北京郵電大學

实验报告

课程名称 计算机组成原理

实验名称

运算器、双端口储存器、数据通路联合实验

班级： 2022211305

学号： 2022212408

姓名： 胡宇杭

学院： 计算机学院 (国家示范性软件学院)

时间： 2024 年 5 月 20 日

目录

1 实验内容和实验环境描述

1.1 实验任务内容和目的

- 利用所学数据链路层原理，自己设计一个滑动窗口协议，在仿真环境下编程实现有噪音信道环境下两站点之间无差错双工通信
- 实现有噪音信道环境下的无差错传输，充分利用传输信道的带宽
- 实现搭载 ACK 技术的 GBN 和选择重传协议
- 实现 ACK 计时器、捎带确认、NAK 等补充功能
- 根据信道实际情况合理地为协议配置工作参数，包括滑动窗口的大小和重传定时器时限以及 ACK 搭载定时器的时限
- 进一步巩固和深刻理解数据链路层误码检测的 CRC 校验技术，以及滑动窗口的工作机理。

1.2 实验环境

本次实验在 WINDOWS 11 下进行，使用 CLion 作为 IDE，gcc 作为编译工具，Ninja 作为生成器

- 系统版本：WINDOWS 11
- 编译器：gcc 8.1.0
- 生成器：Ninja
- IDE：CLion

2 软件设计

2.1 数据结构

2.1.1 帧的种类及结构

帧的结构如下所示：

帧的种类及结构

```
/* FRAME kind In datalink.h */
#define FRAME_DATA 1
#define FRAME_ACK  2
#define FRAME_NAK  3

/*
    DATA Frame
    +=========+========+========+===============+========+
    | KIND(1) | ACK(1) | SEQ(1) | DATA(240~256) | CRC(4) |
    +=========+========+========+===============+========+

    ACK Frame
    +=========+========+========+
    | KIND(1) | ACK(1) | CRC(4) |
    +=========+========+========+

    NAK Frame
    +=========+========+========+
    | KIND(1) | ACK(1) | CRC(4) |
    +=========+========+========+
*/

/* FRAME structure In datalink.c */
struct FRAME {
        unsigned char kind;                             // 帧的种类，一字节
        unsigned char ack;                      // ACK 帧的序号，一字节
        unsigned char seq;                              // 当前帧的序号，一字节
        unsigned char data[PKT_LEN];    // 数据，长度在240到256字节之间
        unsigned int padding;                   // CRC 校验位，四字节
};
```

帧的种类有 3 中，在 **datalink.h** 中定义，分别为 **DATA(1)**、**ACK(2)**、**NAK(3)**，相应的帧组成如上所示

其中，通过查阅指导书，发现网络层的包长固定为 256 字节，故一个数据帧的帧长固定为 $256+1+1+1+4=263$ 字节。其中，**kind**、**ack**、**seq** 均为 1 字节，**data** 为 $240-256$ 字节（实际固定为 256 字节），**CRC** 为 4 字节。

每个字段的含义如下所示（以数据帧为例）：

- **kind**：表示帧种类，有 **DATA**、**ACK**、**NAK** 三种
- **ack**：用于捎带确认，在数据帧中包含所要发送的 **ACK** 帧的数据，即期望帧的前一帧的序号
- **seq**：当前发送帧的序号
- **data**：网络层传来的包的数据
- **padding**：**CRC** 校验位，用于差错检测，在 **ACK**、**NAK** 中填充在 **ACK** 字段的后 4 个字节

2.1.2 使用的静态全局变量 SR

使用的静态全局变量如下：

使用的静态全局变量

```
1  static unsigned char nbuffered = 0;                              // 缓冲区数据包的数量
2  static int phl_ready = 0;                                        // 物理层是否准备好
3
4  /* sender config */
5  static unsigned char next_frame_to_send = 0;    // 下一次要发送的帧
6  static unsigned char sender_buffer[MAX_SEQ + 1][PKT_LEN];   // 发送端缓冲区
7  static unsigned char sender_window_head = 0;                              // 发送端窗口左端
8  static unsigned char sender_window_tail = NR_BUFS - 1;          // 发送端窗口右端
9
10 /* receriver config */
11 static unsigned char frame_expected = 0;                                          // 希望接收的帧
12 static unsigned char receiver_buffer[NR_BUFS][PKT_LEN];         // 接收端缓冲区
13 static unsigned char receiver_window_head = 0;                              // 接收端窗口左端
14 static unsigned char receiver_window_tail = NR_BUFS - 1;    // 接收端窗口右端
15 static int arrived[NR_BUFS];                                // visit 标记数组
16
17 static int no_nak = 1;                                          // 是否发送 NAK
```

对每个变量的说明如下：

- **nbuffered**：目前发送端窗口（缓冲区）内缓存的帧的数量
- **phl_ready**：物理层是否准备好，当且仅当物理层发送队列长度低于 50 字节时为 1
- **next_frame_to_send**：发送端下一帧要发送的帧序号
- **sender_buffer**：发送端缓冲区
- **sender_window_head & sender_window_tail**：显式定义的发送端窗口的头尾
- **frame_expected**：接收端期望接收到的帧序号
- **receiver_buffer**：接收端缓冲区
- **receiver_window_head & receiver_window_tail**：显式定义的接收端窗口的头尾
- **arrived**：标记数组，表示该索引位置是否有未传输到网络层的缓冲数据
- **no_nak**：接收方使用，表示当前期望帧是否发送过 **NAK**

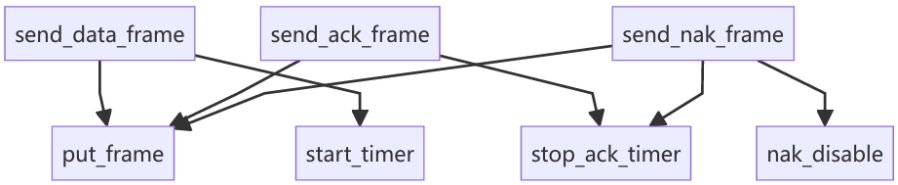
2.2 模块结构

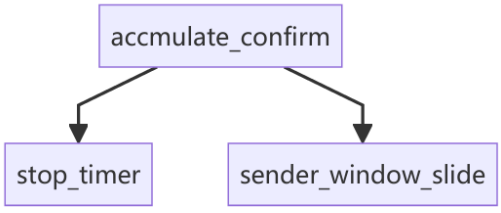
2.2.1 子程序说明

- **static int sender_window_check_inside(unsigned char frame_id)**：判断序号 **frame_id** 是否在发送端窗口内，返回值为 0 代表不在，反之则为在
- **static void sender_window_slide(int step)**：将发送方窗口左移或右移 **step** 位，**step** 的正负代表右或左移
- **static int nak_active(void)**：判断当前是否允许发送 **NAK**，返回值为 0 代表不能，反之则为能发送
- **static void nak_enable(void)**：将 **no_nak** 设置为 1，代表当前允许发送 **NAK**
- **static void nak_disable(void)**：将 **no_nak** 设置为 0，代表当前不允许发送 **NAK**
- **static void accmulate_confirm(unsigned char end_id)**：依次确认发送方窗口在 **end_id** 前的所有帧（不包括 **end_id** 帧）
- **static int receiver_window_check_inside(unsigned char frame_id)**：判断序号 **frame_id** 是否在接收方窗口内，返回值为 0 代表不在，反之则为在

- **static void receiver_window_slide(int step)**：将接收方窗口左移或右移 **step** 位，**step** 的正负代表右或左移
- **static void put_frame(unsigned char *frame, int len)**：**frame** 为指向传入帧的指针，**len** 为该帧的长度，该函数作用为为传入帧生成 **CRC** 校验并发送到物理层
- **static void send_data_frame(unsigned char frame_id)**：发送发送方缓冲区中索引为 **frame_id** 的帧，同时停止 **ACK** 计时器（因为数据帧中包含 ACK）, 并启动 **frame_id** 帧的计时器
- **static void send_ack_frame(unsigned char frame_id)**：发送序号为 **frame_id** 帧的 **ACK**，由于该函数仅在 **ACK** 计时器超时时调用，故不需要显式地停止 **ACK** 计时器
- **static void send_nak_frame(unsigned char frame_id)**：发送序号为 **frame_id** 帧的 **NAK**，同时令当前帧不再发送 **NAK** 并停止 **ACK** 计时器，因为程序中发送 **NAK** 仅会以当前期望帧作为 **frame_id**，因此也有 **ACK** 的作用

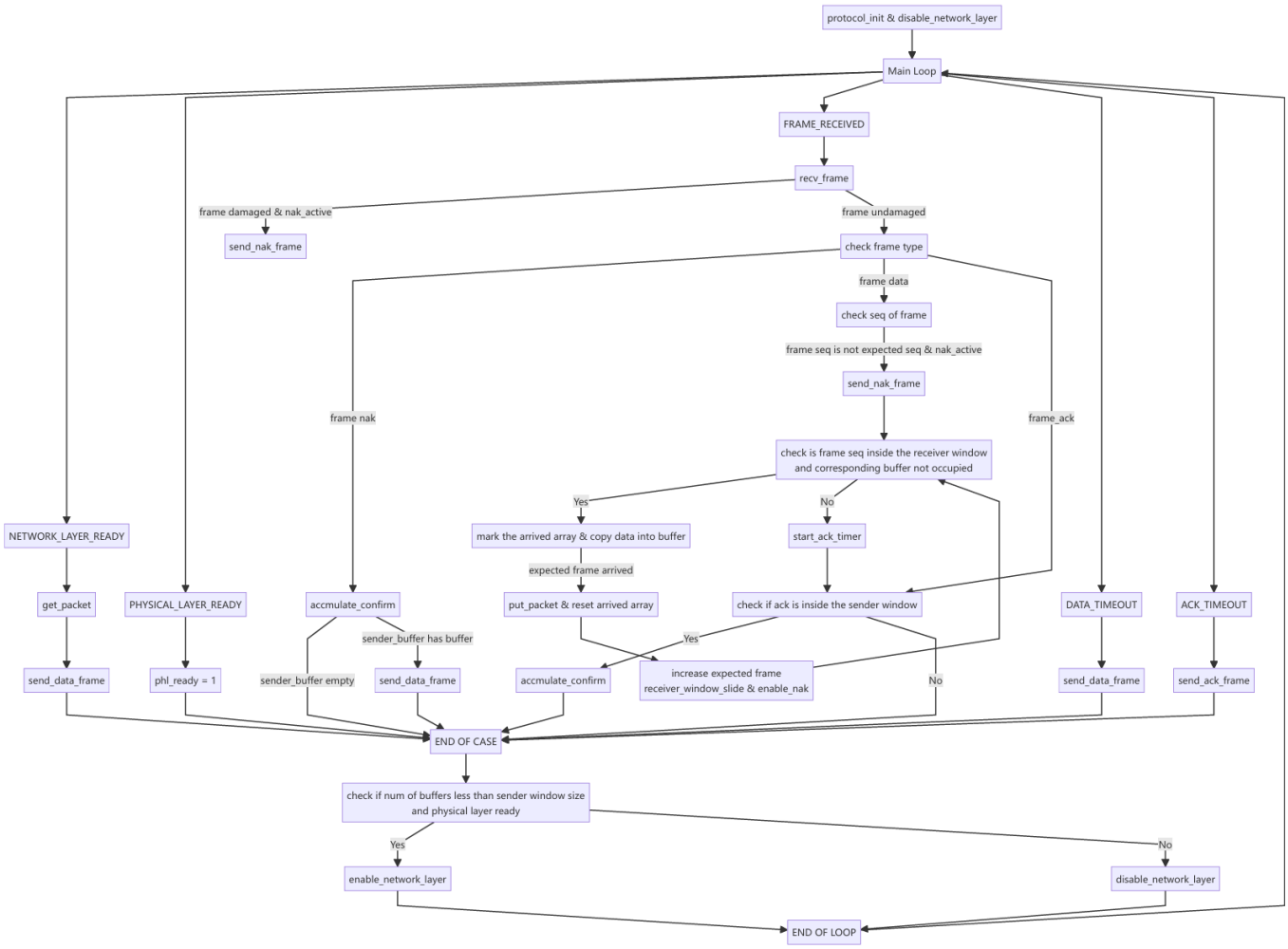
2.2.2 子程序调用图





2.3 算法流程

2.3.1 算法流程图



2.3.2 算法流程描述