http://wiki.jikexueyuan.com/project/java-nio-zh/java-nio-datagramchannel.html

一个Java NIO DatagramChannel是一个可以发送、接收UDP数据包的通道。由于UDP是面向无连接的网络协议，我们不可用像使用其他通道一样直接进行读写数据。正确的做法是发送、接收数据包。

## 打开一个DatagramChannel（Opening a DatagramChannel）

打开一个DatagramChannel你这么操作：

```
DatagramChannel channel = DatagramChannel.open();
channel.socket().bind(new InetSocketAddress(9999));
```

上述示例中，我们打开了一个DatagramChannel，它可以在9999端口上收发UDP数据包。

## 接收数据（Receiving Data）

接收数据，直接调用DatagramChannel的receive()方法：

```
ByteBuffer buf = ByteBuffer.allocate(48);
buf.clear();
channel.receive(buf);
```

receive()方法会把接收到的数据包中的数据拷贝至给定的Buffer中。如果数据包的内容超过了Buffer的大小，剩余的数据会被直接丢弃。

## 发送数据（Sending Data）

发送数据是通过DatagramChannel的send()方法：

```
String newData
    = "New String to wrte to
file..."+System.currentTimeMillis();
ByteBuffer buf = ByteBuffer.allocate(48);
buf.clear();
buf.put(newData.getBytes());
buf.flip();
int byteSent
    = channel.send(buf, new InetSocketAddress("jenkov.com",
80));
```

上述示例会把一个字符串发送到“jenkov.com”服务器的UDP端口80.目前这个端口没有被任何程序监听，所以什么都不会发生。当发送了数据后，我们不会收到数据包

是否被接收的的通知，这是由于UDP本身不保证任何数据的发送问题。

## 链接特定机器地址（Connecting to a Specific Address）

DatagramChannel实际上是可以指定到网络中的特定地址的。由于UDP是面向无连接的，这种链接方式并不会创建实际的连接，这和TCP通道类似。确切的说，他会锁定DatagramChannel,这样我们就只能通过特定的地址来收发数据包。

看一个例子先：

```
channel.connect(new InetSocketAddress("jenkov.com"), 80);
```

当连接上后，可以向使用传统的通道那样调用read()和Writer()方法。区别是数据的读写情况得不到保证。下面是几个示例：

```
int bytesRead = channel.read(buf);
int bytesWritten = channel.write(buf);
```