JPCAPJava中的数据链路层控制 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/252/2021_2022_JPCAP_E2_80_94_E2_80_c104_252502.htm 一．JPCAP简介 众所周知，JAVA语言虽然在TCP/UDP传输方面给予了良好的定义，但对于网络层以下的控制，却是无能为力的。JPCAP扩展包弥补了这一点。 JPCAP实际上并非一个真正去实现对数据链路层的控制，而是一个中间件，JPCAP调用wincap/libpcap，而给JAVA语言提供一个公共的接口，从而实现了平台无关性。在官方网站上声明，JPCAP支持FreeBSD 3.x, Linux RedHat 6.1, Fedora Core 4, Solaris, and Microsoft Windows 2000/XP等系统。 二．JPCAP机制 JPCAP的整个结构大体上跟wincap/libpcap是很相像的，例如NetworkInterface类对应wincap的typedef struct _ADAPTERADAPTER，getDeviceList()对应pcap_findalldevs()等等。 JPCAP有16个类，下面就其中最重要的4个类做说明。 1．NetworkInterface 该类的每一个实例代表一个网络设备，一般就是网卡。这个类只有一些数据成员，除了继承自java.lang.Object的基本方法以外，没有定义其它方法。数据成员NetworkInterfaceAddress[]java.lang.Stringdatalink_description.数据链路层的描述。描述所在的局域网是什么网。例如，以太网（Ethernet）、无线LAN网（wireless LAN）、令牌环网(token ring)等等。java.lang.Stringdatalink_name 该网络设备所对应数据链路层的名称。具体来说，例如Ethernet10M、100M、1000M等等。java.lang.Stringdescription网卡是XXXX牌子XXXX型号之类的描述。例如我的网卡描述：Realtek

RTL8169/8110 Family Gigabit Ethernet NIC booleanLoopback标志这个设备是否loopback设备。byte[]mac_address网卡的MAC地址，6个字节。java.lang.StringName这个设备的名称。例如我的网卡名称
：\Device\NPF_{3CE5FDA5-E15D-4F87-B217-255BCB351CD5}
2．JpcapCaptor 该类提供了一系列静态方法实现一些基本的功能。该类一个实例代表建立了一个与指定设备的链接，可以通过该类的实例来控制设备，例如设定网卡模式、设定过滤关键字等等。数据成员int0dropped_packets抛弃的包的数目。protected intID这个数据成员在官方文档中并没有做任何说明，查看JPCAP源代码可以发现这个ID实际上在其JNI的C代码部分传进来的，这类本身并没有做出定义，所以是供其内部使用的。实际上在对JpcapCator实例的使用中也没有办法调用此数据成员。protected staticboolean[]instanciatedFlag同样在官方文档中没有做任何说明，估计其为供内部使用
。protected staticintMAX_NUMBER_OF_INSTANCE同样在官方文档中没有做任何说明，估计其为供内部使用
。intreceived_packets收到的包的数目方法成员staticNetworkInterface[]getDeviceList() 返回一个网络设备列表。staticJpcapCaptoropenDevice(NetworkInterface interface, intsnaplen, booleanpromisc, intto_ms) 创建一个与指定设备的连接并返回该连接。注意，以上两个方法都是静态方法
。Interface：要打开连接的设备的实例；Snaplen：这个是比较容易搞混的一个参数。其实这个参数不是限制只能捕捉多少数据包，而是限制每一次收到一个数据包，只提取该数据包中前多少字节；Promisc：设置是否混杂模式。处于混杂模式

将接收所有数据包，若之后又调用了包过滤函数setFilter()将不起任何作用；To_ms：这个参数主要用于processPacket()方法，指定超时的时间；voidClose()关闭调用该方法的设备的连接，相对于openDivece()打开连接
。JpcapSendergetJpcapSenderInstance() 该返回一个JpcapSender实例，JpcapSender类是专门用于控制设备的发送数据包的功能的类。PacketgetPacket() 捕捉并返回一个数据包。这是JpcapCaptor实例中四种捕捉包的方法之一
。intloopPacket(intcount, PacketReceiver handler) 捕捉指定数目的数据包，并交由实现了PacketReceiver接口的类的实例处理，并返回捕捉到的数据包数目。如果count参数设为－1，那么无限循环地捕捉数据。这个方法不受超时的影响。还记得openDivice()中的to_ms参数么？那个参数对这个方法没有影响，如果没有捕捉到指定数目数据包，那么这个方法将一直阻塞等待。PacketReceiver中只有一个抽象方法void receive(Packet p)。intprocessPacket(intcount, PacketReceiver handler) 跟loopPacket()功能一样，唯一的区别是这个方法受超时的影响，超过指定时间自动返回捕捉到数据包的数目
。intdispatchPacket(intcount, PacketReceiverhandler) 跟processPacket()功能一样，区别是这个方法可以处于“non-blocking”模式工作，在这种模式下dispatchPacket()可能立即返回，即使没有捕捉到任何数据包
。voidsetFilter(java.lang.Stringcondition, booleanoptimize) .condition：设定要提取的包的关键字。Optimize：这个参数在说明文档以及源代码中都没有说明，只是说这个参数如果为真，那么过滤器将处于优化模式

。voidsetNonBlockingMode(booleannonblocking)如果值为“ true ”，那么设定为“ non-blocking ”模式
。voidbreakLoop()当调用processPacket()和loopPacket()后，再调用这个方法可以强制让processPacket()和loopPacket()停止。