android系统实现上网机制

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **变更（+/-）说明** | **作者** | **版本号** | **日期** | **批准** |
| 1 | 初稿 | 崔慧娟 | V1.0 | 2017-04-17 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

目录

[android系统实现上网机制 1](#_Toc480298876)

[1 Android网络整体结构 3](#_Toc480298877)

[1.1 整体架构图 3](#_Toc480298878)

[２什么是DataConnection 4](#_Toc480298879)

[2.1 DataConnection 类的定义及子类 4](#_Toc480298880)

[2.2 DataConnection 类关键属性定义 5](#_Toc480298881)

[2.3 DataConnection的抽象类关键方法 6](#_Toc480298882)

[2.4 DataConnection 五个数据类之间的状态转化 6](#_Toc480298883)

[3 StateMachine状态机 7](#_Toc480298884)

[3.1 state设计模式 7](#_Toc480298885)

[3.2 StateMachine 核心类 7](#_Toc480298886)

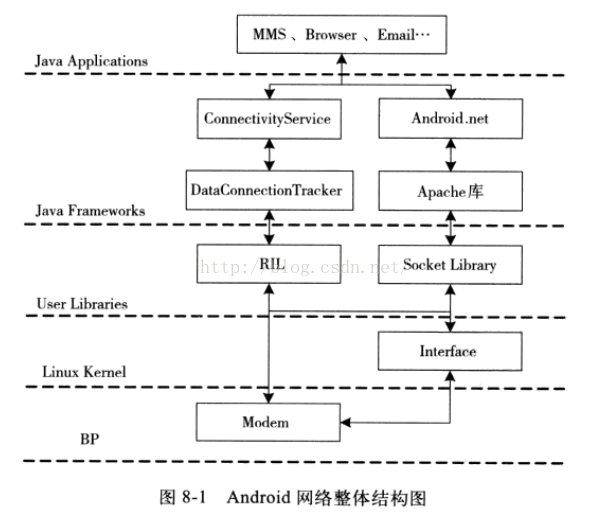
[3.3 StateMachine 运行机制 8](#_Toc480298887)

[3.4 DataConnection 使用State的运行机制 9](#_Toc480298888)

# 1 Android网络整体结构

## 整体架构图

Android 网络结构从上到下依次为 ：Java Applications、Java Frameworks和User Libraries层、linux kernel 四层。



这里为了方便理解将ＢＰ划分为第五层

在底层实现两个虚拟串口：

1、Command 命令通道：建立或断开网络建立

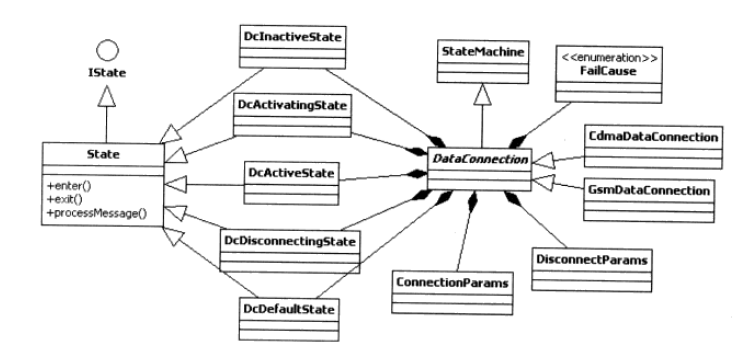
2、Data 数据通道：基于TCP/IP 网络通信协议的数据传输

# ２什么是**DataConnection**

**DataConnection 在Telephony Frameworks 层中用来管理手机上网连接**

Android 最大支持9个数据连接类型，但同一时刻仅有一个是有效的；

DataConnection 继承 StateMachine 类（State设计模式）的抽象类，其子类 GsmDataConnection 和 CdmaDataConnection；



## 2.1 DataConnection 类的定义及子类

**定义了6个内部类：**

DcDefaultState  默认状态定义，作为其他5个状态对象的父对象

DcInactiveState  不活动的状态定义，作为初始化状态，在不使用、断开数据连接、数据连接异常时回到此状态

DcActivatingState  正在激活的状态定义

DcActiveState  活动中的状态定义，连接网络成功

DcDisconnectingState  正在断开的状态定义

DcDisconnectionErrorCreatingConnection  断开失败并且正在创建的状态定义，连接网络失败

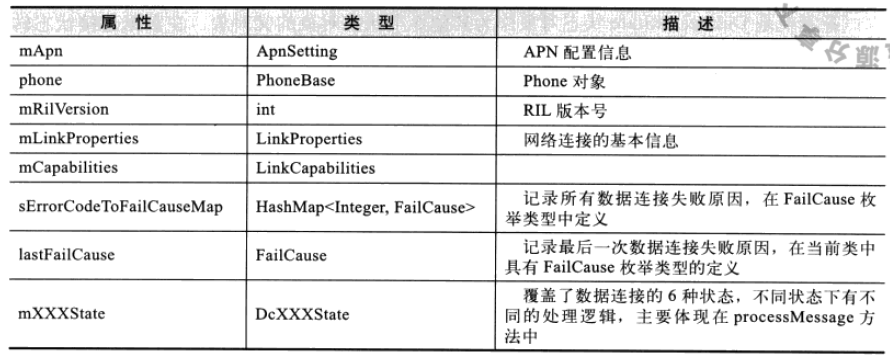
这些内部类都有enter、exit 和 processMessage 三个重要方法

enter ：在状态转换时，进入此状态的操作

exit ：在状态转换时，退出此状态的操作

processMessage ：根据Handler 消息类型进入不同的分支进行处理

## 2.2 DataConnection 类关键属性定义



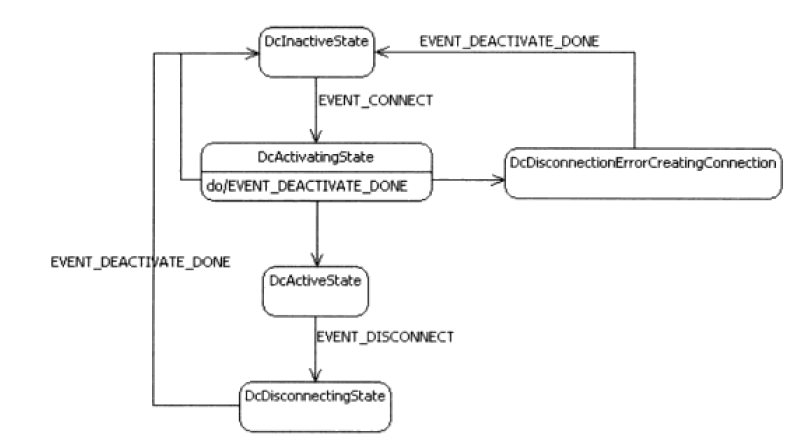
图中需要注意的地方：

1. 共有6个mXXXState对象属性，在数据连接的切换过程中或是不同的状态情况下，由他们的enter、exit、processMessage三个方法采用不通逻辑进行处理，这样减少了对状态的繁琐判断，这正是使用state设计模式的优势所在。
2. mApn保存当前连接的配置信息，APN相关内容后续讲解。
3. 具有phone对象的引用

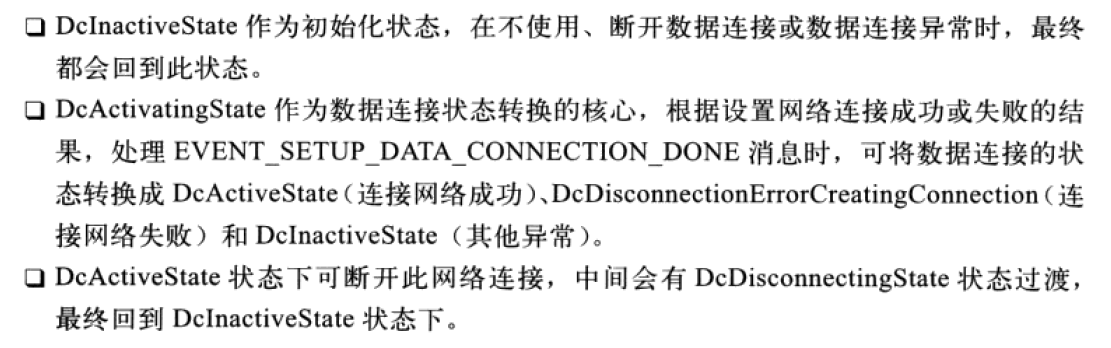
## 2.3 DataConnection的抽象类关键方法



## 2.4 DataConnection 五个数据类之间的状态转化



状态转化过程：

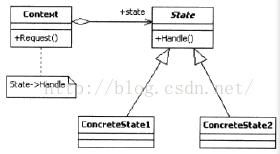


# 3 **StateMachine**状态机

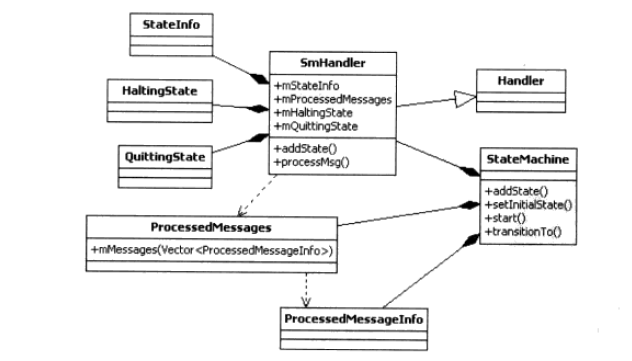
## 3.1 state设计模式

State 设计模式将对象的状态封装成一个对象，在不同状态下，同样的调用执行不同的操作；

Context 上下文维护对个State 子类的实例，每个实例实现该状态的操作；Context只需固定调用State.Handle方法，交由ConcreteState对象的Handle处理；



## 3.2 StateMachine 核心类



StateMachine 类共有6个内部类：

ProcessedMessageInfo  已处理的消息实体类

ProcessedMessages  已处理的消息类

SmHandler  消息处理核心类，负责Handler 消息的发生和接收用来管理和更新State对象

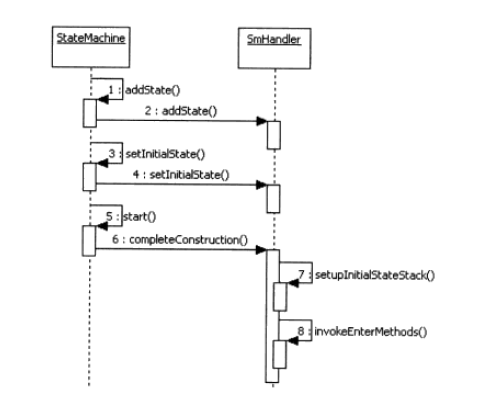
HaltingState  关闭的状态类

QuittingState  正在退出的状态类

StateInfo  状态实体类型

## 3.3 StateMachine 运行机制

StateMachine状态机的运行机制如下图：



StateMachine 运行过程：

addState  增加管理的状态对象

setInitialState  设置状态的初始化对象

start  开始运行状态机

addState 增加管理的状态对象，这些状态对象之间存在着树状关系，一个State对象不会处理的消息交由父节点处理，这是职责链设计模式

StateInfo  addState(State  state,  State  parent)

SmHandler 内部类作为自定义的Handler 消息处理对象

关于StateMachine的详细说明，感兴趣的同学可以自行学习

## 3.4 DataConnection 使用State的运行机制

private DataConnection(PhoneBase phone, String name, int id,

DcTrackerBase dct, DcTesterFailBringUpAll failBringUpAll,

DcController dcc) {

super(name, dcc.getHandler());

setLogRecSize(300);

addState(mDefaultState);

addState(mInactiveState, mDefaultState);

addState(mActivatingState, mDefaultState);

addState(mRetryingState, mDefaultState);

addState(mActiveState, mDefaultState);

addState(mDisconnectingState, mDefaultState);

addState(mDisconnectingErrorCreatingConnection, mDefaultState);

setInitialState(mInactiveState);

mApnContexts = new HashMap<ApnContext, ConnectionParams>();

if (DBG) log("DataConnection constructor X");

}

start() 启动方法的调用则是在子类GSMDataconnection和CDMADataconnection 类完成

# 4 什么是APN

APN的英文全称是Access Point Name，中文全称叫接入点，是您在通过手机上网时必须配置的一个参数，它决定了您的手机通过哪种接入方式来访问网络.

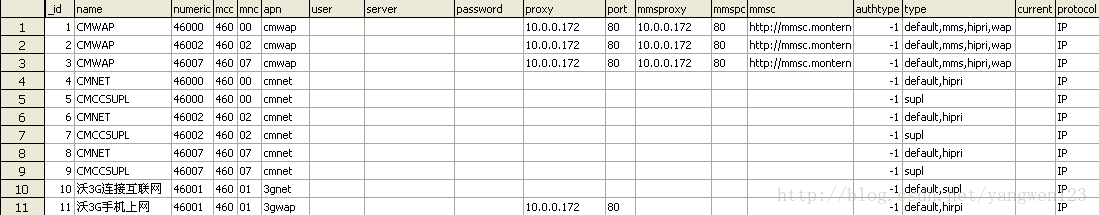
在启动[Android](http://lib.csdn.net/base/android" \o "Android知识库" \t "_blank)手机或者启动Android虚拟设备后，所有的APN配置信息都会保存在telephony.db的SQLite[数据库](http://lib.csdn.net/base/mysql)表名为carriers的表中。我们可以将此数据库文件pull到本地，然后可以查看carriers表的结构和其中的APN配置信息数据，命令如下：

 adb pull /data/data/com.android.providers.telephony/databases/telephony.db D:/telephony.db

 我们可以在D盘中找到生成的名称为telephony.db的数据库。

或者直接手动将此数据库拷贝出来查看。

android通过telephony.db数据库中的 carriers表来保存所有的APN配置信息。



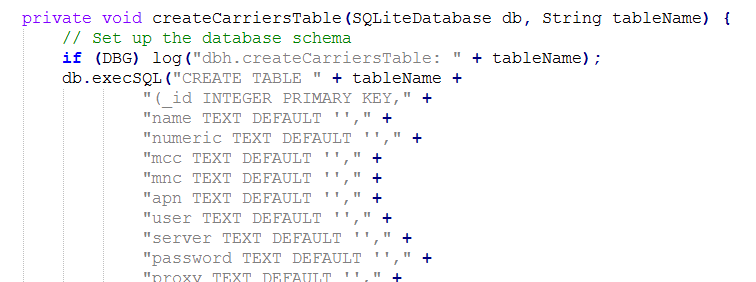
## 4.1 APN配置关键字



存储apn的数据库telephony.db的相关代码在android\packages\providers\TelephonyProvider\

src\com\android\providers\telephony\ TelephonyProvider.java 文件中，

其中创建表的SQL为 ：



虽然字段众多但是我们从用途上可以把apn分为发送彩信和上网两种，其中上网的apn又分为wap和net。

## 4.2 Android支持的apn类型

Android中最多支持5种类型的APN，其功能如下所示：

|  |  |
| --- | --- |
| **类型** | **描述** |
| **default** | 默认数据连接，即浏览器、Email等普通连接 |
| **mms** | 接收和发送彩信使用的数据连接 |
| **supl** | 支持APGS的数据连接 |
| **dun** | 拨号连接 |
| **hipri** | 扩展 |

此表中的数据优先级是由低到高的，即default数据连接的优先级最低，而hipri数据连接的优先级最高。比如在手机上网聊天时，将建立default数据连接；当手机收到一条彩信，因为彩信的数据连接是mms，这时会断开default数据连接而创建mms数据连接，从而能快速接收到此彩信，因为mms比default的数据连接优先级高。因此，在发送和接收彩信的同时不能上网。

## 4.3 Android中如何新增apn信息

Apn信息对于Android设备上网是很重要的，一旦apn信息配置有误，有可能导致设备无法上网。

我们来看一下apn信息是如何配置的。

在系统的/system/etc/ 目录下面有这样的文件 apns-conf.xml 文件



注意： 这里需要注意的是，修改时一定要注意格式正确，这里只有换行符，没有回车符。

配置成功之后，首先将系统里面的

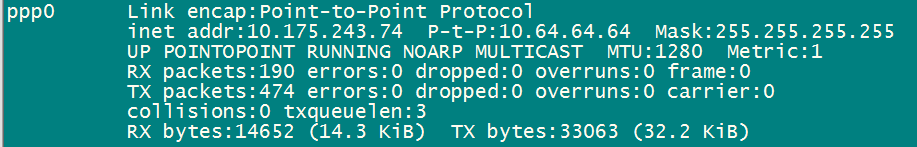
/data/data/com.android.providers.telephony/databases/telephony.db 删除掉，然后再替换掉/system/etc/ apns-conf.xml 再重启系统。 之后再将生成的telephony.db 拿出来，然后用工具查看是否已经生成。

# 5 开机自动建立default默认数据连接

## 5.1 如何查看Android设备的基本上网信息

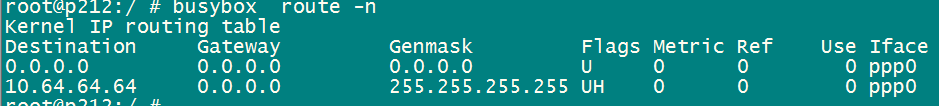
1> 查看当前的网卡信息

命令 ： busybox ifconfig



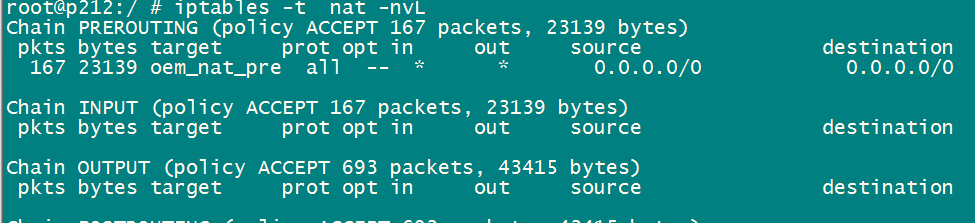
2> 查看当前的路由信息

命令 ： busybox route



3> 查看当前的iptables 信息

命令： iptables -t  nat –nvL



主要分三步完成DataConnection 数据连接的创建和设置：

1、初始化ApnContext，APN 上下文信息

2、SIM 卡加载完成后，设置创建DataConnection 数据连接

3、激活创建的DataConnection 数据连接

## 5.2初始化ApnContext

initApnContextAndDataConnection 获取 defaultEnable 和 networkConfigStrings 参数

networkConfigStrings 方法在 frameworks/base/core/res/res/values/configs.xml 配置文件中定义

onSetDependencyMet ，applyNewState 更新数据连接的状态；