**西安电子科技大学网信院**

**计算机网络原理实验**

**实 验 报 告**

**后退N帧算法模拟**

**班级：**

**姓名：**

**学号：**

**日期：2024年12月20日**

1. 实验目的（包括实验环境、实现目标等等）
2. 实验环境：python3.12.7 pyame模块
3. 实现目标：

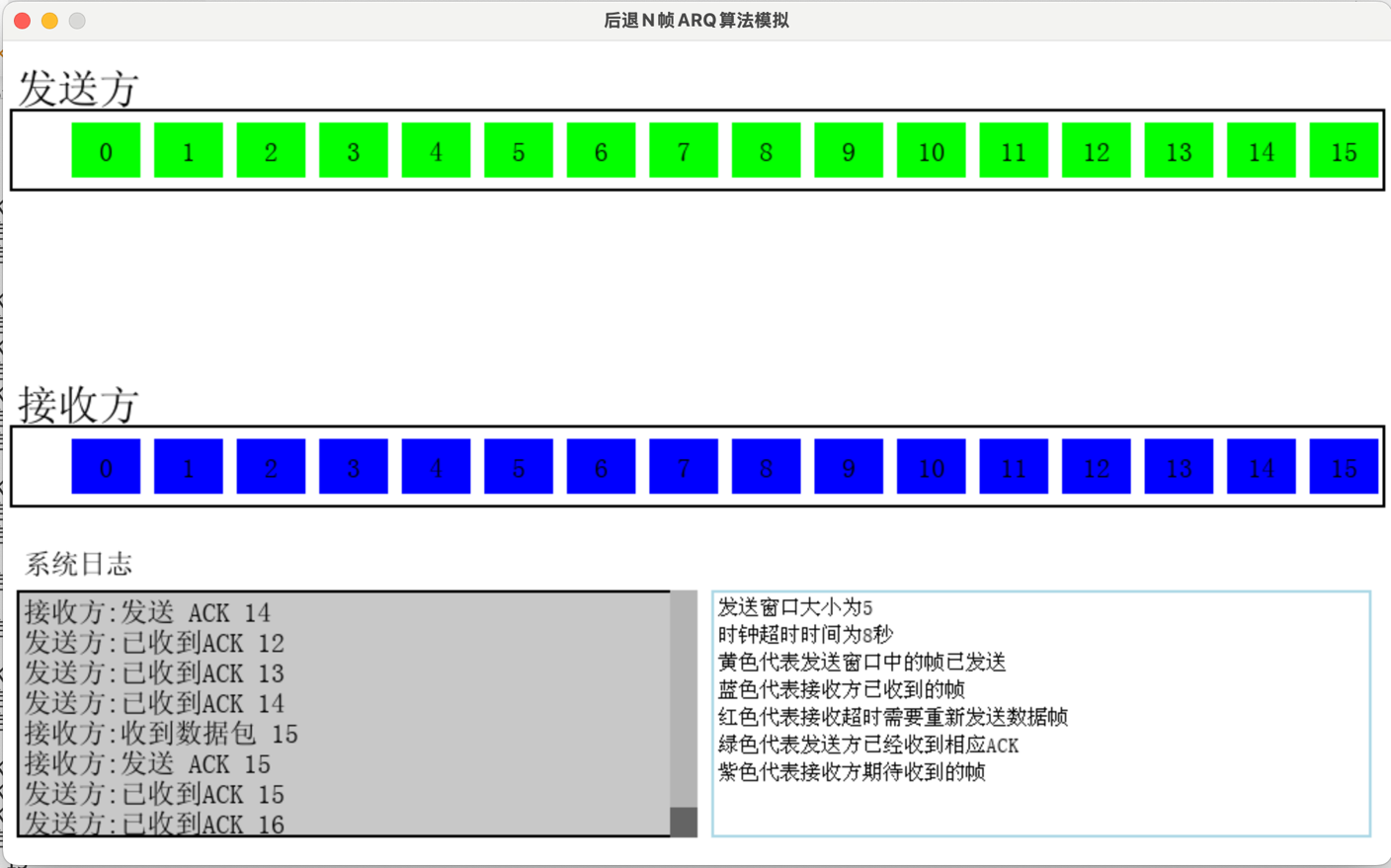
写两个程序，一个发送方S一个接收方R。S程序从上层拿到数据包后要组装成帧，缓存、发送、定时，可以接收ACK应答，删除数据帧，停止定时器。R程序可以接收数据帧，回应ACK，上传数据帧到上层。S程序和R程序需要能模拟所有异常的处理过程，例如数据帧或ACK的丢失、损坏、超时

二、方案设计

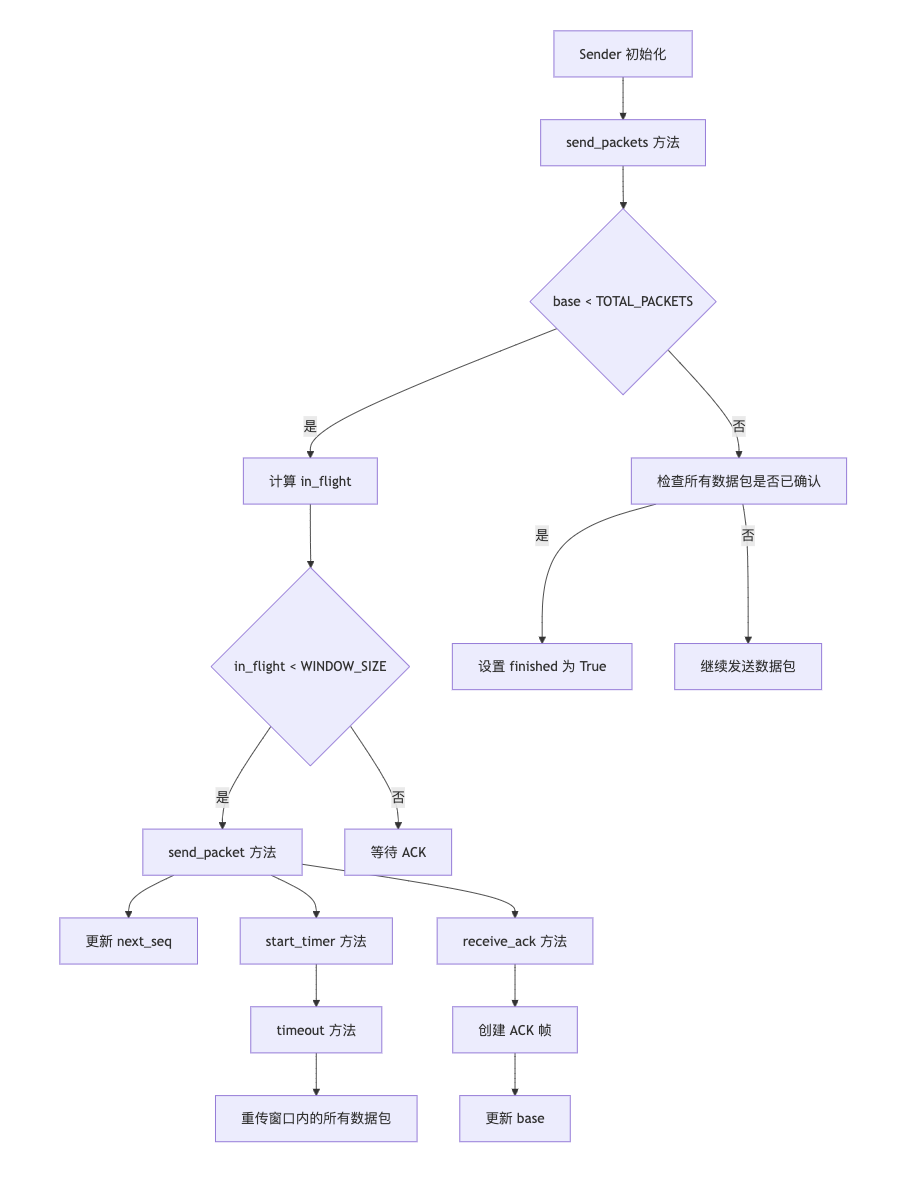
本次程序设计主要包括：成帧、发送方、接收方、记录log这几个类。成帧模块主要负责绘制帧，并且控制其颜色（运用颜色来判断帧的发送、接收、超时等状态）、移动方式和是否被鼠标点按（规定点击帧则该帧失效）。发送方模块主要是实现发送滑动窗口内数据帧以及启动和暂停定时器，并且负责接收ACK以便确定下一个发送帧，回退N帧重传也在此实现。接收方负责传送ACK并且更新下一个希望接收的帧。

设计界面如下：上方为发送方窗口，下方为接收方窗口，左下方是系统日志运用滚动条可以查看实验中每个帧的状态，右下方是提示字段。

在实验中通过点按传送的帧使其失效，相应的帧根据右下方提示中对应颜色更新帧的实时状态，同时通过log日志模块记录整个系统操作变化，最后在实验结束后我们能够拖动左下方滚动条查看实验过程每步操作日志。

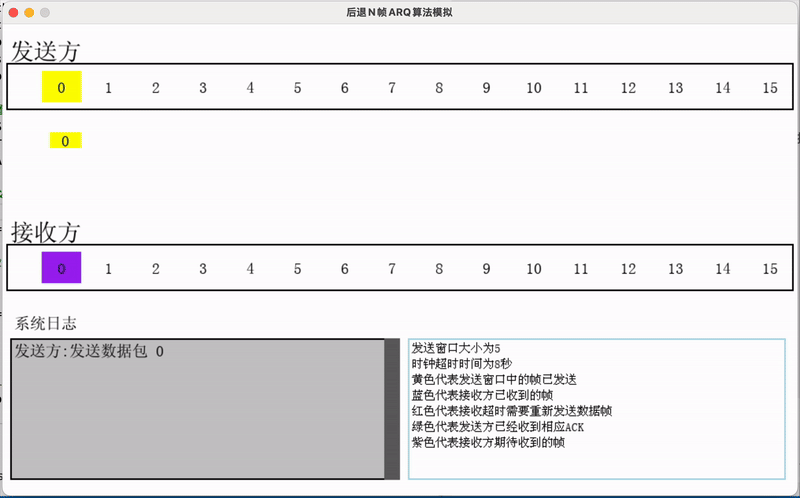


关键模块（发送方）设计流程图如下：



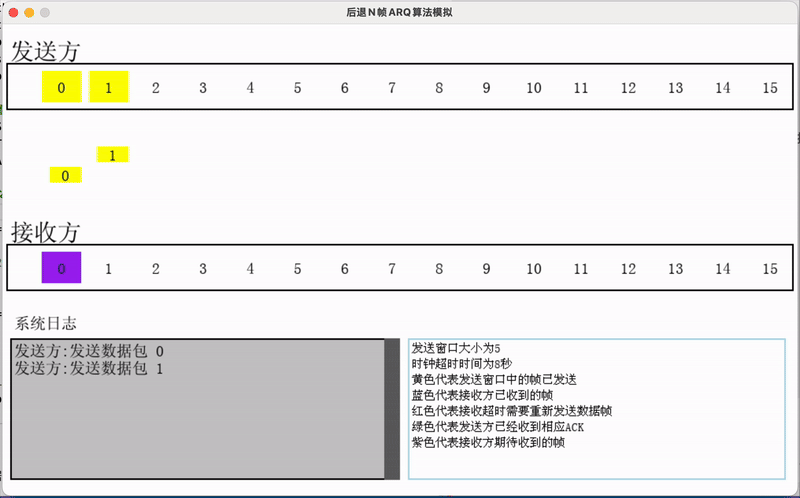
1. 演示动图
2. 正常收发状态

此时发送方和接收方都正常收发数据帧和确认ACK帧。



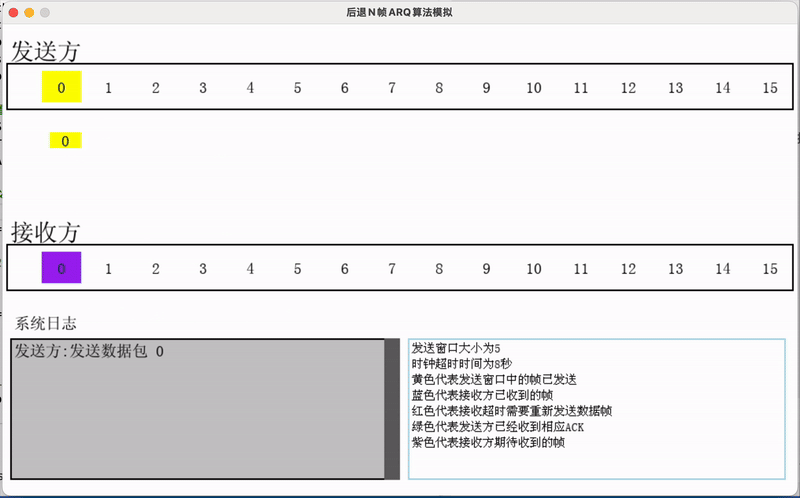
1. 有数据帧丢失状态

此时我们通过鼠标点按发出去的数据帧让其丢失，等到8s之后发送端未收到ACK帧超时（置为红色）重新传送窗格内数据帧。

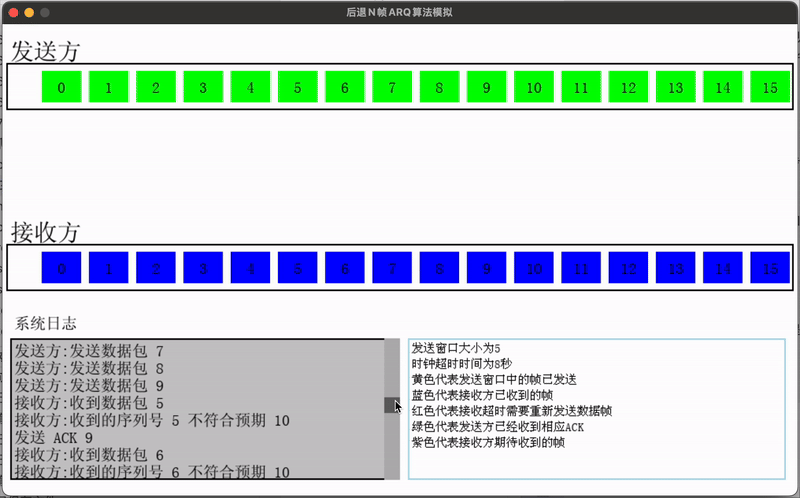


1. 有ACK帧丢失状态

通过用鼠标点按ACK帧使数据帧不能够到达接收方，此时经过8s发送发超时，会重新传送窗格内的数据帧。



最后我们可以拖动滚动条查看系统日志记录



1. 关键代码简介

1. 发送方模块：

Sender 类定义

1. 数据包发送

  def send\_packets(self):

while self.base < TOTAL\_PACKETS and self.running:

self.lock.acquire()

# 计算当前在发送窗口内的未确认帧数量

in\_flight = self.next\_seq - self.base

if in\_flight < WINDOW\_SIZE and self.next\_seq < TOTAL\_PACKETS:

self.send\_packet(self.next\_seq)

self.next\_seq += 1

self.lock.release()

time.sleep(0.5) # 每发送一个帧后暂停0.5秒

else:

self.lock.release()

time.sleep(0.1) # 窗口已满，等待ACK

# 检查是否所有数据包都已确认

self.lock.acquire()

if self.base >= TOTAL\_PACKETS:

if all(status == 'green' for status in self.packets\_status):

self.finished = True

self.lock.release()

break

self.lock.release()

1. 单个数据包发送

绘制新的数据帧，并且在更改颜色为黄色，将发送状态写入日志。

def send\_packet(self, seq):

print(f"发送方:发送数据包 {seq}")

log.add\_message(f"发送方:发送数据包 {seq}")

self.update\_packet\_status(seq, 'yellow')

frame = Frame(seq, 70 + seq \* 60, 130, 250)

self.frames.append(frame)

self.start\_timer(seq)

1. 启动计时器

def start\_timer(self, seq):

timer = threading.Timer(TIME\_OUT, self.timeout, args=(seq,))

timer.start()

self.timers[seq] = timer

1. 超时处理

超时处理更新日志状态，停止所有计时期并且将下一个要发送的数据帧置为窗口起始帧，将超时帧变成红色。

def timeout(self, seq):

self.lock.acquire()

print(f"发送方:数据包 {seq} 超时，重传窗口从数据包 {self.base} 开始")

log.add\_message(f"发送方:数据包 {seq} 超时，重传窗口从数据包 {self.base} 开始")

# 停止所有计时器

for t in self.timers.values():

t.cancel()

self.timers.clear()

# 重发窗口内的所有数据包

self.next\_seq = self.base

for i in range(self.base, self.base + WINDOW\_SIZE):

if i < TOTAL\_PACKETS:

self.update\_packet\_status(i, 'red')

self.lock.release()

1. 接收 ACK

def receive\_ack(self, ack):

self.lock.acquire()

# 创建 ACK 帧

ack\_frame = Frame(ack + 1, 70 + ack \* 60, 250, 130, ack=True)

self.frames.append(ack\_frame)

self.lock.release()

1. 更新数据包状态（颜色）

def update\_packet\_status(self, seq, color):

self.packets\_status[seq] = color

1. 停止发送方

def stop(self):

self.running = False

# 取消所有计时器

self.lock.acquire()

for t in self.timers.values():

t.cancel()

self.timers.clear()

self.lock.release()

2.接收方模块：

Receiver 类定义

1. 设置发送方

def set\_sender(self, sender):

self.sender = sender

1. 数据包接收

def receive\_packets(self):

while self.expected\_seq < TOTAL\_PACKETS and self.running:

time.sleep(0.1) # 每隔0.1秒检查一次

1. 检查数据帧

def check\_frame(self, frame):

if frame.y == frame.dest\_y and not frame.ack and frame.active == False:###

print(f"接收方:收到数据包 {frame.seq}")

log.add\_message(f"接收方:收到数据包 {frame.seq}")

if frame.seq == self.expected\_seq:

self.update\_packet\_status(frame.seq, 'blue')

# 发送 ACK

print(f"接收方:发送 ACK {frame.seq}")

log.add\_message(f"接收方:发送 ACK {frame.seq}")

self.sender.receive\_ack(frame.seq)

self.expected\_seq += 1

else:

print(f"接收方:收到的序列号 {frame.seq} 不符合预期 {self.expected\_seq}")

log.add\_message(f"接收方:收到的序列号 {frame.seq} 不符合预期 {self.expected\_seq}")

# 在 Go-Back-N 中，如果收到不期望的帧，会发送最后一个已确认的 ACK

if self.expected\_seq > 0:

last\_ack = self.expected\_seq - 1

print(f"发送 ACK {last\_ack}")

log.add\_message(f"发送 ACK {last\_ack}")

self.sender.receive\_ack(last\_ack)

return True

elif frame.y == frame.dest\_y and frame.ack and frame.active == False:

print(f"发送方:已收到ACK 帧 {frame.seq}")

log.add\_message(f"发送方:已收到ACK 帧 {frame.seq}")

return True

return False

1. 更新数据包状态

def update\_packet\_status(self, seq, color):

self.packets\_status[seq] = color

1. 停止接收方

def stop(self):

self.running = False

1. 成帧准备模块
2. 移动功能（使发送方的帧向下运动，接收方的帧向上运动）

def move(self):

if self.y < self.dest\_y:

self.y += self.speed

if self.y >= self.dest\_y:

self.y = self.dest\_y

self.active = False

elif self.y > self.dest\_y:

self.y -= self.speed

if self.y <= self.dest\_y:

self.y = self.dest\_y

self.active = False

1. 绘制帧模块

def draw(self):

pygame.draw.rect(screen, self.color, (self.x - 10, self.y, 40, 20))

seq\_text = font.render(str(self.seq), True, BLACK)

screen.blit(

seq\_text,

(self.x +10 - seq\_text.get\_width() // 2, self.y + 10 - seq\_text.get\_height() // 2))

1. 判断帧被鼠标点按

def is\_clicked(self, pos):#规定鼠标点到范围内

rect = pygame.Rect(self.x - 10, self.y, 40, 20)

return rect.collidepoint(pos)

4. 日志模块

1. [add\_message(message)](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)：添加日志消息到[lines](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)列表，并更新滚动条滑块位置。
2. 滚动功能：

[scroll\_up()](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)：向上滚动日志内容。

[scroll\_down()](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)：向下滚动日志内容。

1. [update\_thumb()](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)：更新滚动条滑块的位置和大小。
2. 事件处理[handle\_event(event)](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)：处理鼠标事件，包括点击、释放和拖动，用于滚动条的交互。
3. 绘制功能：[draw(screen)](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)：绘制日志窗口背景、边框、标题、日志内容、滚动条轨道和滑块。

5. 绘制主界面函数draw\_window(sender, receiver)

此函数包括绘制发送方接收方窗口、log日志窗口和文字解释窗口。结合帧移动位置和全局设置的帧率clock.tick(30)我们能够把动画刷新到屏幕上实现后退N帧协议过程模拟。

1. 思考与总结

通过本实验，我们深入理解了后退 N 帧ARQ 协议的工作原理，并掌握了如何在程序中实现该协议。我们成功实现了发送方和接收方程序，并能够模拟各种异常情况。在实验中我发现如果接收方仅仅在收到预想接收的帧时发送ACK，那么当发送方窗口所有数据帧已经被接受方接收但是所有的ACK帧全部丢失时，仅通过发送方的超时判断会进入死循环，接收方也不会收到自己预想的帧。所以说我们需要在接受方收到不符合帧预期帧的时候发送自己预期的帧，帮助发送方更新下一个发送的帧。