

# **Design Document**

## **Network Toolkit Application**

**COMP208 Group 29**

Group Members: Minhao Jin, Pengcheng Jin, Weihao Jin,  
Yaokang Li, Shuheng Mo, Jinyao Xu

## **Contents**

<b>1. Summary of proposal .....</b>	<b>4</b>
<b>1.1 Background .....</b>	<b>4</b>
<b>1.2 Objectives .....</b>	<b>5</b>
<b>1.3 Changes to specification .....</b>	<b>5</b>
 <b>2. Design .....</b>	<b>6</b>
<b>2.1 Anticipated Components of the System .....</b>	<b>6</b>
<b>2.1.1 Network Components .....</b>	<b>6</b>
<b>2.1.2 Map Components .....</b>	<b>7</b>
<b>2.1.3 UI Components .....</b>	<b>7</b>
 <b>2.2 Data Storage .....</b>	<b>9</b>
<b>2.3 Object-Oriented Design .....</b>	<b>10</b>
<b>2.3.1 Use Case Diagram .....</b>	<b>10</b>
<b>2.3.2 Interaction chart .....</b>	<b>14</b>
<b>2.3.3 Class Diagram .....</b>	<b>18</b>

<b>2.4</b>	<b>Interface Design</b>	<b>19</b>
2.4.1	internal interface	19
2.4.2	external interface	19
<b>2.5</b>	<b>User Interface Design</b>	<b>22</b>
2.5.1	Target User	22
2.5.2	The Main Idea of Screen Design	22
2.5.3	Interaction style	22
2.5.4	Organization and Layout of Screens	23
2.5.5	The Design of Icon and Theme Color	24
2.5.6	Animation Design	25
2.5.7	User Interface Prototyping	26
<b>2.6</b>	<b>Algorithm Design and implementation method of core functions</b>	<b>29</b>
2.6.1	Android ‘Traceroute’ Implementation Method and Algorithm	29
2.6.2	Implementation Method for UI	30
2.6.3	Implementation Method for Animation on Map	32
2.6.4	Implementation Method for Scanning Devices under LAN	33
2.6.5	Implementation Method for Sketching Topological Graph	34
<b>2.7</b>	<b>Network Protocol</b>	<b>34</b>
<b>2.8</b>	<b>Evaluation Design</b>	<b>35</b>
2.8.1	Evaluation Methodology for User Interface	35

2.8.2	Test Form .....	36
2.8.3	Test Design .....	36
3.	Review Against Plan .....	37
4.	Reference.....	38
5.	Signature.....	38

# 1. Summary of proposal

## 1.1 Background

This design document will include all the design of our network toolkit application project and demonstrate the motivations and effectiveness of design components. As the start of the document, this section will introduce some relevant background information about functions that will be provided in our application:

- IP address geolocation: This is basically the geographical location of a IP address in the real world.
- Ping: Ping is a computer network administration software utility used to test the reachability of a host on an IP network. It sends messages to the designated IP address and obtains the information from the returned ICMP message (in ICMP packet) to test the reachability of a host [1].
- Traceroute: Traceroute is a computer network diagnostic command for displaying route and measuring transmission delays of packets in an IP network. This command was widely used by network developers to diagnose exact network error [2].
- LAN and net state: LAN stands for local area network. Local area network refers to a network that connects computers in a limited area such as campus, corporation, labs, or office rooms together. Network technologies such as Ethernet and WIFI are well-known LAN technologies in modern network development [3]. Net state is a cluster of information about devices under the same LAN. It includes information on connection between each network layer, router tables, network interface detail, and network protocol statistics. (Precisely, IP addresses, router information, hostname of devices connected to routers, etc.)

As this application was implemented to provide users with utilitarian network functions, implementation of network command Ping and traceroute was indeed significant. Network will work on sending commute messages among servers, third-party APIs and user's devices (If user's phone was not connected to network this application should return an error message).

## 1.2 Objectives

- To develop an Android network toolkit application.
- To develop a toolkit application that allows users to use basic network diagnostic functions Ping, traceroute and searching IP geolocation.
- To develop a toolkit application that contains a LAN manager to manage all the information of devices which connected to the same router. The information includes IP address, MAC address, hostname, devices' name, etc.
- To develop a toolkit application that will be able to provide network topological graph of the family network to the user.

## 1.3 Changes to specification

A new network tool called 'LAN Manager' will be added based on the original specification.

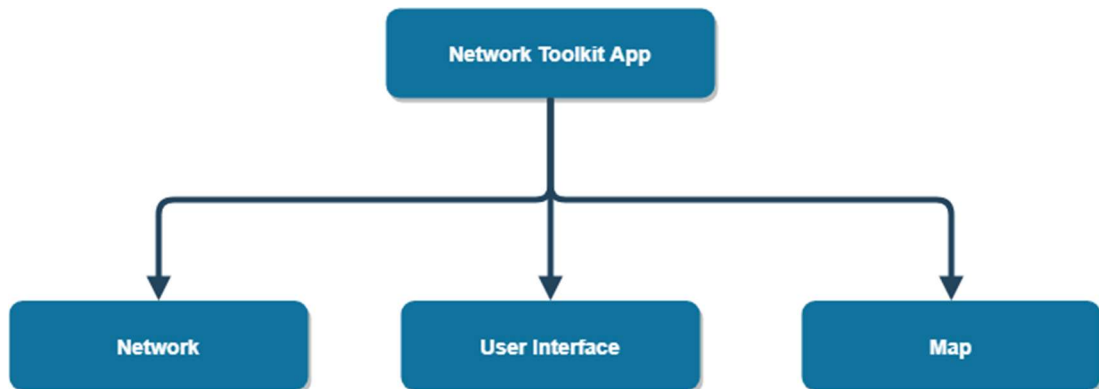
Functional requirement for 'LAN Manager':

If the android device is connected to the network,

1. This tool should automatically find and return the IP address of the router which it connects to.
2. This tool should display all the IP and MAC addresses of devices under the currently connected router.
3. This tool should be able to generate a picture to sketch the topological structure of the current LAN.
4. If this tool is used under a private network, it should provide the user with a list of devices it detected and allow the user to confirm which of the devices belong to him/her.
5. If a device is confirmed by the user, then it is registered. This tool should record and store the device's name, IP address and MAC address in the register table.
6. If an unregistered device connected to user's private network, this tool should send a warning dialog to the user to inform that a strange device connected to his/her network.

## 2. Design

### 2.1 Anticipated Components of the System



**Figure 1: Anticipated components of system**

#### 2.1.1 Network Components

- **Ping**  
As this application was designed as a network toolkit, component Ping will work as part of the network diagnosis functions. By sending packets to destination host and receiving ICMP message, Ping checks the connection quality between clients and servers and returns the necessary information. (e.g. Packet lost rate, TTL and so on.) In this project, Ping will be used to retrieve information about the hosts on trace path and therefore contribute to the implementation of traceroute on Android platform. Also, this component will be used to Ping (or broadcasting) all devices under the same subnet to retrieve information about these devices (IP address, hostname, etc.).
- **System method to lookup ARP table**  
ARP stands for Address Resolution Protocol. It works as a cache to store MAC address and IP address and mapping MAC address to corresponding IP address under the same LAN. When routers (other similar hosts) sending messages to destination IP, information about this IP address will be stored in ARP table. If this router wants to reach the same IP address, it will lookup it in its own ARP table and therefore to save time for broadcasting. In this project, ARP table provides information about the IP/MAC address of the hosts which used to visit this router. Users log family member's devices in the application, then this application will check whether un-logged devices have connected to this router and then report an alert.
- **HTTP GET**  
GET method is the core method of HTTP protocol, it requests a page (could be in HTML format) from server [4]. In this application, HTTP GET method will be used to obtain the content on the web page from the third-party API and convert them into required representation. Functions

based on the HTTP GET method will then extract useful information and pass it to those methods which handle IP geolocation function.

### 2.1.2 Map Components

- **Google map basic drawing:**  
Google map basic drawing is one of the core components of visualizing IP address. With the help of this component, the application will be able to display a basic view of a world map and the geolocation of an IP address can be pointed out on the map based on given latitude and longitude.
- **Line drawing:**  
To get better visual effects than the default google map, the android animator will be used to draw dynamic polyline on google map. With the support of the animator, lines will have 'directions', flowing from the start point to the endpoint.

### 2.1.3 UI Components

#### Input Controls

Element	Description	Use case
EditText	A text file for user to enter text. By 'get()' method, the program could get its content.	<ul style="list-style-type: none"> <li>• Get IP/Domain name</li> <li>• Get package number for Ping tool</li> </ul>
RadioButton	It's a set of selections for user to select one option	<ul style="list-style-type: none"> <li>• For user to choose to authorize the selected device or not</li> </ul>

#### Navigational Components

Element	Description	Use case
ImageButton	An icon implemented by a button with an image. Users could click the image as clicking a standard button.	<ul style="list-style-type: none"> <li>• The diagnose tools</li> <li>• The 'clear button' in input box</li> <li>• The 'back button' for users to back to previous activity</li> <li>• The 'enter button' for users to send the input to program and display the diagnosis result in text.</li> <li>• The 'map button' for user to send the input to program and display the diagnosis result visually in google map</li> </ul>
ImageView	It's a UI element for developer to display image resources. In this program, it could make UI more attractive and easier to use.	<ul style="list-style-type: none"> <li>• Display an image of 'magnifier' on the left side of every input box.</li> </ul>
Switch	With this button, users could change the state between	<ul style="list-style-type: none"> <li>• Change the registration state of devices in lan manager tool</li> </ul>

	two options by clicking or dragging the toggle	
--	--	--

### Information Components

Element	Description	Use case
Notification	It could remind users with messages outside the application's UI	<ul style="list-style-type: none"> <li>• Remind users of unregistered device's connection.</li> </ul>
TextView	An element to display text to user	<ul style="list-style-type: none"> <li>• Display the icon's name under an image button.</li> <li>• Display the text of diagnose results.</li> </ul>
Dialog	The dialog is a window which could be designed to display additional information or ask users to decide.	<ul style="list-style-type: none"> <li>• Display the warning dialog if users input invalid IP, domain name or package number.</li> </ul>
Toast	This view could provide feedback with a popup after the user's operation.	<ul style="list-style-type: none"> <li>• Display the user's IP with its geolocation information</li> </ul>

### Layout

Categories	Description
LinearLayout	A layout which could arrange views horizontally or vertically
FrameLayout	A layout designed to display child view(s) in a blocked area.
ConstraintLayout	This layout allows developers to place and resize widgets flexibly.



## 2.2 Data Storage

### 2.2.1 Internal storage

Since no database design was required for this project, the data storage method for this application will be internal storage (or app-specific storage). External storage is not chosen because it needs the support of SD card. Therefore, to ensure our application can run on mobile phones with or without SD card, we choose to save the file in internal storage.

### 2.2.2 Write to file

- Saving file directory: "data/data/packageName/file.txt" on mobile phone
- Saving content: Each device's attributes will be recorded in one line and split by semicolon: "IP;MAC;HostName;Brand". For instance, "192.168.0.1;40:5a:c9:bb:6d:db; My iPhone; Apple". Therefore, if n devices' information should be saved, a n-lined text will be generated correspondingly, and the text will be saved in the file.
- Overwriting: for every writing operation, the whole file will be overwritten.

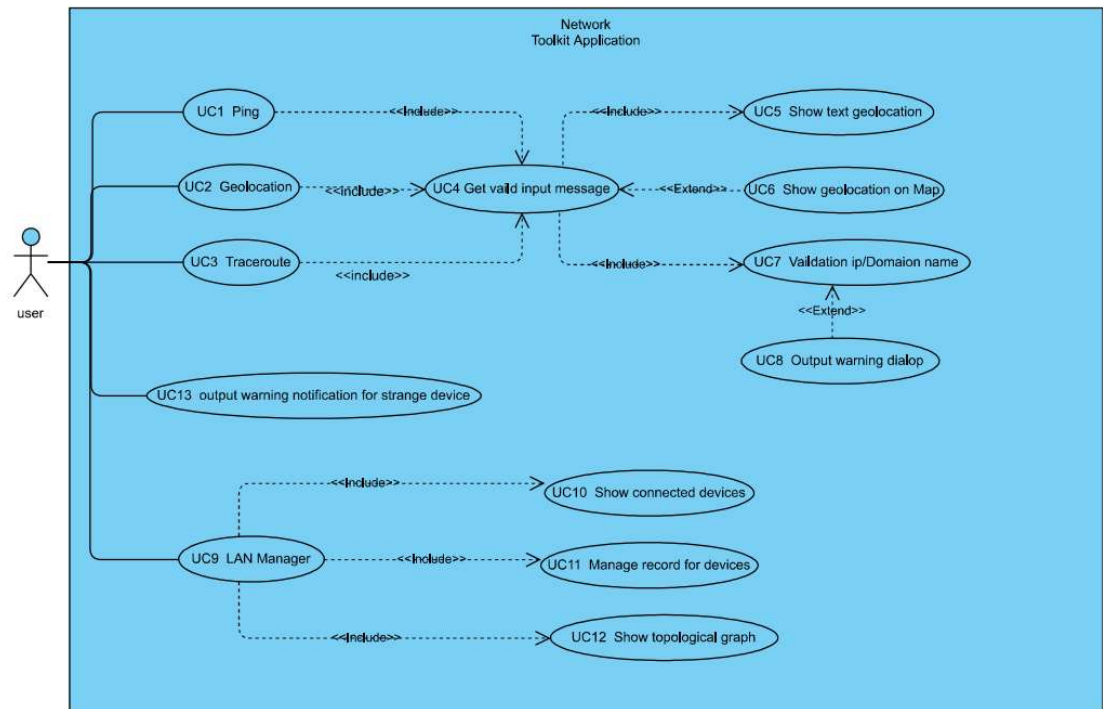
### 2.2.3 Read from file

- Reading file directory: "data/data/packageName/file.txt" on mobile phone
- Reading method: The text file will be read line by line. After one line comes in, it will be split by semicolon into 4 parts, which are IP address, MAC address, Host Name and Brand Name, respectively. In this case, this line "192.168.0.1;40:5a:c9:bb:6d:db; My iPhone; Apple " will be regarded as the table below.

IP address	MAC address	Host Name	Brand Name
192.168.0.1	40:5a:c9:bb:6d:db	My iPhone	Apple

## 2.3 Object-Oriented Design

### 2.3.1 Use Case Diagram



**Figure 2: Use-case diagram**

#### Description:

ID	UC1
Name	Ping
Description	The user selects Ping button to get into functions for Ping
Pre-condition	Application in the menu interface
Event flow	1. Jump to the interface of inputting information for Ping 2.Include UC4 3.Return to the menu interface
Extension points	
Triggers	Ping button is clicked
Post-condition	Two IP objects are created Google Map is created

ID	UC2
Name	Geolocation
Description	User selects Geolocation button to get into functions for Geolocation
Pre-condition	Application in the menu interface
Event flow	1. Jump to the interface of inputting information for Geolocation 2.Include UC4 3.Return to menu interface
Extension points	
Triggers	Geolocation button been clicked
Post-condition	An IP object is created Google Map is created

ID	UC3
Name	Traceroute
Description	The user presses Traceroute button to get into functions for Traceroute
Pre-condition	Application in the menu interface
Event flow	1. Jump to the interface of inputting information for Traceroute 2.Include UC4 3.Return to menu interface
Extension points	
Triggers	Traceroute button been clicked
Post-condition	Several IP objects are created Google Map is created

ID	UC4
Name	Get valid input message
Description	Get the required message corresponding to the function user selects from the user's input
Pre-condition	Application in the interface for inputting information
Event flow	1.Get the IP/Domain name user enters 2.Get the number of packages to be sent user enters if Ping is clicked 3.Include UC7 4.Include UC5
Extension points	UC6
Triggers	Text field for inputting been clicked
Post-condition	IP objects are created Google Map is created

ID	UC5
Name	Show text geolocation
Description	Present the information contained in IP objects to the user in text
Pre-condition	Application in the interface for inputting information
Event flow	1. Get information from IP objects 2. Show text information in the format corresponding to the function clicked
Extension points	
Triggers	IP objects are created
Post-condition	Text information showed in the interface for inputting information

ID	UC6
Name	Show geolocation on Map
Description	Present a Google Map generated by IP objects to the user
Pre-condition	Application in the interface for inputting information
Event flow	1.Get information from IP objects 2.Draw markers on Google Map 3.Draw lines between markers if number of IP objects is larger than 1
Extension points	
Triggers	Map button been clicked

Post-condition	Application in the map interface Google Map is created
----------------	---

ID	UC7
Name	Validation IP/Domain name
Description	Check the format of input correct or not
Pre-condition	Application in the interface for inputting information
Event flow	1.Check the format of input 2.Use all the information got from the user to generate the result of the corresponding function clicked 3.Using the result to generate IP objects to contain all the information
Extension points	UC8
Triggers	The IP/Domain name has been input
Post-condition	IP objects are created

ID	UC8
Name	Output warning dialog
Description	Give the user a warning if the input format is incorrect
Pre-condition	Application in the interface for inputting information
Event flow	1. Present a warning in the interface 2.Clear the input text field 3.Let user re-enter message
Extension points	
Triggers	Input format is incorrect
Post-condition	A warning appears in the interface

ID	UC9
Name	LAN Manager
Description	The user presses LAN Manager button to get into functions for LAN Manager
Pre-condition	Application in the menu interface
Event flow	1. Jump to the interface for LAN Manager 2.Include UC10 3.Include UC11 4.Include UC12 5.Return to menu interface
Extension points	
Triggers	LAN Manager button is clicked
Post-condition	The application will jump to the LAN Manager tool's interface

ID	UC10
Name	Show connected devices
Description	When it comes to the LAN Manager tool, all the IP and MAC address of devices under the current LAN will be shown
Pre-condition	Application in the LAN Manager interface
Event flow	1. There is a chunk in a line for one connected device to show its information 2. Click the chunk, the user could see more details

Extension points	
Triggers	
Post-condition	The connected devices are shown with their MAC and IP address

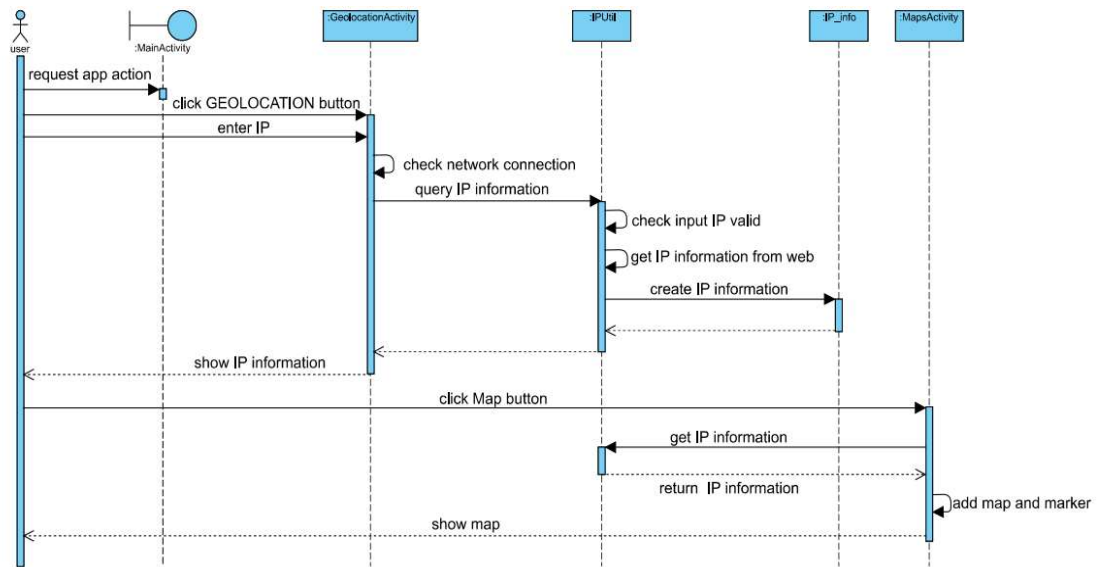
ID	UC11
Name	Manage record for devices
Description	The user selects one device's chunk to choose to authorize it or not
Pre-condition	Application in the LAN Manager interface
Event flow	1. User clicks one of the device items 2. There will be a selection window for the user to choose to authorize it 3. Save the record
Extension points	
Triggers	One of the devices been clicked
Post-condition	The record for authorization will be updated

ID	UC12
Name	Show topological structure
Description	The user presses 'show structure' button to get the topological structure of the current LAN
Pre-condition	Application in the LAN Manager interface
Event flow	1. User clicks the 'show structure' button 2. The tool will generate a picture to illustrate the structure 3. Close the picture
Extension points	
Triggers	'Show structure' button been clicked
Post-condition	The picture of the topological structure is shown

ID	UC13
Name	Output warning dialog for a strange device
Description	If an unregistered device connects to the network, there will be a notification to remind the user
Pre-condition	The application is running in the background
Event flow	A popup notification will remind the user with this device's information
Extension points	
Triggers	An unregistered device connects to the network
Post-condition	

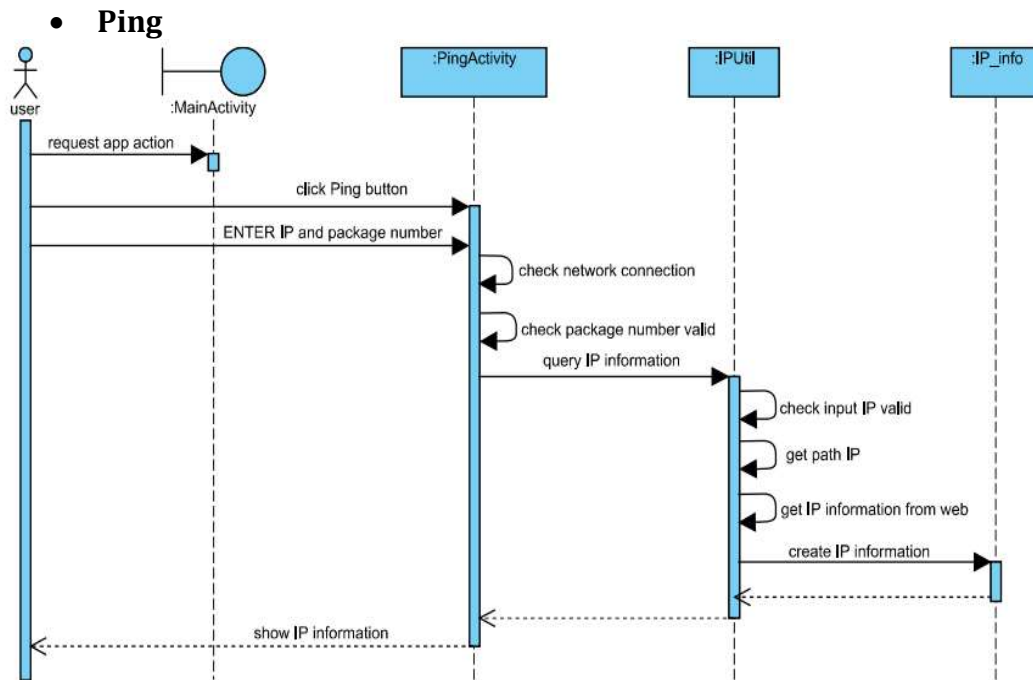
### 2.3.2 Interaction chart

- Geolocation



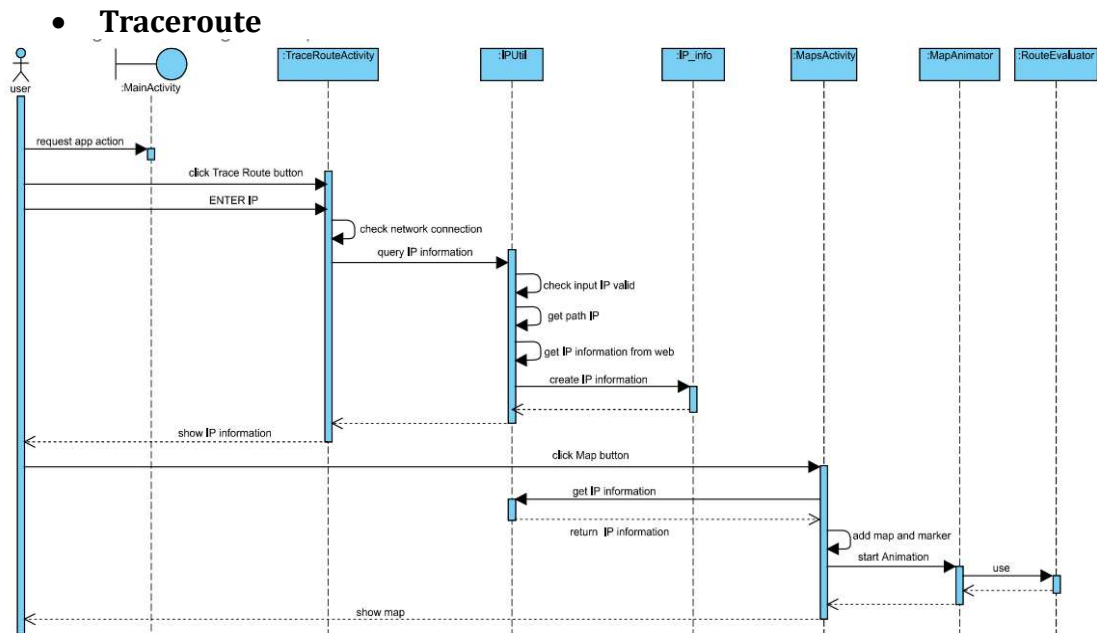
**Figure 3: IP geolocation function**

1. After the user open this application, *MainActivity* is invoked.
2. Then user clicks 'GEOLOCATION' button and enters IP address in *GeolocationActivity*.
3. *GeolocationActivity* will check this device whether connecting the internet.
4. The input IP address will be sent from *GeolocationActivity* class to *IPUtil* class.
5. *IPUtil* class will check IP validation, and find detailed information (city, country, continent, latitude, longitude) for this IP from the third-party API.
6. Then *IPUtil* class will use *IP\_info* class as constructor to store IP information.
7. *GeolocationActivity* class will get IP information, and show it to user.
8. Then if user clicks Map button, Map class will be invoked and get IP information from *IPUtil* class to add markers on map using latitude and longitude.
9. Finally, the map will be shown to user.



**Figure 4: Ping function**

1. After the user open this application, *MainActivity* is invoked.
2. Then user clicks Ping button, enter IP address and packet number in *PingActivity*s.
3. *PingActivity*s will check this device whether connecting the internet.
4. The input IP address will be sent from *PingActivity* class to *IPUtil* class.
5. *IPUtil* class will check IP validation, get path IP and retrieve detailed information (city, country, continent, latitude, longitude) for this IP from the third-party API.
6. Then *IPUtil* class will use *IP\_info* class as constructor to store IP information.
7. *PingActivity* class will get IP information, and show it to user.

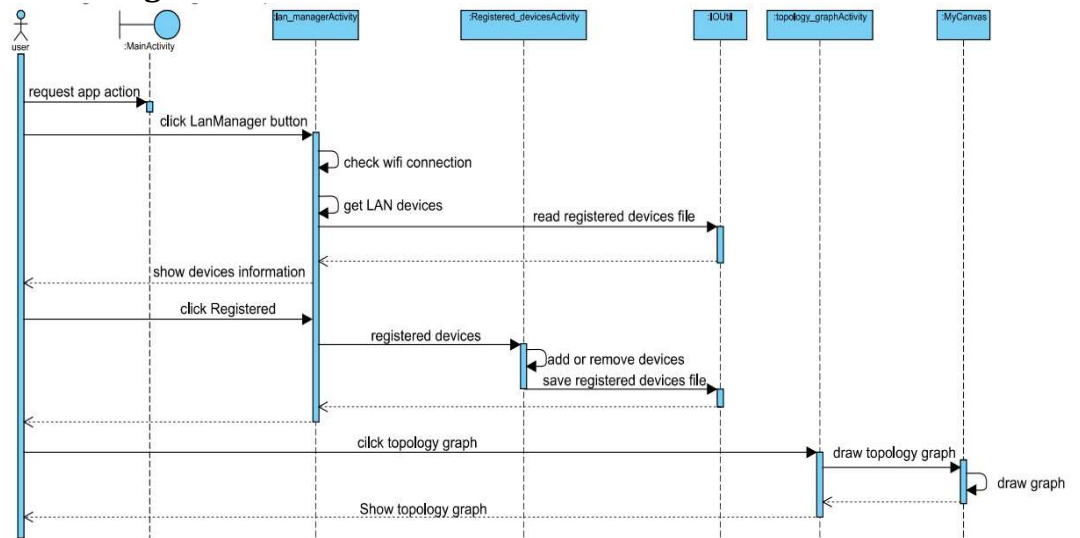


**Figure 5: Traceroute function**

1. After the user open this application, *MainActivity* is invoked.
2. Then user clicks Traceroute button, enter IP address into *TraceRouteActivity*.
3. *TraceRouteActivity* will check this device whether connecting the internet.
4. The input IP address will be sent from *TraceRouteActivity* to *IPUtil* class.
5. *IPUtil* class will check IP validation, get path IP and find detail information (city, country, continent, latitude, longitude) for this IP for this IP from the third-party API.
6. Then *IPUtil* class will use *IP\_info* class as constructor to store IP information.
7. *TraceRouteActivity* class will get IP information, and show it to user.
8. Then if user clicks Map button, *Map* class will be invoked and get IP information from *IPUtil* class to add markers on map using latitude and longitude.
9. Animation line will be added on Map between different marker, so Map class will call *MapAnimator* class to draw it.
10. *MapAnimator* class will use *RouteEvaluator* class to compute distance between each animation drawing point and each marker.
11. Finally, the map will be shown to user.



- **LAN manager**



**Figure 6: LAN manager function**

1. After the user open this application, *MainActivity* is invoked.
2. Then user clicks Lan Manager button to *lan\_managerActivity* class.
3. *lan\_managerActivity* class will check WIFI connection in this device, then queries devices information under the same LAN.
4. *lan\_managerActivity* class will through *IOUtil* class to read registered devices file. Then compare with queries devices information under the same LAN to show information together to user.
5. If user choose one device and click Registered button, *lan\_managerActivity* class will get this information and send to *Registered\_devicesActivity* class.
6. *Registered\_devicesActivity* class will add or remove this device in application, then save it by *IOUtil* class to file.
7. If user click topology graph button, *topology\_graphActivity* class will get this information, then send it to *MyCanvas* class to draw topology class.
8. Finally, topology graph will show to user.

## 2.3.3 Class Diagram

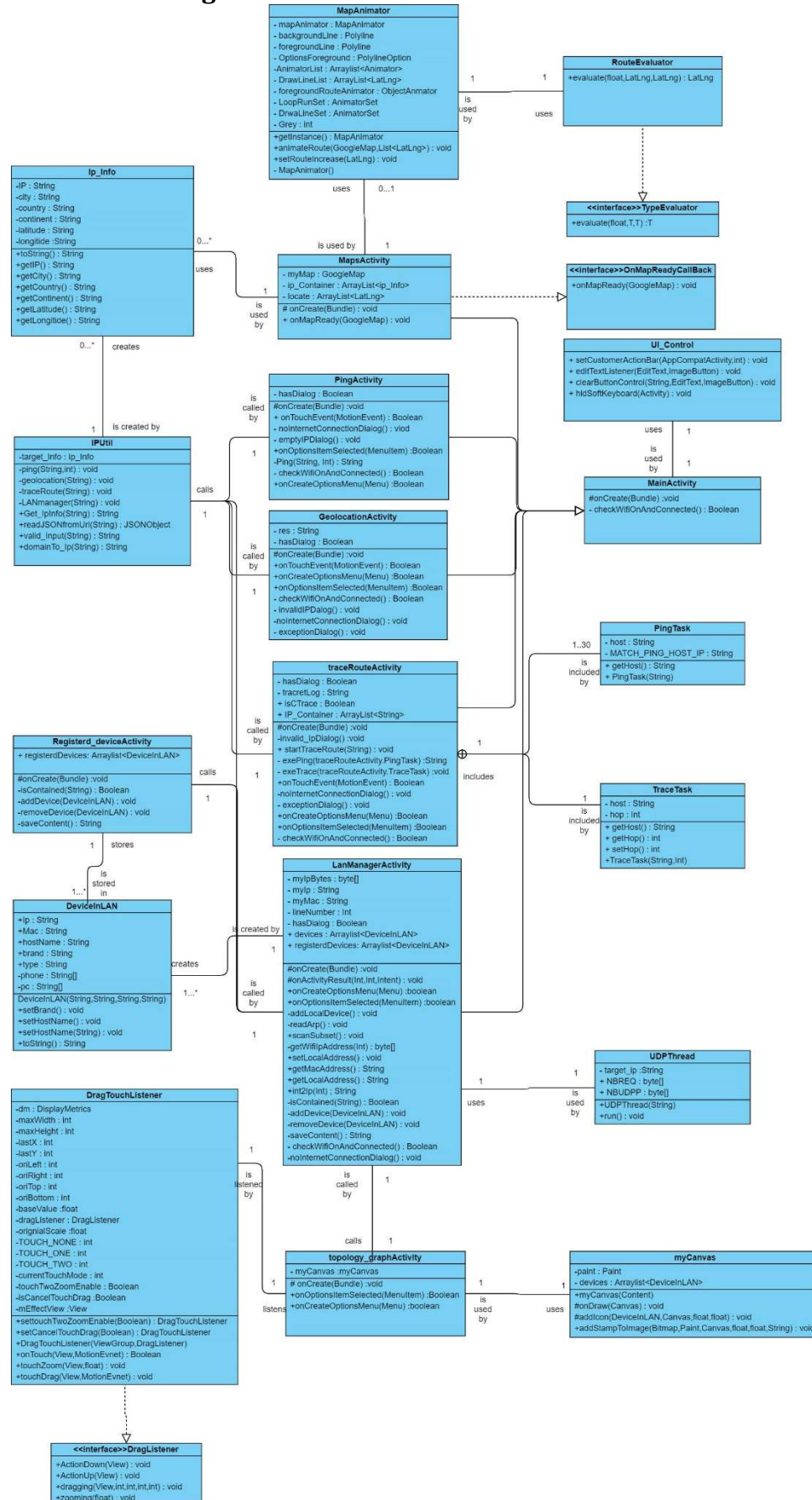


Figure 7: Class diagram

## Description

Our project will start at the main activity which can four methods: Ping, *TraceRoute*, Geolocation, and *LANManager* by user's selection. The first three methods have its own activity to collect the input message from the user and send them to *IPUtil*.

In *IPUtil* class, the validation of all the input message will be checked first. If false, the project will back to activity class and display the warning message. Otherwise, the corresponding method will be called to create objects containing specific information to be displayed as a result by processing the input message in the method. Moreover, *Ip\_Info* will also be used by Map to display a more intuitive result on *GoogleMap*.

In Map class, *OnMapReadyCallBack* Interface is implemented and the *onMapReady* method is overridden to create our Google map. Besides the foundational method provided in *GoogleMap* API, the *MapAnimator* class is used to add animation on the map. When adding animation, the *RouteEvaluator* class is used which implements *TypeEvaluator* and overrides the evaluation method to realize the animation of the drawing line on the map.

The last method: *LANManager* will use methods in *UDPThread* class to scan the LAN to get required information and pass them to *DeviceInLAN* class to create an array to store them as devices. Those devices which have been favor by the user will be passed to *registerd\_devicesActivity* to store those temporary devices in our application. The *LANManger* class can also create a topology graph of the devices got from LAN by using methods in *topology\_GraphActivity* class. After getting the information passed by *LANManger* class, the *topology\_GraphActivity* class will use methods in *myCanvas* class to draw a corresponding topology graph. Then a listener created by *DragTouchListener* class which implements the *DragListener* interface will apply to that graph to realize the effect of dragging and zooming.

## 2.4 Interface Design

### 2.4.1 internal interface

There are two internal Interfaces used in our project.

- The first one is '*OnMapReadyCallBack*' which is an essential component of building a Google Map. All of the drawings on Google Map should be overridden in the '*OnMapReady*' method in this Interface.
- The second Interface is '*TypeEvaluator*' which is used in the animator class to define the way of the input value of animator changes. In our project, we override the 'evaluate' method in this Interface by changing the input element type from T to *LatLng* and defining a calculation formula to realize the animation of drawing a line from the start point to the end point.

### 2.4.2 external interface

- **Google map API**

The basic map scene and adding marker behavior in the map could be supported by Google map android API. API key credentials will be got from the Google Cloud Platform Console and then add in 'google\_maps\_api.xml'. Finally, through instance, Google map will embed to our application.

- **IP geolocation API**

Valid information about a IP address is critical for this application's IP geolocation function. To obtain the required details about the IP addresses,

third-party API (<https://extreme-ip-lookup.com/>) will be called. This API works to retrieve legal details about an IP address and return it in JSON format. The application will then extract useful information out of JSON and display it to the user.

Browsing <https://extreme-ip-lookup.com/> to call this API's function. This is a legal and free API which developers can write URL as 'extreme-ip-lookup.com/json/IP\_address' to get JSON data about the IP address of the host.

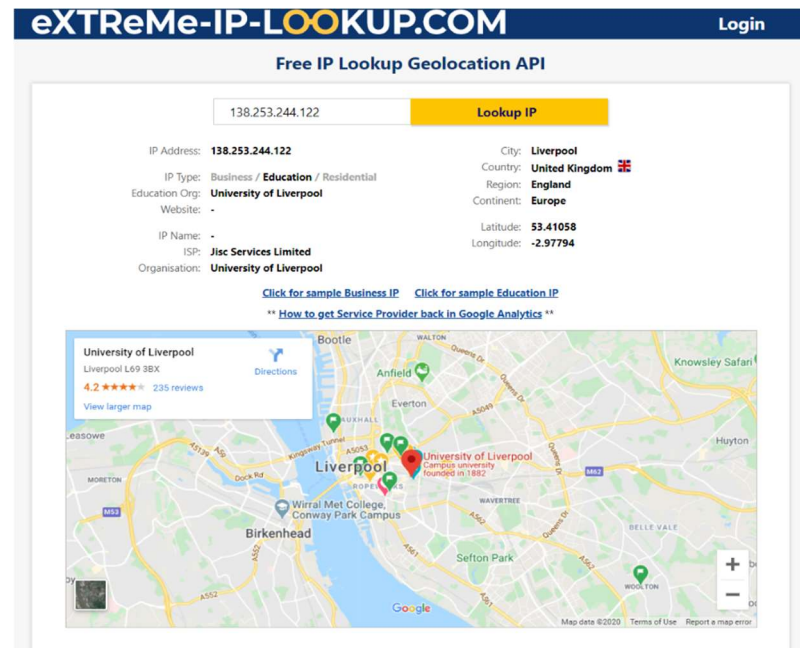


Figure 8: eXTReMe-IP-LOOKUP.COM screenshot [5]

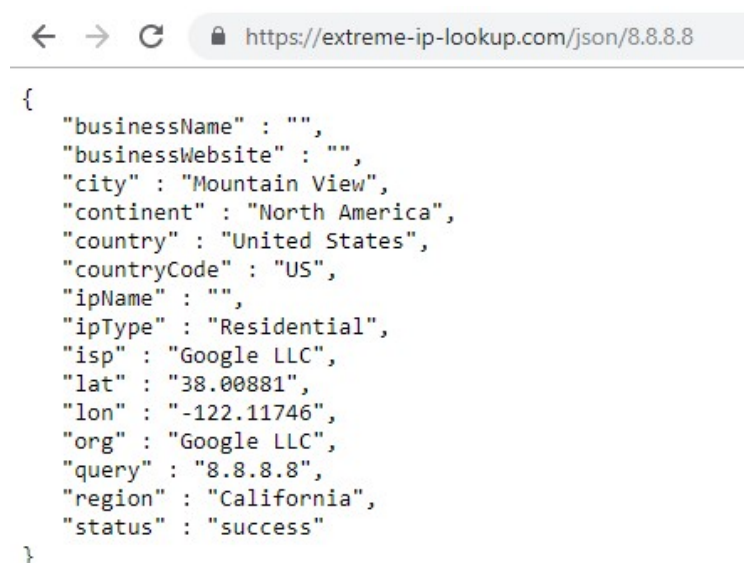


Figure 9: JSON data about IP of the host [5]

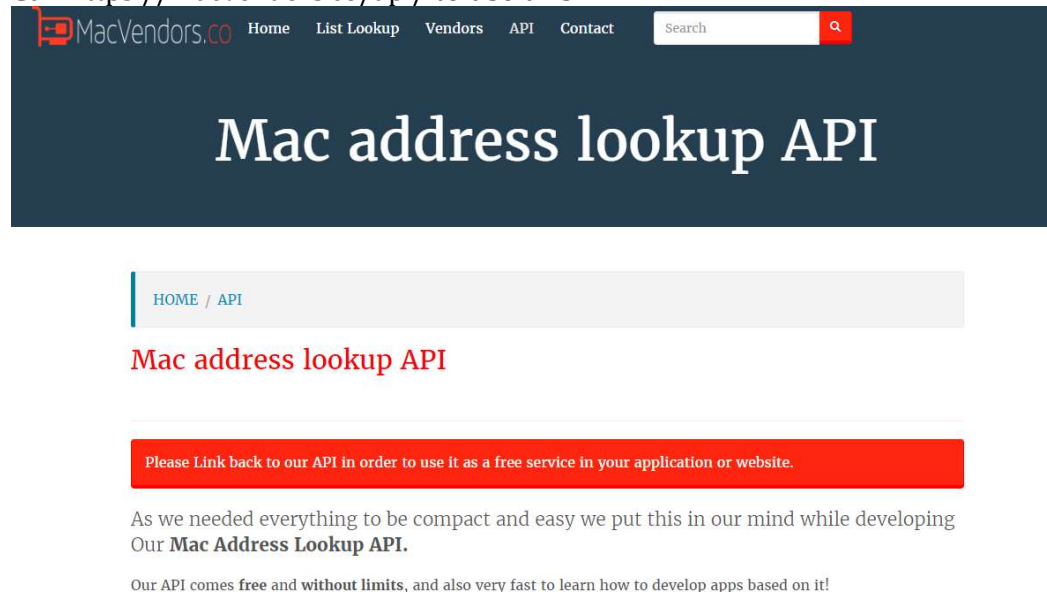
For instance, enter Google's server IP 8.8.8.8 to figure out the IP geolocation of

this server. By entering [extreme-ip-lookup.com/json/8.8.8.8](http://extreme-ip-lookup.com/json/8.8.8.8) we will be able to get JSON format information about this IP address as demonstrated in Figure above. In the following development, we will use HTTP and several sub-functions to obtain the required information such as 'lat' (latitude) and 'lon' (longitude) from this JSON data and operate this information for relative functions.

- **Mac vendors lookup API**

This API is used for getting one device's company/brand name through its MAC address. It will be used in LAN manager. After this tool gets the MAC address of another device in the same subnet, it can inform the user about the type of that device.

Call <https://macvendors.co/api/> to use this API.



**Figure 10: macvendors screenshot [6]**

For instance, enter <https://macvendors.co/api/08:74:02:00:00:00>, MacVendors lookup API will return the information about MAC address '08:74:02:00:00:00' in a JSON format as Figure 9 shows.

```
{
  "result": {
    "company": "Apple, Inc.",
    "mac_prefix": "08:74:02",
    "address": "1 Infinite Loop,Cupertino CA 95014,US",
    "start_hex": "087402000000",
    "end_hex": "087402FFFFFF",
    "country": "US",
    "type": "MA-L"
  }
}
```

**Figure 11: JSON data about MAC address [6]**

## 2.5 User Interface Design

### 2.5.1 Target User

Since this is an internet diagnose tool based on the android system, the target customers are mainly android users. In this application, it includes four core functions which are 'Searching IP Geolocation', 'Ping', 'Traceroute' and 'LAN Manager'. It could be used to process simple internet diagnose for common users when they have internet error with their android device. As for engineers, every IP information from diagnosis result will be shown with its geolocation information on google map. With a clear visual network structure, the application could also have professional applications such as network measurement and topology.

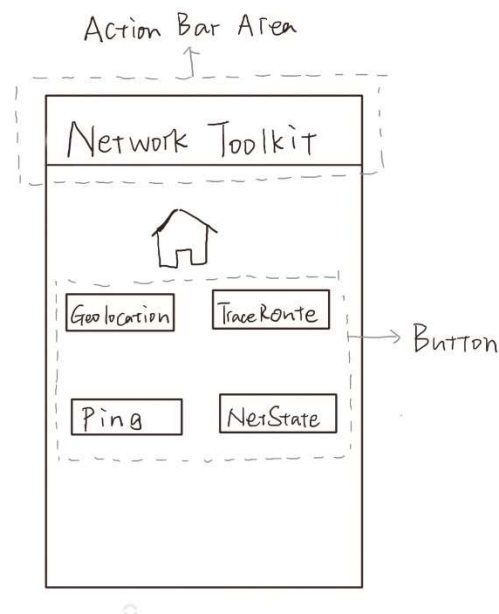
### 2.5.2 The Main Idea of Screen Design

Due to the case that target users include common customers who are not computer science majored, the diagnosis tools' interface will be designed as simple as possible for them to operate. For example, when users want to use the 'Ping' tool, they only need to input the package number and target IP. The rest command will be implemented internally.

### 2.5.3 Interaction styles

- Menu selection

The menu selection style will be applied in main activity's layout and action bar area. In the main activity, users could choose which tool to use by clicking the image button. The other helper functions such as 'get my IP' or 'show on map' will be placed in the action bar.



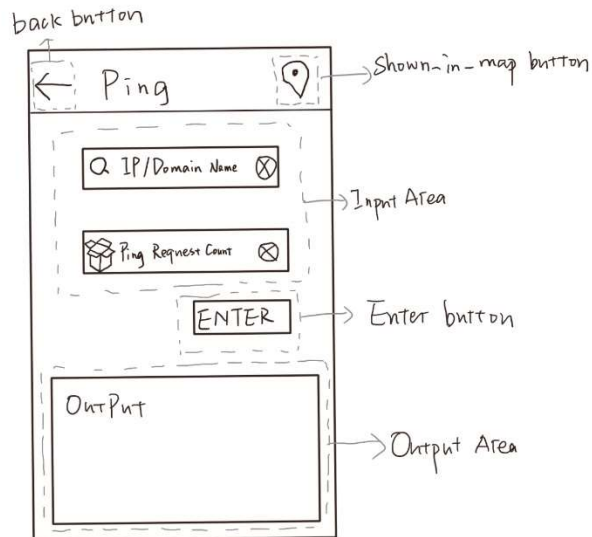
**Figure 12: Menu selection interaction: Prototype of the menu**

- Form fill-in

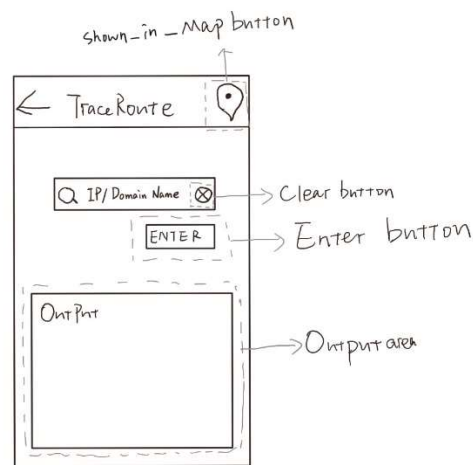
When it comes to the input section, the form fill-in style will be chosen. It will be used to get package number and IP/Domain name. By clicking the 'enter' or 'map' button, the result of diagnosis will be shown in the text or map view.



Additionally, if the input area is not empty, there will be a 'clear button' for the user to easily empty the input. If the input area is empty, this button will be hidden.



**Figure 13: Form fill-in interaction: the prototype of the 'Ping' tool**



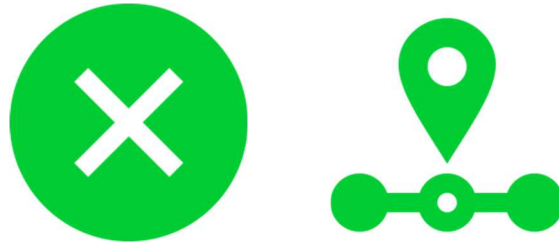
**Figure 14: Form fill-in interaction: the prototype of 'Traceroute' tool**

#### 2.5.4 Organization and Layout of Screens

In every activity's screen, there will be a custom-design action bar at the top of the screen. It will include the title of current activity, the 'back' button to jump to the previous activity and a toolbar area. Then the rest of the screen is filled with the current activity's layout. The detailed is shown in the prototype design.

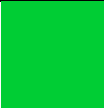



### 2.5.5 The Design of Icon and Theme Color

The resources of icons' images will be chosen from <https://iconmonstr.com/>. The Iconmonstr is a free source of simple icons run by a senior designer named Alexander Kahlkopf.



**Figure 15: Some icon resources from Iconmonstr [7]**

As for theme color, the green and black is chosen to simulate the theme of 'The Matrix'. The table of expected color is shown below.

Preview	Name	Hexadecimal code	Use case
	coderGreen	#00CC33	<ul style="list-style-type: none"><li>• Most of Image buttons</li><li>• Loading animation</li><li>• Traceroute animation</li><li>• Activity title</li></ul>
	darkCoderGreen	#00B82E	<ul style="list-style-type: none"><li>• Image and image button in input area</li><li>• The input text and output text</li></ul>
	backgroundBlack	#1A1A1A	<ul style="list-style-type: none"><li>• Application's background</li></ul>
	grayFrame	#303030	<ul style="list-style-type: none"><li>• Input area's background</li><li>• The folded list's background of tool bar</li><li>• Action bar's background</li></ul>

**Figure 16: The table of expected color**



### 2.5.6 Animation Design

- **'Loading' animation**

The resource of loading animation is added in Gradle file

- Dependencies:

*implementation 'com.github.yuweiguocn:SquareLoading:v1.1.0'*

- Repositories:

*maven { url "https://jitpack.io" }*

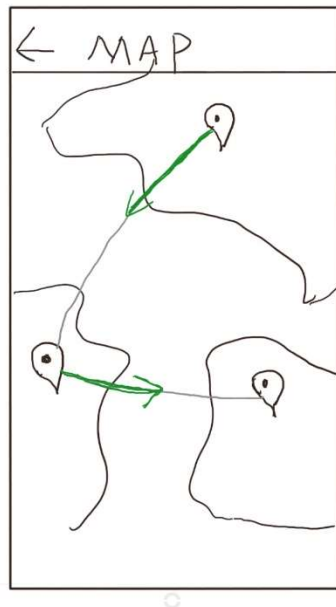
In the application, it will be shown like this:



**Figure 17: The preview of loading animation**

- **Traceroute animation on Google Map**

In 'Traceroute' tool, every gateway will be connected with a line. To make the traceroute sequence clearer, there will be an animation of drawing a line that will cover on the original gray line between gateways. A green line will start at the user's geolocation and end at the target IP's geolocation to show the process of 'traceroute'.



**Figure 18: The expected animation effect of traceroute on map**

### 2.5.7 User Interface Prototyping

In the beginning, we divide UI of this application into 5 different sections, which respectively are the main menu, traceroute page, Ping page, My geolocation page and map page. (figure 17).

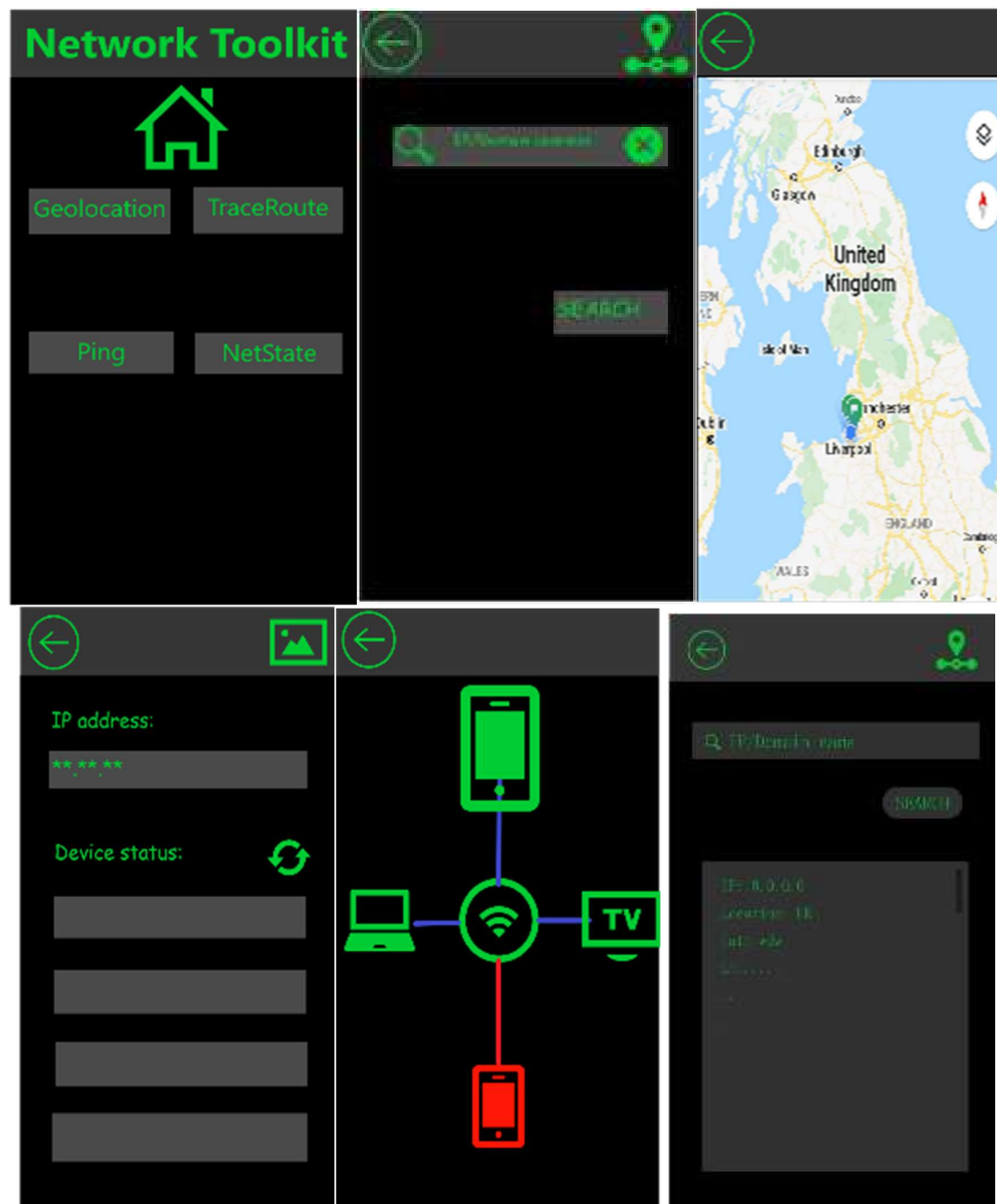


Figure 19: User interface activities prototypes

Then we simulate the process of this application using the storyboarding technique.

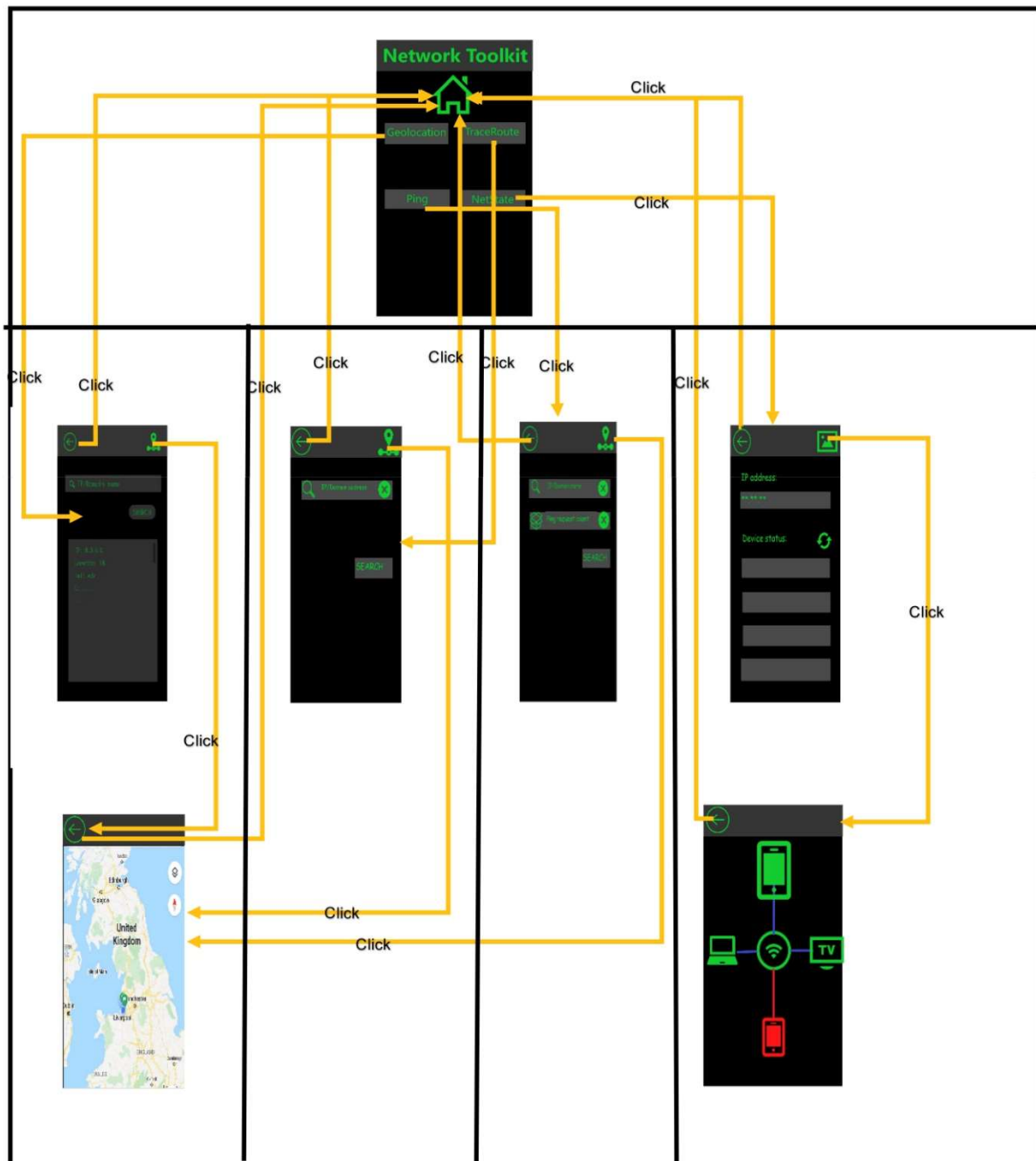


Figure 20: Main storyboard

- Explanation:
  - After clicking the 'Geolocation' button in Main UI, application should jump to the IP geolocation page. Then user should type in their target IP address or domain address then the application will provide the information about this target. And if the user clicks the 'map' button on the right-top. The application will jump to the 'MAP' page, which will show the geolocation info.
  - After clicking the 'TraceRoute' button in Main UI, application should jump to the 'TraceRoute' page. Then user should type in their target IP address or domain address then the application will provide the information about this target. And if the user clicks the 'map' button on the right-top. The application will jump to the 'MAP' page, which will visualize the geolocation information about all the IP addresses which are on the route to the target address.
  - After clicking the 'Ping' button in the main menu, the application should jump to the Ping page. Then user should type in their target IP or domain address and the ping request count. Then the application will provide the information about this target. And if the user clicks the 'map' button on the right-top. The application will jump to the 'MAP' page, which will visualize the geolocation information about this ping.
  - After clicking the 'LAN Manager' button in the main menu, the application should jump to the LAN manager page. Then the application will provide information about the user's network status, such as how many people are using your network (WIFI) and which one is not the trusted device. Click the refresh button can refresh the page. And if the user clicks the 'Show in Picture' button on the right-top. The application will jump to the 'Picture' page, which will visualize the status of network.

## 2.6 Algorithm Design and implementation method of core functions

### 2.6.1 Android 'Traceroute' Implementation Method and Algorithm

Traceroute is a common command in Linux operation systems. However, given the fact that android does not include a traceroute command, we decide to implement this command from the base, using 'Ping' command [8].

- **Principle:**

'TTL' is a header in the IP packet which stands for 'Time for live' or 'hop limit'. It represents the number of hops a packet can take before it is dropped. Each node in the network that the packet transmits through will subtract 1 from the TTL. When TTL falls to 0, it will be discarded and the device which drops the packet will return an 'ICMP TTL Exceeded in Transit' message back to the packet sender, including its IP address. In essence, 'traceroute' sends packets with gradually increasing TTL values until the packet reaches the target, as the process shows in the figure below.

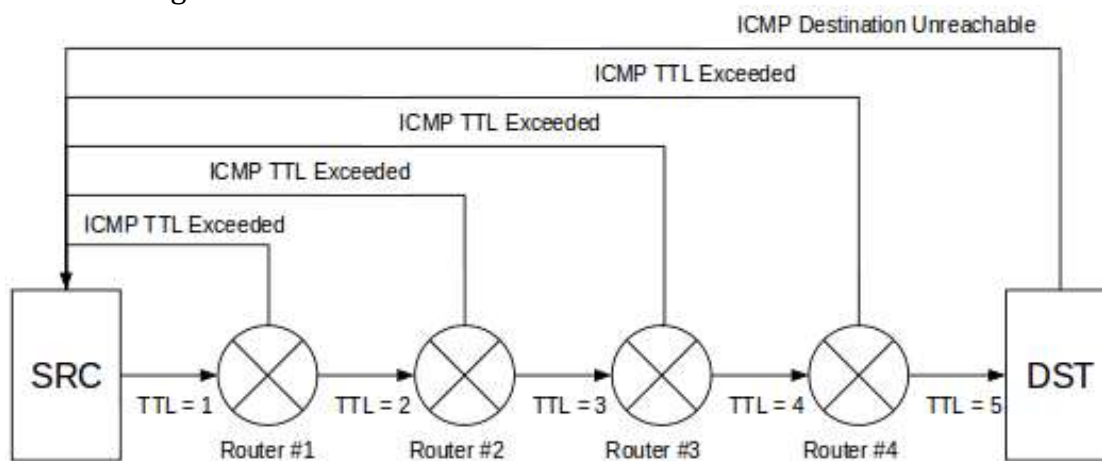


Figure 21: Principle of traceroute

- **Algorithm design:**

Input: a valid IP address or domain name as a traceroute target.

Output: a sequence of IP addresses of hops from the local host to the target.

Pseudo code example:

$Max\_TTL \leftarrow 255$

$Current\_TTL \leftarrow 1$

$Target \leftarrow 8.8.8.8$

$IP\_sequence = \{\}$

While ( $Current\_TTL \leq Max\_TTL$ ) do

    Execute command: 'ping -c 1 -t  $Current\_TTL$   $Target$ '

$msg \leftarrow$  Read in ICMP message when TTL expires

$IP \leftarrow$  IP extracted from  $msg$

    Add  $IP$  to  $IP\_sequence$

    If  $IP == Target$

        Break

    Else

$TTL = TTL + 1$

End

Return  $IP\_sequence$

- **Pseudocode Explanation:**

1.  $Max\_TTL \leftarrow 255$   
Since the TTL field is 8 bits long, so max hops would be 255. Set the maximum hop limit of a packet to 255. If a packet cannot arrive at the target within 255 hops, the target will be regarded as unreachable.
2.  $Current\_TTL \leftarrow 1$   
Set the initial TTL of a packet to 1.
3.  $Target \leftarrow 8.8.8.8$   
Set the target IP address as 8.8.8.8.  
(Note: 8.8.8.8 is an example, any valid IP address or domain name will be accepted)
4.  $IP\_sequence = \{\}$   
Initialize an empty vector for storing a sequence of IP addresses
5. In the while loop, if  $Current\_TTL$  does not exceed  $Max\_TTL$   
Send 1 packet to the target host with TTL value  $Current\_TTL$ .  
Store the ICMP message in  $msg$ .  
Extract IP address from  $msg$  which is exactly the IP address of the device which discards this packet.  
Add the extracted IP to  $IP\_sequence$   
Judge whether the extracted IP equals the target IP. If they are equal, which means the target is reached within  $Max\_TTL$ . If not, increase  $Current\_TTL$  by 1 and loop again.
6. Return  $IP\_sequence$   
Finally return the results of 'traceroute'.

## 2.6.2 Implementation Method for UI

- **Keyboard Control**

The input function will be implemented with the 'EditText' fragment in the Android studio. In our application, the soft keyboard will only appear when the user clicks the input textbox. And the soft keyboard will automatically disappear when the program changes the layout content on the screen. Here are two other cases need to be implemented by ourselves to hide the keyboard.

- If the user clicks the other blank area on the screen, the program will get the currently focused view and its unique token of this view. If neither of them is null, the focus before clicking could not be a blank area. Then the soft keyboard will be hidden.
- When user directly clicks the button to start searching, the program will check the currently focused view again. Due to the case that the keyboard will only appear when the user clicks the textbox, the view must be null if the keyboard is not shown on the screen. As a result, the keyboard will disappear if the view is not null.

- **input listener**

As that we want to implement a 'clear button' beside every input area, an 'edit-text-listener' becomes necessary to check the content of 'EditText' fragments. The content itself will not change unless the user clicks the 'enter' button on their soft keyboard during the typing. Due to this case, the listener needs to check the content before changing, on changing and after changing. If either of them is not empty, the 'clear button' will be visible. Otherwise, the button will be

hidden.

- **on-click-listener**

For every button, there is a listener to listen to the user's click action. The program will set new actions for buttons by implementing a new on-click event for them.

- **action bar**

To achieve more custom-design UI and functions, the android's action bar needs to be replaced. The program will implement a custom-design layout for every activity action bar. It will mainly include:

- the 'back button' – An image button used to back to the previous interface.
- the activity title – A text of the current interface's title.
- the toolbar area – It's an area of buttons which includes some shortcut function such as showing geolocation information on google map. This area will be set on the right side of the action bar.

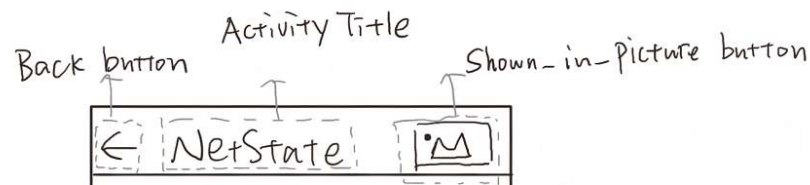


Figure 22: a preview of one custom design layout of the action bar

- **dynamic loading of output views**

During the output of diagnose result, the UI should be changed dynamically by adding designed layout. In the diagnose processing, it could be implemented by adding views in a UI thread with inflate method. Essentially, every single view is a user-defined xml layout file. The file includes textview and other UI components to satisfy the required output format.

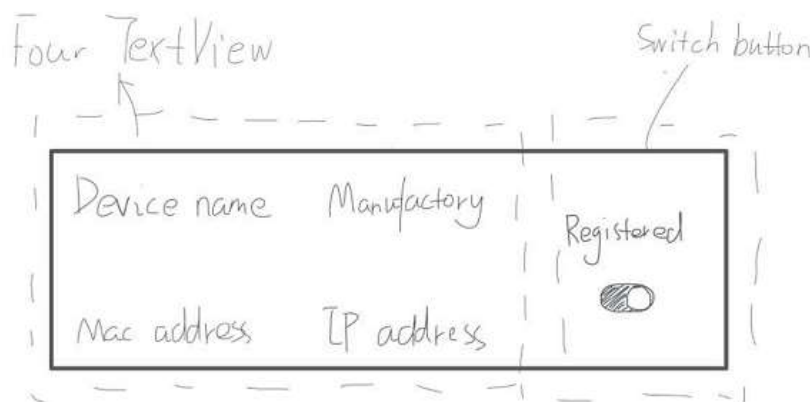


Figure 23: a preview of one user-defined layout of the lan manager output

### 2.6.3 Implementation Method for Animation on Map



Figure 24: Brief process of the whole animation

The whole animation will be consisted of two simple animation to give users a feeling that there is data transferred between points in an ordered sequence.

The first animation will draw two same lines: background line and foreground line from the first point to the last point by overriding the evaluator provided in API to smoothly change the coordinate of a point in the line and creating an object animator using the evaluator to start the animation.

The second animation will be an animation set which will first change the color of the foreground line and then reduce its length from start point to end point smoothly by time to provide the dynamic visual effects (The color of background line and foreground line is different so that the foreground line seems like running when changing its length). The whole animation will loop when its listener calls the end of the animation.

### 2.6.4 Implementation Method for Scanning devices under LAN

The core function of LAN Manager tool is to scan the whole local network area. We decide to read ARP cache in local device for obtaining the MAC address of all the other devices in the same subnet.

Suppose there are two devices A and B in the same subnet. At first, the local ARP table of A is empty. After A sends an ARP request to B and B sends an ARP reply to A, the ARP table in A will be updated. It will contain the MAC address of B. Therefore, if we want to scan the whole subnet and get all the devices' MAC addresses, we should broadcast the ARP query to the whole subnet.

Steps for broadcasting the ARP query:

1. Get the private IP address of local host. (Suppose its IP is 192.168.0.54)
2. Extract network address and subnet address which is 192.168.0.
3. Send ARP queries from 192.168.0.1 to 192.168.0.255.
4. If the IP address is reachable, the device which owns that IP address will send ARP reply back to the sender. Then the local ARP table will be updated.


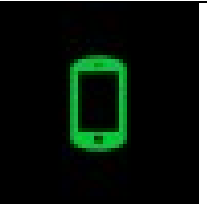
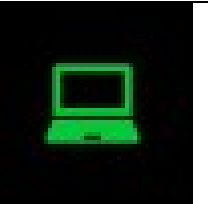

Notice: the scanning process may cost much time if ARP queries are sent



sequentially. Therefore, each ARP query will be sent in a new tread so that multiple queries can be sent at the same time and time for processing will be shortened.

### 2.6.5 Implementation Method for sketching topological graph

After reading the local ARP table, we can obtain the MAC address of every device in the current subnet. The brand name of these devices can be found via Mac vendors lookup API mentioned in 2.4.2. Then each device will be classified into one of the 4 types which are router, mobile phone, Laptop/Desktop, and unknown devices. Corresponding icons for each type will apply on sketching topological graph.

Types	Router	Mobile phone	Laptop/Desktop	Unknown devices
Icons				

Here are the classification rules:

1. Router: a device with last number of IP address is 1 (e.g. 192.168.0.1).
2. Mobile phone: based on its brand name.
3. Laptop/Desktop: based on its brand name.
4. Unknown devices: devices with unknown brand names.

Sketching steps:

1. Put the router icon at the center of the canvas.
2. Put all the other devices' icons will be evenly distributed on a circle around the router icon. The center of this circle is the center of canvas and radius is  $0.5 * \text{width of canvas}$ .
3. Add IP address and brand name of that device below its icon.

## 2.7 Network Protocol

- **IP**

Internet protocol is a significant communication protocol in internet protocol suite. It ensures the routing functions to implement so-called internetworking, also known as the 'Internet'. In this project, IP works as a fundamental protocol to implement the message packet transmission between hosts inside the network. To transmit packets from source hosts to destination hosts, IP need IP address to locate those hosts in the internet.

- **HTTP**

Hypertext Transfer Protocol is the foundation of data transmission in the World Wide Web. Hypertext document usually includes hyperlinks and relevant resources types that user will be able to access. In this project, we use HTTP/1.1 to extract useful information from eXTReMe-IP-Lookup API on the web page.

- **ICMP**

Internet Control Message Protocol is a supporting protocol in internet protocol suite. It was used by network devices such as routers and servers to send error messages or operational information to indicate success or failure of connections between different IP addresses. In this project, it works to return operational information of IP addresses in Ping and Traceroute functions.

- **ARP**

ARP stands for Address Resolution Protocol. It works as a cache to store MAC address and IP address and mapping MAC address to corresponding IP address under the same LAN. When routers sending messages to destination IP, information about this IP address will be stored in the ARP table. LAN Manager will retrieve information in the ARP table and display it to the user.

- **DHCP**

Dynamic configuration protocol is a network management protocol that used on IP networks whereby a DHCP server dynamically assigns an IP address to each device on a network so they can communicate with other IP networks. When device connects to the network it will be automatically assigned an IP address and this is a critical part when looking up all IP addresses of devices under the same LAN.

## 2.8 Evaluation Design

### 2.8.1 Evaluation Methodology for User Interface

In the design process, we planned to evaluate the UI from three perspectives: attractiveness, easy-to-use and error tolerance (bug-free). All these aspects will help to improve the whole quality of the user interface.

Firstly, the evaluation of UI attractiveness should involve collecting advice from users who have experience in using our UI. We will hand out 50 questionnaire and let the 50 users who experienced our UI to finish this questionnaire. Then we collect the questionnaires and modify our UI according to advice on the questionnaires.

The attractive degree of UI questionnaire template		
Version of UI:		Date:
Which UI	If attractive	Advice

Secondly, whether the UI is easy-to-use, this aspect can be shown by the average time that a user who never used our UI, can handle 80% of our functions for the first time. To obtain this information, we need to have observation of the user at work with the UI and 'think aloud' about how they are trying to use the UI to accomplish particular tasks (around 80% of the functionalities). Invite around 100 volunteers, count the time they used to handle 80% of the functions and calculate the average time they used. Set the expected time to 3 hours and compare the actual result with the expected time. Also, we can ask the users about the difficulty of using our UI to see if the existing UI is too difficult to handle. While doing tests, use the UI test form to record the details to fix the bugs.

Thirdly, bug-free and error tolerance of UI. We plan to designate one people in our group to test UI from three aspects: whether the page can be loaded correctly every time, the note in text box was displayed whenever we enter the page if it can jump to correct page after we input different types of inputs and click the certain button. Then we record the bug while we testing. (As form shown in figure 2.6.2.) What's more, the tolerance can be shown by two methods. The first one is to click one button 100 times in a high frequency. Then observe whether there will be errors. The second one is to click the whole screen randomly for 100 times to see if the application will crash.

After we fixed the bug, repeat the above operation until no more bug occurs. Every time we make modifications to UI, we need to do this in case the modification generates more bugs.

## 2.8.2 Test Form

Test plan template							
Test date:		Test Version:		Tester name:		Test component:	
No.	Name of case	Description	Input data	Action	Expected output	Actual output	Success/Fail
1							
2							
3							

This test form aimed at inspecting the output of particular functions in the project is the expected output. When set up a test, members should specify its date, what this test is about (name of case), general description about the test case, who is testing this and how did he test it. Eventually, they should also describe the output of the test, report the final result (success or fail). All functions in network, UI and Map design will be tested in the future.

## 2.8.3 Test Design

Test component	Test cases/ input data
Geolocation search	Textbox for target host: <ol style="list-style-type: none"> <li>1. Null</li> <li>2. Valid IP address</li> <li>3. Invalid IP address</li> <li>4. Valid domain name</li> <li>5. Invalid domain name</li> </ol>
Ping	Textbox for target host: <ol style="list-style-type: none"> <li>1. Null</li> <li>2. Valid IP address</li> <li>3. Invalid IP address</li> <li>4. Valid domain name</li> <li>5. Invalid domain name</li> </ol> Textbox for packet number: <ol style="list-style-type: none"> <li>1. Negative integer</li> <li>2. Zero</li> <li>3. Positive integers</li> <li>4. Decimals (float/double)</li> <li>5. Other invalid ASCII characters</li> </ol>
Traceroute	Textbox for target host: <ol style="list-style-type: none"> <li>1. Null</li> <li>2. Valid IP address</li> <li>3. Invalid IP address</li> <li>4. Valid domain name</li> <li>5. Invalid domain name</li> </ol>
LAN Manager	<ol style="list-style-type: none"> <li>1. Read records of the devices from the log file.</li> <li>2. Update records of the devices to log file.</li> </ol>

### 3. Review Against Plan

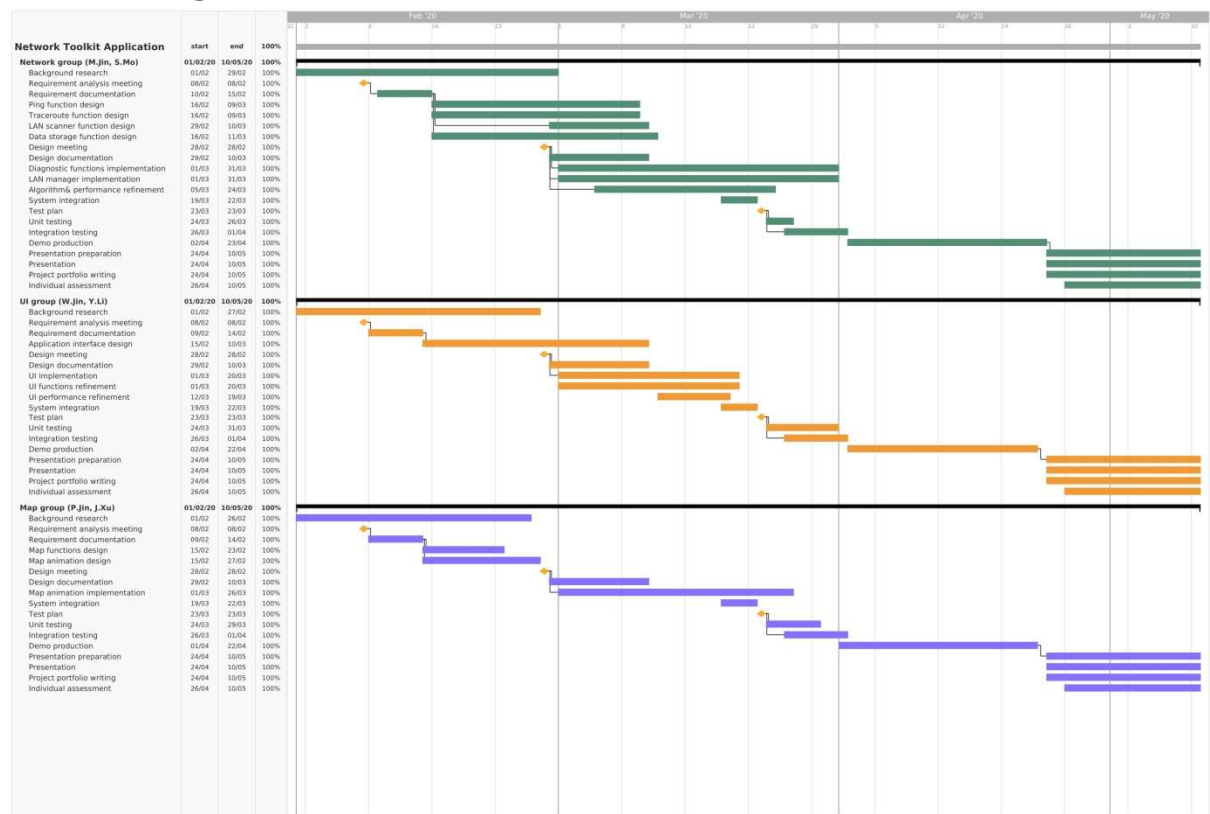



Figure 25: Gantt Chart

#### Description:

Team assignment	Group members
Network group	Minhao Jin, Shuheng Mo
UI group	Weihao Jin, Yaokang Li
Map group	Pengcheng Jin, Jinyao Xu

As mentioned in section 1.3, we have introduced a new LAN Manager functions for the network toolkit application therefore there will be significant modification to the task scheduling (Gantt chart) of our project. For the previous version of our Gantt chart, details such as task dependencies and time scheduling for specific tasks were not specified. In the current Gantt chart, we have assigned enough time to both network group and UI group for developing a new LAN Manager function in the application and re-arranged the time for the following tasks. Also, dependencies between specific tasks have been indicated on the chart by a solid line between task boxes.  Diamond shapes like this indicate important milestones and most tasks after the milestone should have a dependency on it. Another feature that worth distinction is that we added an area with a percent completion view of each task, which helps to show the overall progress of each task.

#### 4. References

- [1]'Ping (networking utility)', *En.wikipedia.org*, 2020. [Online]. Available: [https://en.wikipedia.org/wiki/Ping\\_\(networking\\_utility\)](https://en.wikipedia.org/wiki/Ping_(networking_utility)). [Accessed: 07- Mar- 2020]
- [2]'Traceroute', *En.wikipedia.org*, 2020. [Online]. Available: <https://en.wikipedia.org/wiki/Traceroute>. [Accessed: 07- Mar- 2020]
- [3]'Local area network', *En.wikipedia.org*, 2020. [Online]. Available: [https://en.wikipedia.org/wiki/Local\\_area\\_network](https://en.wikipedia.org/wiki/Local_area_network). [Accessed: 07- Mar- 2020]
- [4]'Hypertext Transfer Protocol', *En.wikipedia.org*, 2020. [Online]. Available: [https://en.wikipedia.org/wiki/Hypertext\\_Transfer\\_Protocol#Request\\_methods](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#Request_methods). [Accessed: 07- Mar- 2020]
- [5]'Free IP Lookup Geolocation API', *eXTReMe-IP-Lookup.com*, 2020. [Online]. Available: <https://extreme-ip-lookup.com/>. [Accessed: 09- Mar- 2020]
- [6] "Mac address lookup API | Mac Vendors", *Macvendors.co*, 2020. [Online]. Available: <https://macvendors.co/api/>. [Accessed: 23- Apr- 2020]
- [7]'iconmonstr - Free simple icons for your next project', *Iconmonstr.com*, 2020. [Online]. Available: <https://iconmonstr.com/>. [Accessed: 09- Mar- 2020]
- [8]'Reading a Traceroute', *Maxcdn.com*, 2020. [Online]. Available: <https://www.maxcdn.com/one/tutorial/how-to-read-a-traceroute/>. [Accessed: 09- Mar- 2020]

#### 5. Signatures

Minhao jin:		Pengcheng Jin:	
Weihao Jin:		Yaokang Li:	
Shuheng Mo:		Jinyao Xu:	