**RDT 第一周实验报告**

**学 号 3020244294**

**姓 名 石子跃**

**学 院 智能与计算**

**专 业 网络安全**

**年 级 2020**

**任课教师 仇超**

**2022年 4月 30日**

### 1. 协议需求分析

#### Stop-and-Wait

**需求**: 如何检测丢包，需要增加一种新的协议机制，比特交错协议。

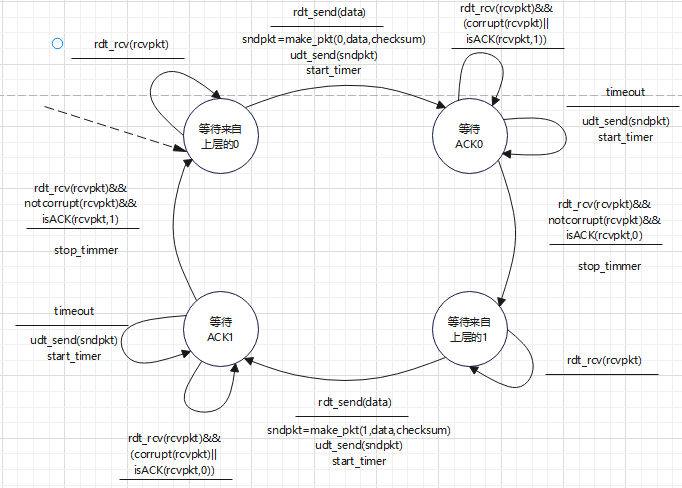
**所适用的场景**: 这是一个功能正确的协议。但是性能方面不佳，特别是在今天的高速网络中更是如此。定义发送方的利用率为：发送方实际发送比特进信道的那部分时间与发送时间之比，分析表明效率极低。而且我们还忽略了底层协议处理时间，路由器的处理和排队时延等问题，这将进一步增加时延，使其性能更加糟糕。发送方这时发送数据包每次只能发送一个，并且必须等待这个数据包的ACK，才能发送下一个。虽然在效率上比较低，带宽利用率明显较低，不过在网络环境较差，或是带宽本身很低的情况下，还是适用的。

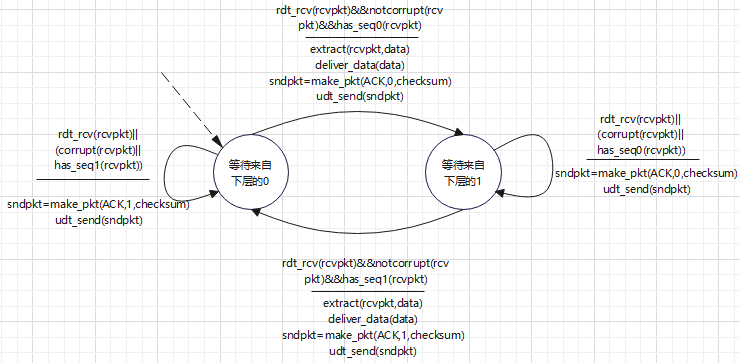
### 2. 协议的设计

#### Stop-and-Wait

**设计原理**：停等协议(stop－and－wait)，这时接受方的窗口和发送方的窗口大小都是1，1个比特就够表示了，所以也叫比特滑动窗口协议。发送方这时自然发送每次只能发送一个，并且必须等待这个数据包的ACK，才能发送下一个。

**FSM 图**：





### 3. 协议的实现

#### Stop-and-Wait

**主要数据结构**：

struct Sender

{

  int state;//当前01比特顺序

  int acknum;//

  float time\_out;//rtt最长等待时间

  struct pkt last\_pkt;//上一个传输的数据包

} A;

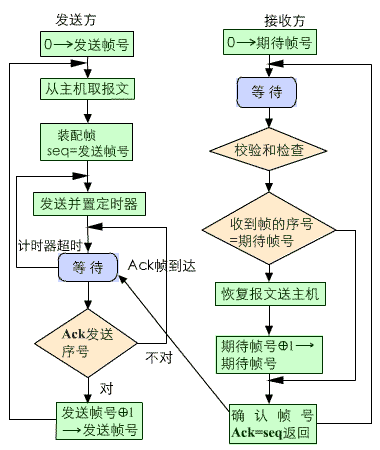
struct Reciver

{

  int seqnum;//01交错

} B;

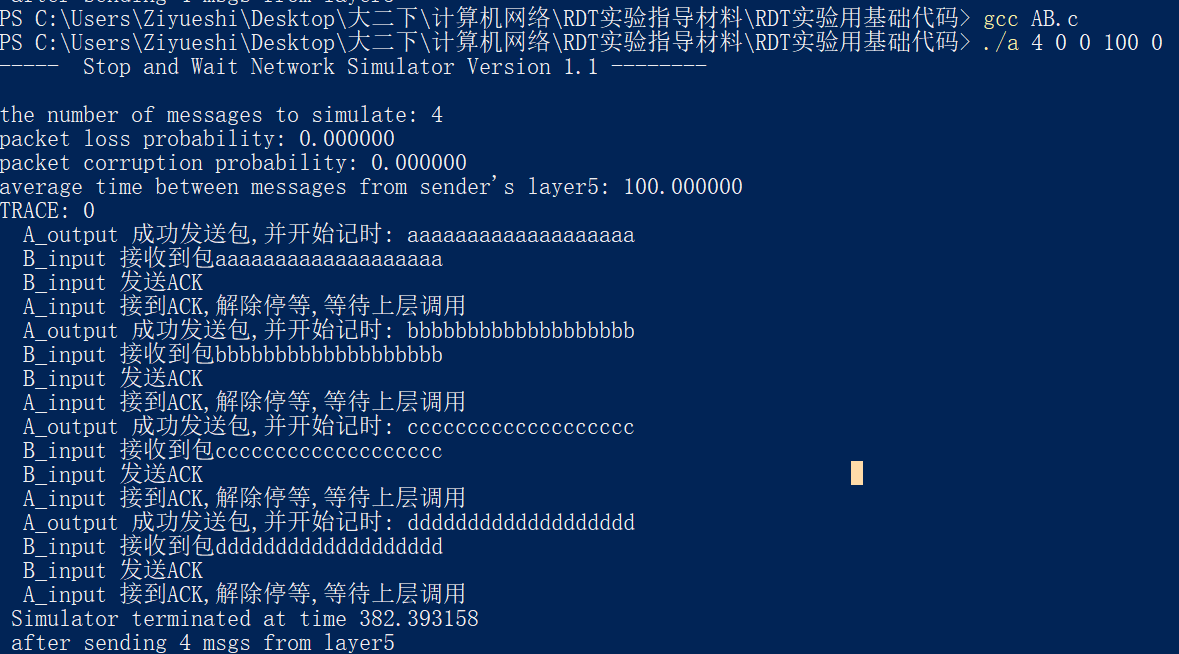
**流程图**：



### 4. 协议功能测试

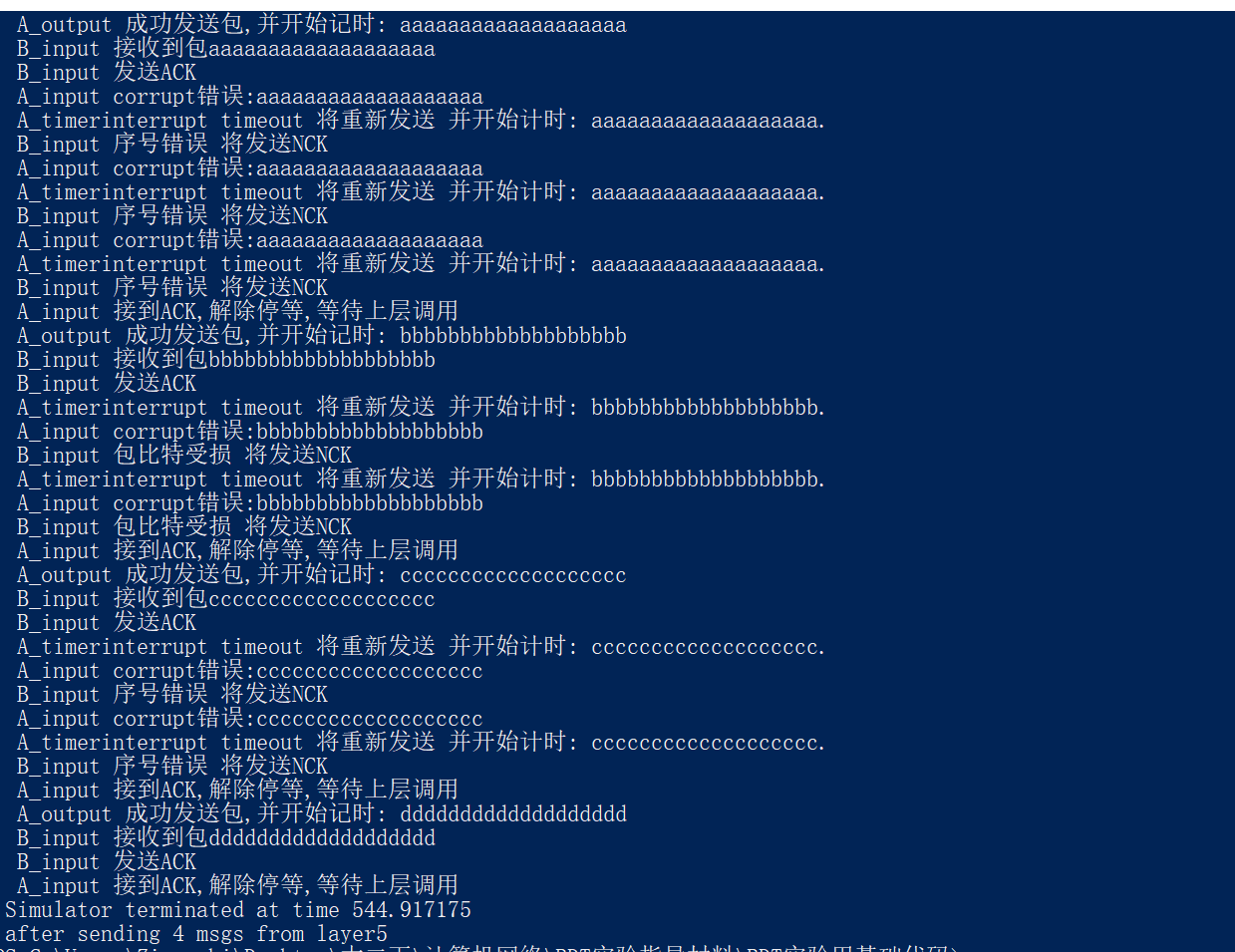
#### Stop-and-Wait

1) 无 error 无 loss；



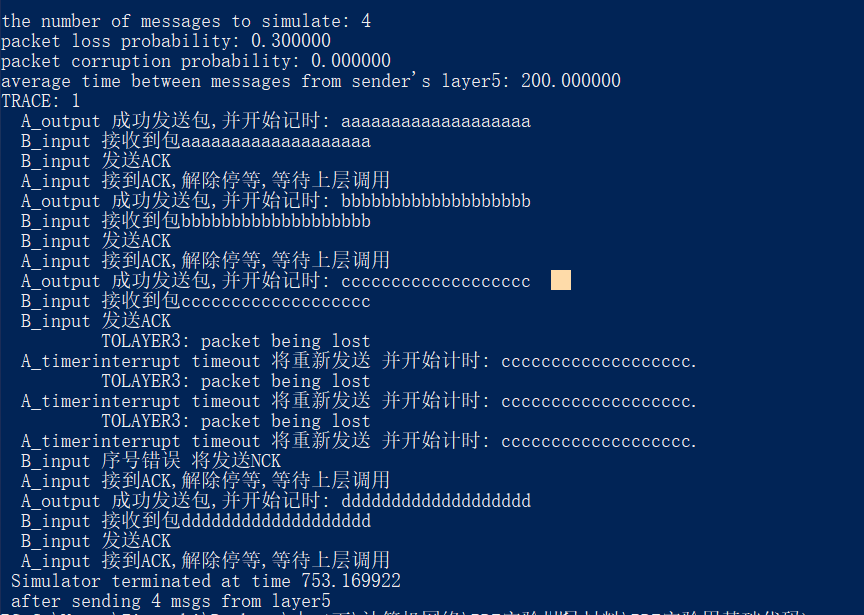
符合FSM，当A发送一个数据包后要等待Binput发送的ACK，由于此测试五error，无loss，所有顺序正确。

2) 数据包、ACK 包出现 error；



发送a时，B返回的ACK包出现错误，A会重新发送a,结果B检查到比特交错错误,会发送nck，这时又有A的时间超时又发送了一次a，最后接到了ack.于是A端口可以继续接受顶层的调用，发送b，可能是由于设置的超时时间太短总容易触发超时，后面的情况如上所述。

3) 数据包、ACK 包有 loss；



前两个和第四个包很正常，但是c居然lost了三次，每一次都在超时后重新发送了c,但是此时B已接受c，所以B返回NCK，这时A收到后会解除限制接受上层调用。

### 总结

1. 深入理解可靠数据传输RDT的基本原理。

2. 掌握Stop-and-Wait协议的设计方法，能够使用FSM图进行设计。

3. 掌握Stop-and-Wait协议的实现方法。

**本周任务完成表**（以第一周为例）。

在“完成”“没完成”列对应打“√”

|  |  |  |  |
| --- | --- | --- | --- |
| 本周任务  要求 | 完成 | 没完成 | 备注 |
| 协议需求分析。  分别阐述两种协议的需求和所适用的网络场景。 | √ |  |  |
| 协议设计。  使用FSM工具分别设计两种协议的发送端和接收端的协议流程，并给出相应FSM图。 | √ |  |  |
| 协议实现。  根据FSM图在仿真框架下分别实现Stop-and-Wait和Go-Back-N两种协议。分别说明两种协议实现的具体方法、主要数据结构，给出流程图或者伪代码。 | √ |  |  |
| 实验结果及分析。  1）能够按要求给出协议功能测试用例，并展示结果。测试要求详见下文“补充说明”部分。  2）能够对结果进行合理的分析说明。 | √ |  |  |