****



**RDT实验报告**

**学生姓名 李丰杰**

**班 级 计科四班**

**学 号 3019244196**

1. 协议算法

本次实验实现了Stop-and-Wait和Go-Back-N两种基于传输层的单向传输协议算法。实验提供了网络仿真过程的模拟环境、数据报的数据结构、数据报收发的具体代码。在此基础上，只需要实现每个节点的传输层代码，即只需设计传输协议即可。

1.1 协议算法原理

Stop-and-Wait协议是基于停等策略的协议，发送方每发完一个分组都要停止发送，等待接收方的确认信息，当正确收到确认信息后再发送下一分组。在教材及现实中对该算法有很多种版本的实现，本次实现的是教材中的“rdt3.0协议”，其特点是：全面考虑了比特错误、丢包、乱序等信道的潜在错误、不使用NAK。

我们以问题为导向来介绍该协议的具体原理与过程。

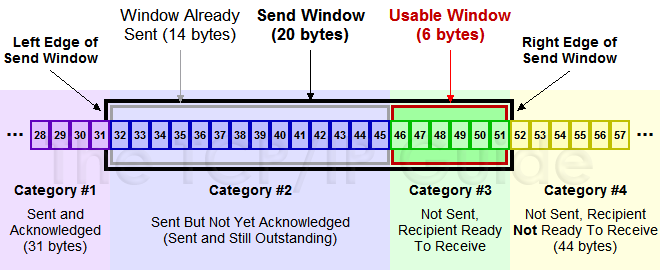
为了解决比特错误，该停等协议要有检测数据完整性的数据，所以在数据报中引入了校验和字段、序列号和ACK号字段。校验和可用来检测传输的数据是否被损坏，在检测到损坏后，接收方进行相应的响应让发送方重传；但如果接收方响应的ACK或者NAK(虽然本协议中没有NAK)损坏，发送方该如何响应？为了简化对该错误的处理过程，我们引入了序列号和ACK号。发送方只需要对发送信息编码(在停等协议中只需要1bit)，接收方就可以检查序列号来确定该数据是否为一次重传，而不需要频繁通话或者增加更多的校验和位数。

为了解决丢包问题，该停等协议需要实现超时重传的功能。因此，为了实现基于时间的重传机制，需要一个计时器(本次实验中已经提供，只需调用即可)。发送方应该做到：每发送一个分组(包括重传分组)便启动定时器、对计时器中断(超时)能正确反应(重传)、停止计时。在本次实验中，我们只需要调用对应的函数即可。

Stop-and-Wait是一个停等协议很容易实现，但是他最大的问题也是因为他是一个停等协议。可以看到，Stop-and-Wait协议在每发一个数据报后就停止，对网络的利用率十分低下，即使是在一个高速信道中，因为协议自身问题，他的效率也不会有太大的提升。

为了解决这个问题，我们提出了流水线可靠传输协议的概念，即允许发送方发送多个分组而无须等待确认。这也带来了新的问题：必须增加序列号范围(1bit的序列号不够用)；协议的发送方和接收方需要缓存多个分组(发送方至少需要缓存那些发送但没有确认的分组，接收方至少缓存正确接收的分组)；错误处理需要针对多个分组进行，而根据错误处理的分类,我们提出了Go-Back-N协议。

Go-Back-N协议虽然允许发送方可以连续发送多个分组，但他对当前已发送但未确认分组的数量有限制(上限为N)。发送方中的分组可以分为4部分：已经发送并确认的分组、已经发送但未确认的分组、亟待发送的分组、不可发送的分组。根据下面的示意图可以形象地将其描绘成**滑动窗口协议。**



图：GBN协议中发送方视角(滑动窗口)

在GBN中，发送方必须响应三类事件：上层调用、收到ACK、超时事件。对于数据包损坏的情况可以继续沿用停等协议的方法。在接收ACK时，GBN协议采用累积确认的方法，即发送方在接收到序号为n的ACK后会认为在n之前(包括n)的所有分组都成功被接收方收到。

在接受方中，如果他收到一个有序且正确的数据报后，他会对它进行处理传递给上层。在所有其他情况下，都将数据包丢弃，并对发送方发回对应的信息。即若分组n丢失，后续分组接收方都会直接丢弃掉。这种方法可以让接收方不需要缓存任何失序分组，只需要维护下一个按序接收的分组的序号即可。

1.2协议算法执行的过程

针对Stop-and-Wait算法，发送方的执行过程可以根据下面的有限状态机来表示。