任务：为网络仿真增加信源支持和最简单的“存储转发”机制

目标：本次迭代完成后，事实上已经实现了Flooding协议中最基础的功能，即对网络中任意一个结点而言，它在收到一个message之后，能够再转手将其转发出去。只不过这里的转发（Forward）并不是选择一个邻居结点发送出去，而是直接广播（Broadcast）给所有邻居所有邻居结点。至于信源，本质上是对应用层传感监测功能的模拟，技术上就是一个能够间歇性发送消息（Message）的模块，以使得整个仿真过程能够长久运行下去。其中的每个message的属性，由信源模块负责填充，对其中的接收结点标识，可以用随机数发生器生成。好的信源模拟，应能尽可能真实的反映实际中传感器模块的功能和行为。

要求

R1：创建一个 Message 类，代替以前字符串类型的message。

目前的代码中用一个字符串来表示在网络中传输的字符串。但是要做Flood仿真，势必涉及到要在message中保存一些属于网络层Flood协议需要利用的属性，例如source、destination、ttl、maxlifetime、payload等。一个变通的办法是在传输时把这些属性全部编码成一个字符串，这样就可以利用目前底层的传输机制。事实上，message当前尽管是string类型，但是应视之为 字符数组，可以自行规定第几个字符表示何种含义。

为了支持上述做法，可以在 Message 类中增加 encode 和decode两个方法，encode方法负责将当前Message的各个属性编码成一个字符串，decode则逆向操作之。至于编码后的格式，是采用二进制还是json格式自行选择。如果希望编解码过程的代码尽可能简单通用，可以选择JSON格式，尽管它的效率比较低，但胜在互联网应用开发中常用，参考资料较多。

R2：review代码，凡是出现收发送字符串message的地方，改为{Message}.encode()或{Message}.decode()。

并调试程序直至通过。

以上两步改动不大，但很重要，对下面进入实质性的Flood协议开发至关重要，无法也不能跳过。

R3：创建类 SimuSensorApplication，模拟传感器生成数据的功能，可以从TestBroadcastNode类中的嵌入类 BroadcastApplication 拷贝代码。这就是信源。

SimuSensorApplication 内部包含一个随机生成数据包（即message)的功能。每次调用其 generateMessage()时，就可以得到一个新的数据包。

目前 TestBroadcastNode第157行：

root.sendMessage( "test message", bcApp );

今后这句代码应该写成如下形式

root.sendMessage(simusensor.generatedMessage().encode())代替。

即待发送数据来自 sensor。类似的，相应接收message的代码中，应该在收到原始message（此时仍为字符串形态）后调用 {Message}.decode()方法，将其中编码的各个属性值解码到Message类中，然后再进行可视化显示。

以上改动完成后，应调试测试通过。这一步很重要。

R4：创建 FloodRouterApplication，继承自Application

其功能是实现message的接收和广播转发。

注意需要定义两个队列 rxque<Message>、txque<Message>。其中，rxque和txque分别是当前结点设备的收取和发送队列。队列元素为Message类型 或 string类型。不妨先用 string 类型。

收到的数据首先进入rxque，设置队列的目的是因为可能短时间内有多个message到达，queue的引入可以有效降低丢包概率。然后为该类增加一个 evolve() 方法，在evolve方法中去检查rxque是否非空，如非空则取出头部message对象并填入txque等待被转发走。

本仿真因为比较简单，所以txque如果取消、只设置一个能容纳一个message对象的txbuf应该也可以。

可参考 TestBroadcastNode 中的嵌入类 BroadcastApplication。

R5：创建 FloodNode，继承自Mica2Node，可仿效TestBroadcastNode

但要用FloodRouterApplication代替TestBroadcastNode类中的嵌入类 BroadcastApplication再测试仿真。

至此，我们应该实现了一个仅具有数据包复制和广播转发功能的flood协议及其测试。