

# 天津大学

## 《计算机网络》课程设计报告



### RDT 第二周：Go-Back-N 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 Go-Back-N 协议的理解。
3. 掌握 Go-Back-N 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。

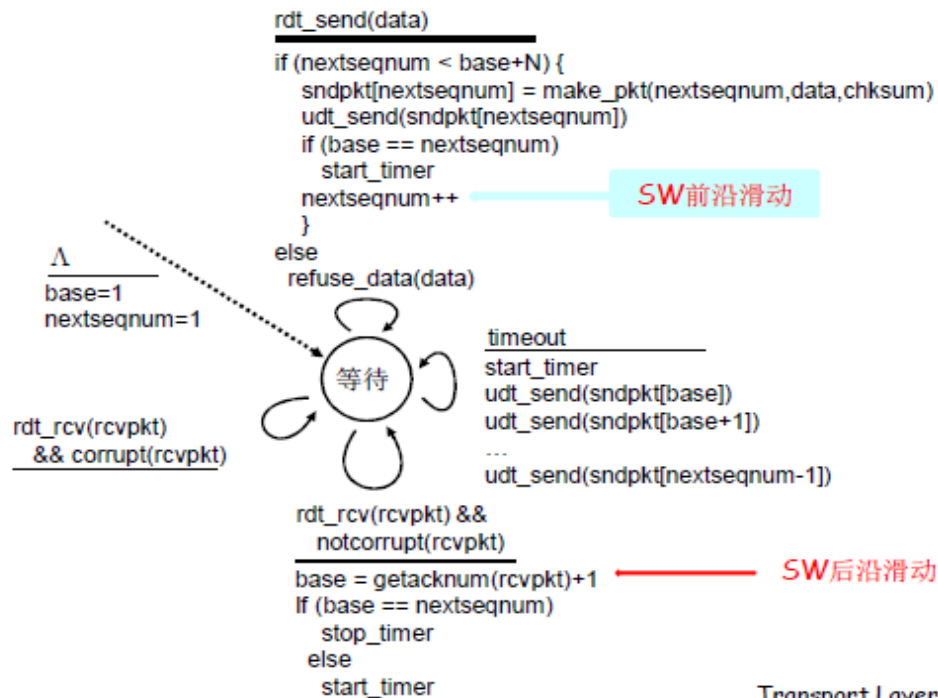
为了方便试验人员进行实验，本实验还提供了以下实现好的过程供试验人员调用：`starttimer()`和`stoptimer()`：试验人员可以通过调用此函数启动和停止节点的计时器。`tolayer3(calling_entity, packet)`：试验人员可以通过调用此函数时，传入需要发送的 `packet`，将数据包传递至网络层。这之后的传输工作将由框架自动完成。`tolayer5(calling_entity, message)`：试验人员可以通过调用此函数时，传入 `message`，将消息传递至应用层。这之后仿真框架将检查数据是否完好、是否按序到达。

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。

### Go-Back-N 协议原理和过程

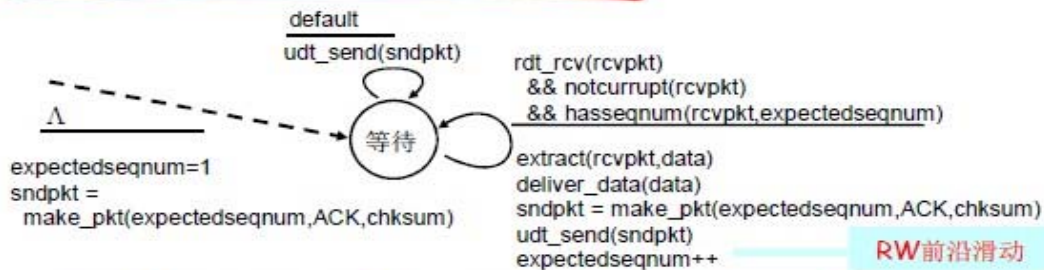
当接收方检测出时序的信息后，要求发送方重发最后一个正确接受的信息帧之后的所有未被确认的帧；或者当发送方发送了  $n$  个帧后，若发现该  $n$  帧的前一帧在计时器超时区间内仍未返回其确认信息，则该帧被判定为出错或丢失，此时发送方不得不重新发送该出错帧及其后的  $n$  帧。

## GBN: 发送方扩展的FSM



Transport Layer 3-67

## GBN: 接收方扩展的FSM



### □ 只发送ACK: 对顺序接收的最高序号的分组

- 可能会产生重复的ACK
- 只需记住 **expectedseqnum**; 接收窗口=1
  - 只一个变量就可表示接收窗口

### □ 对乱序的分组:

- 丢弃 (不缓存) → 在接收方不被缓存!
- 对顺序接收的最高序号的分组进行确认-累计确认



Transport Layer 3-68

对于 GBN 协议，要设置一个发送缓冲区，而且大小是 1，这个缓冲区中用存放已经发送但是没有得到确认的分组。对于接收端来说，只能顺序接收。从发送端来看，一旦一个分组没有发送成功，那么就要返回到这个没有发送成功的分组上，重新开始发送。

## 算法实现规划

在 rdt3.0 的基础上增添一个缓存数组来存储所有已发送未确认以及将要发送的分组，为超时重传作准备。同时修改超时事件为重新发送所有已发送但未确认的分组。最后分别为发送方和接收方设置“滑动窗口”模式和累积确认机制。

## 实现具体细节

实现具体细节即按照规划将功能具体落实到代码上。比如将发送方收到的每一次上层调用的消息存储到数据报数组中。按照当前窗口情况按序发送。超时处理函数修改为用循环体依次发送所有需要重传的数组。接收方根据收到数据报的正确性和序列号来进行下一步操作。

## 仿真的结果

最终还是经过大量打印过程信息帮助调试来实现 GBN 协议。按照实验文档，测试条件为 20messages，丢包率 0.2，损坏率 0.2，trace 设置为 2。  
对仿真过程中的各种情况分析：

## 算法分析

我在代码中加入了 Makefile 在运行的时候：

```
$ mingw32-make  
$ gbn
```

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

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

## 数据包和 ACK 正常

```
C:\Users\MSI-PC\Desktop\This PC\study\计算机网络\GBN>gbn
Enter the number of messages to simulate: 10
Enter packet loss probability [enter 0.0 for no loss]:0
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: buffered packet (seq=1): aaaaaaaaaaaaaaaaaa
  send_window: send packet (seq=1): aaaaaaaaaaaaaaaaaa

EVENT time: 5.960295, type: 2, fromlayer3 entity: 1
  B_input: recv packet (seq=1): aaaaaaaaaaaaaaaaaa
  B_input: send ACK (ack=1)

EVENT time: 8.038575, type: 1, fromlayer5 entity: 0
  A_output: buffered packet (seq=2): bbbbbbbbbbbbbbbbbb
  send_window: send packet (seq=2): bbbbbbbbbbbbbbbbbb

EVENT time: 8.459425, type: 2, fromlayer3 entity: 0
  A_input: got ACK (ack=1)
Warning: attempt to start a timer that is already started
  A_input: timer + 15.000000
```

如果数据包和 ACK 都正常的话，会正常运行。



## 数据包或 ACK-LOSS

```
C:\Users\MSI-PC\Desktop\This PC\study\计算机网络\GBN>gbn
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:0
  A_output: buffered packet (seq=1): aaaaaaaaaaaaaaaaaa
  send_window: send packet (seq=1): aaaaaaaaaaaaaaaaaa
  B_input: rcv packet (seq=1): aaaaaaaaaaaaaaaaaa
  B_input: send ACK (ack=1)
  A_output: buffered packet (seq=2): bbbbbbbbbbbbbbbbbb
  send_window: send packet (seq=2): bbbbbbbbbbbbbbbbbb
  A_input: got ACK (ack=1)
Warning: attempt to start a timer that is already started
  A_input: timer + 15.000000
  B_input: rcv packet (seq=2): bbbbbbbbbbbbbbbbbb
  B_input: send ACK (ack=2)
  A_output: buffered packet (seq=3): cccccccccccccccccc
  send_window: send packet (seq=3): cccccccccccccccccc
  A_input: got ACK (ack=2)
Warning: attempt to start a timer that is already started
  A_input: timer + 15.000000
  B_input: rcv packet (seq=3): cccccccccccccccccc
  B_input: send ACK (ack=3)
  A_timerinterrupt: resend packet (seq=3): cccccccccccccccccc
  A_timerinterrupt: timer + 15.000000
  A_output: buffered packet (seq=4): dddddddddddddddddd
  send_window: send packet (seq=4): dddddddddddddddddd
  A_input: got ACK (ack=3)
Warning: attempt to start a timer that is already started
  A_input: timer + 15.000000
  A_output: buffered packet (seq=5): eeeeeeeeeeeeeeeeeee
  send_window: send packet (seq=5): eeeeeeeeeeeeeeeeeee
  B_input: not the expected seq. send NAK (ack=3)
  A_output: buffered packet (seq=6): ffffffffffffffffffff
  send_window: send packet (seq=6): ffffffffffffffffffff
  A_output: buffered packet (seq=7): gggggggggggggggggggg
  send_window: send packet (seq=7): gggggggggggggggggggg
  A_output: buffered packet (seq=8): hhhhhhhhhhhhhhhhhhhh
  send_window: send packet (seq=8): hhhhhhhhhhhhhhhhhhhh
  B_input: rcv packet (seq=4): dddddddddddddddddd
```

丢失的情况处理也比较简单，因为没有接收到所以不会有动作，双方都将继续等待直到下一事件发生。

## 数据包或 ACK 出错

```
C:\Users\MSI-PC\Desktop\This PC\study\计算机网络\GBN>gbn
Enter the number of messages to simulate: 10
Enter packet loss probability [enter 0.0 for no loss]:0
Enter packet corruption probability [0.0 for no corruption]:0.1
Enter average time between messages from sender's layer5 [ > 0.0]:5
Enter TRACE:0
  A_output: buffered packet (seq=1): aaaaaaaaaaaaaaaaaa
  send_window: send packet (seq=1): aaaaaaaaaaaaaaaaaa
  B_input: rcv packet (seq=1): aaaaaaaaaaaaaaaaaa
  B_input: send ACK (ack=1)
  A_output: buffered packet (seq=2): bbbbbbbbbbbbbbbbbbb
  send_window: send packet (seq=2): bbbbbbbbbbbbbbbbbbb
  A_input: got ACK (ack=1)
Warning: attempt to start a timer that is already started
  A_input: timer + 15.000000
  B_input: rcv packet (seq=2): bbbbbbbbbbbbbbbbbbb
  B_input: send ACK (ack=2)
  A_output: buffered packet (seq=3): cccccccccccccccccc
  send_window: send packet (seq=3): cccccccccccccccccc
  A_input: got ACK (ack=2)
Warning: attempt to start a timer that is already started
  A_input: timer + 15.000000
  B_input: rcv packet (seq=3): cccccccccccccccccc
  B_input: send ACK (ack=3)
  A_timerinterrupt: resend packet (seq=3): cccccccccccccccccc
  A_timerinterrupt: timer + 15.000000
  A_output: buffered packet (seq=4): dddddddddddddddddd
  send_window: send packet (seq=4): dddddddddddddddddd
  A_input: got ACK (ack=3)
Warning: attempt to start a timer that is already started
  A_input: timer + 15.000000
  A_output: buffered packet (seq=5): eeeeeeeeeeeeeeeeeee
  send_window: send packet (seq=5): eeeeeeeeeeeeeeeeeee
  B_input: not the expected seq. send NAK (ack=3)
  A_output: buffered packet (seq=6): ffffffffffffffffffff
  send_window: send packet (seq=6): ffffffffffffffffffff
  A_input: got NAK (ack=3). drop.
  B_input: rcv packet (seq=4): dddddddddddddddddd
  B_input: send ACK (ack=4)
  A_timerinterrupt: resend packet (seq=4): dddddddddddddddddd
  A_timerinterrupt: resend packet (seq=5): eeeeeeeeeeeeeeeeeee
```

如上图所示，超时事件发生后将依次传送窗口内未确认的所有分组，同时重新启动计时器。

## 分析总结

GBN 允许发送多个分组，这也使得网络事件的复杂性大大增加。在短时间内数据包的数量也迅速增加，给编程调试带来了很大的困难。但是还是本着 rdt3.0 的原则来思考问题，将复杂的情况归纳总结并设计出统一的处理方式，来简化我们对复杂事件的处理。同时，程序运行过程中的信息也会因为数据包的数量增多而大大增多，导致调试过程的信息太过冗杂。