

# 天津大学

## 《计算机网络》课程设计报告



RDT 第一周：Stop-and-Wait 协议实现

学 号 6319000359

姓 名 张明君（留学生）

学 院 智能与计算学部

专 业 计算机科学与技术

年 级 2019 级

任课教师 石高涛

2022 年 05 月 01 日

## 一、实验目的

1. 深入体会可靠数据传输的思想和理念。
2. 加深对 Stop-and-Wait 协议的理解。
3. 掌握 Stop-and-Wait 协议的具体实现方式。

## 二、实验内容

编写传输层代码，实现单向传输情景下的 Stop-and-Wait 和 Go-Back-N 两种协议。为专注于协议的开发，本次实验已经提供了诸如网络仿真过程的模拟代码、数据报具体收发的代码、节点除传输层以外的各层功能代码等大量基础代码。这些基础代码已经构建起了一套完善的网络仿真环境。

只有每个节点的传输层代码部分留空，需要实验人员自行填补，具体包括节点传输层的初始化操作、节点接收到应用层消息的处理过程、节点接收到网络层数据包的处理过程、节点计时器到时的响应过程等内容。

为了方便试验人员进行实验，本实验还提供了以下实现好的过程供试验人员调用：`starttimer()`和`stoptimer()`：试验人员可以通过调用此函数启动和停止节点的计时器。`tolayer3(calling_entity, packet)`：试验人员可以通过调用此函数时，传入需要发送的`packet`，将数据包传递至网络层。这之后的传输工作将由框架自动完成。`tolayer5(calling_entity, message)`：试验人员可以通过调用此函数时，传入`message`，将消息传递至应用层。这之后仿真框架将检查数据是否完好、是否按序到达。

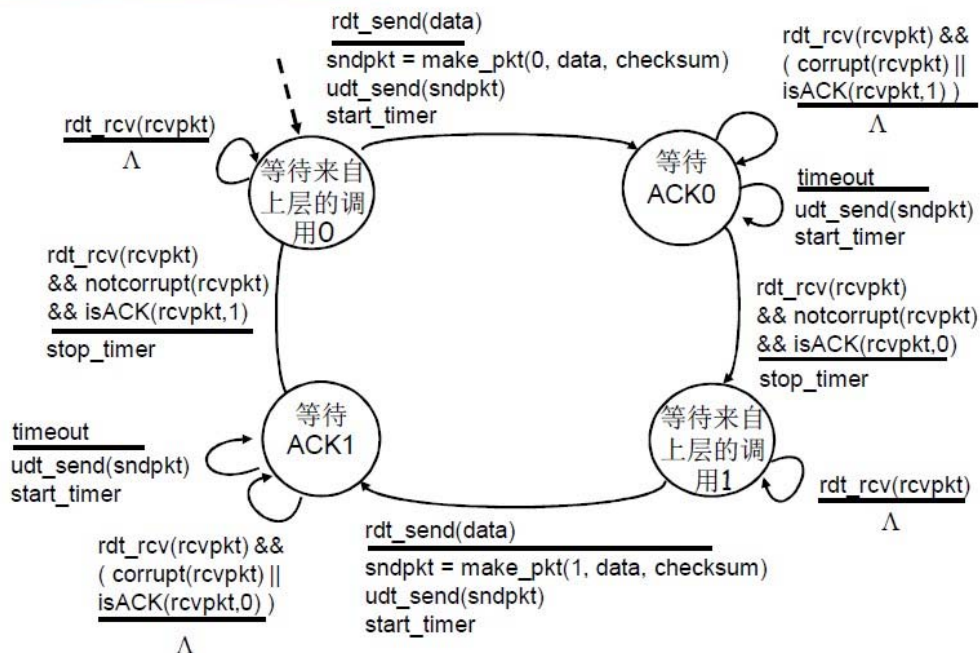
实验人员完成所有节点的算法后，运行网络仿真，记录仿真过程和结果，并对结果进行分析和总结。

## Stop-and-Wait 协议原理和过程

stop-and-wait 协议基于 RDT3.0 来实现，也就是在在最接近真实的网络环境的情况下实现两个实体之间的信息的传输。在这种情况下，下层信道，在本实验中也就是啊网络层，可能对丢失分组，这其中包括但不限于数据或者是 ACK。对于这些情况，RDT3.0 给出了解决方法。在发送端设定一个计时器，如果数据发送出去之后，在超时之前没有受到 ACK，无论是由于丢包了还是 ACK 在传输的过程中被扰乱了，发送端都会重新来发送数据。

Stop-and-wait 协议的过程可以由一下这个 FSM 来表示：

## rdt3.0 发送方



发送方等待来自上层 0 的调用，发送完信息之后出于等待 ACK0 的阶段，如果出现了超时或者是包被破坏 的阶段，就再次发送数据包，再次出于等待 ACK0 的阶段。如果数据发送从成功，就切换状态，出于等待 上层 1 的调用，发送完数据包的之后，切换状态到等待 ACK1 的阶段，如果遇到了超时重传或者 ACK 错误，那么就重新发送数据。如果数据发送成功就进入等待来自下一个状态，也就是等待来自上层 1 的调用。

## 算法实现规划

在给出的代码框架中，给出了 A、B 两个实体的发送函数，接收函数和计时器。为了能够更好的规范实体，这里只考虑了由 A 发送数据到 B 的过程，反向的过程是一样的，所以我在代码中定义了两个实体 A、B。

由于实体 A 是发送方，所以实体 A 的结构体中包含了所处的状态，序列号，预估等待时间（用于计算超时），以及上一个成功发送的数据包。B 作为接收方，就只需要一个序列号。

## 实现具体细节

1. 首先，定义计算校验和的公式，将这个 package 中的序列号和确认号都加起来，再加上负载中数据 的值，这样能够保证，在计算校验和的时候把一个数据包中所有的数据都包含了。
2. 定义 A 实体的输出：首先，只有当 A 实体正处于等待调用的时候才能发送数据，然后构造一个 package，把 message 中的数据 copy 到 package 中，再给数据包中设置这个数据的序列号。
3. 在 A 实体的输入端口：如果传入给 A 的数据报的校验和与 A 接收到以后计算出来的不一样那么就出错了。或者 A 接收到的 ack 和 A 的 seq 不一样也会报错。
4. 对于 B 的设计和 A 相同。

## 实现过程中遇到的问题

在一开始的时候，没有定义 A 和 B 的结构体，在设置传输的时候就设置了很多变量，管理起来很麻烦。在 设置 A 的输出函数的时候，当 A 传输数据完毕之后没有更改 A 的状态，导致 A 一直处于等待 ACK 的状态。所以在设置的时候，要多次设置 print 语句。

## 网络仿真的过程

网络仿真即实现相应函数功能后直接整体运行框架代码即可实现网络仿真过程。每次仿真过程需要设置相应参数来模拟各种情况：传递信息的总数、出现比特差错的比例、出现分组丢失的比例以及上层数据到发送方的时间间隔。按照实验文档，终测试条件为 10messages，丢包率 0.1，损坏率 0.3，trace 设置为 2。

## 仿真的结果

在最终仿真测试中，算法对于各种情况的处理均符合预期要求，针对各种事件的运行情况见下一节的内容。

## 算法分析

我在代码中加入了 Makefile 在运行的时候：

```
$ mingw32-make
$ rdt
```

这是我在 Window 的 cmd 中测试的结果：

例如：

```
Enter the number of messages to simulate: 10
Enter packet loss probability [enter 0.0 for no loss]:0.1
Enter packet corruption probability [0.0 for no corruption]:0
Enter average time between messages from sender' s layer5 [ > 0.0]:5
Enter TRACE: 2
```

数据报和 ack 均没有发生 loss 和 error:

```
EVENT time: 0.467849, type: 1, fromlayer5 entity: 0
  A_output: send packet: aaaaaaaaaaaaaaaaaa

EVENT time: 5.960295, type: 2, fromlayer3 entity: 1
  B_input: rcv message: aaaaaaaaaaaaaaaaaa
  B_input: send ACK.

EVENT time: 8.038575, type: 1, fromlayer5 entity: 0
  A_output: not yet acked. drop the message: bbbbbbbbbbbbbbbbbbb

EVENT time: 8.459425, type: 2, fromlayer3 entity: 0
  A_input: acked.
```

可以看到发送方收到上层数据后发送并启动定时器，在时限内接收方接收到正确的数据且为当前期望序号。接着反馈一个 ack 到发送方，发送方确认接收方已正确接收分组后改变状态为等待上层调用。

数据包或 ACK 出现 error、数据包或 ACK 的序列号不是期望序号:

```
C:\Users\MSI-PC\Desktop\This PC\study\计算机网络\RDT>rdt
----- Stop and Wait Network Simulator Version 1.1 -----

Enter the number of messages to simulate: 10
Enter packet loss probability [enter 0.0 for no loss]:0.1
Enter packet corruption probability [0.0 for no corruption]:0
Enter average time between messages from sender's layer5 [ > 0.0]:5
Enter TRACE:2

EVENT time: 0.467849, type: 1, fromlayer5 entity: 0
  A_output: send packet: aaaaaaaaaaaaaaaaaa

EVENT time: 5.960295, type: 2, fromlayer3 entity: 1
  B_input: rcv message: aaaaaaaaaaaaaaaaaa
  B_input: send ACK.

EVENT time: 8.038575, type: 1, fromlayer5 entity: 0
  A_output: not yet acked. drop the message: bbbbbbbbbbbbbbbbbbb

EVENT time: 8.459425, type: 2, fromlayer3 entity: 0
  A_input: acked.

EVENT time: 11.610157, type: 1, fromlayer5 entity: 0
  A_output: send packet: cccccccccccccccccc
  TOLAYER3: packet being lost
```

这类错误处理方式基本一致，都是由于接收到的数据包不满足要求，故无法到下一个阶段。数据包出错，接收方会检测到比特差错，然后向发送方发 NAK(即上一个序号的 ACK)。若是 ack 发生了 error，发送方检测到错误后不进行任何动作，等待直到超时。

数据报或 ack 发生 lost

```
EVENT time: 0.467849, type: 1, fromlayer5 entity: 0
  A_output: send packet: aaaaaaaaaaaaaaaaaa

EVENT time: 5.960295, type: 2, fromlayer3 entity: 1
  B_input: recv message: aaaaaaaaaaaaaaaaaa
  B_output: send ACK.
    TOLAYER3: packet being corrupted

EVENT time: 8.038575, type: 1, fromlayer5 entity: 0
  A_output: not yet acked. drop the message: bbbbbbbbbbbbbbbbbbbb

EVENT time: 8.459425, type: 2, fromlayer3 entity: 0
  A_input: packet corrupted. drop.

EVENT time: 13.693045, type: 1, fromlayer5 entity: 0
  A_output: not yet acked. drop the message: cccccccccccccccccc

EVENT time: 14.138920, type: 1, fromlayer5 entity: 0
  A_output: not yet acked. drop the message: dddddddddddddddddd

EVENT time: 15.467849, type: 0, timerinterrupt entity: 0
  A_timerinterrupt: resend last packet: aaaaaaaaaaaaaaaaaa.

EVENT time: 19.019745, type: 1, fromlayer5 entity: 0
  A_output: not yet acked. drop the message: eeeeeeeeeeeeeeeeeeee

EVENT time: 20.904539, type: 2, fromlayer3 entity: 1
  B_input: not the expected seq. send NAK.
    TOLAYER3: packet being corrupted
```

若是 ack 发生了 error，发送方检测到错误后不进行任何动作，等直到超时。

## 分析总结

在以上实验过程中，可以发现网络仿真中会出现各种复杂的事件和情况，但是经过思考和分析，这些事件都有很多共通之处，并在分析了我们的需求，可以将很多情况归为一类进行处理。这样思路也变得清晰起来。同时在仿真过程中增加大量打印信息的语句可以帮助我们精确追踪到仿真情况，方便调整。