**Android客户端与PC服务器实现Socket通信**

Posted on 2013年05月20日 by U3d / [Unity3D 基础教程](http://www.unitymanual.com/category/manual/unity3d-%e5%9f%ba%e7%a1%80%e6%95%99%e7%a8%8b)/被围观 17 次

Android终端持续扫描AP信息并发送给服务器端的实现。首先基于TCP协议在Android终端和PC两端之间形成网络虚拟链路。使用ServerSocket创建TCP服务器端，然后在Android客户端使用Socket的构造器来连接服务器。其中Android终端通过WIFI连接和PC处于同一局域网。

1. PC服务器启用ServerSocket

两个通信实体在建立虚拟链路之前，需要有一方先准备好，主动接受来自其他通信实体的连接请求。

使用ServerSocket对象监听来自客户端的Socket连接。

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 1 | *//创建ServerSocket对象* |
| 2 |  |
| 3 |  |
| 4 | ServerSocket ss = new ServerSocket(30000); |
| 5 | *//监听来自客户端的请求* |
| 6 | **while**(**true**){ |
| 7 | Socket s = ss.accept(); |
| 8 | … |
| 9 | } |

如果没有连接，则将一直处于等待状态。当接收到连接请求后，获取消息到输入流，并保存到文件。

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 01 | *//接收客户端消息* |
| 02 |  |
| 03 |  |
| 04 |  |
| 05 | BufferedReader **in** = new BufferedReader(new InputStreamReader(client.getInputStream())); |
| 06 | **String** str; |
| 07 |  |
| 08 | BufferedWriter bw = new BufferedWriter(new FileWriter("D:/ApInfo"+ (i++) +".txt")); |
| 09 | **while** ((str = **in**.readLine()) != **null**) { |
| 10 | System.**out**.println(str); |
| 11 | bw.write(str); |
| 12 | bw.newLine(); |
| 13 | } |

2. Android终端使用Socket通信

客户端使用Socket的构造器连接服务器，指定服务器IP和端口号就可以了。

Socket s = new Socket(“192.168.1.100”, 30000);

这样服务器端的accept()方法就得到响应，从而向下执行，服务器端和客户端就形成了一对互相连接的Socket。再进行通信时就没有服务器和客户端之分了，都是通过输入输出流进行通信。

详细步骤：

采用Handler和TimerTask来定时扫描AP信息并发送给服务器端。TimerTask规定了到达指定的时间所要进行的任务。

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 1 | TimerTask task = new TimerTask(){ |
| 2 |  |
| 3 | **public** **void** run() { |
| 4 | Message message = new Message(); |
| 5 | message.what = 1; |
| 6 | handler.sendMessage(message); |
| 7 | } |
| 8 |  |
| 9 | }; |

handler传递message内容：

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 01 | Handler handler = new Handler(){ |
| 02 |  |
| 03 | **public** **void** handleMessage(Message msg) { |
| 04 | **switch** (msg.what) { |
| 05 | **case** 1: |
| 06 | *// 执行定时器时间到了之后由handler传递的任务* |
| 07 | **break**; |
| 08 | } |
| 09 | super.handleMessage(msg); |
| 10 | } |
| 11 |  |
| 12 | }; |

因为需要持续执行扫描任务，所以启用新线程执行定时任务。

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 01 | *//启动单独线程定时向服务器发送AP信息* |
| 02 | *//by wayne from www.cnblogs.com/dwayne* |
| 03 |  |
| 04 | new Thread(){ |
| 05 |  |
| 06 | @**Override** |
| 07 | **public** **void** run() { |
| 08 | *// TODO Auto-generated method stub* |
| 09 | timer.schedule(task, 2000,10000); *//在2秒后每10秒执行一次定时器中的方法* |
| 10 | } |
| 11 |  |
| 12 | }.start(); |

接下来扫描AP信息并发送给服务器端，然后将结果保存。

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 1 | WifiManager wifiManager=(WifiManager) getSystemService(WIFI\_SERVICE); |
| 2 | wifiManager.startScan(); |
| 3 | mWifiList = wifiManager.getScanResults(); |

由WifiManager说明可知，它可以用于处理已配置的网络，当前连接的网络及AP信息的扫描等情况。

向服务器发送消息：

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 1 | socket = new Socket("192.168.1.211",30000); |
| 2 | *//向服务器端发送消息* |
| 3 | PrintWriter **out** = new PrintWriter( new BufferedWriter( new OutputStreamWriter(socket.getOutputStream())),**true**); |
| 4 | **out**.println(message); |

其中message为获取的AP信息，测试收到的信息格式为：

SSID: ICIS\_LAB, BSSID: 1c:af:f7:9a:65:e4, capabilities: [WPA-PSK-TKIP+CCMP], level: -80, frequency: 2437