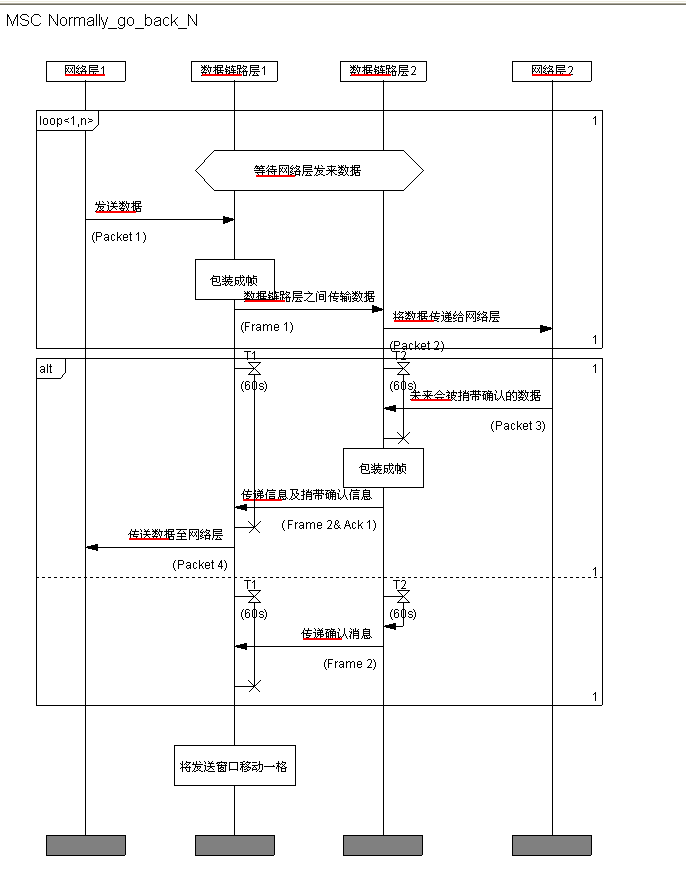
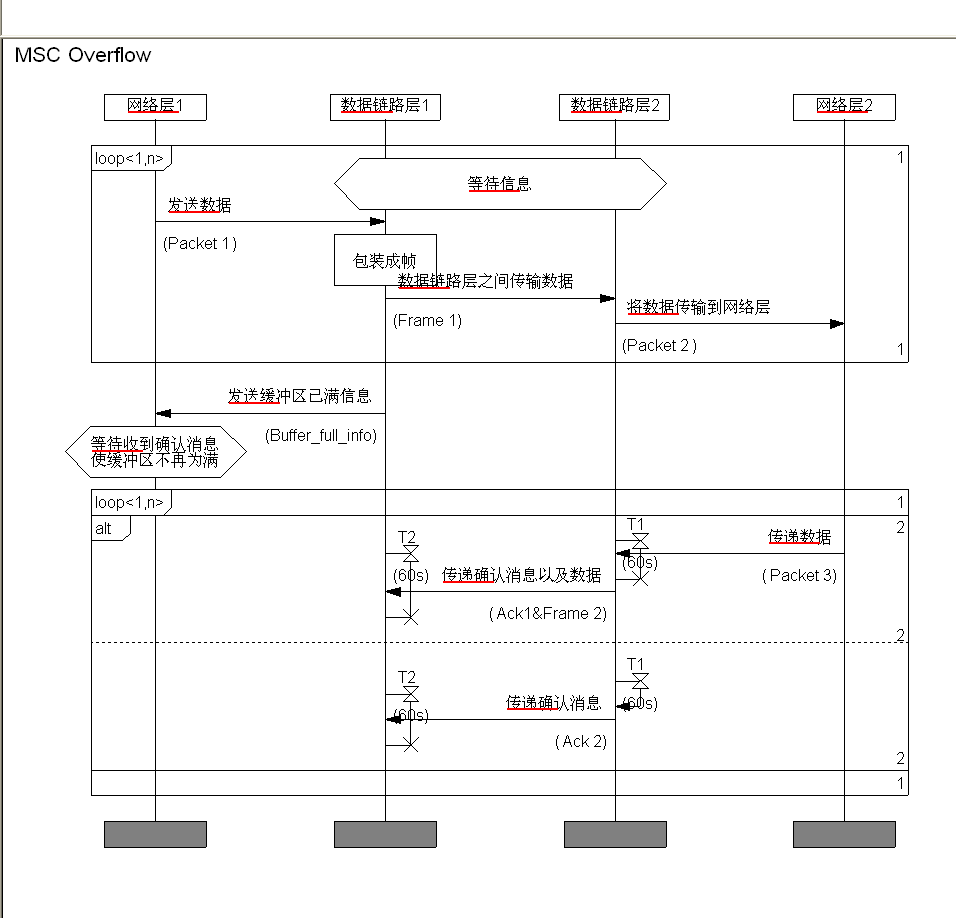
正常情况：

首先数据链路层等待网络层发来数据，当收到数据后将数据包装成帧，并且将其传输至另一端的数据链路层和网络层。此过程会循环多次，下面就进入了二选一的情况也就是是否会出现捎带确认的消息，使用一个定时器，如果在定时器时间范围内数据链路层2收到了来自网络层的消息，那么将这个消息和Ack消息一起包装成帧，然后将这些信息传递给对方数据链路层，这时就完成了捎带确认的工作，结合课上的学习，这里的捎带确认避免了只传递确认信息不传递数据造成的浪费。同时还有另一种情况就是网络层2在一段时间内没有发送消息的需求，等待一定的时间，当定时器超时就证明最近一段时间没有需要传输的数据，这时直接向数据链路层1传递Ack消息即可。当完成这些操作以后将发送窗口往后移一格。



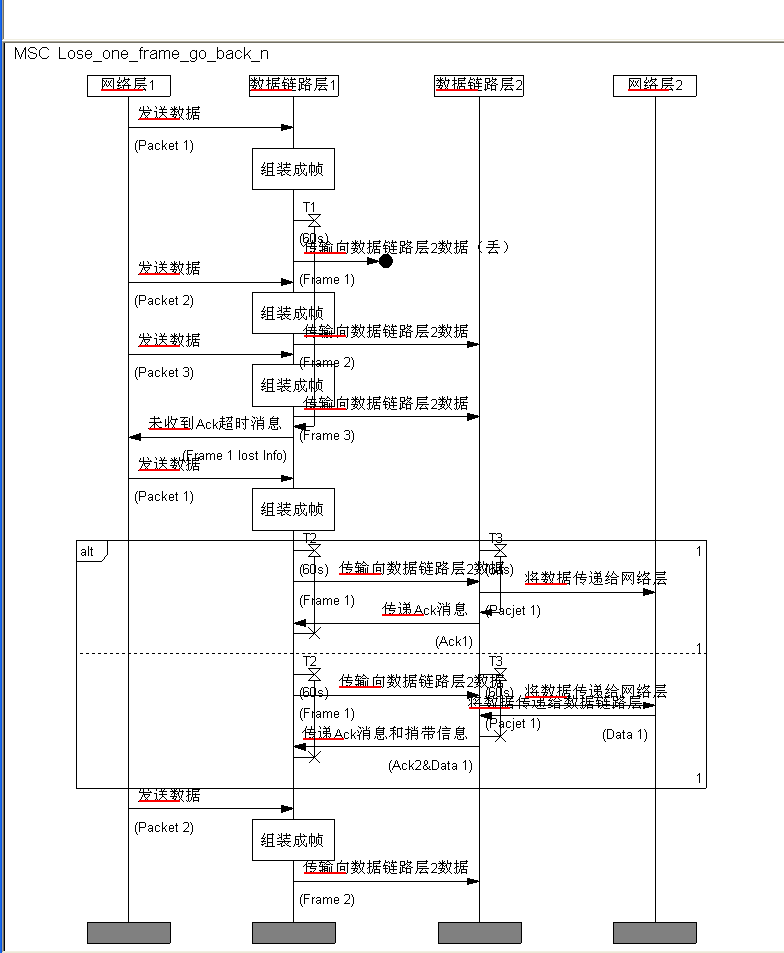
缓存区满后的处理

与上一张正常情况的MSC图类似，也是多次循环等待消息，从网络层1传递数据给数据链路层1，在数据链路层之间传递数据，最终将数据传递给网络层，多次循环这个读取消息的操作。当发现缓冲区已满的时候数据链路层1会自动发出缓冲区已满的消息，这个时候网络层进入了等待状态，当数据链路层收到确认消息之后减少了缓冲区的数据以后才可以继续传输数据。获得确认消息的方法和之前相同，也是有两种方式，即是否会使用捎带确认的消息，使用一个定时器，如果在定时器时间范围内数据链路层2收到了来自网络层的消息，那么将这个消息和Ack消息一起包装成帧，然后将这些信息传递给对方数据链路层，这时就完成了捎带确认的工作，结合课上的学习，这里的捎带确认避免了只传递确认信息不传递数据造成的浪费。同时还有另一种情况就是网络层2在一段时间内没有发送消息的需求，等待一定的时间，当定时器超时就证明最近一段时间没有需要传输的数据，这时直接向数据链路层1传递Ack消息即可。



丢失一帧：

丢失帧的情况比较能体现回退n的操作，首先发送一条数据，也是从网络层发送到数据层，组装成帧以后向对方数据链路层发送数据，这个时候假设丢失了这条数据，也就是Frame1，当刚刚丢失的时候网络层1和数据链路层1是无法发现丢失数据的。这时网络层将在不知情的情况下继续发送数据，直至第一帧在等待Ack的定时器超时的时候会发现，一直未收到响应消息。在此之前我们假设已经发送了packet2和packet3两个数据包，但是网络层收到数据链路层发出的一直没有收到对方反馈的Ack消息的时候就知道Packet1没有传送到，于是就会重发Packet1，并且继续按照顺序重新发送Packet2和Packet3（虽然之前已经成功发送了2和3，但是由于回退n的特性，会在重发1的时候重新发送2和3，确定1接受后才能接受2）。



Ack消息超时：

除了之前描述的丢失普通的帧以外，还有可能会丢失Ack消息，与之前一样也是首先等待发送数据，当网络层将数据传送给数据链路层的时候包装成帧在数据链路层之间进行传输，并最终传输给网络层，这个时候数据链路层2应当回复给数据链路层1Ack消息，或者捎带确认的消息，加入这个Ack消息丢失，就会导致在等待回复消息的定时超时，数据链路层1就会认为对方没有收到消息或者Ack消息丢失，它会从缓冲区里重新发送消息给数据链路层2然后继续等待响应消息。这个过程也是有捎带确认和直接返回Ack消息的丢失两种情况。

