**电子科技大学**

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

**最小网元设计阶段二报告**



专 业 计算机科学与技术（互联网+）

学 号 **2018080901001，2018080901023，2018080901026**

姓 名 陈秉辉，余明哲，郑仪涛

目录

[一． 层级结构 3](#_Toc22502615)

[二． 帧结构 5](#_Toc22502616)

[三． 控制信息 6](#_Toc22502617)

[四． 结果展示 8](#_Toc22502618)

[五． 可改进的地方 10](#_Toc22502619)

1. 层级结构

项目提供了一个物理层的模拟软件，我们就在物理层的模拟软件的上面加上了两层，数据链路层和网络层。

示意图如下：

网络层2

数据链路层2

物理层

物理层

网络层1

数据链路层1

1. 网络层

我们设置的网络层的主要作用就是让使用者输入一个字符串（这个字符串可以是“hello”这些，也可以是由文件或者图片转化成的二进制流信息），然后将这个字符串信息按照一个字符八个二进制位依据ASCII码来转化成一个比特串，例如字符a，ASCII码值为97，对应的八位二进制位为01000001，这样构造字符的一一对应关系（也由于使用了ASCII码来编，设置的网络层是无法传输中文信息的，因此不能输入中文）。

由于两边的网络层是对等的，所以是一样的，可以选择两种状态，分别是接收状态和发送状态，接收状态就只需要等待对应下层的数据链路层发送比特信息之后按照与刚才编码的反过程解码即可。

网络层的部分代码：



设置端口，初始化的过程。

1. 数据链路层

这层也具有两种状态，接收状态和发送状态。

在接收状态下，数据链路层接收来自下层物理层的信息，其中该信息以扩展帧的形式被传输到链路层，在该帧被确认之后，链路层将返回一个应答ack1，表示让对方传下一帧，若该帧出错，则返回一个应答ack0，表示让对方重传，在链路层把该扩展帧转化为帧，然后再转化为比特流的形式，最后再将所有比特流整合一并传输给对应的网络层。

在发送状态下，数据链路层接收来自网络层的比特流信息，并将之转化为帧，之后再将帧相应的扩展（具体下面帧结构会讲到），然后一帧一帧地发送给物理层，发送之后等待对方的应答信息来决定要重传还是继续传下一帧，当所有帧发送完成并且都接收到了对方的确认信息之后，链路层会返回一个所有数据都已正确的传输给对方的信息给自己的网络层，同时，网络层接收该信息，此次信息传输完成。

示意图：字符a的传输为例，图中帧（图里无）和扩展帧分别为：

01111110-00000000-01000001-10111110-01111110

01111110-00000000-01000001-101111100-01111110

网络层2

数据链路层2

物理层

物理层

网络层1

数据链路层1

a

01000001

扩展帧

a

01000001

扩展帧

1. 帧结构

帧结构示意图：

帧头8位

序号8位

数据内容32位

校验码8位

帧尾8位

帧头、帧尾：为01111110，用来帧同步。

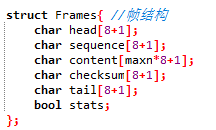
序号：用来记录帧序号，防止乱序，范围0-255，其中序号0代表结束帧。

数据内容：最大32位，4个字节，用来存储要传输的数据，其中每八位算一个字符。

校验码：方式为因特网校验和，用来差错控制。

由于帧头、帧尾用于帧同步，因此在除了帧头帧尾以外不能含有序列01111110，所以在帧头帧尾之间只要出现5个连续的1，那么就在这后面加一个0，也就形成了扩展帧，而传到另一边之后要把扩展帧转变回为帧。

定义的帧结构代码：

 其中，布尔型的stats代表该帧

是否使用，true为使用，false表

示没被使用。

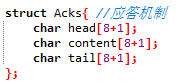
**其他结构：**

应答机制的Ack结构：

帧头、帧尾：一样是01111110，用于同步。

内容：Ack1内容为111110111，Ack0内容为00000000，其中，Ack1表示接收的帧验证正确，Ack0表示接收的帧错误，要求重传。

代码：



还有比特结构，不多赘述。

1. 控制信息
2. 帧同步

由于通过物理信道后，传输的数据前后可能会加上一些干扰的01串，因此需要帧同步来找出帧的位置，这里我们用帧头帧尾来解决这个问题，帧头帧尾都为01111110，我们可以保证帧内不包含和帧头帧尾一样的序列，而随机加上的干扰串出现和帧头帧尾相同序列的概率也非常小，因此我们只需要通过在帧的两头加上帧头和帧尾就可以实现帧同步。

1. 差错控制

由于信道一般用模拟信号传输，且会有其他的一些噪音等的干扰存在，因此信道都会存在一定的误码率，也就是可能会把原来的0干扰之后变为1，或者将1干扰为0，甚至可能某位的0或者直接被干扰而被丢弃，这里我们不考虑最后一种情况，因此物理层的模拟软件也是没有比特丢弃这种可能的。而前面的两种情况我们用因特网校验码来判断传输过来的某个帧是否发生了误码，我们把除了帧头帧尾之外的所有序列都考虑进入校验码（不考虑帧头帧尾的原因是，1.如果帧头或者帧尾出错，那么大概率就找不到帧头或帧尾，也就无法识别出内容，那么这种情况显然要求重传;2.帧头帧尾序列都不变，加入到校验码也就等于没加入），下面举例来说明：

帧(出去帧头帧尾部分)：00000000（序号）-11111111（内容）

那么校验码为：（序号+内容）再取反（注意：这里的加法是循环加法），也就是00000000

可以得到整个帧为：00000000-11111111-00000000

传到对方之后，也做一遍循环加法（但不取反），如果加上传过来的帧的校验码的和如果是11111111，也就是255的话，就说明没出错。

如果序号或者内容或者校验码的某位或者某几位出错了，明显会被校验出来，此时要求重传即可。

1. 流量控制

其实这部分我们不是特别理解，流量控制就是可能一次数据发送过多导致一些数据会被丢弃，因此所有的信息传输要在最大传输范围内，因此我们选择的方法是一个帧一个帧的发送，在发送完一个帧之后等待对方发送应答信息之后再发送下一个帧。

具体的时序图如下所示：

帧i

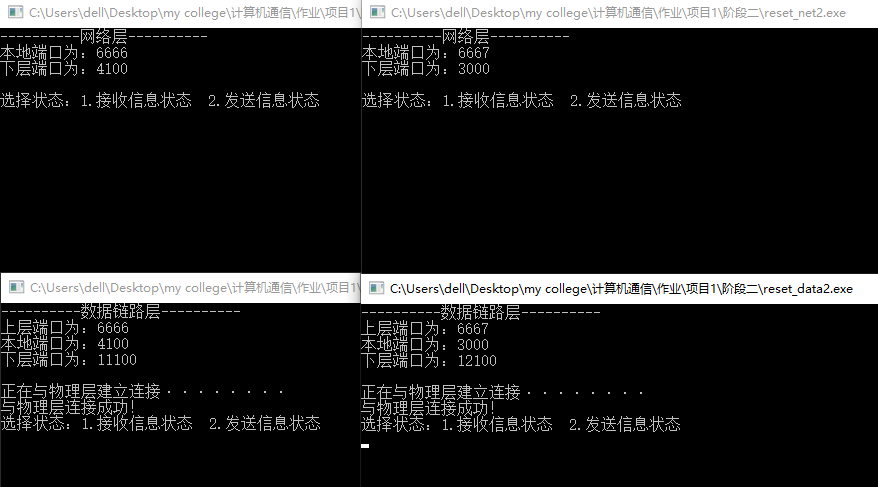
Ack

帧i+1

Ack

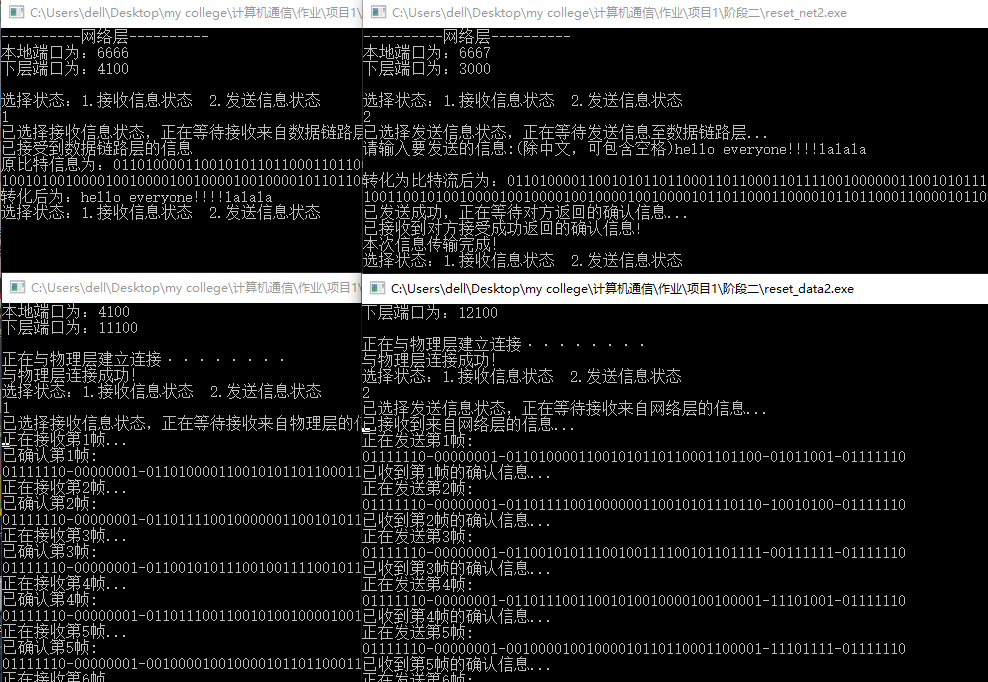
帧i+2

1. 结果展示



上图为初始状态，每个层都有两个状态，接收或者发送，也绑定了自己的端口，而对于数据链路层，会先与物理层进行连接，但是同边的层，也就是在同一边的网络层和数据链路层要选择相同的状态，而剩下的状态另一边要选择。

接下来选完状态之后，在发送状态的网络层输入一个字符串（可包含空格，但是不能包含中文），如下图：



左边为接收状态，右边为发送状态，可以看到在发送的那一段输入了一个字符串之后，数据链路层开始互动，最后在对方的网络层接收到了相同的信息，完成了这一次的信息传输，可以再次选择状态以开始下一次的信息传输。

1. 可改进的地方

由于时间的关系，阶段二的任务虽然算是完成了，但是仍然有非常多可以改进的地方，在此记录下来，之后希望可以完成这些改进。

1. 每个层都需要选择状态，较麻烦

因为目前还没用到select函数，因此必须要选择状态来确定接下来的一系列使用sendto函数或者recvfrom函数，这个问题可以通过select函数或者多线程（一个线程只管接收，一个只管发送）来解决。

1. 无法传输中文信息

由于编码方式选择的是ASCII码，因此只能包含有ASCII码定义了的字符，后面可以通过utf-8的编码方式来解决这个问题。

1. 手动输入而非自动传输或者可以选择图片等等

现在还是只能通过手动输入一个字符串来传输，希望之后可以通过一些函数可以选择将一些图片等转成比特流而传输或者可以自动传输一些特定的信息。