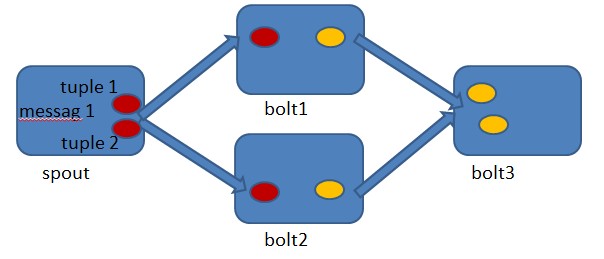
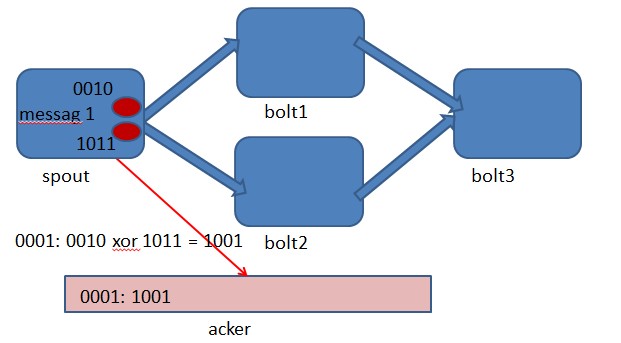
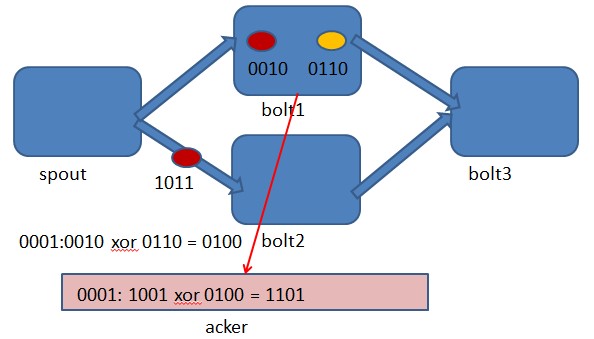
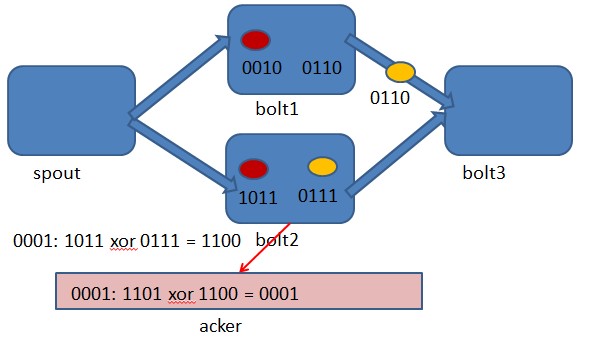
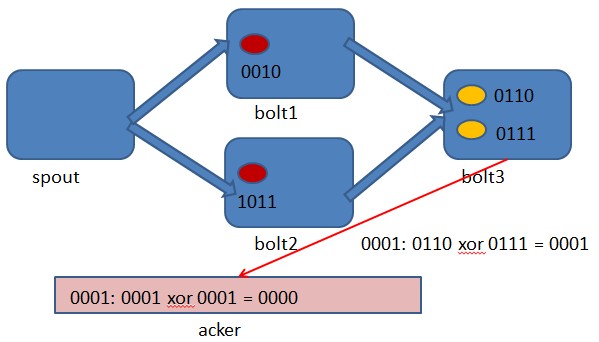
* **Storm记录级容错的基本原理**

首先来看一下什么叫做记录级容错？storm允许用户在spout中发射一个新的源tuple时为其指定一个message id, 这个message id可以是任意的object对象。多个源tuple可以共用一个message id，表示这多个源 tuple对用户来说是同一个消息单元。storm中记录级容错的意思是说，storm会告知用户每一个消息单元是否在指定时间内被完全处理了。那什么叫做完全处理呢，就是该message id绑定的源tuple及由该源tuple后续生成的tuple经过了topology中每一个应该到达的bolt的处理。举个例子。在图4-1中，在spout由message 1绑定的tuple1和tuple2经过了bolt1和bolt2的处理生成两个新的tuple，并最终都流向了bolt3。当这个过程完成处理完时，称message 1被完全处理了。  
[](http://www.searchtb.com/wp-content/uploads/2012/08/message10.jpg)  
图4-1

在storm的topology中有一个系统级组件，叫做acker。这个acker的任务就是追踪从spout中流出来的每一个message id绑定的若干tuple的处理路径，如果在用户设置的最大超时时间内这些tuple没有被完全处理，那么acker就会告知spout该消息处理失败了，相反则会告知spout该消息处理成功了。在刚才的描述中，我们提到了”记录tuple的处理路径”，如果曾经尝试过这么做的同学可以仔细地思考一下这件事的复杂程度。但是storm中却是使用了一种非常巧妙的方法做到了。在说明这个方法之前，我们来复习一个数学定理。

A xor A = 0.

A xor B…xor B xor A = 0，其中每一个操作数出现且仅出现两次。

storm中使用的巧妙方法就是基于这个定理。具体过程是这样的：在spout中系统会为用户指定的message id生成一个对应的64位整数，作为一个root id。root id会传递给acker及后续的bolt作为该消息单元的唯一标识。同时无论是spout还是bolt每次新生成一个tuple的时候，都会赋予该tuple一个64位的整数的id。Spout发射完某个message id对应的源tuple之后，会告知acker自己发射的root id及生成的那些源tuple的id。而bolt呢，每次接受到一个输入tuple处理完之后，也会告知acker自己处理的输入tuple的id及新生成的那些tuple的id。Acker只需要对这些id做一个简单的异或运算，就能判断出该root id对应的消息单元是否处理完成了。下面通过一个图示来说明这个过程。  
[](http://www.searchtb.com/wp-content/uploads/2012/08/ack10.jpg)  
图4-1 spout中绑定message 1生成了两个源tuple，id分别是0010和1011.  
[](http://www.searchtb.com/wp-content/uploads/2012/08/ack20.jpg)  
图4-2 bolt1处理tuple 0010时生成了一个新的tuple，id为0110.  
[](http://www.searchtb.com/wp-content/uploads/2012/08/ack30.jpg)  
图4-3 bolt2处理tuple 1011时生成了一个新的tuple，id为0111.  
[](http://www.searchtb.com/wp-content/uploads/2012/08/ack40.jpg)  
图4-4 bolt3中接收到tuple 0110和tuple 0111，没有生成新的tuple.

可能有些细心的同学会发现，容错过程存在一个可能出错的地方，那就是，如果生成的tuple id并不是完全各异的，acker可能会在消息单元完全处理完成之前就错误的计算为0。这个错误在理论上的确是存在的，但是在实际中其概率是极低极低的，完全可以忽略。