Network Access in Android

Network Status, Network Connectivity

A.R. Kazemi

Network in Android

- Access to Local Network or Internet
  - Networking is based on TCP/IP protocol
  - Do data transfer via
    - Direct low level transfer layer protocols (TCP or UDP)
    - Or using application level protocols (http, https, rtp, ftp, …)
- Network Operations
  - Getting Network status info
    - Is device connected to network?
    - Which network type (WiFi, Mobile data)?
  - Connecting to the network
  - Transferring the data (Download/Upload)

Network Access Permissions

- Add following permissions to the App Manifest
  - Internet access
  - Checking the network status

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="org.myorganization.myapplication" >

  <uses-permission android:name="android.permission.INTERNET"></uses-permission>
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"></uses-permission>
  ...
  <application>
    ...
  </application>
</manifest>
```

Slides By Alireza Kazemi,
Content From: https://developer.android.com/training/basics/network-ops/connecting.html &
https://www.tutorialspoint.com/android/android_network_connection.htm

Checking Network Connection Status

- Use the system service ConnectivityManager

```
ConnectivityManager check = (ConnectivityManager)
    this.context.getSystemService(Context.CONNECTIVITY_SERVICE);
```

- Get network status info
  - For all available networks

```
NetworkInfo[] info = check.getAllNetworkInfo();
```

```
for (int i = 0; i<info.length; i++){
   if (info[i].getState() == NetworkInfo.State.CONNECTED){
      ...
   }
}
```

  - For Active network:

```
ConnectivityManager connectivityManager =
        (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo networkInfo = connectivityManager.getActiveNetworkInfo();
```

Checking Network Type

- Wi-Fi

- Mobile Data

```
NetworkInfo networkInfo =this.getActiveNetworkInfo();
if (networkInfo.getType() == ConnectivityManager.TYPE_WIFI)
    ...
if (networkInfo.getType() == ConnectivityManager.TYPE_MOBILE)
    ...
```

Connecting to a Host/URL

- Connection to a HTTP server

```java
String link = "http://www.google.com";
URL url = new URL(link);
```

```java
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.connect();
```

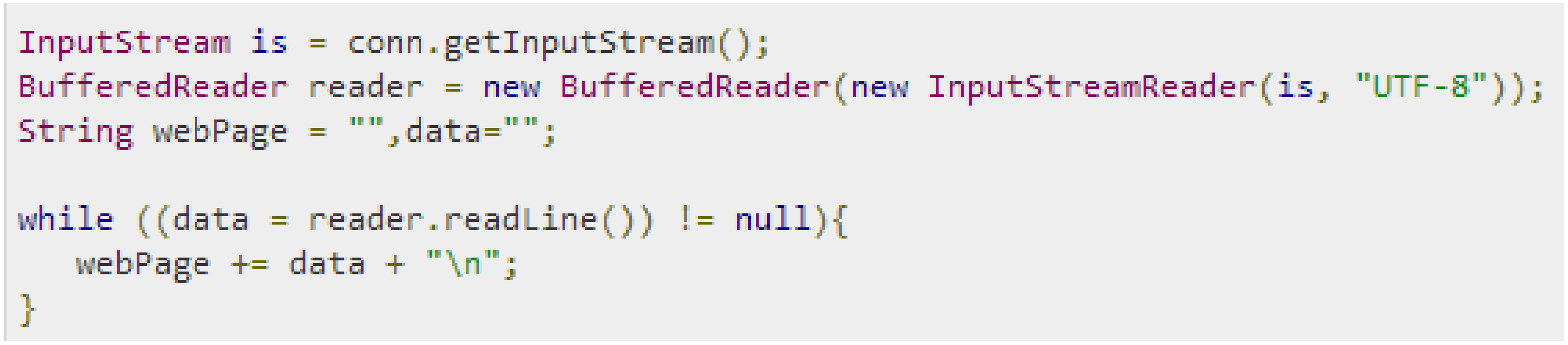
- Secure Connection to n HTTP server over SSL/TLS

```java
HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
conn.connect();
```

Slides By Alireza Kazemi,
Content From: https://developer.android.com/training/basics/network-ops/connecting.html &
https://www.tutorialspoint.com/android/android_network_connection.htm

Transfer (Read) Data from the Connection

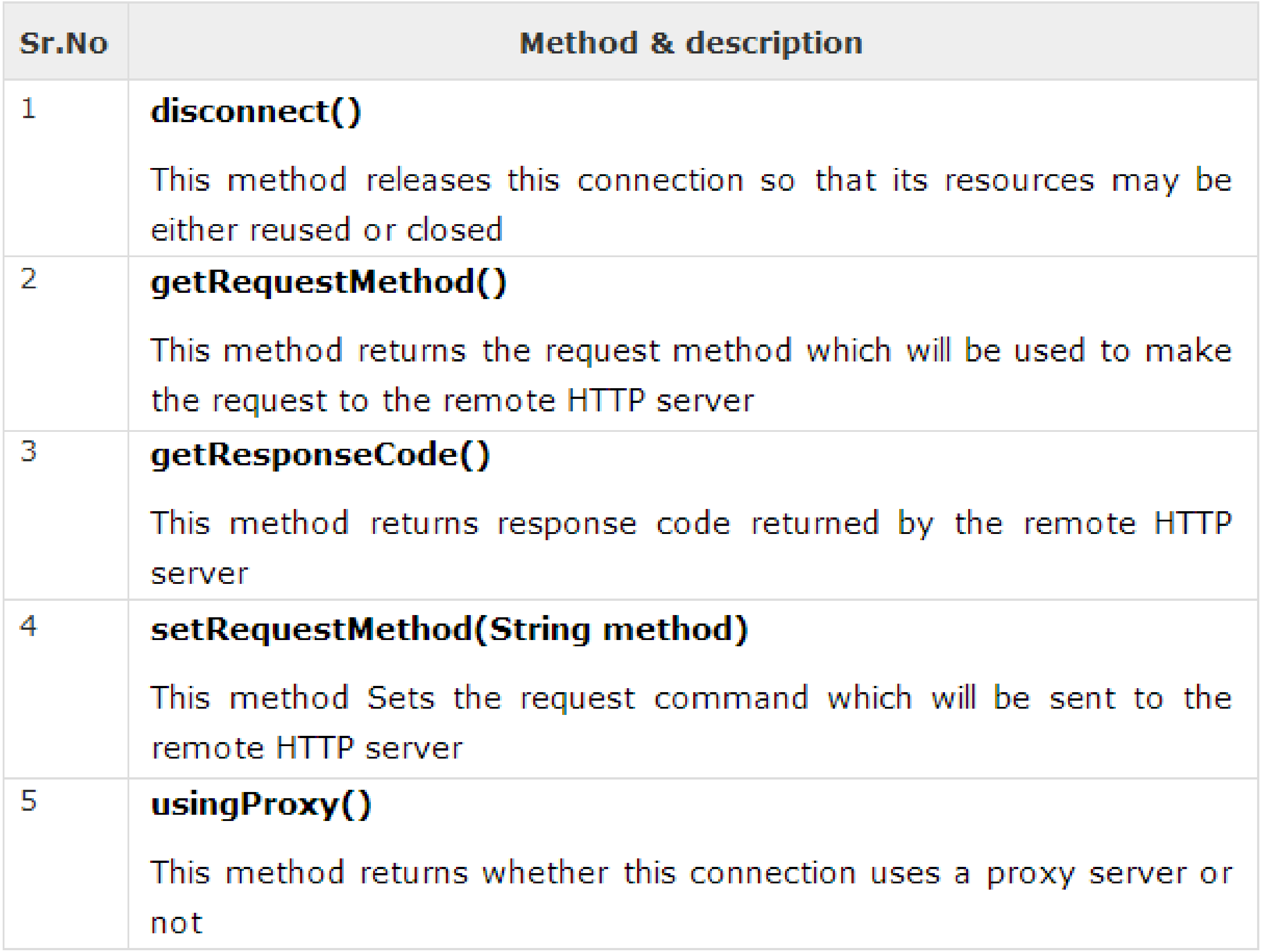
- First get the input stream from the connection
- Then create an input stream reader & a buffered reader
- Finally read data until no more data is available



```java
InputStream is = conn.getInputStream();
BufferedReader reader = new BufferedReader(new InputStreamReader(is, "UTF-8"));
String webPage = "",data="";

while ((data = reader.readLine()) != null){
   webPage += data + "\n";
}
```

Other Useful methods from the Connection



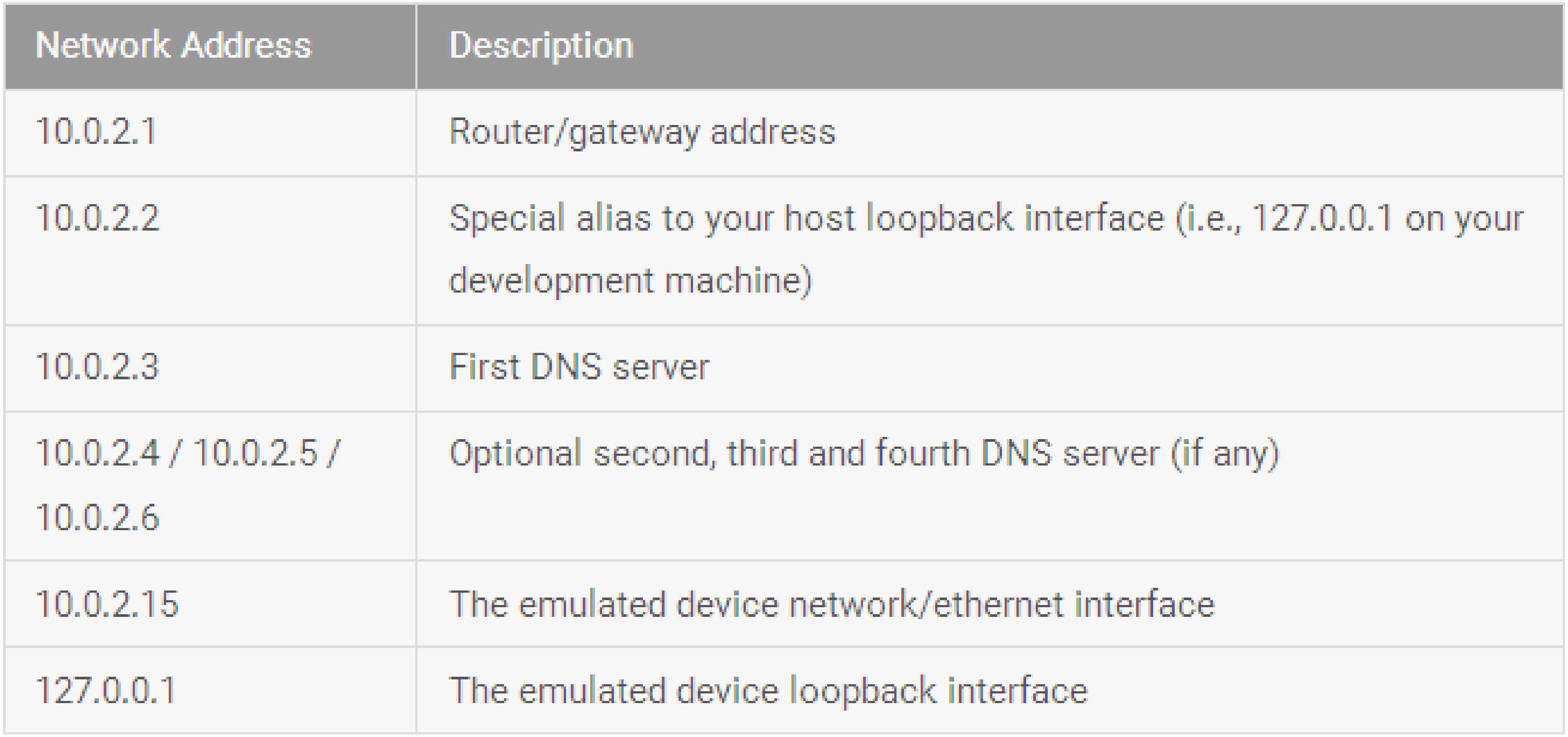
Sr.No	Method & description
1	**disconnect()** This method releases this connection so that its resources may be either reused or closed
2	**getRequestMethod()** This method returns the request method which will be used to make the request to the remote HTTP server
3	**getResponseCode()** This method returns response code returned by the remote HTTP server
4	**setRequestMethod(String method)** This method Sets the request command which will be sent to the remote HTTP server
5	**usingProxy()** This method returns whether this connection uses a proxy server or not

Slides By Alireza Kazemi,
Content From: https://developer.android.com/training/basics/network-ops/connecting.html &
https://www.tutorialspoint.com/android/android_network_connection.htm

Access To the Network from Android Virtual Device (AVD)

- How to connect from an AVD to the development machine or Internet:
  - Each AVD instance is not directly connected to the Developer machine (Host) OS
  - It is behind a **Virtual Router** which can map/forward some IP/Port addresses
  - The Virtual Router manages IP the pre-allocated address range 10.0.2.xx IP address range

AVD Network Address Space Pre Allocated Addresses



Network Address	Description
10.0.2.1	Router/gateway address
10.0.2.2	Special alias to your host loopback interface (i.e., 127.0.0.1 on your development machine)
10.0.2.3	First DNS server
10.0.2.4 / 10.0.2.5 / 10.0.2.6	Optional second, third and fourth DNS server (if any)
10.0.2.15	The emulated device network/ethernet interface
127.0.0.1	The emulated device loopback interface

Slides By Alireza Kazemi,
Content From: https://developer.android.com/training/basics/network-ops/connecting.html &
https://www.tutorialspoint.com/android/android_network_connection.htm

Sending a voice call or SMS to another emulator instance

- Dialing another AVD
  - Launch the dialer app on the originating emulator instance.
  - As the number to dial, enter the **console port number** of the instance you'd like to call. You can determine the console port number of the target instance by checking its window title
  - Press "Dial". A new inbound call appears in the target emulator instance.
- Sending an SMS to an AVD
  - It is same as dialing: just use the **AVD console port number** as the receiver phone number.