深 圳 大 学 实 验 报 告

|  |  |  |  |
| --- | --- | --- | --- |
| 课程名称 | 计算机网络 | | |
| 实验名称 | 实验3：可靠传输机制 | | |
| 学院 | 计算机与软件学院 | | |
| 专业 | 软件工程（腾班） | | |
| 指导教师 | 张磊 | | |
| 报告人 | 黄亮铭 | 学号 | 2022155028 |
| 实验时间 | 2024年4月10、17日 | | |
| 提交时间 | 2024年4月24日 | | |

教务处制

# 实验目的与要求

实验目的

1. 理解网络传输过程中可能出现的差错；
2. 理解网络传输中的稳定传输机制和差错处理方法；
3. 理解面向无连接的稳定传输机制；
4. 了解TCP稳定传输机制；
5. 了解滑动窗口机制；
6. 理解面向无连接的稳定传输机制；
7. 理解和掌握不同网络情况下，不同稳定传输机制的差异；
8. 通过对传输机制进行比较。

实验要求

1. 理解和掌握可靠传输机制的原理；
2. 基于提供的资料和代码，完成任务要求；
3. 按照任务要求进行实验，并按要求绘制实验结果图；
4. 对实现代码和实验结果进行截图展示；
5. 撰写实验报告。

# 实验过程

任务1：rdt2.1理解

任务目标及要求

1. 理解rdt2.1的状态转移机制及代码实现；
2. 学会利用clumsy模拟器在不同网络状态下进行实验，并通过作图进行分析；
3. 解释和说明rdt2.1的状态转移机制；
4. 利用clumsy模拟器模拟网络状态，在不同的网络状态下进行实验，并记录相关数据；
5. 将实验数据作图展示，并进行分析。

任务步骤

* + 1. **理解rdt2.1状态转移机制**

分别解释和说明发送方(server)和接收方(client)的状态转移过程

发送方：① 在状态为“等待来自上层的调用 偶序包”时，如果上层发送数据data，那么发送方会将ACK(0)、数据data和校验和checksum封装，并将其发送给接收方，此时状态变为“等待ACK或NAK”；② 在状态为“等待ACK或NAK”时有两种状态转移方式：1）接收到接收方发送来的数据，但信息损坏或NAK，我们将重新发送封装好的偶序包数据，状态保持不变；2）接收到接收方发送来的数据，没有信息损坏并且接收到ACK，此时不做任何处理，只是状态变为“等待来自上层的调用 奇序包”；③ 在状态为“等待来自上层的调用 奇序包”时，如果上层发送数据data，那么发送方会将ACK(1)、数据data和校验和checksum封装，并将其发送给接收方，此时状态变为“等待ACK或NAK”；④ 在状态为“等待ACK或NAK”时有两种状态转移方式：1）接收到接收方发送来的数据，但是信息损坏或NAK，我们将重新发送封装好的偶序包数据，状态保持不变；2）接收到接收方发送来的数据，没有信息损坏并且接收到ACK，此时不做任何处理，只是状态变为“等待来自上层的调用 偶序包”；⑤ 重复上述过程，直到没有上层数据需要传输。

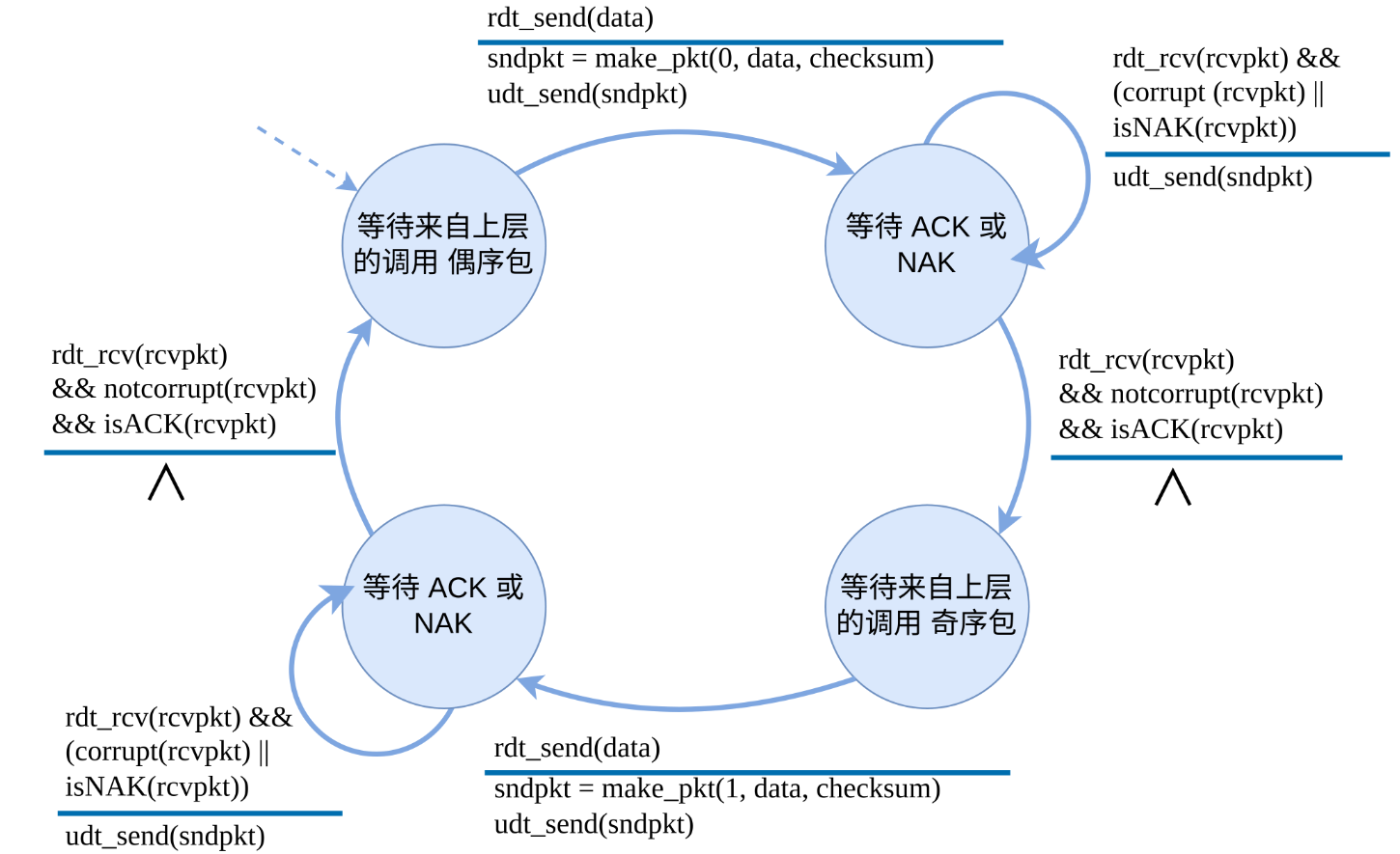


图1：发送方状态机

接收方：① 在“等待来自下层的调用 偶序包”状态时有三种状态转移方式：1）收到来自下层的数据，发现有中断信息，则发送由NAK和校验和checksum组成的数据包给发送方，要求重传数据，状态维持不变；2）收到来自下层的数据，没有信息损坏，但是发现序列号是奇数，同样发送由NAK和校验和checksum组成的数据包给发送方，要求重传数据，状态维持不变；3）收到来自下层的数据，没有信息损坏，并且序列号是期望的偶数，接收方将从数据（rcvpkt）中提取data，将data交付给上层，同时发送由ACK核对校验和组成的数据包给发送方；② 在“等待来自下层的调用 奇序包”状态时有三种状态转移方式：1）收到来自下层的数据，发现有中断信息，则发送由NAK和校验和checksum组成的数据包给发送方，要求重传数据，状态维持不变；2）收到来自下层的数据，没有信息损坏，但是发现序列号是奇数，同样发送由NAK和校验和checksum组成的数据包给发送方，要求重传数据，状态维持不变；3）收到来自下层的数据，没有信息损坏，并且序列号是期望的奇数，接收方将从数据（rcvpkt）中提取data，将data交付给上层，同时发送由ACK核对校验和组成的数据包给发送方；

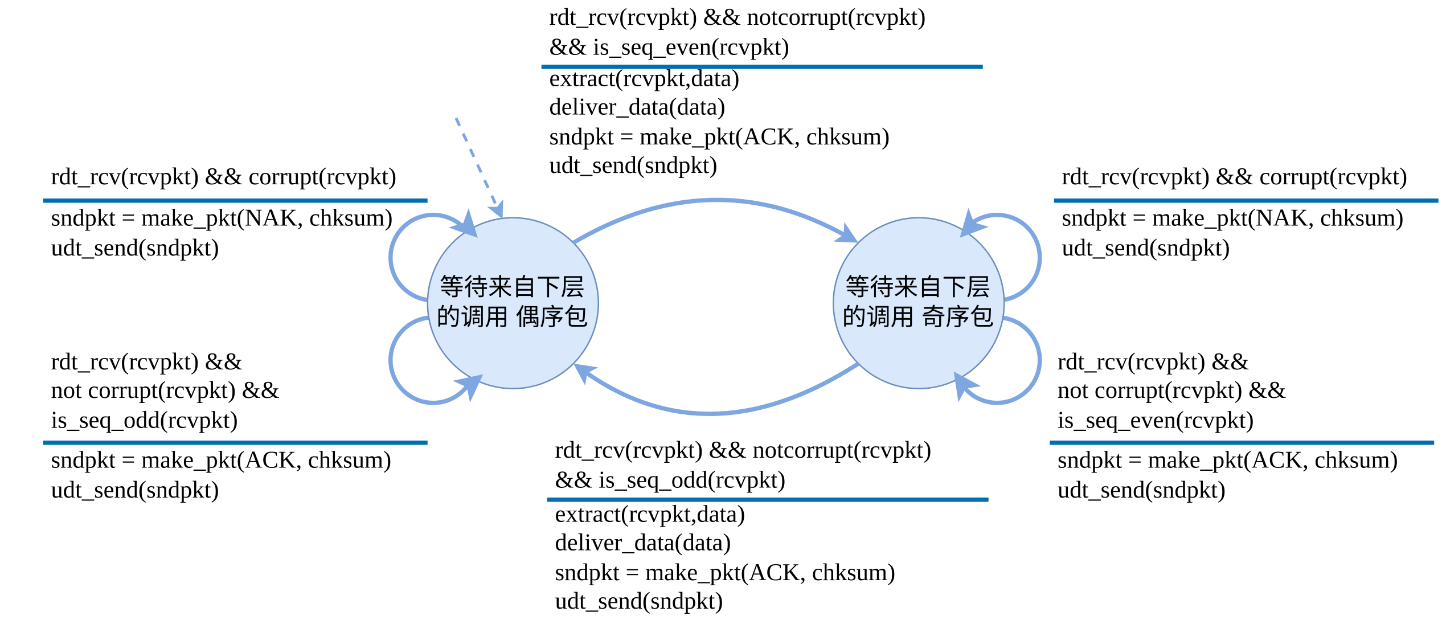


图2：接收方状态机

* + 1. **理解rdt2.1状态转移代码实现**

基于所给出的client.c和server.c代码，理解下述函数的使用方法

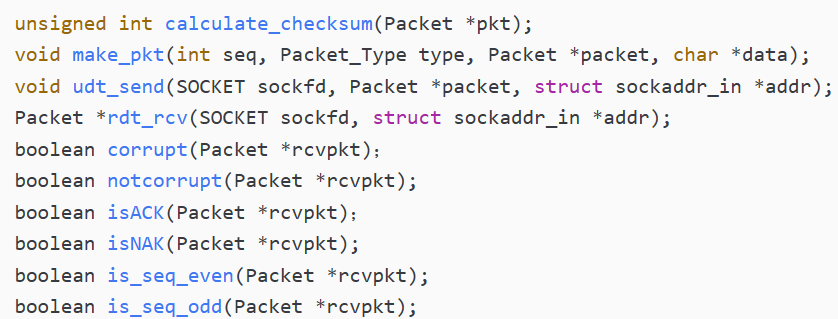


图3：函数声明

* 1. unsigned int calculate\_checksum(Packet \*pkt)

计算给定数据包的校验和。它遍历数据包的每个字节，将它们相加，跳过校验和字段本身的计算，最后返回计算得到的校验和值。

* 1. Packet\* make\_pkt(int seq, Packet\_Type type, char \*data)

这个函数用于创建一个数据包。它接受序列号seq、数据包类型type和数据data，并根据这些信息构建一个数据包。如果数data为NULL，则分配一个空的数据包；否则，将数据复制到数据包中。最后，它计算并填充数据包的校验和，并返回指向该数据包的指针。

* 1. void udt\_send(SOCKET sockfd, Packet \*packet, struct sockaddr\_in \*addr)

不可靠的数据传输发送操作。它将数据packet复制到缓冲区中，并使用send函数发送数据包。如果发送失败，它会打印错误消息；否则，打印成功发送的消息。

* 1. Packet \*rdt\_rcv(SOCKET sockfd, struct sockaddr\_in \*addr)

这个函数用于接收数据包。它使用recvfrom函数接收数据包，并将其存储在缓冲区中。如果接收失败，它会打印错误消息并返回 NULL；否则，返回指向接收到的数据包的指针。

* 1. boolean corrupt(Packet \*rcvpkt)

用于检查接收到的数据包是否损坏。它通过比较数据包的校验和和重新计算的校验和来确定数据包是否损坏。如果损坏，打印错误消息并返回 TRUE；否则返回 FALSE。

* 1. boolean notcorrupt(Packet \*rcvpkt)

用于检查数据包是否未损坏。如果数据包未损坏，返回 TRUE；否则返回 FALSE。

* 1. boolean isACK(Packet \*rcvpkt)

发送方用于检查接收到的数据包是否是 ACK 类型的数据包。如果是，打印相应的信息并返回 TRUE；否则返回 FALSE。

* 1. boolean isNAK(Packet \*rcvpkt)

发送方用于检查接收到的数据包是否是 NAK 类型的数据包，是则打印相应的信息并返回TRUE，否则返回FALSE。

* 1. boolean is\_seq\_even(Packet \*rcvpkt)

用于检查接收到的数据包的序列号是否为偶数。如果是，返回 TRUE；否则返回 FALSE。

* 1. boolean is\_seq\_odd(Packet \*rcvpkt)

与boolean is\_seq\_even(Packet \*rcvpkt)相反，这个函数用于检查接收到的数据包的序列号是否为奇数。

* + 1. **在模拟环境下运行代码并分析**
  + 学习clumsy网络模拟器

在Filtering中添加过滤器语法以设置过滤器（实验中请采用默认，outbound and loopback）其次选择网络模拟功能；最后点击Start启动模拟器，结束后点击Stop结束模拟器功能；

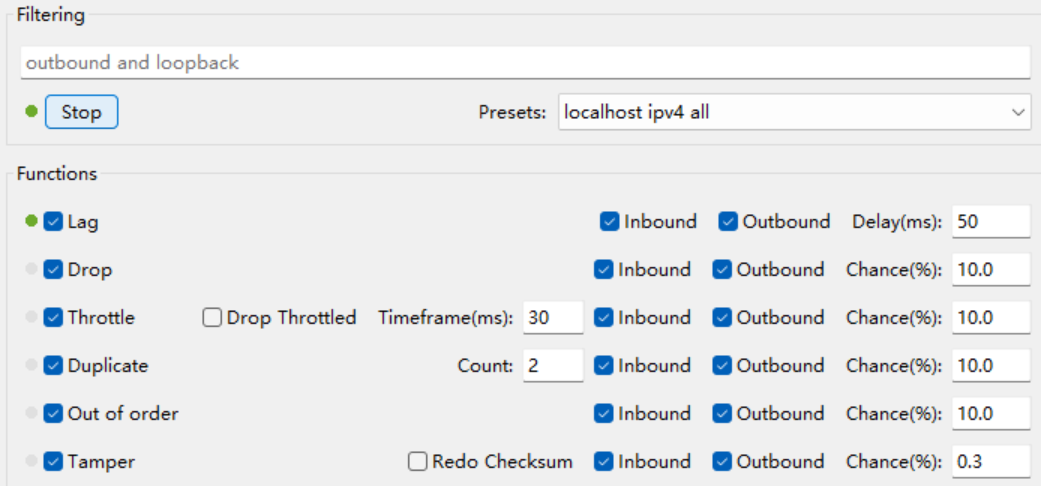
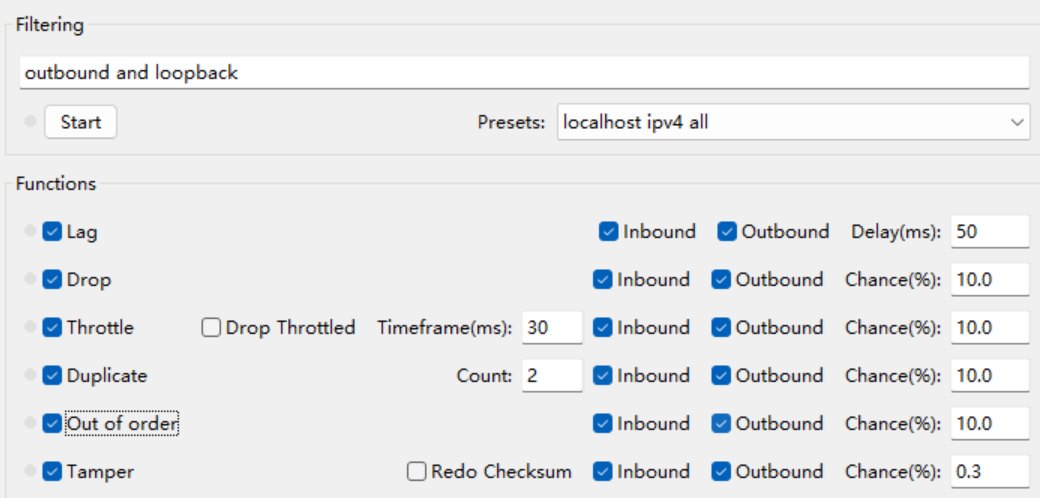


图4：开启和关闭网络模拟

* + 在模拟网络环境下进行实验分析

实验条件如下图所示，Lag=50ms，Throttle发生几率为10%，统一延迟发出时间为50ms，Tamper分别取0.3%、0.5%、1%、2%、3%、5%、10%、20%、30%、50%、70%。

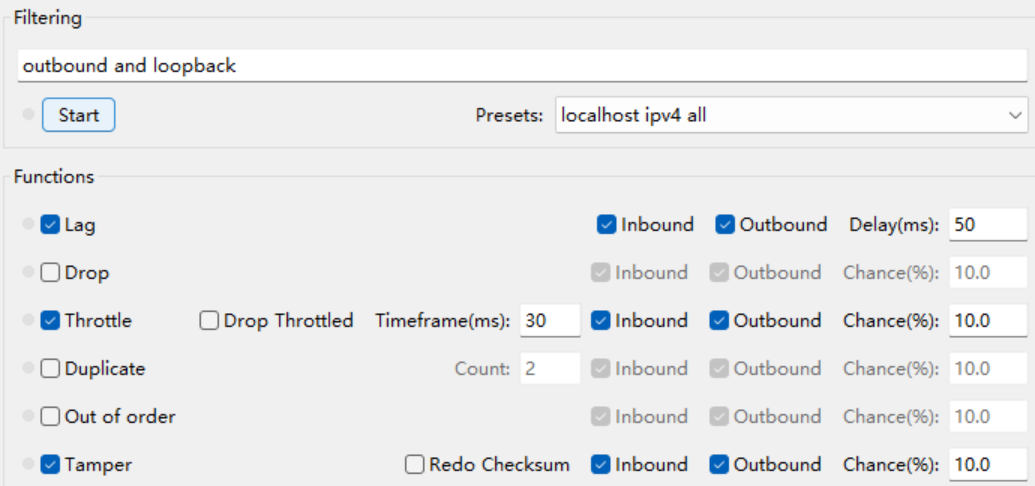


图5：实验条件

运行提供的rdt2.1 client和server程序，并记录数据包损坏数量和ACK、NAK损坏数量。

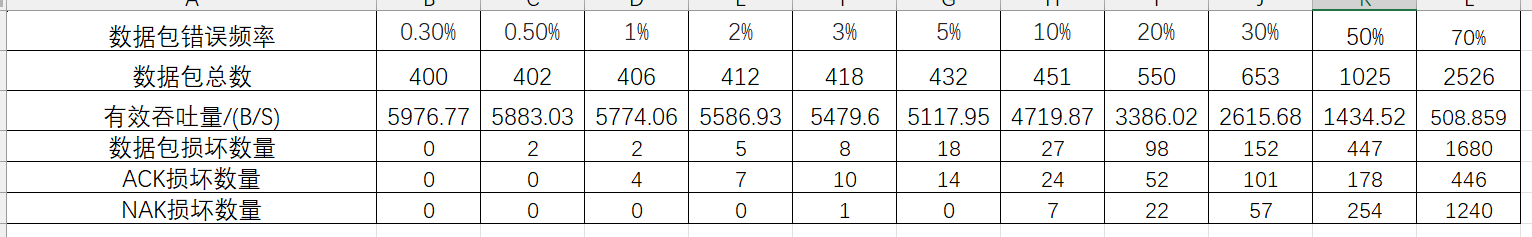


图6：数据图

基于实验结果，作图（数据包总数(OverHead)-数据包错误频率(Tamper rate)、有效吞吐量(Goodput)-数据包错误频率图）并分析调丢包对网络的影响。

丢包对网络的影响：丢包率较低时，只会导致网络延迟轻度增加。丢包率较高时，会导致网络性能严重下降。

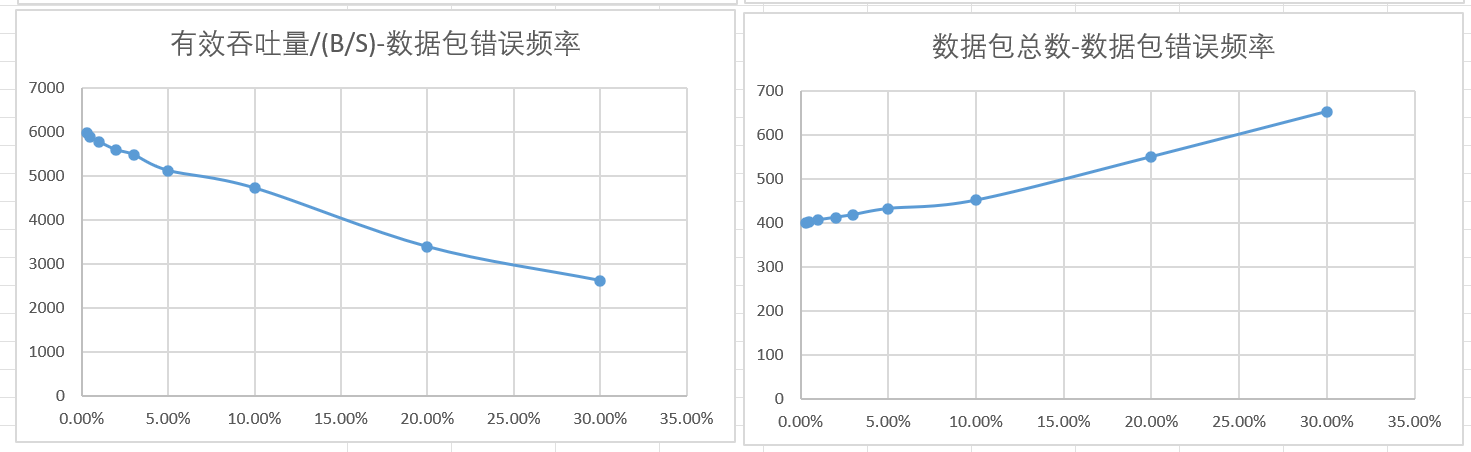


图7：数据包错误频率为0.3%-30%

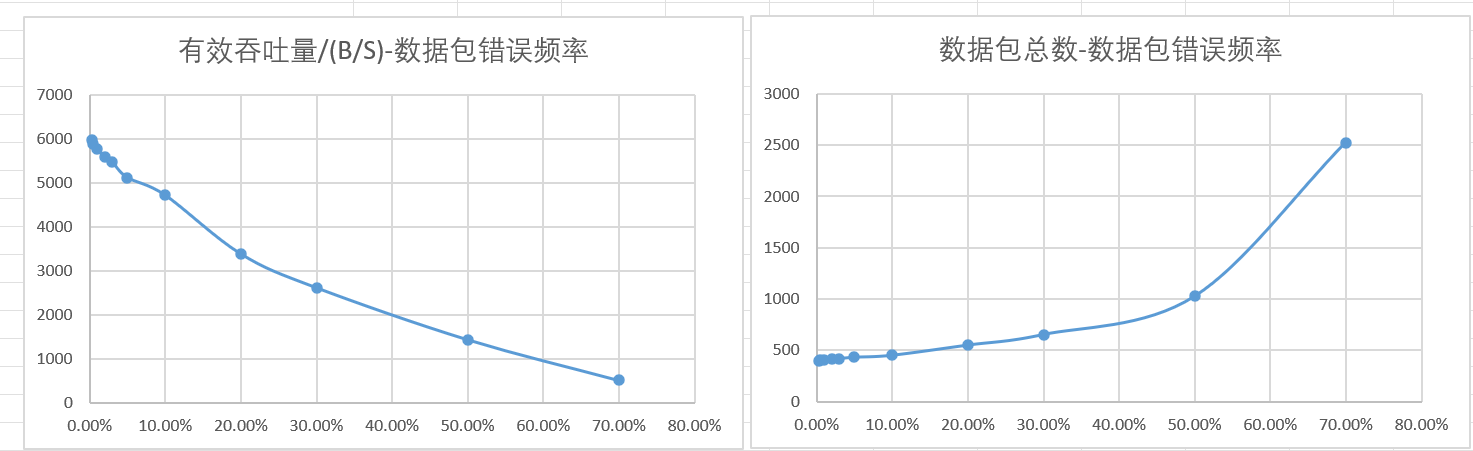


图8：数据包错误频率为0.3%-70%

* 问题1：当数据包发生错误时（在一定的数据包错误频率情况下），重发单个数据包是否能解决问题？从数据包发生错误到恢复正常发送状态的过程是怎样的？

重发单个数据包不能解决问题。过程：接收方检测到错误，它会发送NAK或者丢弃该数据包，这将被发送方认为是数据包丢失。发送方通过接收到的ACK或重传后的数据包确认传输成功，然后继续发送下一个数据包。

* 问题2：根据实验结果图，分析数据包总数、有效吞吐量与数据包错误频率之间的数学关系（线性关系？指数关系？）

数据包总数-数据包错误频率图在数据包错误频率为0.3%-30%时呈现出现线性关系，但在0.3%-70%时呈现出指数关系；有效吞吐量-数据包错误频率图在两种数据包错误频率范围均呈现出线性关系。

任务2：rdt2.2实现

任务目标及要求

1. 理解rdt2.2的状态转移机制并完成代码实现；
2. 利用clumsy模拟器在不同网络状态下进行实验，并通过作图进行分析；
3. 解释、说明和实现rdt2.2的状态转移机制；
4. 利用clumsy模拟器模拟网络状态；
5. 在不同的网络状态下进行实验，并记录相关数据；
6. 将实验数据作图展示，并进行分析；

****任务步骤****

1. **理解rdt2.2状态转移机制**
2. **分别解释和说明发送方(server)和接收方(client)的状态转移过程**

发送方：① 在状态为“等待来自上层的调用 偶序包”时，如果上层发送数据data，那么发送方会将seq(0)、数据data和校验和checksum封装，并将其发送给接收方，此时状态变为“等待ACK偶序包”；② 在状态为“等待ACK偶序包”时有两种状态转移方式：1）接收到接收方发送来的数据，但信息损坏ACK为奇序包，我们将重新发送封装好的偶序包数据，状态保持不变；2）接收到接收方发送来的数据，没有信息损坏并且接收到ACK，此时不做任何处理，只是状态变为“等待来自上层的调用 奇序包”；③ 在状态为“等待来自上层的调用 奇序包”时，如果上层发送数据data，那么发送方会将seq(1)、数据data和校验和checksum封装，并将其发送给接收方，此时状态变为“等待ACK 奇序包”；④ 在状态为“等待ACK 奇序包”时有两种状态转移方式：1）接收到接收方发送来的数据，但是信息损坏ACK为偶序包，我们将重新发送封装好的奇序包数据，状态保持不变；2）接收到接收方发送来的数据，没有信息损坏并且接收到ACK，此时不做任何处理，只是状态变为“等待来自上层的调用 偶序包”；⑤ 重复上述过程，直到没有上层数据需要传输。

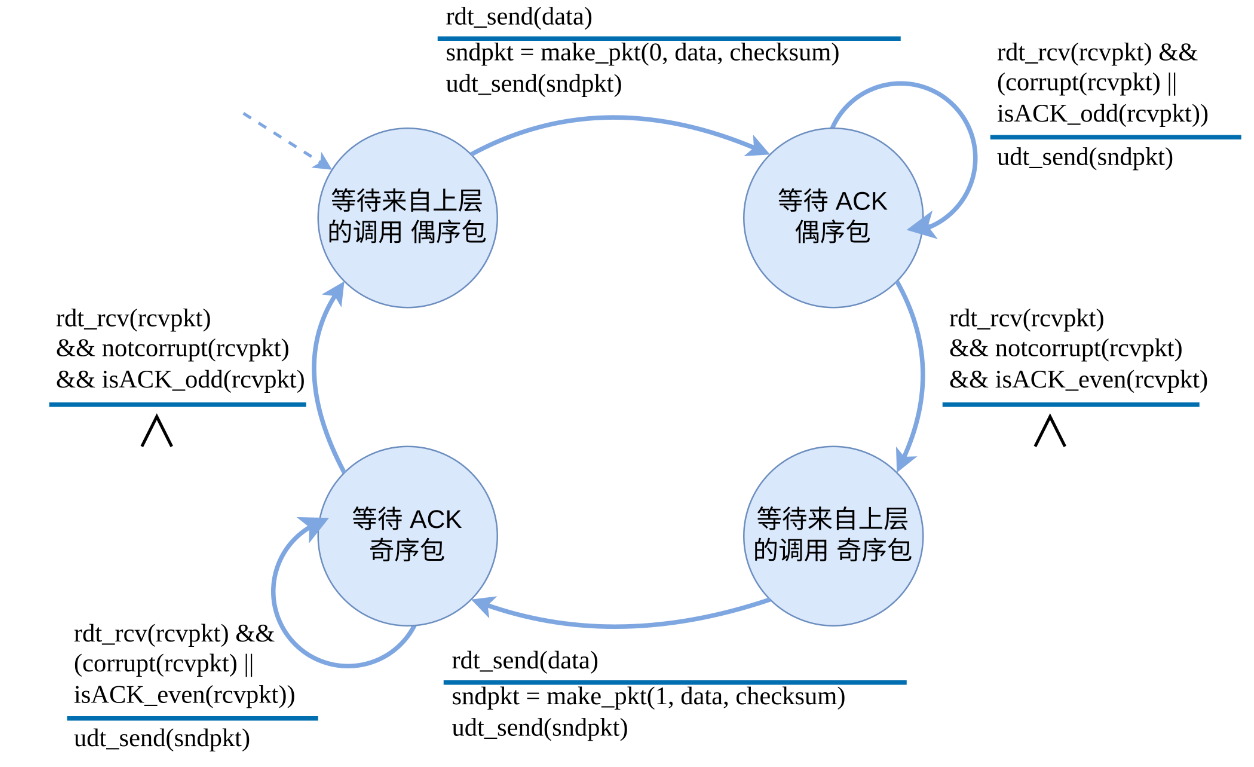


图9：发送方状态机

接收方：① 在状态为“等待来自下层的调用 偶序包”时，有两种状态转移方式：1）信息损坏或者收到seq为奇序时，将重新发送ACK(1)给接收方；2）信息未损坏并且收到seq为偶序时，状态转移为“等待来自下层的调用 奇序包”；② 在状态为“等待来自下层的调用 奇序包”时，有两种状态转移方式：1）信息损坏或者收到seq为偶序时，将重新发送ACK(0)给接收方；2）信息未损坏并且收到seq为奇序时，状态转移为“等待来自下层的调用 偶序包”；

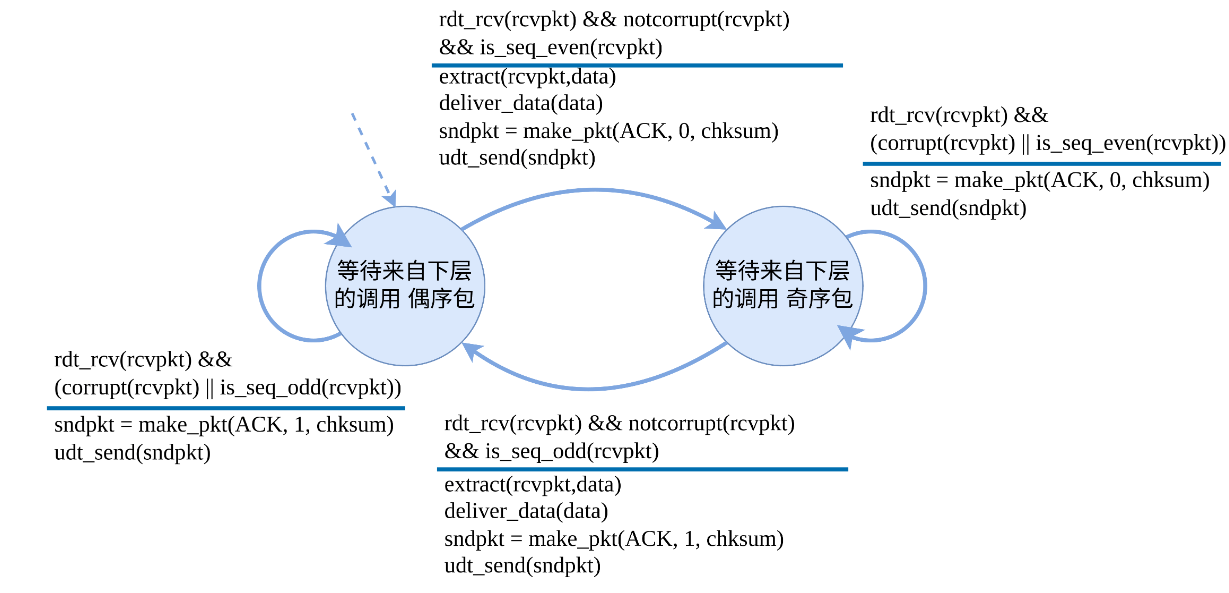


图10：接收方状态机

1. **实现rdt2.2机制功能**

发送端修改状态名称。将状态修改如下图。

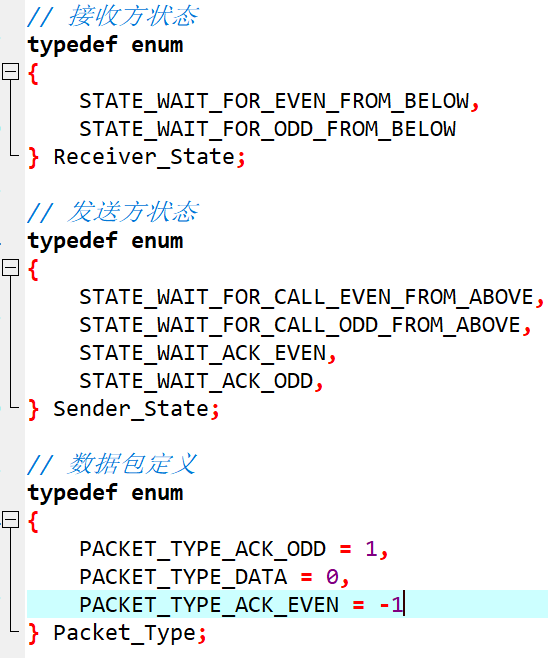
****

图11：发送方状态

发送端修改ACK发送内容（判断ACK奇偶性、实现isACKOdd、isACKEven函数）

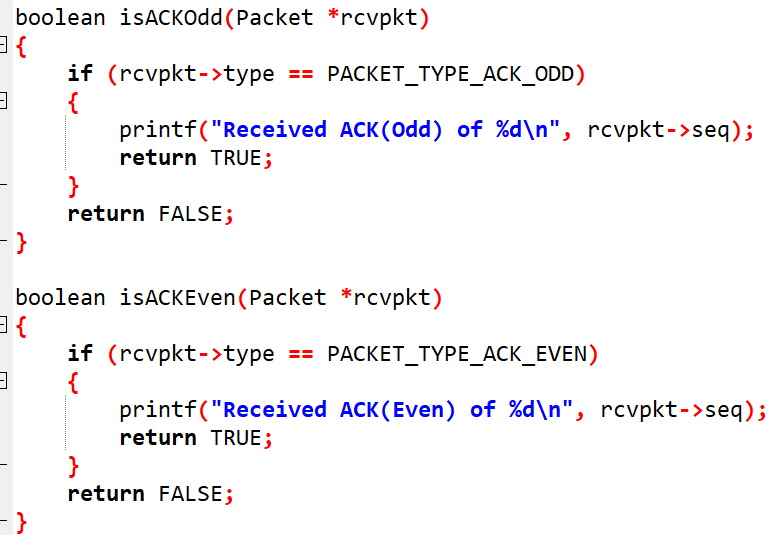


图12：isACKOdd、isACKEven函数代码

发送端修改状态转移判断条件，见下图。主要是将选择分支的条件改为与状态机一致。

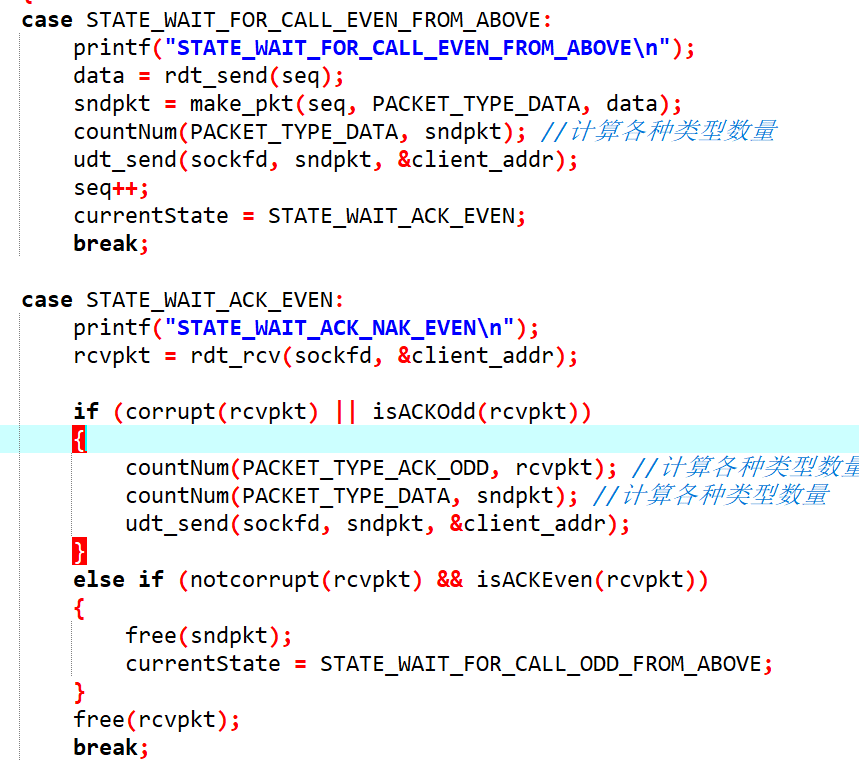
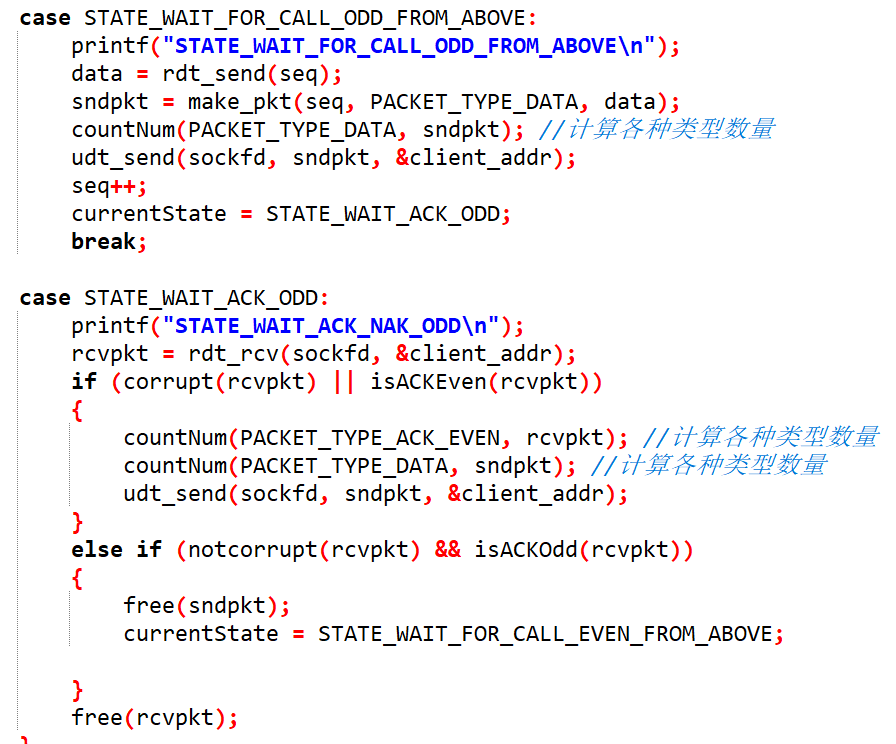


图13：修改状态转移判断条件

接收端针对rdt2.2和rdt2.1的区别，更改rdt2.1的代码中的状态转移判断条件，符合rdt2.2的需求。

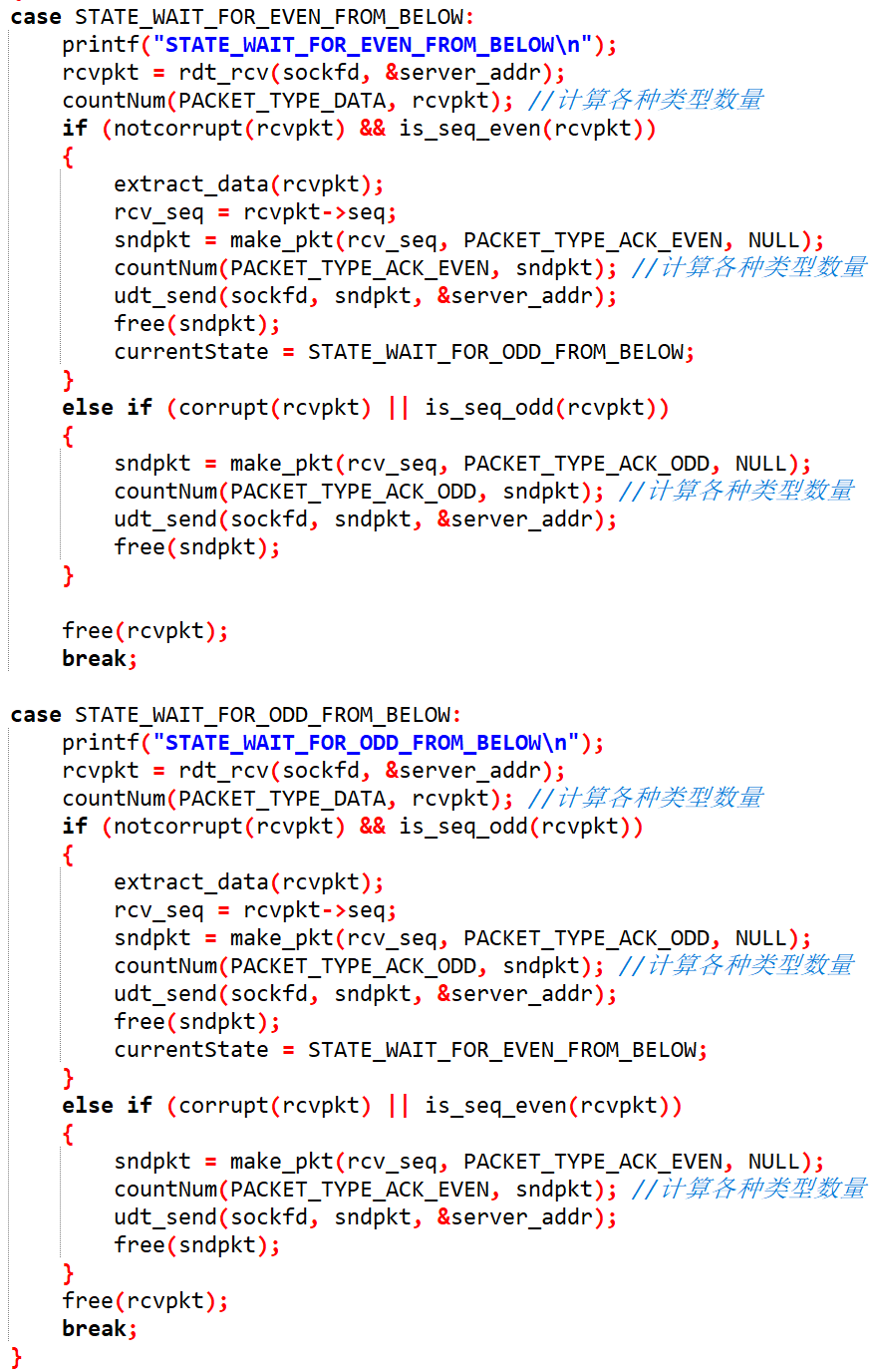


图14：修改状态转移判断条件

接收端修改数据包发送类型以及去除和NAK有关的发送内容及状态转移条件及事件。

在状态为“等待来自下层的调用 偶序包”时，发送ACK（1）。

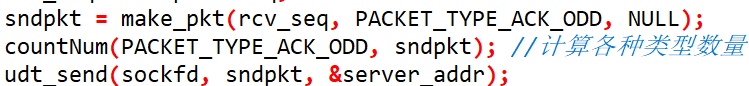


图15：代码实现

在状态为“等待来自下层的调用 奇序包”时，发送ACK（0）

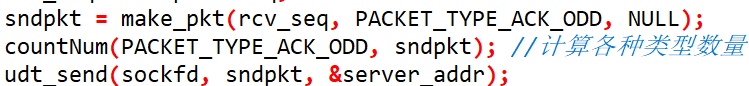


图16：代码实现

1. **分析rdt2.1和rdt2.2的机制差异**

1）连续三个包满足以下条件：接收方正常接收数据包、接收方返回的ACK发生错误、发送方重发数据包再次发生错误。在以上三种情况中，rdt2.1和rdt2.2的处理方法分别是什么？哪一种机制在出现这种状况时表现较好？为什么？

答：发送方重发数据包再次发生错误后（假设接收全部成功），rdt2.1中的发送方接收到NAK，再发送一个已被接收的数据包，接收方会发送一个ACK给发送方，发送方在接收到ACK后才会发送正确的数据包。而rdt2.2总的发送方接收到ACK之后，发送方可以直接发送正确的数据包给接收方。

Rdt2.2在出现这种状况时表现较好。因为rdt2.1发送两倍于rdt2.2的确认信息（ACK+NAK）。

2）在模拟网络环境下进行实验分析

实验条件如下图所示，Tamper分别取0.3%、0.5%、1%、2%、3%、5%、10%、20%、30%、50%、70%。

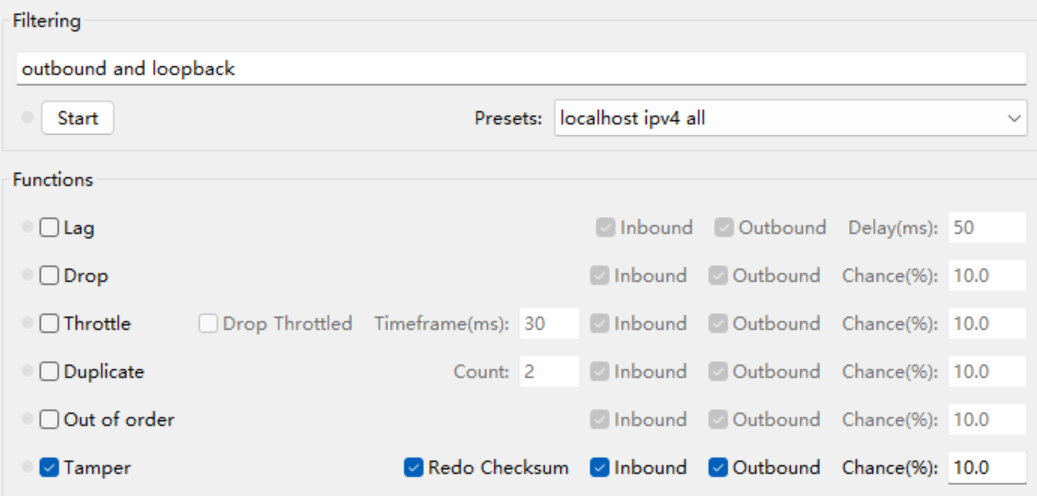


图17：实验条件

运行提供的rdt2.2 client和server程序，并记录数据包损坏数量和ACK、NAK损坏数量。

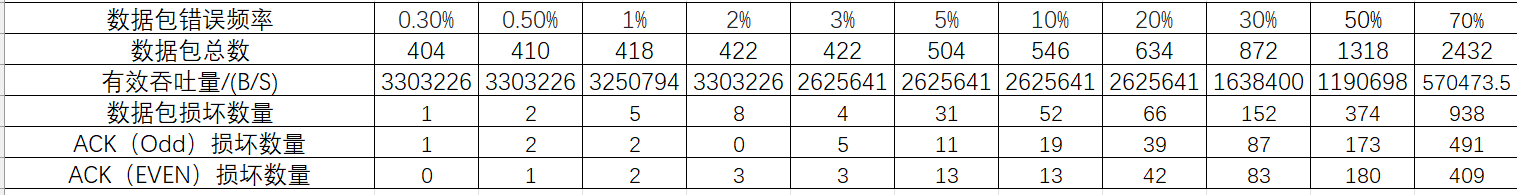


图18：数据图

基于实验结果，作图（数据包总数-数据包丢失频率(丢包率，Drop rate)、有效吞吐量-丢包率）并分析调丢包对网络的影响，同时与任务1中所绘制的图像进行对比，说明两种机制下实验的差异，并描述产生差异可能的原因。

丢包对网络的影响：丢包率较低时，只会导致网络延迟轻度增加。丢包率较高时，会导致网络性能严重下降。

对比图像可知，rdt2.1的有效吞吐量随着丢包率的上升呈现线性下降的趋势，而rdt2.2的有效吞吐量随着丢包率的上升先保持平稳，然后出现线性下降趋势。两者的的数据包总数随丢包率上升的趋势几乎相同，丢包率较低时为线性增长，丢包率较高时为指数增长。

差异的原因：随数据包出现错误增高时，因为rdt2.1发送两倍于rdt2.2的确认信息（ACK+NAK），所以rdt2.1的有效吞吐量下降速率较rdt2.2更快，且发送的数据包总数也相对较多。

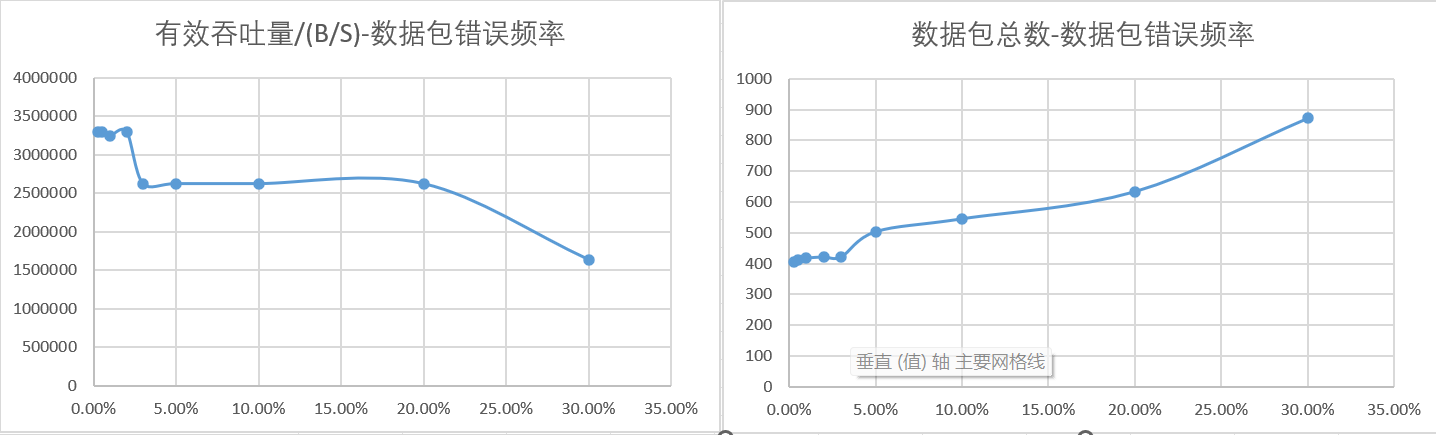


图19：数据包错误频率为0.3%-30%

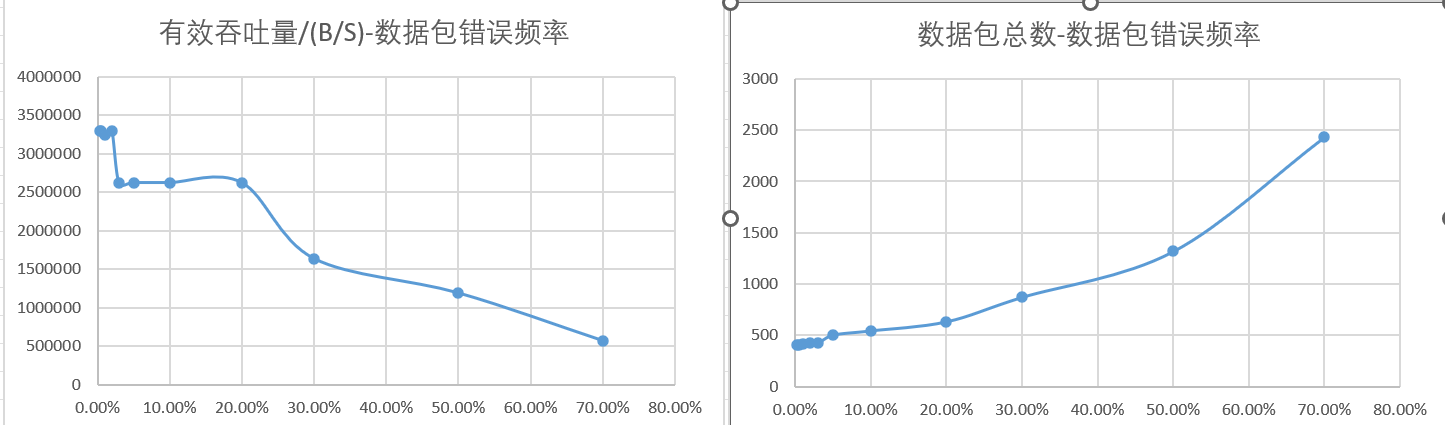


图20：数据包错误频率为0.3%-70%

任务3：rdt3.0实现

任务要求及目标

1. 理解丢包事件对传输造成的影响；
2. 理解rdt3.0的状态转移机制并完成代码实现；
3. 利用clumsy模拟器在不同网络状态下进行实验，并通过作图进行分析；
4. 解释、说明和实现rdt3.0的状态转移机制；
5. 利用clumsy模拟器模拟网络状态；
6. 在不同的网络状态下进行实验，并记录相关数据；
7. 将实验数据作图展示，并进行分析；

任务步骤

1. 理解rdt3.0发送方状态机
   1. 在状态为“等待来自上层的调用偶序包”时，1）如果收到来自接收方的数据包，不做任何处理；2）如果收到来自上层的数据，则将其和序列号、校验和等数据封装，然后发给接收方，同时启动定时器，状态转移为“等待ACK 偶序包”。
   2. 在状态为“等待ACK 偶序包”时，1）如果来自接收方的信息没有损坏或者ACK为奇序，不做任何处理；2）如果超时，则重新发送数据包；3）如果来自接收方的信息没有损坏并且ACK为偶序，则停止定时器，状态转移为“等待来自上层的调用 奇序包”。
   3. 在状态为“等待来自上层的调用奇序包”时，1）如果收到来自接收方的数据包，不做任何处理；2）如果收到来自上层的数据，则将其和序列号、校验和等数据封装，然后发给接收方，同时启动定时器，状态转移为“等待ACK 奇序包”。
   4. 在状态为“等待ACK 奇序包”时，1）如果来自接收方的信息没有损坏或者ACK为偶序，不做任何处理；2）如果超时，则重新发送数据包；3）如果来自接收方的信息没有损坏并且ACK为奇序，则停止定时器，状态转移为“等待来自上层的调用 偶序包”。

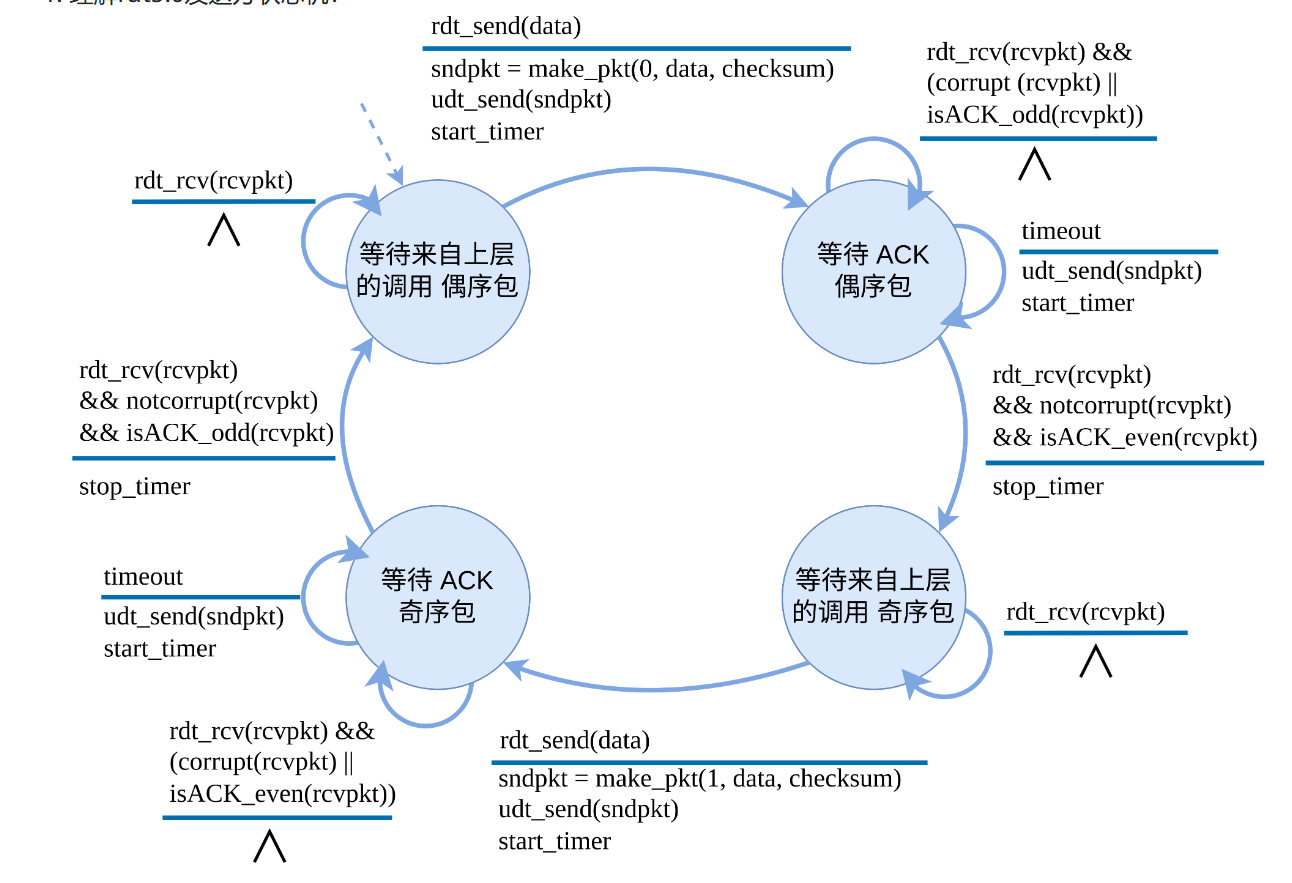


图21：rdt3.0发送方状态机

1. 理解rdt3.0接收方状态机
2. 在状态为“等待来自下层的调用 偶序包”时，1）收到来自发送方的数据包信息没有损坏或者序列号为奇数，则发送ACK（1）给发送方，期望收到序列号为偶数的数据包；2）如果收到来自发送方的数据包信息没有损坏并且序列号为偶数，则提取数据包中的有效数据，将数据发送给上层，然后发送ACK（0）给发送方，状态转移为“等待来自下层的调用 奇序包”。
3. 在状态为“等待来自下层的调用 奇序包”时，1）收到来自发送方的数据包信息没有损坏或者序列号为偶数，则发送ACK（0）给发送方，期望收到序列号为奇数的数据包；2）如果收到来自发送方的数据包信息没有损坏并且序列号为奇数，则提取数据包中的有效数据，将数据发送给上层，然后发送ACK（1）给发送方。

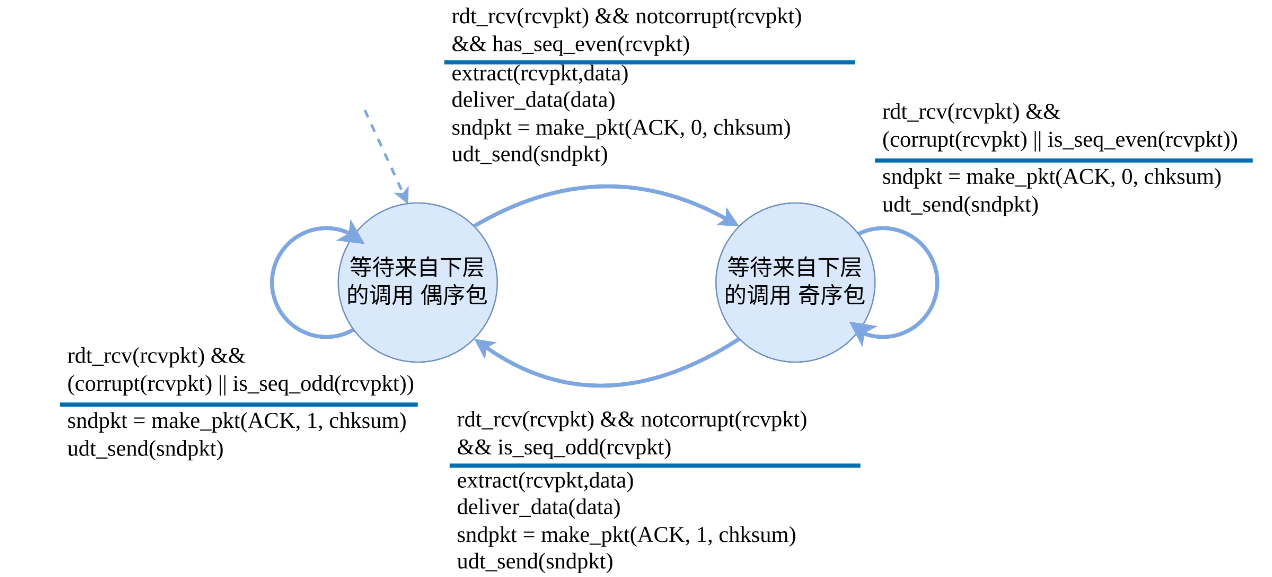


图22：rdt3.0接收方状态机

1. 了解定时器

**在rdt3.0的发送方中，触发timeout事件时，定时器的内容是否有区别？如果有，有什么样的区别？**

答：定时器的内容有区别。在不同的状态中超时会重传不同的数据包，数据包的内容不一样。

**每次触发timeout事件时，是否会改变发送方的状态？**

答：不会改变发送方的状态，发送方会重新发送超时的数据包，直接收方接收正确的ACK。

**基于对定时器的理解，完成定时器函数。**

超时之后重新传输数据包给接收方，并重新启动定时器。注意要统计发送的数据包总数。

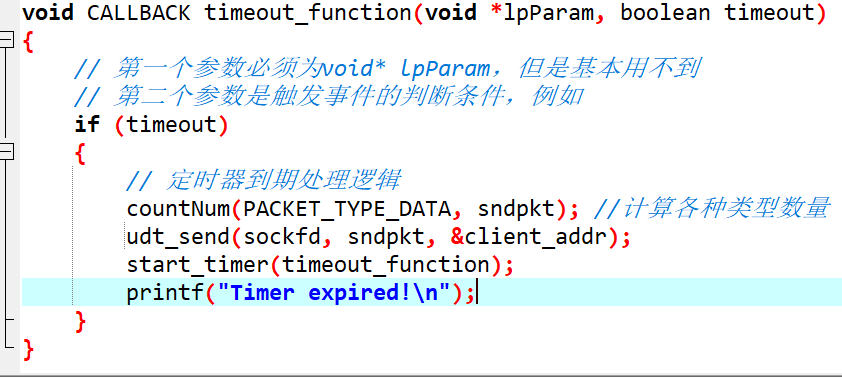
****

图23：定时器函数

**实现rdt3.0机制。**

发送方代码（见下图），逻辑和发送方状态机的描述一致。

图24：rdt3.0发送方代码

确实有超时的数据包，通过超时重传后成功被接收，证明机制逻辑正确。

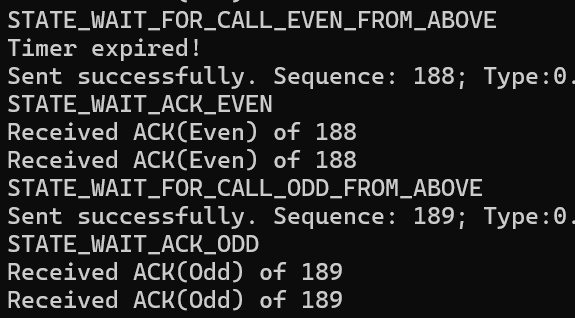


图25：代码结果

接收方代码与rdt2.2的接收方代码相同。

1. 分析rdt3.0和rdt2.1、rdt2.2的差异
2. **在数据包错误的情况下，rdt3.0和rdt2.2中的现象是什么？有什么区别？**

答：rdt3.0和rdt2.2中的接收方都会发送上一次正确接收的ACK给发送方。

区别在于：rdt3.0对接收方的ACK不做处理，直到定时器超时，rdt3.0发送方才重新传输数据包给接收方，并重新启动定时器；rdt2.2则会直接发送正确的数据包给接收方。

**2）在数据包丢失的情况下，rdt3.0是如何处理的？**

答：在数据包丢失的情况下，rdt3.0的发送方会一直等待接收方的回应，直到超时，然后重新传输数据包给接收方。

**3）使用clumsy模拟数据包丢失状况，作图（数据包总数-数据包丢失频率(丢包率，Drop rate)、有效吞吐量-丢包率）并分析丢包对网络的影响。（参数设置：Tamper设置为1%，丢包率(Drop)设置为0.3%、0.5%、1%、2%、3%、5%、10%、20%、30%。）**

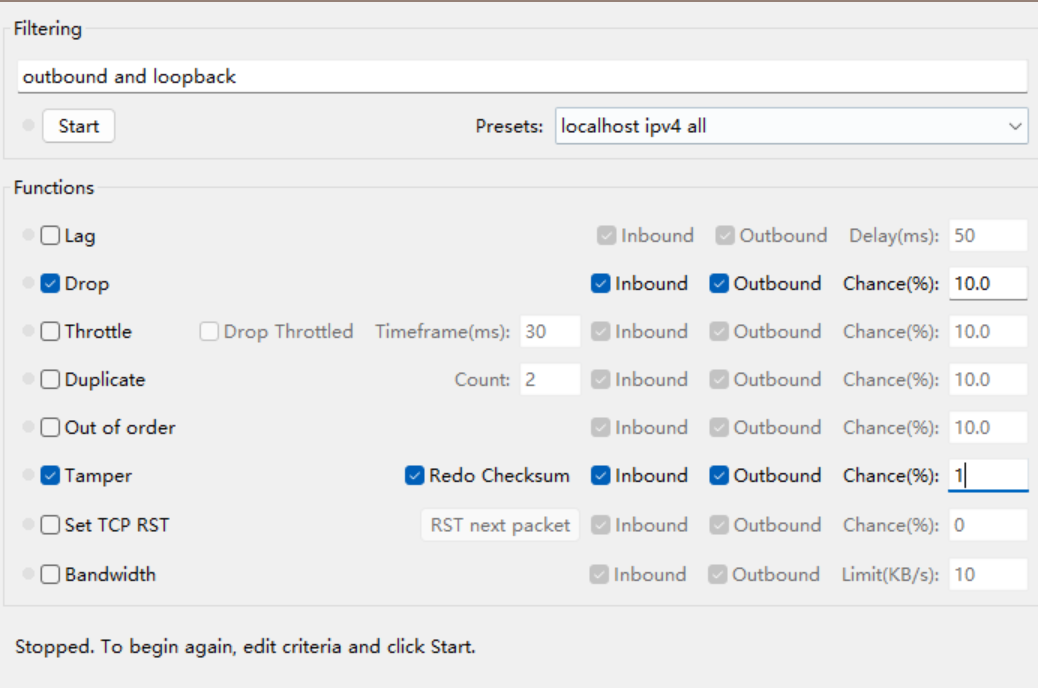
****

图26：实验参数

**4）根据实验结果图，分析数据包总数、有效吞吐量与丢包率之间的数学关系；**

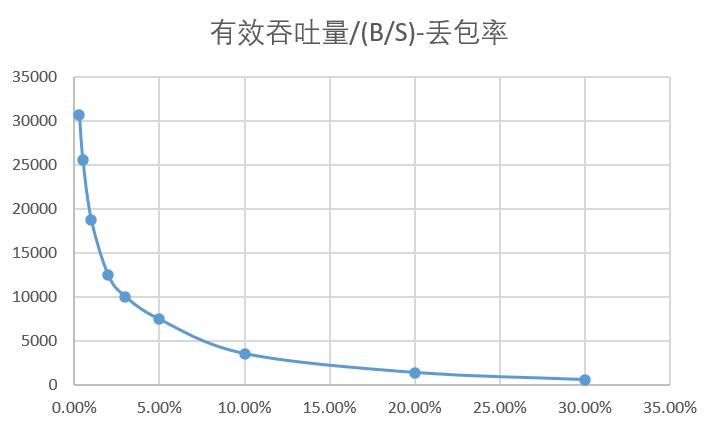


图27：有效吞吐量和丢包率的关系

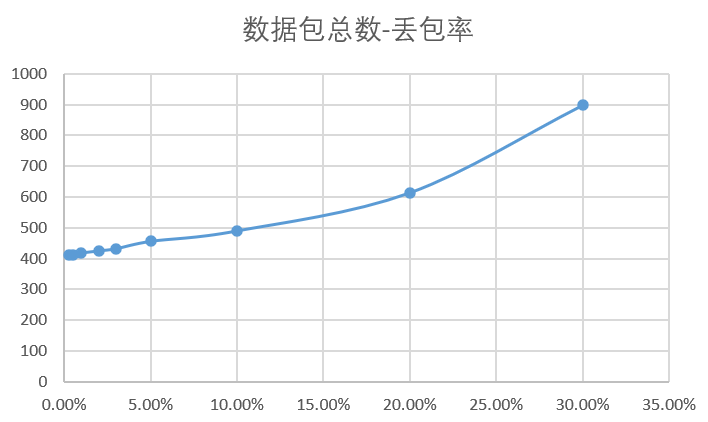


图28：数据包总数和丢包率的关系

**有效吞吐量和丢包率之间为指数关系。**在丢包率较低时，丢包率升高，有效吞吐量下降迅速；在丢包率较高时，丢包率升高，有效吞吐量下降较慢，最后趋于0。

**数据包总数和丢包率之间为线性关系。**但是，在丢包率较高时，随着丢包率的升高，数据包总数的增加速度也更快，可以合理猜测在丢包率为50%~70%时数据包总数和丢包率之间的关系可能由线性关系变为指数关系。

任务4：回退N步（GBN）机制实现

任务目标及要求

1. 理解并实现GBN机制；
2. 利用clumsy模拟器在不同网络状态下进行实验，并通过作图进行分析
3. 解释、说明和实现GBN机制；
4. 利用clumsy模拟器模拟网络状态；
5. 在不同的网络状态下进行实验，并记录相关数据；
6. 将实验数据作图展示，并进行分析；

任务步骤

1. **理解GBN状态转移机制。**

GBN发送方状态机：

1. 初始化发送端，设置基序号和下一个序号的数值。
2. 收到来自上层的数据报文时，发送方会判断下一个序号是否在滑动窗口内。如果在，发送方会将下一个序列号和数据以及校验和等信息分装，然后发送给接收方，此时判断基序号是否等于下一个序列号，如果等于则重启定时器并让下一个序列号加一；如果不在，则拒绝将来自上层的数据发送给接收方。
3. 在定时器超时时，发送方会发送从基序号开始，到下一个序列号前的所有数据包。
4. 接收到来自接收方的确认信息并且确认信息无误时，发送方会将基序号变为接收到的序列号加1，然后判断基序号是否等于下一个序列号，如果等于就停止定时器；如果不等于则重新启动定时器。
5. 接收到来自接收方的确认信息但是确认信息损坏时，发送方对此不做任何处理。

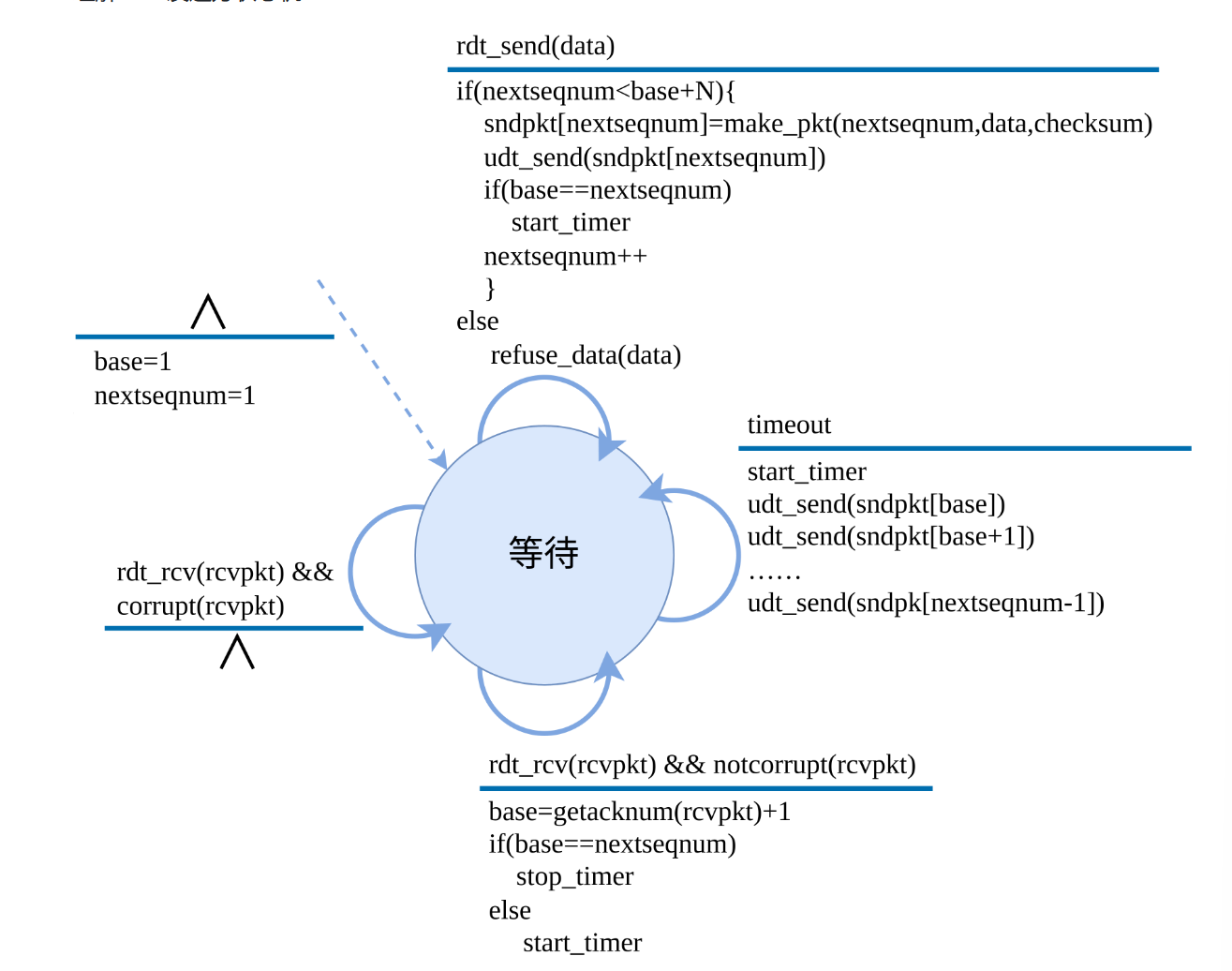


图29：GBN发送方状态机

GBN接收方状态机：

1. 初始化接收端，设置期望序列号并将0、ACK、校验和封装成确认信息，准备发送给发送方。
2. 如果接收到来自下层的数据包，数据没有损坏并且附带的序列号时期望序列号，接收方将提取有效数据并将数据发送给上层。同时将期望序列号、ACK和校验和封装成数据包发送给发送方，期望序列号加一。
3. 如果不是上述操作，发送上一次已经封装好的数据包给发送方。

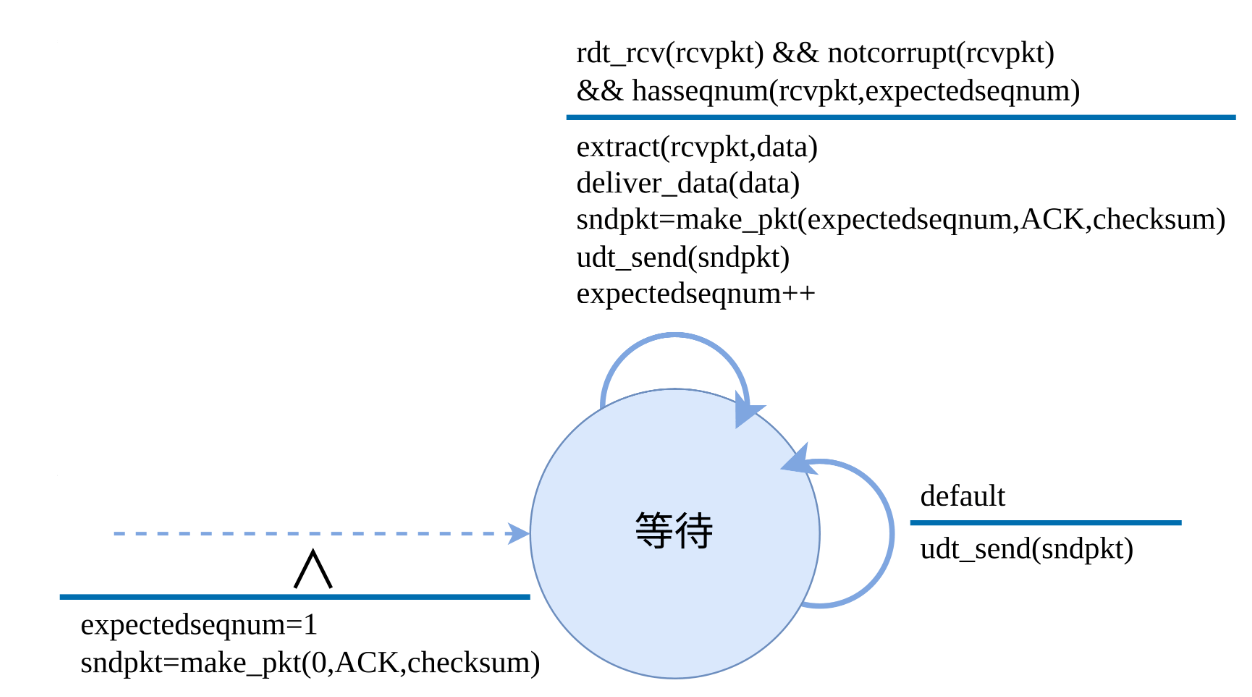


图30：GBN接收方状态机

1. **回答问题。**
2. **在GBN机制中，如何利用ACK序号对数据包进行判断？**

答：GBN机制采用累计确认的方式，当前的ACK序号（假设为N）代表接收方已经正确接收序列号为N及序列号为N以前的所有数据包。

1. **解释和说明GBN机制如何对乱序数据包的情况进行处理的？**

答：GBN机制中的接收方会丢弃所有乱序的数据包，直到发送方的定时器超时，发送方重新传输所有未被确认的分组。

1. **在给出GBN机制为何能够完成rdt3.0机制中的所有条件，最后如何将所有状态合并为一个状态的？**

答：**发送方：**rdt3.0状态机中的“等待来自上层的调用 偶序包”和“等待来自上层的调用 奇序包”可以合并为GBN中的滑动窗口的下一个序列号发送数据包，即两个状态可以合并为一个状态。Rdt3.0状态机中的“等待ACK 偶序包”和“等待ACK 奇序包”可以合并为GBN中的滑动窗口中的基序号等待确认ACK，即两个状态合并为一个状态。此时还剩余两个状态：等待上层数据的发送状态和等待确认信息的接收状态。两者都依靠滑动窗口。因此，可以以滑动窗口为媒介，将两个状态合并为一个状态，得到GBN机制。**接收方：**“等待来自下层的调用 偶序包”和“等待来自下层的调用 奇序包”可以合并为等待下层的调用，即两个状态合并为一个状态。

1. **实现GBN机制。**

在发送方新增变量base、nextseqnum，具体作用同GBN状态转移机中的说明。将原本指向数据包地址的指针sndpkt改为指向数据包地址的指针数组的数组指针，因为此时我们不再是发送一个数据包就停止发送，而是发送多个数据包。具体代码见下图。

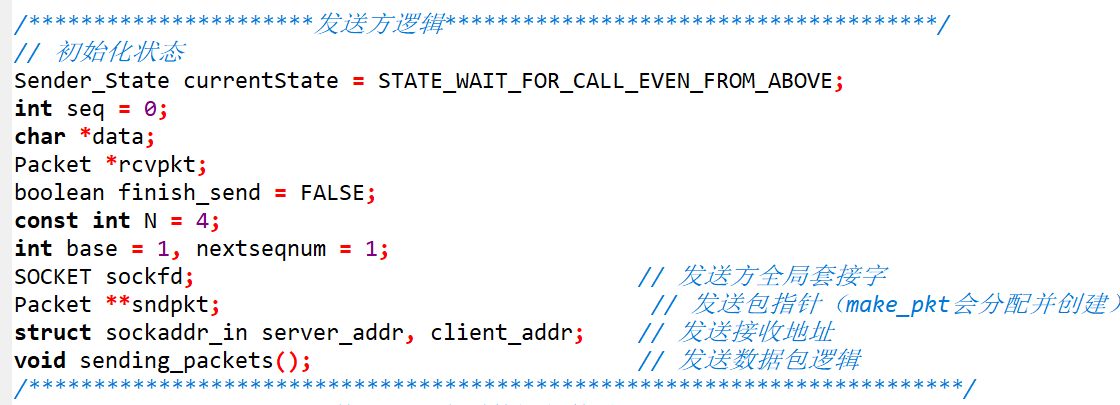


图31：GBN发送方全局变量

GBN发送方不再区分ACK的奇偶性。因此，统计不同类型的数据包函数也需要更改，具体见下图。

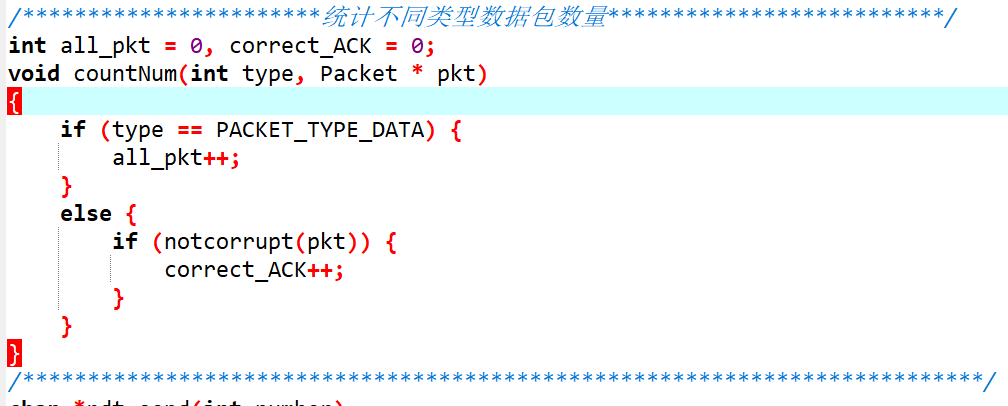


图32：GBN统计数据包函数

发送前的准备：为存放数据包的指针的数组开辟相关内存，同时启动一个计数器（计算有效吞吐量）。请注意，要在发送结束后释放申请的内存。

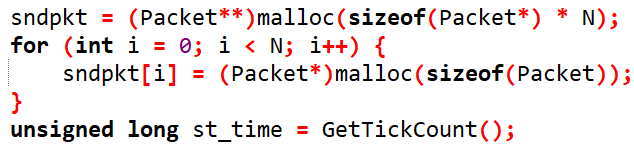


图33：发送前的准备

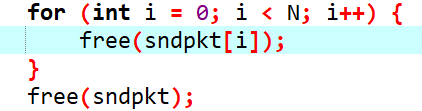


图34：释放内存

发送逻辑，首先判断当前发送序列号是否大于或等于总发包数，如是则说明发送结束，退出发送循环，否则继续发送数据包。

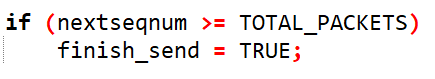


图35：判断条件

主要发送逻辑与GBN状态机中的描述相同。

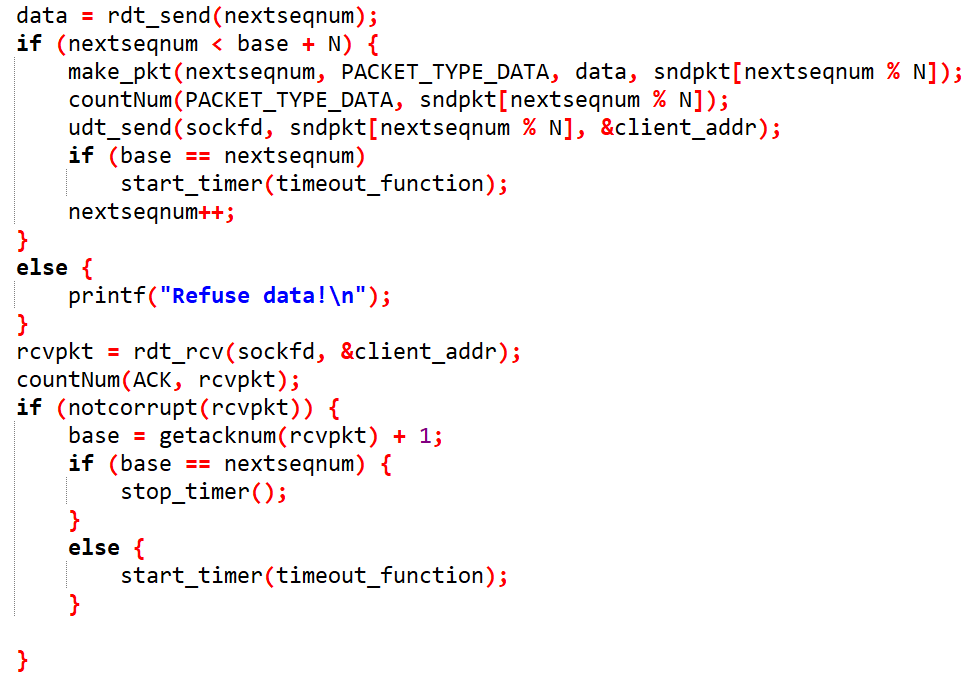


图36：主要发送逻辑

发送结束后统计相关信息，并释放占用的内存，以防内存泄漏。

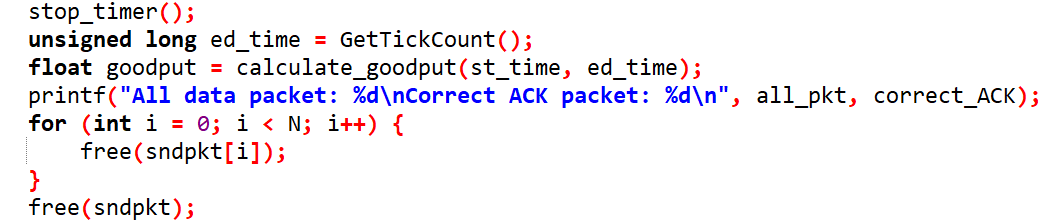


图37：结束前工作

定时器超时时，相应的函数处理内容见下图。

首先启动定时器，然后重发滑动窗口内的所有内容。

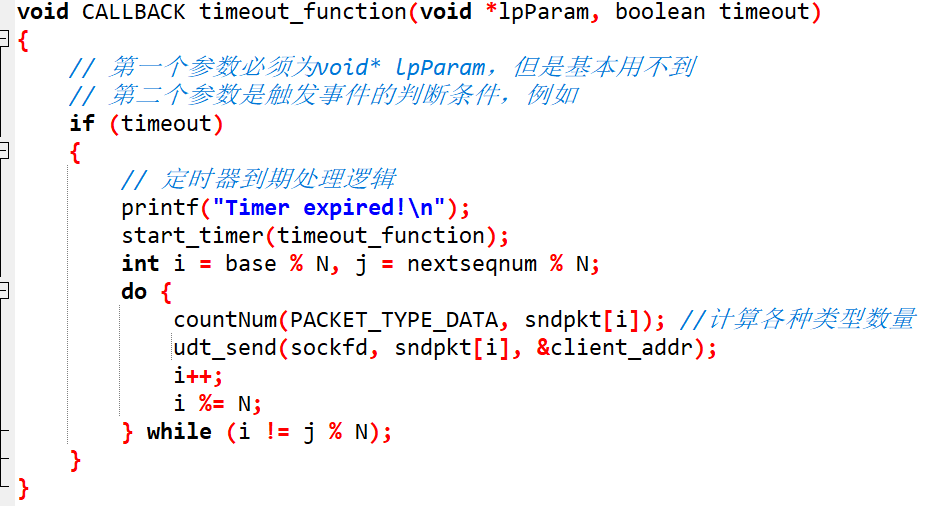


图38：超时函数

以下是GBN机制的接收方的逻辑。

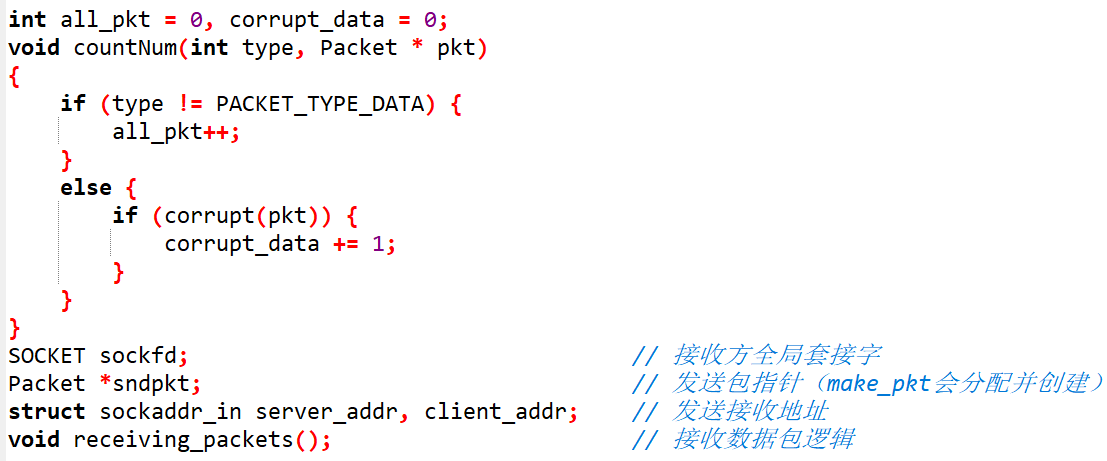


图39：统计函数和全局变量定义

初始化接收端状态：为发送数据包开辟相关空间，定义rcv\_seq、expectedseqnum和封装初始数据包。

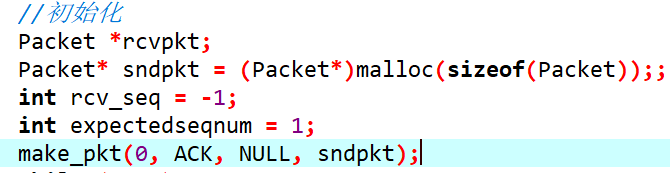


图40：初始化接收端

主要接收逻辑与GBN状态机描述的接收端的动作相同。

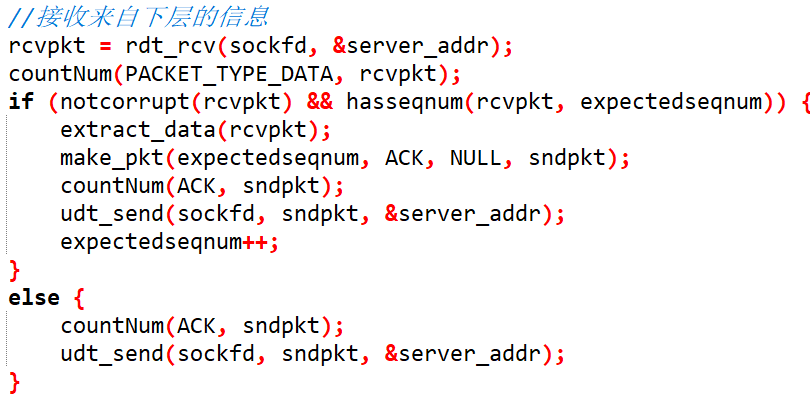


图41：接收数据和发送确认ACK

输出统计结果并释放占用的内存，避免内存泄漏。

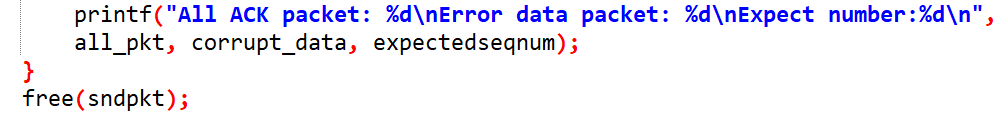


图42：结束前的工作

对rdtv4.h文件的修改：为了方便管理内存，我将make\_pkt函数中为发送数据包（sndpkt）开辟的内存统一为在发送端和接收端初始化阶段开辟，并在发送和接收完成之后统一释放。

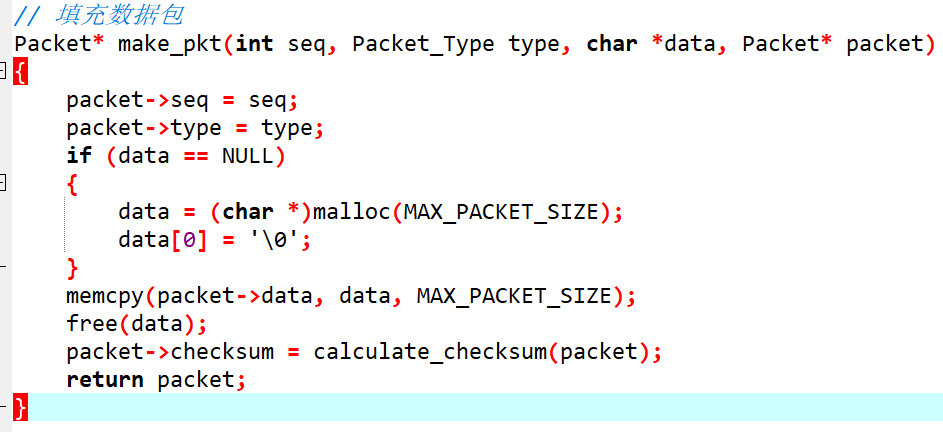


图43：make\_pkt函数

任务5：面向无连接的可靠传输机制（GBN）

任务目标及要求

1. 理解并实现面向无连接的可靠传输机制；
2. 利用clumsy模拟器在不同网络状态下进行实验，并通过作图进行分析；
3. 解释、说明和实现基于NAK的稳定传输机制
4. 利用clumsy模拟器模拟网络状态；
5. 在不同的网络状态下进行实验，并记录相关数据；
6. 将实验数据作图展示，并进行分析；

任务步骤

1. 理解GBN状态转移机制。
2. 理解面向无连接GBN发送方状态机。
3. 首先定义expectedseqnum和将0，ACK及校验和封装成数据包。
4. 接收到来自上层的数据包后将其发送给接收方。
5. 收到来自接收方的信息时，如果信息没有损坏并且是NAK，就重新发送数据包。
6. 接收来自接收方的信息时，如果信息没有损坏并且不是NAK时，不做任何处理。

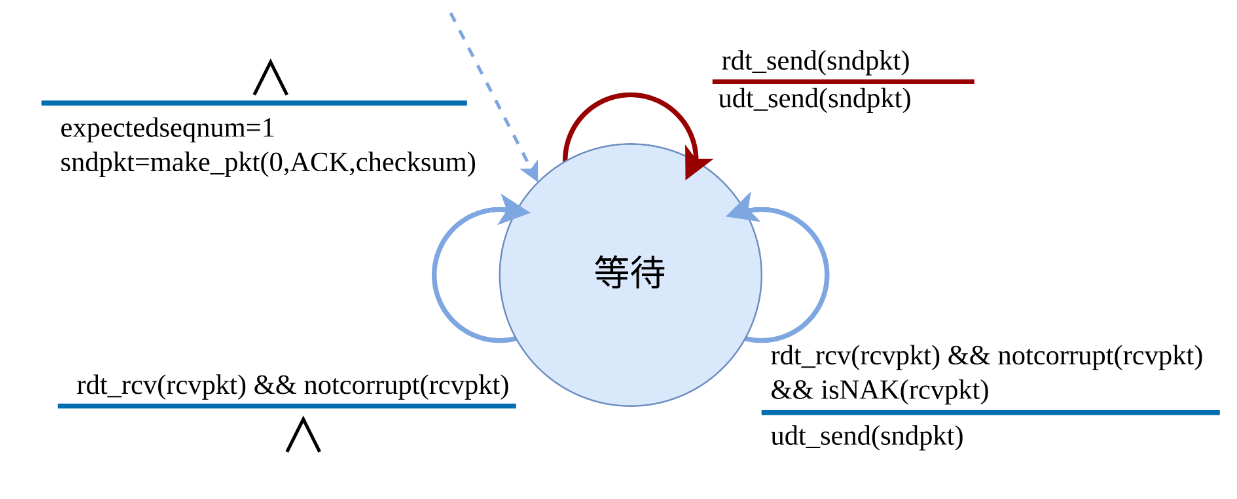


图44：发送方状态机

1. 理解面向无连接GBN接收方状态机。
2. 首先定义base、nextseqnum。
3. 如果收到来自下层的信息但是信息损坏，就发送NAK和base给发送方。
4. 如果收到来自下层的信息，信息没有损坏并且base和nextseqnum+1相等，就将base赋值为nextseqnum+1，并重新启动定时器。
5. 如果base>nextseqnum+1，就发送NAK给发送方。
6. 如果超时，就发送NAK给发送方并启动定时器。
7. 如果不是以上任意一个状态，则不做任何处理。

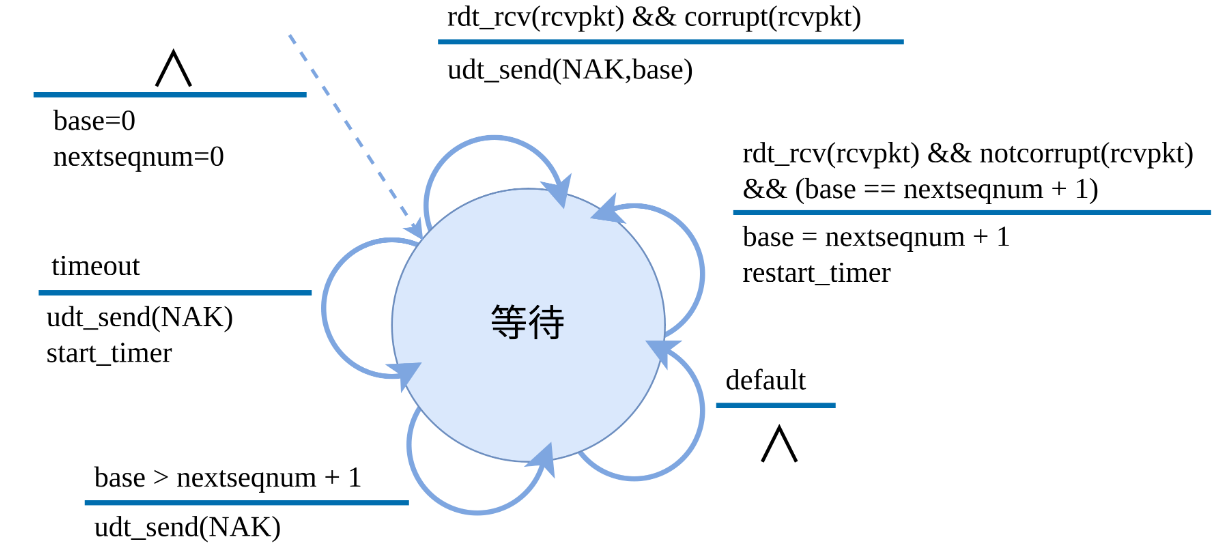


图45：接收方状态机

1. 理解面向无连接基于NAK机制的稳定传输机制细节。
   * + **理解停等协议和非停等协议的区别：在停等协议中，发送方只有确定接收方已经正确接收当前发生的数据包时，才会继续发送新的数据，而在非停等协议中，发送方可以持续发送数据，直到触发错误处理事件（数据包出错、丢包等）；那么，面向无连接基于NAK机制的稳定传输机制属于哪种协议？它应该如何触发错误处理事件？**

面向无连接基于NAK机制的稳定传输机制属于非停等协议。

触发错误处理事件：如果数据包发生错误，接收方将会发送NAK及相关信息给发送方，发送方会采取一定的应对措施。

* + - **在面向无连接的数据传输过程中，接收方并不需要使用ACK向发送方确认数据传输成功，而是使用NAK告知发送方数据发送失败，那么在这种情况下，数据发送的过程是怎样的？请画出无连接数据传输过程中的时序图，并分析NAK机制和ACK机制的区别**；

数据发送过程：接收方检测出数据包发生错误，发送NAK给发送方，发送方收到NAK之后重新发送数据包。

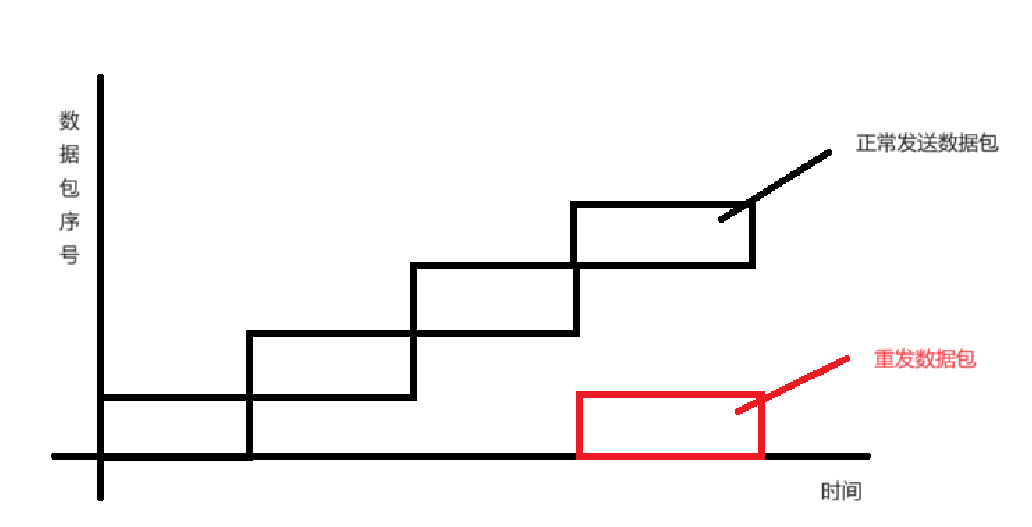


图46：时序图

NAK机制和ACK机制的区别：NAK机制只有在数据包发生错误或超时时才会发送NAK，而ACK对于每一个正确发送的数据包都会发送ACK。NAK机制更加激进，只要没有收到NAK就可以继续发送数据，因此吞吐和时延较ACK机制会更好。在高速网络上，NAK一般较少，因此也无需担心回退N步引起吞吐量下降。在高速网络上，丢包或数据错误概率小，大量ACK反而给网络带来负担，即使采用滑动窗口，发送方也容易到达发送瓶颈，导致一定的时延。

* + - **在无连接数据传输过程中，一旦发生数据内容错误的情况，在不使用ACK的情况下，可否用NAK机制处理？如何处理？**

可以使用NAK机制处理。如果是数据包内容错误的情况，接收方会发送NAK给发送方，由发送方重新发送相应的数据包。如果是NAK内容出现错误的情况，则接收方在定时器超时后重新发送NAK和启动定时器。

* + - **接收端如何发现丢包？**

**💡注意考虑下面两个情况： a. 序号间隔（收到的包不是已经接受过包的下一个顺序包） b. 如何判断最后一个包是否丢失？**

对于情况a，如果base比nextseqnum+1大，则判断为发生丢包。对于情况b，则需要依靠定时器判断是否丢包，如果定时超时则判断最后一个包丢失。

* + - **在无连接数据传输过程中，当NAK数据包本身发生错误或丢失时，NAK机制是如何处理的？**

**💡3b 和 4 是否可以使用相同的机制？**

接收方在定时器超时之后会重新发送NAK并启动定时器。

* + - **当发送方收到NAK数据包时，应如何操作？**

重新发送上次收到的NAK（base）或从头开始（未收到）发送数据包直到当前发送的数据包的序列号。

* + - **当接收方已正确接收丢失数据包的重传时，应如何操作？**

更新相应的变量，然后进入等待状态。等待来自下层的数据包或定时器超时，再执行相应的代码片段。

* + - **当发送方将丢失数据重传完毕后，下一步应该怎么做？**

进入等待状态，等待来自上层或下层的数据，并进行对应的处理。

1. 了解并行事件处理模式。

请根据对面向无连接基于NAK机制的稳定传输理解，回答以下问题：

**在面向无连接基于NAK的稳定传输机制中，哪些事件需要进行并行处理？**

定时器超时的重传、接收NAK和正常发送数据包。

**为什么面向无连接基于NAK的稳定传输机制无法使用停等的方式进行传输？**

只有在数据错误或者丢包时接收方才会给发送方发送消息，因此无法使用停等的方式进行传输。

1. 编程实现。

根据发送方状态机要求，添加了如下变量：rcvpkt用于接受来自接收方方的NAK信息，expectedseqnum用于存储需要重发的位置，all\_data\_pkt和correct\_NAK\_pkt用于统计数据包的数量。

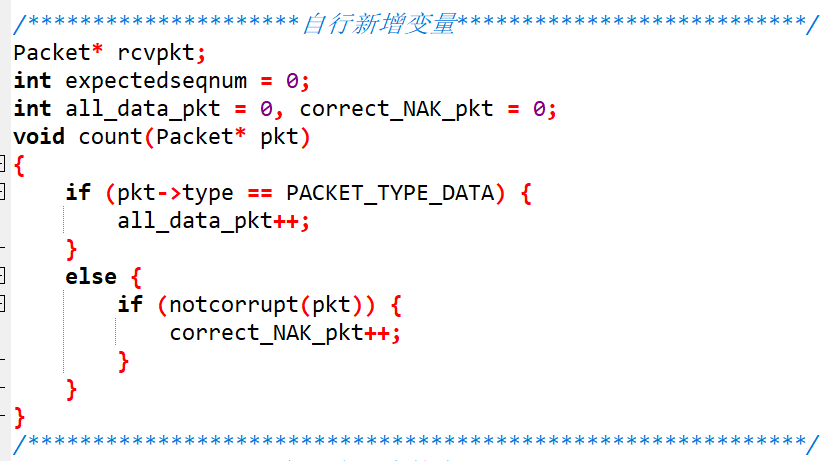


图47：新增内容

对字段进行填充，具体内容与rdt和GBN中的字段内容保持一致。

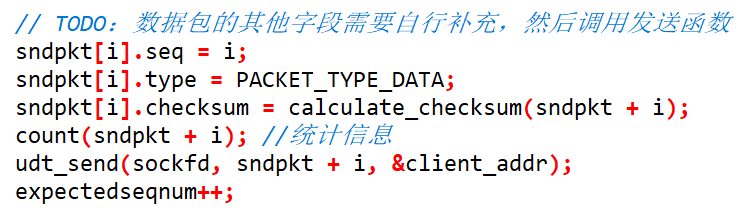


图48：填充数据包字段

发送方收到NAK信息，并将相应的数据包进行重发（我们只需要重发base到expectedseqnum中的数据包即可）。

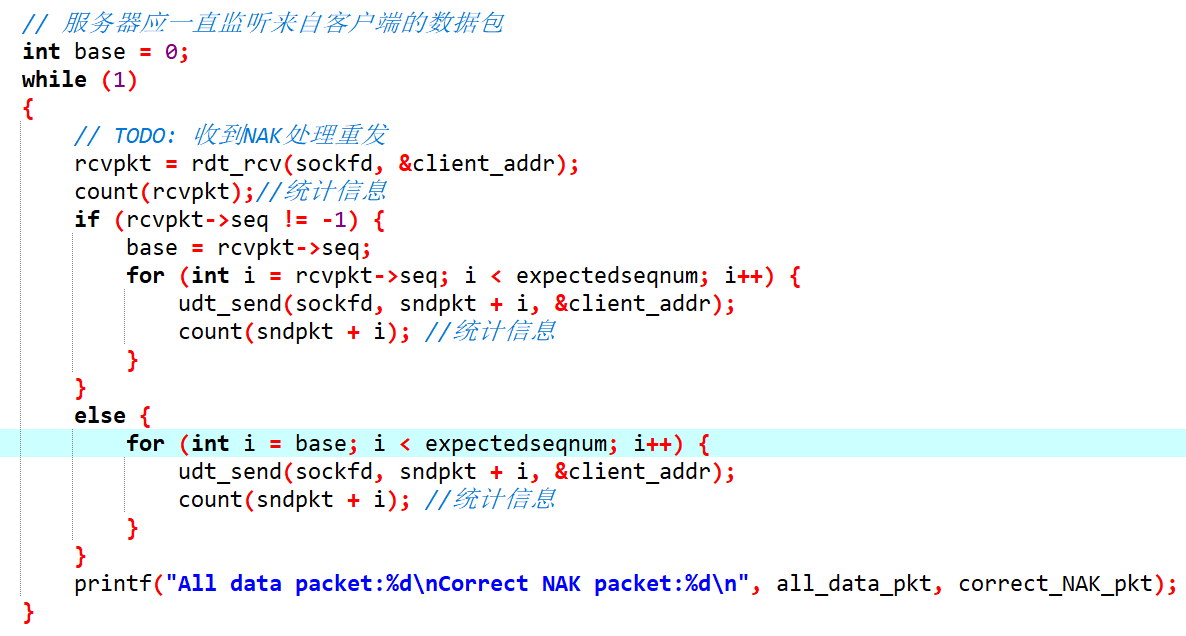


图49：收到NAK处理重发

根据接收方状态要求，添加了如下变量：sndpkt用于发送NAK给发送方，base和nextseqnum的作用和状态机中的描述一致。此外，新增了变量all\_NAK\_pkt和correct\_data\_pkt以及函数count，用于统计数据包信息。

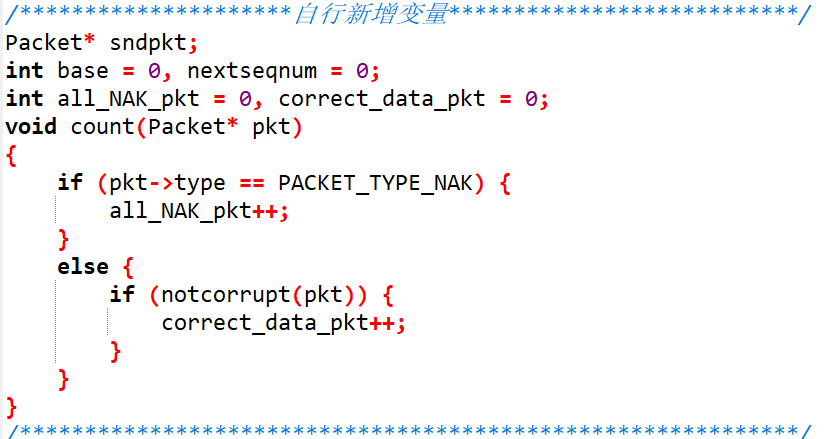


图50：新增内容

接收来自下层的数据，然后执行相应的处理代码。再LOOP2中添加数据是否传输完成的判断和最大接收序列号的更新，其余逻辑与接收方状态相对应。具体代码见下图。

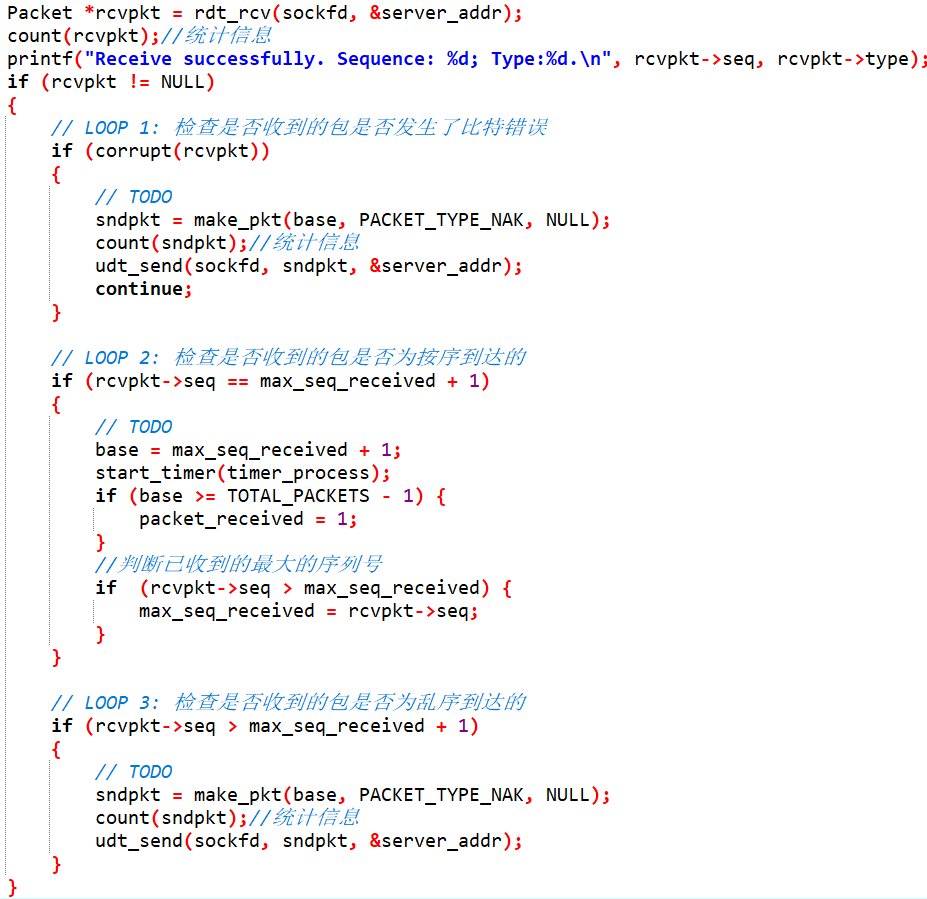


图51：接收数据逻辑

定时器超时函数如下图所示。

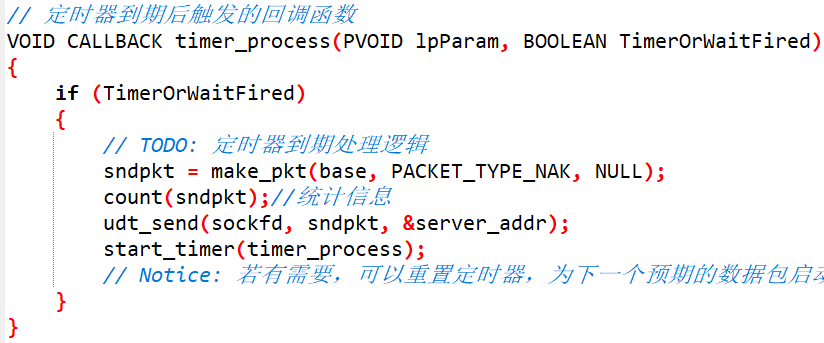


图52：超时函数

任务6：量化分析

任务目标及要求

1. 对比分析基于ACK的数据传输机制（GBN）和基于NAK的无连接数据传输机制；
2. 理解两种机制的差异；
3. 利用clumsy模拟器在不同网络状态下进行实验，并通过作图进行分析；
4. 利用clumsy模拟器模拟网络状态；
5. 在不同的网络状态下进行实验，并记录相关数据；
6. 将实验数据作图展示，并进行分析；

任务步骤

1. 对比两种机制在数据包错误情况下的差异。

**使用clumsy模拟数据包错误状况，并记录以下数据：**

**数据发送过程中的OverHead（发送端总发包量 + 接收端总发包量，即包括ACK/NAK包、丢失包、重传包等）**

**数据发送过程的Goodput（（有效数据包数量 × 数据包大小）÷ 数据发送总时间；有效数据包不计算ACK/NAK和丢失重传包）**

**clumsy模拟参数设置如下： Drop设置为1%，Out of Order设置为1%，数据包错误频率设置为0.3%、0.5%、1%、2%、3%、5%、10%、20%、30%。**

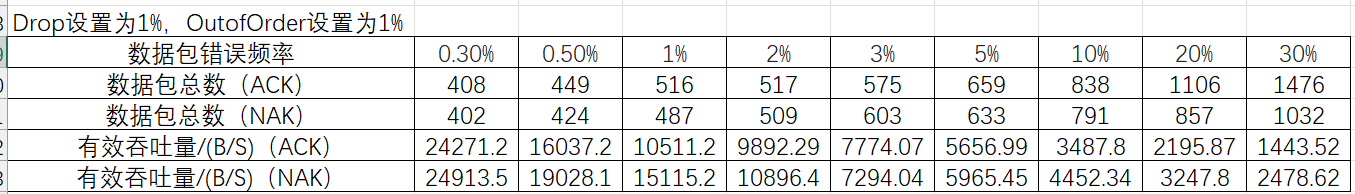


图53：数据图

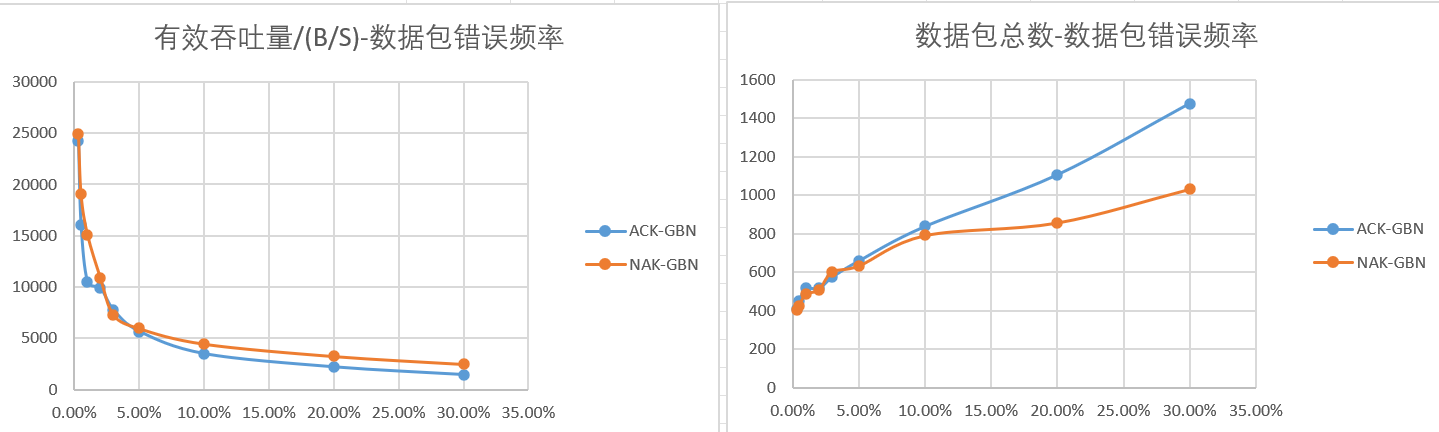


图54：数据可视化

1. 对比两种机制在数据包丢包情况下的差异。

**使用clumsy模拟数据包错误状况，并记录以下数据：**

**数据发送过程中的OverHead（发送端总发包量 + 接收端总发包量，即包括ACK/NAK包、丢失包、重传包等）**

**数据发送过程的Goodput（（有效数据包数量 × 数据包大小）÷ 数据发送总时间；有效数据包不计算ACK/NAK和丢失重传包）**

**clumsy模拟参数设置如下： Tamper设置为1%，Out of Order设置为1%，丢包率设置为0.3%、0.5%、1%、2%、3%、5%、10%、20%、30%。**

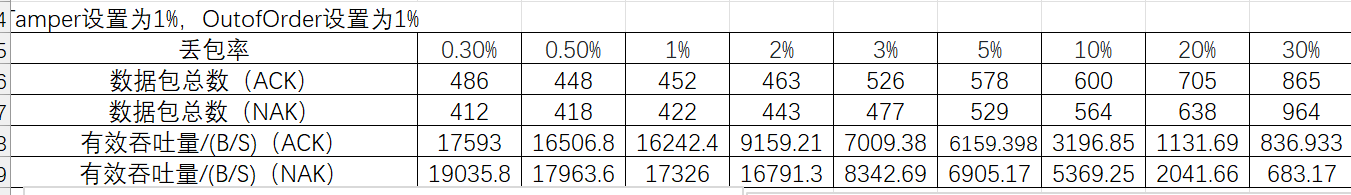


图55：数据图

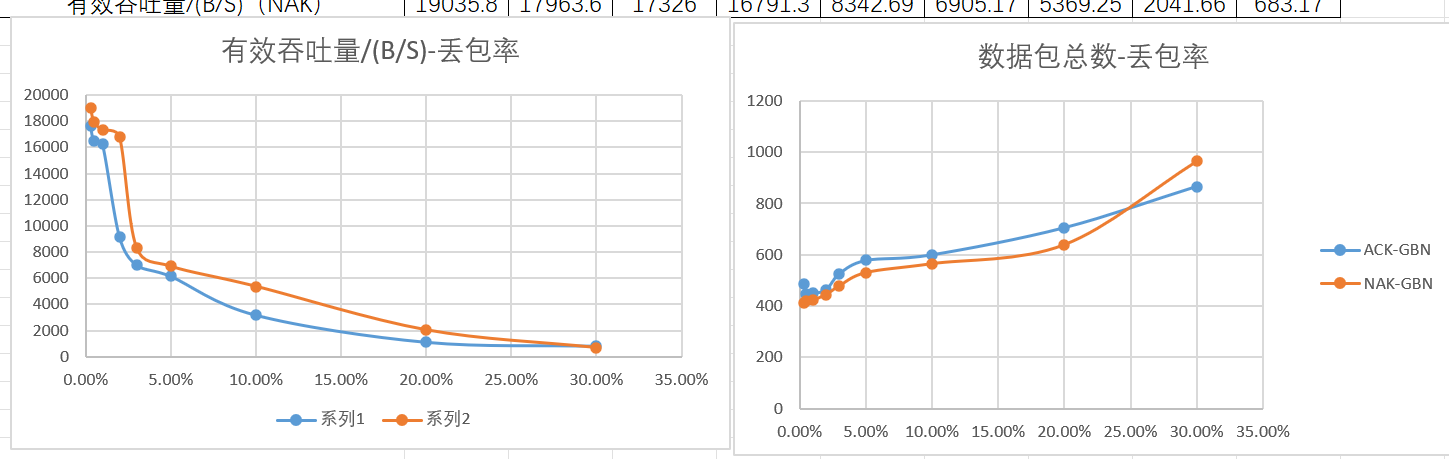


图56：数据可视化

1. 对比两种机制在数据包乱序情况下的差异。

**使用clumsy模拟数据包错误状况，并记录以下数据：**

**数据发送过程中的OverHead（发送端总发包量 + 接收端总发包量，即包括ACK/NAK包、丢失包、重传包等）**

**数据发送过程的Goodput（（有效数据包数量 × 数据包大小）÷ 数据发送总时间；有效数据包不计算ACK/NAK和丢失重传包）**

**clumsy模拟参数设置如下： Drop设置为1%，Tamper设置为1%，乱序数据包比例设置为0.3%、0.5%、1%、2%、3%、5%、10%、20%。**

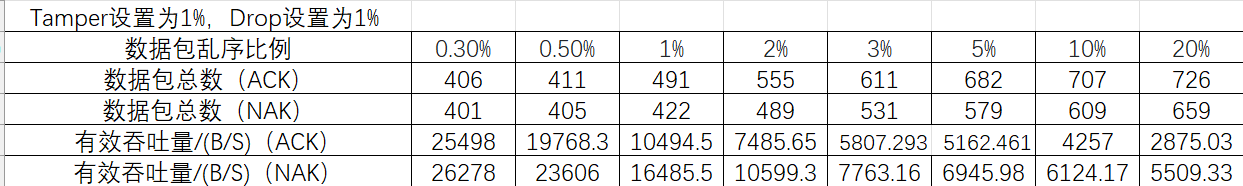


图57：数据图

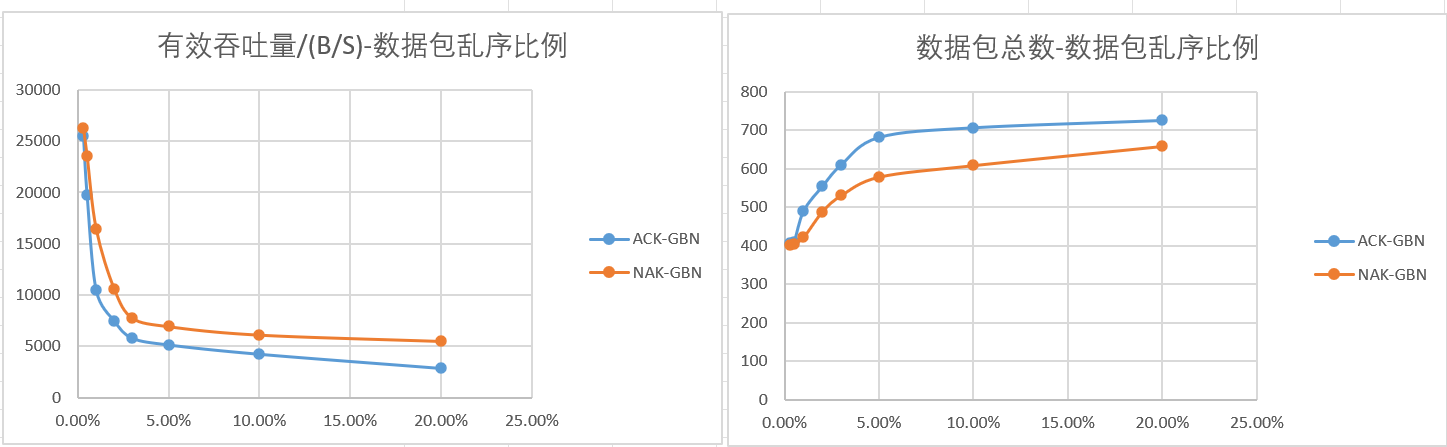


图58：数据可视化

1. 实验分析。
2. 经三次控制变量实验发现基于NAK的GBN机制优于基于ACK的GBN机制。
3. 有效吞吐量与数据包错误频率、数据包丢包率、数据包乱序比例均为指数关系。数据包总数与数据包错误频率和数据包总数与丢包率基本呈线性关系。数据包总数与数据包乱序比例呈对数关系，即数据包乱序比例较低时呈线性关系，数据包乱序比例较高时，数据包总数增长速率逐渐减缓。
4. 在本次实验中，基于NAK的GBN机制基本全面优于基于ACK的GBN机制。但是在丢包率较高时，基于ACK的GBN机制可能会优于基于NAK的GBN机制。

# 实验结果

**下图如有实验过程或实验结果，则相应任务的变量取值均为10%。**

Rdt2.1实验过程见下图。

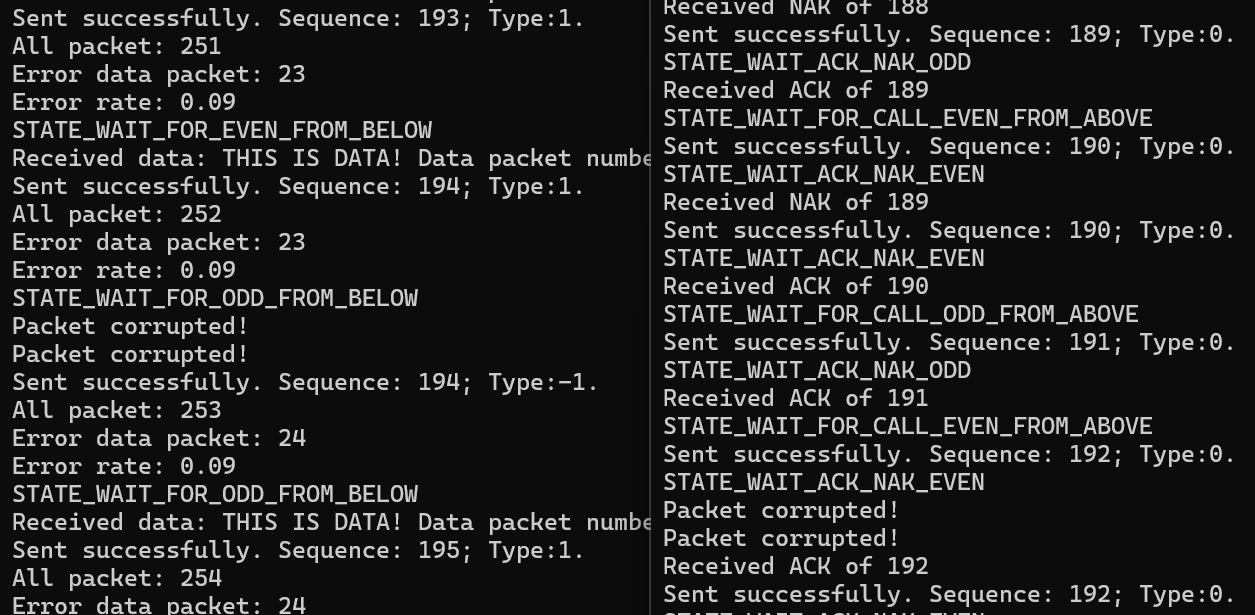


图59：实验过程

Rdt2.1实验结果见下图。



图60：实验结果

Rdt2.1可视化实验结果见下图。

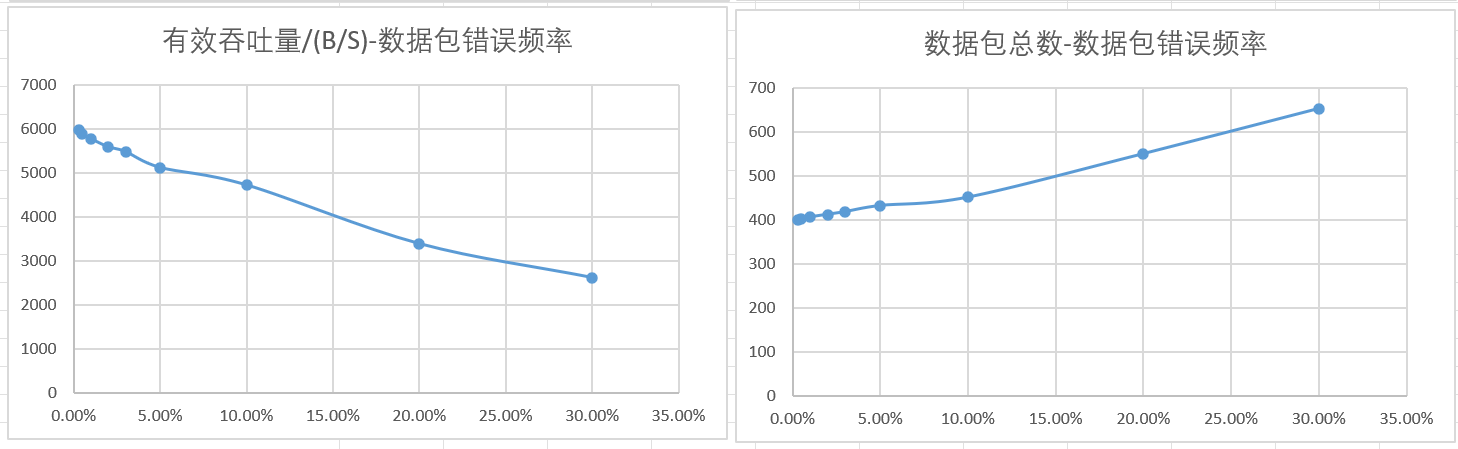


图61：可视化结果

Rdt2.2实验过程见下图。

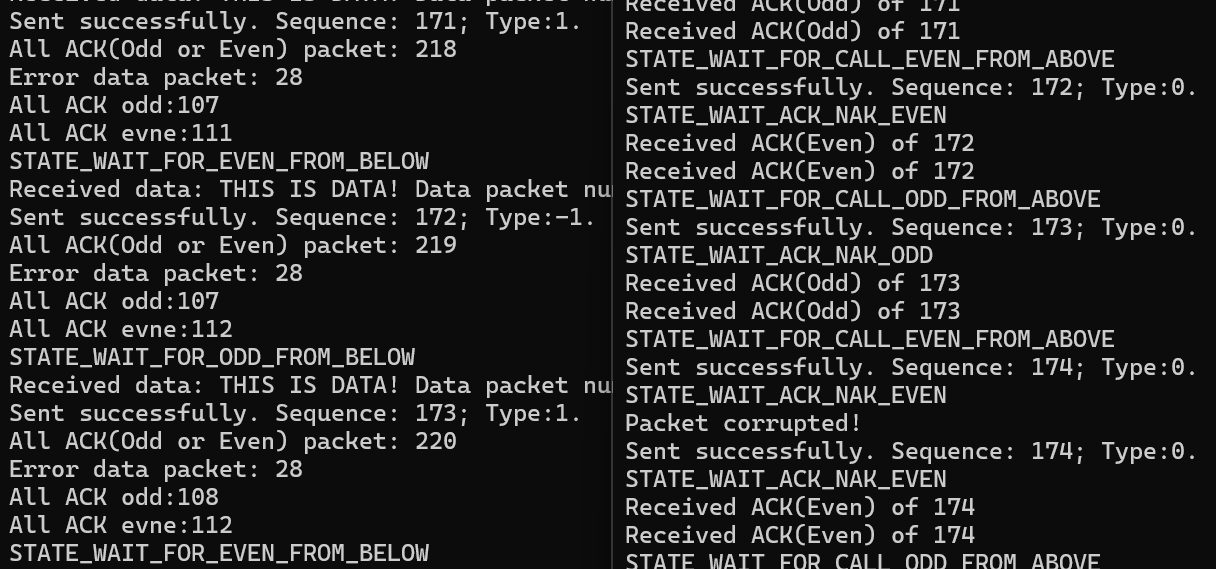


图62：实验过程

Rdt2.2实验结果见下图。

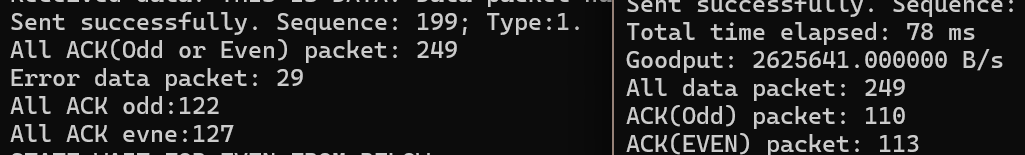


图64：实验结果

Rdt2.2可视化实验结果见下图。

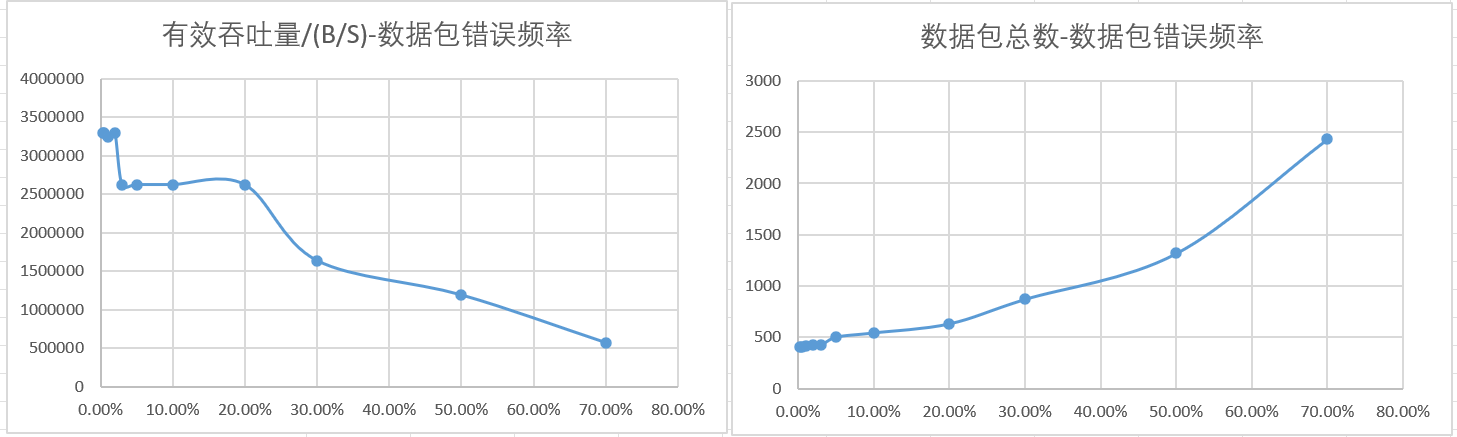


图65：可视化结果

Rdt3.0实验过程见下图。

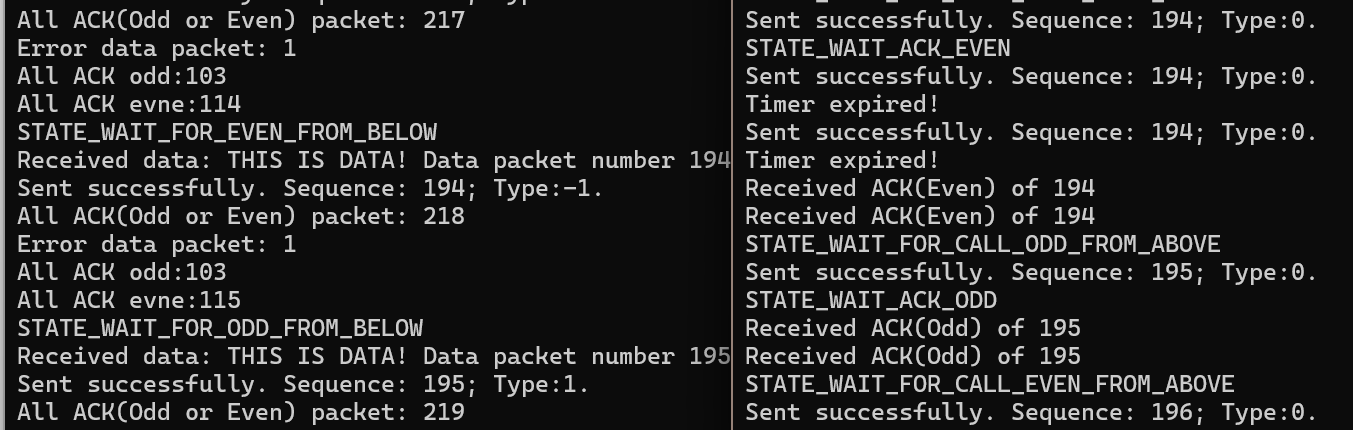


图66：rdt3.0实验过程

Rdt3.0实验结果见下图。

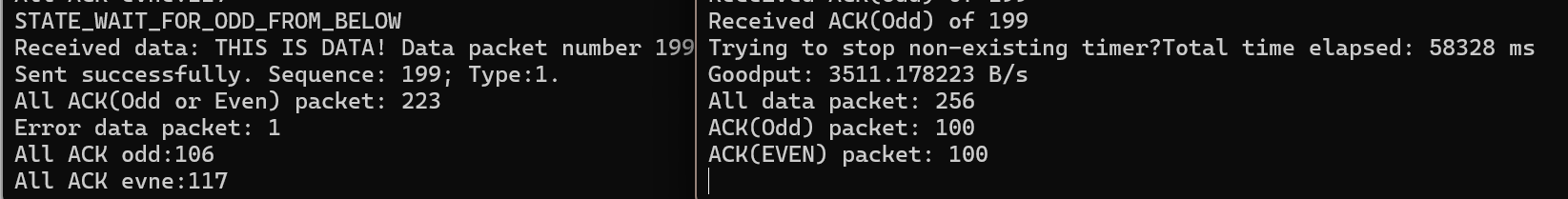


图67：rdt3.0实验结果

Rdt3.0可视化实验结果见下图。

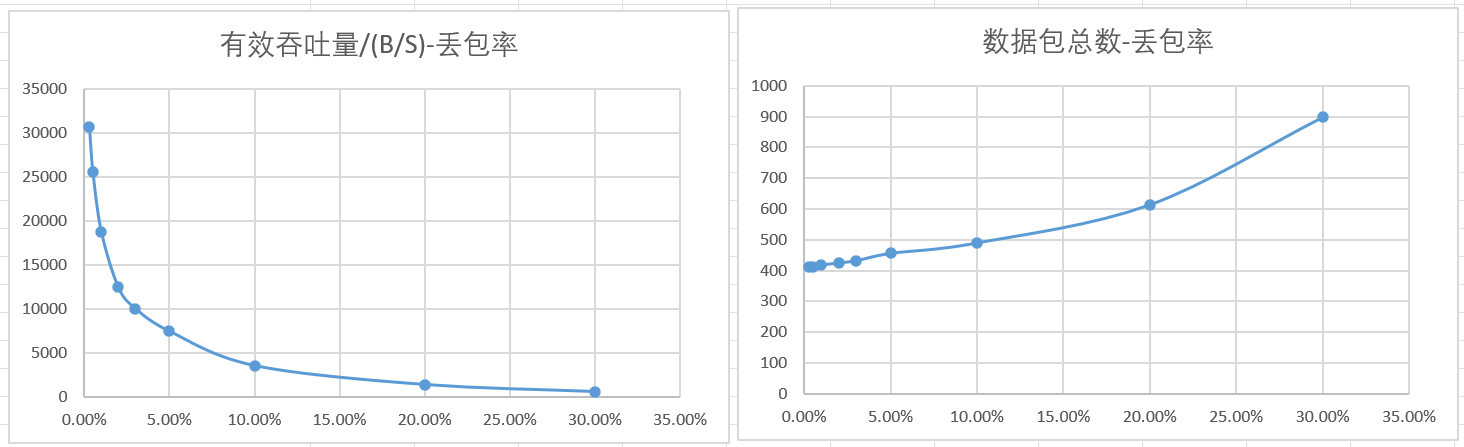


图68：可视化结果

GBN实验过程见下图。

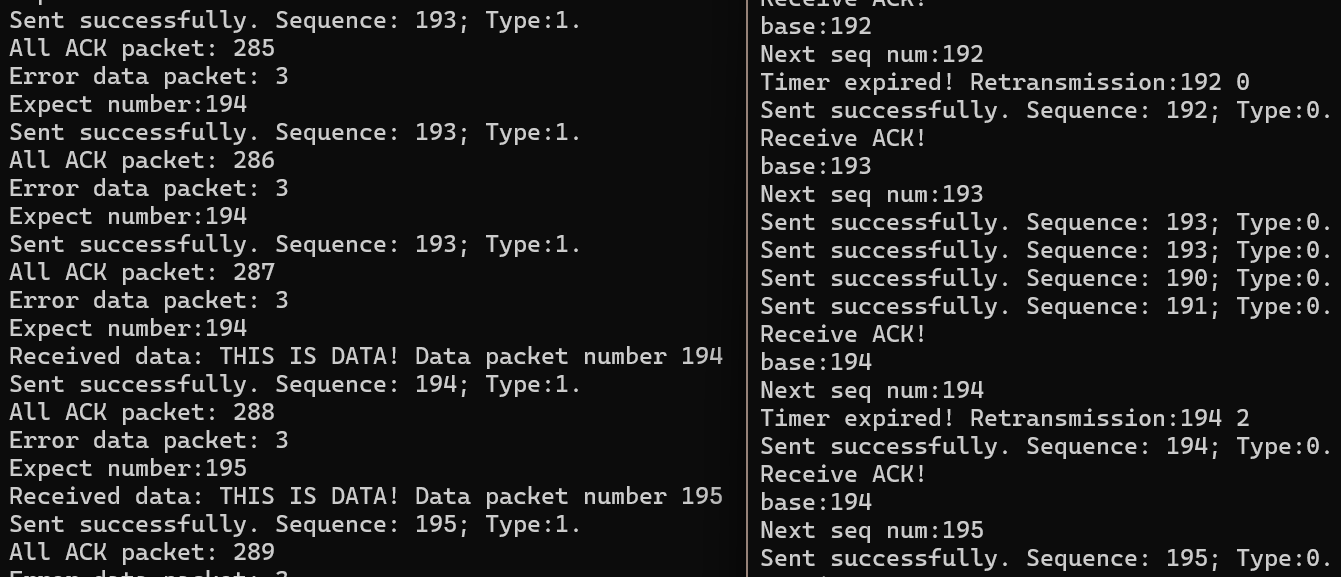


图69：GBN实验过程

GBN实验结果见下图。

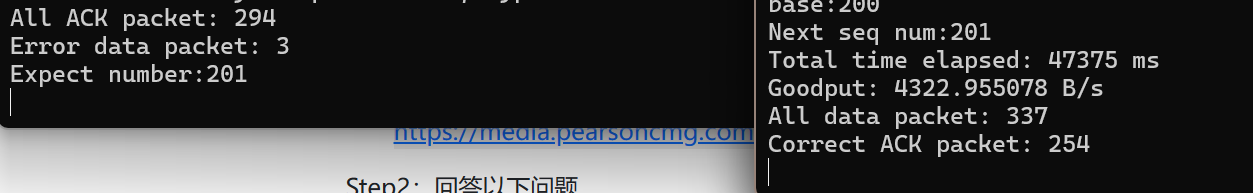


图70：GBN实验结果

GBN可视化实验结果见下图。

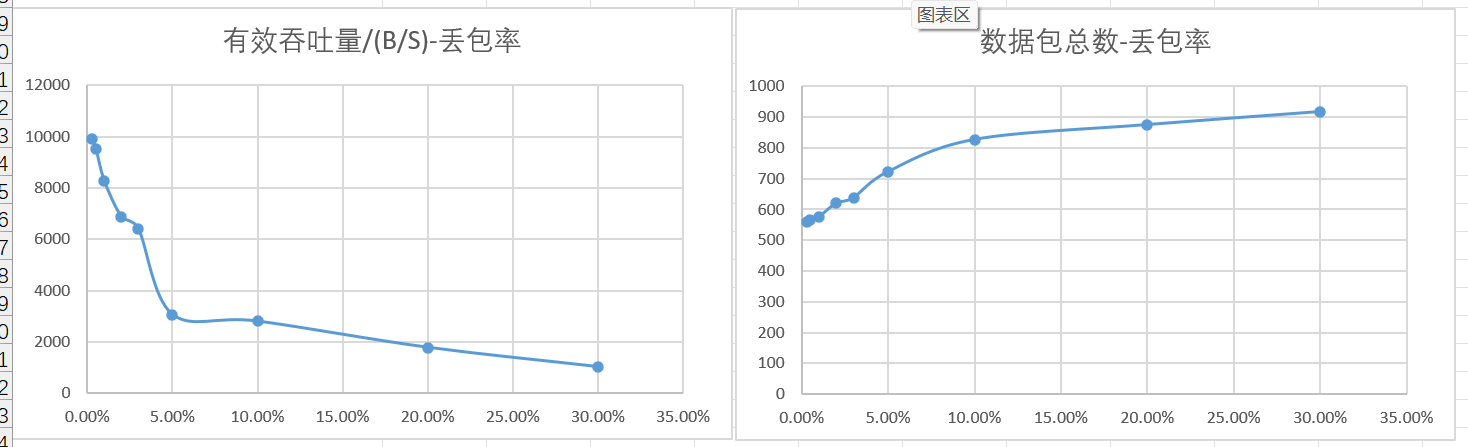


图71：可视化结果

量化分析的可视化结果

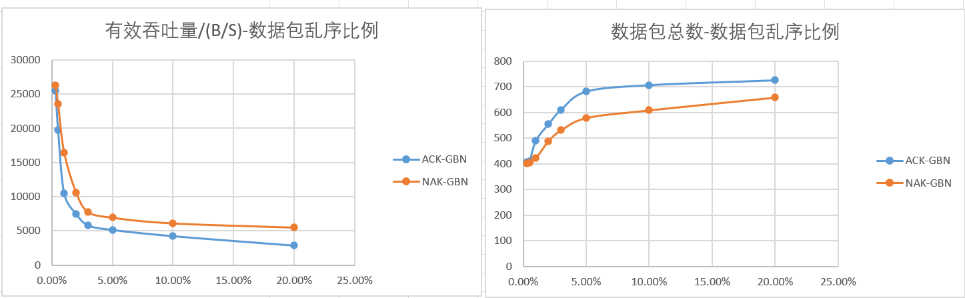


图72a：可视化结果

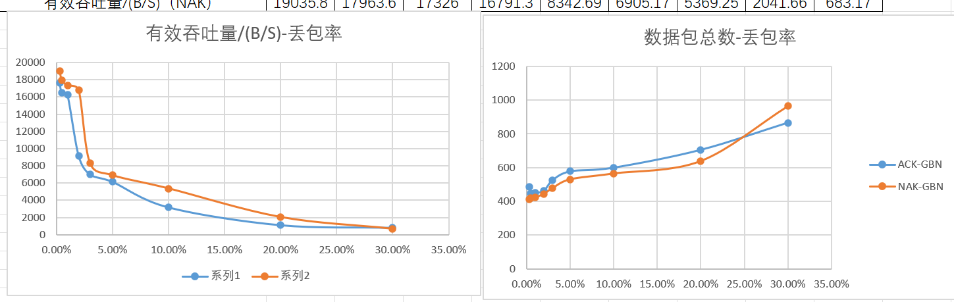


图72b：可视化结果

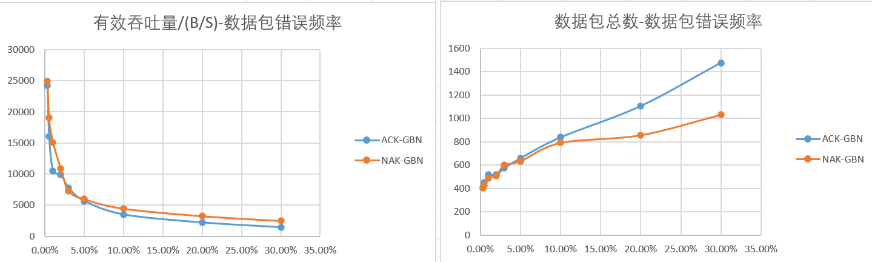


图72c：可视化结果

# 实验分析

1. 在实现任务4：回退N步（GBN）机制的时候，程序在运行的过程中总会中断退出，并返回值3221226505。经上网查阅资料发现该返回值代表内存访问冲突。然后仔细阅读代码发现，发送方应该在超时重发之前启动定时器，但是我的代码是在重发之后启动定时器。这会与接收确认消息ACK启动定时器的步骤同时进行，导致程序同时访问一块内存，造成错误。后更改超时重发定时器的位置，程序正确运行，不再出现错误。
2. 在本次实验中，由于clumsy软件的随机性以及没有取多次平均值，导致实验结果并不完全精确。

# 实验总结

* 1. 通过阅读相关资料，我理解了网络传输过程中可能出现的差错、网络传输中的稳定传输机制和差错处理方法。
  2. 通过查阅资料，我熟悉了面向无连接的稳定传输机制，了解了滑动窗口机制，理解了面向无连接的稳定传输机制。
  3. 通过本次实验，我初步理解和掌握不同网络情况下，不同稳定传输机制的差异。
  4. 通过本次实验，我理解和掌握可靠传输机制的原理。

# 思考题

指导教师批阅意见

成绩评定

指导教师签字：

年 月 日

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。