeply understand THE ONE. It is good for us to know the running process before we go deeply into THE ONE project codes. Specifically, after I trace the code of THE ONE, I found that there are five important steps for THE ONE to work.

- Firstly, THE ONE offers us two modes to run this simulator. The first one is based on UI (When we run our application in Figure 5 through Eclipse) and the second is based on terminal, namely print out commands and alert information in terminal without user interface (in THE ONE, it called batch mode). In this report, I mainly focus on UI mode (In fact, when we go through the code, batch mode and UI mode are 99% similar). Once, we start our THE ONE application. "DTNSim.java" in "core" package will be first called. Because "main" function is here.
- Secondly, "DTNSim.java" will call function "initSettings". This function will read values from "default_settings.txt" files if we don't set up a configuration file. In "default_settings.txt", there are parameters for creating the DTN network scenarios. such as how many nodes are in DTN network, what's the range of the transmission of each node and so on.
- Afterwards, based on the parameters, "DTNSim.java" will call "DTNSimGUI.java" to initial the scenarios and simulate the DTN network. DTNSimGUI is a subclass from DTNSimUI. We can find the definition of each function from DTNSimUI. Once DTNSimGUI start. It will frequently update the scenarios (SimScenarios.java manage the scenarios). When the scenarios are frequently updated by THE ONE. The nodes in this scenarios will move.
- At the same time, once there are two nodes meet each other in the scenarios and they can connect to each other. based on the settings (range, category of network interface and so on) from the configuration file. ActiveRouter.java or PassiveRouter.java will be called. Both of the two router is a parent class of routing. If we want to implement our routing algorithm. we need to inherit one of these two routing class. Then, implement the functions from either ActiveRouter.java or PassiveRouter.java. From the sample of THE ONE, it uses "EpedemicRouter" which was inherited from ActiveRouter.java as an example.
- Finally, we can see that ActiveRouter. java is inherited from MessageRouter.java. MessageRouter.java has responsibilities to manage the buffer that save and manage the messages during the process of the node moving. From the code of MessageRouter.java, it is clear to see that, it uses a fixed buffer to save messages. If I want to implement

iSNUC idea (a set of idea to manage buffer, iSNUC suggests use a non-fix buffer to save message) in it. I should modify this class. Finally, THE ONE finish a DTN network simulation.

5 Future Work

In the later future, I plan to continue learning "THE ONE" and try to understand the work flow deeply. After that, I will try to create a case to help learn THE ONE simulator. At the same time, I will give some experiments data from THE ONE simulator.

6 Conclusion

THE ONE is a good DTN simulator. Not only it can simulate DTN environment. But also it includes many famous DTN routing algorithm. From the code, I can easily find out how some DTN routing algorithms work. Simultaneously, THE ONE project has a very good design pattern. It is very flexible and highly extendable. It is a good example for programmer to design project.

References

[1] Ari Keranen. Opportunistic network environment simulator. May 2008.

[2] Barun Saha. The one - dtn simulator tutorial. `http://delay-tolerant-networks.blogspot.ca/p/one-tutorial.html`. Last accessed April 12nd, 2013.