CS 456/656 - Assignment 2 (Fall 2011)

Device-to-device Communication

Distributed: Oct. 4, 2011 Due: Oct. 21, 2011 11:59:59 pm

1 Introduction

The presence of multiple network interfaces on today's mobile devices is creating new opportunities for mobile application developers. Traditionally, mobile applications have followed a client-server paradigm, where the client application on the mobile device communicates exclusively with a server on the Internet or within a cellular network. Consequently, for two devices to communicate, data must be relayed through a remote server even if the devices are physically adjacent.

In this assignment, we introduce a new class of mobile application that exploits the enormous untapped wireless capacity that exists **between** mobile devices. You will develop a simple application where devices communicate *opportunistically* with other neighbouring devices to exchange some data. This assignment will be done by teams of **two** students.

2 Assignment

In this assignment each team will design and implement a simple device-to-device file sharing application. Each device will periodically scan for neighbouring devices, establish a TCP connection with a neighbouring device, and exchange a listing of files present on the device's micro SD card. In this assignment, you may reuse parts of previous assignments, but you may *not* use any libraries that are not part of the standard Android OS API. The assignment includes a simple service based tool called the 'Bluetooth Lookup Service' (BLS) that must be run in parallel with your application. This tool is described in Section 3.

This assignment can be divided into five phases. The design and implementation details of each phase are up to you; however, you must document your decisions.

2.1 Phase 1: neighbour detection

In this phase, the application must scan for the Bluetooth MAC addresses of neighbouring Bluetooth devices and obtain a list of Bluetooth MAC addresses.

2.2 Phase 2: Bluetooth Lookup Service (BLS) query

After obtaining a list of Bluetooth MAC addresses for neighbouring devices, the application must query the BLS server to determine the WLAN IP address of each neighbouring device (there may be more than one neighbour, but only some of them may be willing to communicate with your device). The query process is further described in Section 3.3.

2.3 Phase 3: Connection setup

Once your application has retrieved a list of WLAN IP addresses of neighbouring Bluetooth devices, it should select one of the devices and connect to it over TCP, optionally repeating this process for each available, willing neighbour.

Given that every student in the class will be using the BLS, your application may frequently detect and connect to devices that implement a different server-client communication protocol over TCP. If the connection fails, your application may re-attempt the connection or attempt to connect to another device. The implementation details are yours, but be sure to document your decisions.

2.4 Phase 4: Listing exchange

After establishing a TCP connection with a neighbouring device, each device must send a listing of files present on its micro SD card. Please fully document the decisions that you make in the design and implementation of your file listing exchange protocol.

2.5 Phase 5: Evaluation

In addition to the completed application, you must submit a trace file that demonstrates the application in use. The trace file must contain an example of usage of your system, i.e., document each phase above. Specifically, the records in the trace file should be of the following form:

```
<time>: SCAN_BEGIN
<time>: SCAN_DETECT:<Bluetooth MAC 1>
.
.
<time>: SCAN_DETECT:<Bluetooth MAC N>
<time>: SCAN_END
<time>: BLS_QUERY: <Bluetooth MAC 1>
<time>: BLS_REPLY: <timestamp>, <WiFi LAN IP>, <WiFi WAN IP> or FAILURE
.
.
<time>: BLS_QUERY: <Bluetooth MAC N>
<time>: BLS_REPLY: <timestamp>, <WiFi LAN IP>, <WiFi WAN IP> or FAILURE
<time>: CONNECT: <WiFi IP>, SUCCESS or FAILURE
<time>: FILE LIST: <WiFi IP>, <filename1>,...,<filename N>
```

2.6 User interface

The application should also have a simple user interface that satisfies the following **minimum requirements**:

- A 'Start Scan' button that initiates a new scan on the Bluetooth radio.
- A text field that indicates the time of the last Bluetooth scan.
- A text field that displays the Bluetooth MAC address of the device itself.
- A text field that displays the WLAN IP address of the device itself.

- A text field/list that displays the list of Bluetooth MAC addresses of devices detected in the last scan performed.
- A *form of user input*, which lets the user connect to one of the devices (TCP server-client communication) from the list.
- A 'Start/Stop' button or menu item that enables/disables the file-list exchange TCP server.
- A text field that indicates the general status of the application. For example, the user should be aware that the application is scanning for neighbours, connecting to a neighbouring device, or performing an BLS lookup.
- After connecting to a neighbouring device, the application should display a screen that lists the files on the neighbouring device.

You can make the following assumptions :

- The WiFi radio has been enabled by the user. You therefore do not need to programmatically invoke the *ConnectivityManager* or *WifiManager* to enable the WiFi radio.
- The /SDCard/ file system root is always available. However, you will need your application to acquire WRITE_EXTERNAL_STORAGE privileges to write to your trace file. Note that, while attempting to write to a file from your application, your device should be un-mounted (USB storage turned off) from your computer's file system in order to be able to write to a trace file on the /SDCard/ (device's external storage).

You can use any other privileges in your application required for accessing the Bluetooth radio *BLUETOOTH, BLUETOOTH_ADMIN*, Socket operations, HTTP Web requests - *INTERNET*. Your application must run in the background when the user presses the home button.

2.7 Bonus

For a maximum of 15% bonus, you may extend your application to allow files to be exchanged between devices. After exchanging a listing of files, each device should have the ability to select a single file from the neighbour's list of files and request the file from the other device. The other device should vibrate to notify the user that a request is being made and prompt the user for permission to transmit the file. If the user accepts the request, the file should be transferred to the requesting device. Once the exchange is complete, the requesting device should vibrate and display a dialog box to notify the user that the exchange is complete.

3 Provided tools

You will need to install on each Android device, a service based application called the **Bluetooth Lookup Service (BLS)** client. The BLS is designed to provide energy efficient device discovery on mobile devices that cannot receive broadcasted packets. Conventional ad hoc applications detect neighbouring computers by periodically broadcasting UDP packets to all devices on its subnet. For example, iTunes detects neighbouring iTunes clients by broadcasting a UDP packet (containing its IP address) on port 5353. Other instances of iTunes then listen for broadcast packets on port 5353. When a 'neighbouring' iTunes client is detected, they may establish a TCP connection to reliably exchange media.

Unfortunately, energy constrained devices such as Android mobile devices, typically do not listen for broadcast packets. Doing so would require the device to keep its WiFi radio on at all times, which would rapidly

consume the device's battery. By not listening to broadcast packets, devices cannot discover each other using the WLAN interface alone. The BLS has been designed to overcome this necessary limitation.

3.1 How it works

The BLS relies on both the Bluetooth and WiFi networks to perform device discovery. At startup the BLS determines the device's fixed Bluetooth MAC address. Periodically (every 30 seconds), the device's Bluetooth MAC address is then uploaded to a server running on the university's network along with its WLAN IP address. The BLS server then records the Bluetooth MAC address, the WLAN IP address, the WAN IP address, and the current timestamp to a persistent database. Other devices may then query the BLS server to determine the IP addresses for any given Bluetooth MAC address. This technique is analogous to the Domain Name Service (DNS), with the exception that the DNS hostname has been substituted for a Bluetooth MAC address.

Note: If the WiFi radio n the device is disabled then the BLS client application would not be able to update your WLAN IP address on the BLS server. However, the BLS client has been designed such that it will turn on your WiFi radio and Bluetooth radio while updating the IP addresses on the BLS server. But, your device needs to be connected to a WiFi network in order to have an IP address associated with it.

Note that, the BLS client application will make sure that your device is Bluetooth discoverable at all times. This is essential for letting neighbouring devices discover your device and attempt a connection to it over TCP. While running in the background, the BLS client application will prompt you (atmost every two minutes) to confirm Bluetooth discoverablility of your device. Alternatively, you can install *Available System Updates* on your device which will let you keep your device Bluetooth discoverable indefinitely. The update will NOT affect your application development/deployment in any other way.

Privacy disclaimer: The course administration does not maintain a record of the Bluetooth MAC address of each student's device. The course administration does not maintain historical records of IP address updates. Submitting a new IP address update will always replace the previous (stale) entry.

3.2 How to get the BLS client on your device

- On your Android device, proceed to *Settings* → *Applications*, and check the *Unknown sources* option. Additionally, you may select suitable options under *Settings* → *Applications* → *Development*, which are useful when developing/debugging applications.
- Launch the device's browser, and access the following URL :

  `http://blizzard.cs.uwaterloo.ca/~rayman/BLS.apk`

  The Android OS will detect it as an Android application file and download it to your device. Proceed to *Downloads* and install the *BLS* application.
- Reboot your device. The BLS client service will start running on your device (as a background service) and update your Bluetooth MAC and IP addresses on the BLS server. Wait for your device to acquire an IP address, following which, you will be able to see a *BLS Update toast* message on your screen. Additionally, you may see a prompt message to turn on the Bluetooth discoverablility of your device.

3.3 How to use the BLS in your application

When the Bluetooth MAC address of a neighbouring device has been detected, an application may query the BLS server through an HTTP GET request with a single 'btmachash' parameter to the following web service:

`http://blow.cs.uwaterloo.ca/cgi-bin/bls_query.pl`

'btmachash' is obtained from a MAC address (ommiting all ':' and converting to lower case) by obtaining its *SHA-1* digest/hash encoded on base 64. For example, to query the details of a neighbouring device with the Bluetooth MAC address `38:E7:D8:46:4E:B4`, the following HTTP GET request would be used:

`http://blow.cs.uwaterloo.ca/cgi-bin/bls_query.pl?btmachash=BZfsOzipNH31YkVePR99THW2ZIY=`

where,
btmachash : *Encode_On_Base_64 (SHA-1_Hash_Digest (*38e7d8464eb4*))*

If found on the BLS server's database, the script will return five lines of text in the following format :

```
<Bluetooth MAC>\n
<LAN IP address>\n
<WWAN IP address>\n
<epoch timestamp (seconds)>\n
<human readable timestamp>\n
```

The following code segment shows how to obtain base 64 encoded SHA-1 hash :

```
MessageDigest md = MessageDigest.getInstance("SHA-1");
md.update(input);
String BtMacHash = Base64.encodeToString(md.digest(), 0);
```

If an entry for the device is not found on the BLS server, it will return the following:

`Bluetooth MAC hash: BZfsOzipNH31YkVePR99THW2ZIY= not found`

For example, consider a device with a Bluetooth MAC address `00:23:7A:11:D4:53` roaming on a private network within the University's campus network (129.97/16) using NAT. While on a private network, the device has a private IP address in the 10.0.2/24 range. The IP address of the NAT is 129.97.7.145. Following is the output of a BLS query for this device:

```
00237a11d453
10.0.2.3
129.97.7.145
1240861061
2011-08-27 15:37:41.642
```

However, suppose the device leaves the private network and associates with the *uw-secured* wireless network and receives a public IP address within the range 129.97/16. Following is the new output of a BLS query:

```
00237a11d453
129.97.137.142
129.97.137.142
1240861165
2009-04-27 15:39:25.738
```

Once you have obtained the IP address of a neighbouring device using the BLS, you may begin to communicate with it over **TCP on port 62009**. neighbouring devices may attempt to communicate to your device on the same lines. At any time you may verify that the BLS client is correctly updating your WLAN IP address by entering the above query URL into your web browser with your own device's Bluetooth MAC address's SHA-1 hash (base 64 encoded).

4 What to turn in

You must submit the following using the Unix *submit* utility:

- Complete application source Android project including the source code, resources, generated files, assets and your bin/ directory (containing your application-apk file) as a single tar/zip package using the unix *submit* utility available in the undergraduate computing environment.
- A trace output (*trace.txt*) for the communication trace of your device, showing one successful scan and file listing.
- A PDF document that describes your application, indicates what you have completed (or not completed), problems encountered, and any other comments that you wish to make.

  Please limit the document to **five** pages, with 1 inch margins, and no less than 12pt font. The document may also include screenshots of your application. You may use the open source software Android Screenshots and Screen Capture : http://ashot.sourceforge.net/.

The unix *submit* utility has been configured to accept:

```
*.zip
*.tar
*.pdf
trace.txt
```

To submit all of the files in the current directory, use the command: *submit cs456 a2 .*

5 Grading scheme

This assignment will be evaluated *in person* by a TA, to whom each team will do a demonstration. To ensure that the version of the code running on your device is the same as the version you turned in, the TA will load the devices with the submitted code. TAs will also look through the source code at the time of the demo.

Grades will be assigned as follows:

Scanning	Application runs and correctly detects neighbouring devices	25%
BLS integration	BLS query and reply completed successfully	15%
Connection	Ability to connect to another device	20%
Listing	Ability to exchange a file listing	15%
Design decisions	Adequate justification of design decisions	25%
Bonus	A maximum of 15% will be added for successfully completing of the bonus component.	15%

6 Question, comments, or problems

Please post general questions or comments to the discussion group 'Assignment 2' on Waterloo Learn at:

https://learn.uwaterloo.ca

Remember: You must use the WiFi or Bluetooth interfaces on the device to transmit data. The GSM interface should **not** be used in this assignment.