# 

# 源程序清单

## GoBackN协议

Datalink.h//头文件

#define FRAME\_DATA 1

#define FRAME\_ACK 2

#define FRAME\_NAK 3

#define DATA\_TIMER 2000 //Data帧定时

#define MAX\_SEQ 7 //帧序号空间,此时为窗口0~7

#define inc(k) if(k<MAX\_SEQ) k=k+1; else k=0//模MAX\_SEQ加1

**typedef** enum**{**FALSE**,**TRUE**}**boolean**;**

**typedef** unsigned char seq\_nr**;**//帧序号或ack序号

**typedef** struct **{**

unsigned char data**[**PKT\_LEN**];**

**}**packet**;**//数据包结构

**typedef** struct FRAME **{**

unsigned char kind**;** /\* FRAME\_DATA ,ack,nak\*/

seq\_nr ack**;** //帧种类

seq\_nr seq**;** //本帧序号

packet info**;** //数据

unsigned int padding**;** //填充字段

**}**frame**;**

**typedef** int event\_type**;**

GoBack.cpp//cpp文件主程序

#include <stdio.h>

#include <string.h>

#include "protocol.h"

#include "datalink.h"

static int phl\_ready **=** 0**;**//物理层是否准备好

//发送窗口是否滑动的判别函数

//between( ack.expected, r.ack, next\_frame\_to\_send );

static boolean between**(**seq\_nr a**,** seq\_nr b**,** seq\_nr c**)**

**{**

**if** **(((**a **<=** b**)** **&&** **(**b **<** c**))** **||** **((**c **<** a**)** **&&** **(**a **<=** b**))** **||** **((**b **<** c**)** **&&** **(**c **<** a**)))**

**return** TRUE**;**

**else**

**return** FALSE**;**

**}**

static void put\_frame**(**unsigned char **\***frame**,** int len**)**

**{**

**\*(**unsigned int **\*)(**frame **+** len**)** **=** crc32**(**frame**,** len**);**

send\_frame**(**frame**,** len **+** 4**);**

phl\_ready **=** 0**;**//将数据发送到物理层后将物理层置为忙

**}**

static void send\_data\_frame**(**seq\_nr frame\_nr**,** seq\_nr frame\_expected**,** packet buffer**[])**

**{**//发送函数Send (next\_frame\_to\_send, frame\_expected, buffer[])

frame s**;**

s**.**kind **=** FRAME\_DATA**;**

s**.**info **=** buffer**[**frame\_nr**];**//插入数据包进帧中

s**.**seq **=** frame\_nr**;**//插入序号进帧

s**.**ack **=** **(**frame\_expected **+** MAX\_SEQ**)** **%** **(**MAX\_SEQ **+** 1**);**//捎带ack

dbg\_frame**(**"Send DATA %d %d, ID %d, Windows %d, PHL Queue %d\n"**,** s**.**seq**,** s**.**ack**,** **\*(**short **\*)**s**.**info**.**data**,** MAX\_SEQ**,** phl\_sq\_len**());**

//输出帧信息

put\_frame**((**unsigned char **\*)&**s**,** 3 **+** PKT\_LEN**);**

start\_timer**(**frame\_nr**,** DATA\_TIMER**);**

**}**

int main**(**int argc**,** char **\*\***argv**)**

**{**

seq\_nr next\_frame\_to\_send **=** 0**;**//发送窗口上限,初始期望到达的帧序号

seq\_nr ack\_expected **=** 0**;**//发送窗口下限,下一次期望到达的ack

seq\_nr frame\_expected **=** 0**;**//接受窗口

frame r**;**

packet buffer**[**MAX\_SEQ **+** 1**];**//数据缓存区

seq\_nr nbuffered **=** 0**;**//控制发送的数据帧<= 发送窗口最大值

//初始缓存区中无数据包

event\_type event**,**arg**;**//事件类型

int len **=** 0**;**//记录帧长

protocol\_init**(**argc**,** argv**);**//协议初始化

lprintf**(**"Designed by Pei Zixiang, build: " \_\_TIME\_\_ " " \_\_DATE\_\_" \n"**);**

disable\_network\_layer**();**

**while** **(**1**)**

**{**

event **=** wait\_for\_event**(&**arg**);**

**switch** **(**event**)**

**{**

**case** NETWORK\_LAYER\_READY**:**//网络层准备就绪,有待发送的分组

get\_packet**(**buffer**[**next\_frame\_to\_send**].**data**);**//从网络层获得新包

nbuffered**++;**

send\_data\_frame**(**next\_frame\_to\_send**,** frame\_expected**,** buffer**);**

inc**(**next\_frame\_to\_send**);**//发送窗口上限前移

**break;**

**case** PHYSICAL\_LAYER\_READY**:**//物理层准备就绪

phl\_ready **=** 1**;**//物理层状态置1

**break;**

**case** FRAME\_RECEIVED**:**//物理层接受到一帧

len **=** recv\_frame**((**unsigned char **\*)&**r**,** **sizeof(**r**));**

//计算接受到的帧长,从物理层获得一帧

**if** **(**len **<** 5 **||** crc32**((**unsigned char **\*)&**r**,** len**)** **!=** 0**)**

**{**//帧被破坏,或CRC32校检失败,丢弃坏帧

dbg\_event**(**"\*\*\*\* Receiver Error, Bad CRC Checksum\n"**);**

**break;**

**}**

dbg\_frame**(**"Recv DATA %d %d, ID %d\n"**,** r**.**seq**,** r**.**ack**,** **\*(**short **\*)**r**.**info**.**data**);**

**if** **(**r**.**seq **==** frame\_expected**)**

**{**//帧按序到达

put\_packet**(**r**.**info**.**data**,** len**-**7**);**

inc**(**frame\_expected**);**

**}**

**while** **(**between**(**ack\_expected**,** r**.**ack**,** next\_frame\_to\_send**))**

**{**

nbuffered**--;**

stop\_timer**(**ack\_expected**);**

inc**(**ack\_expected**);**

**}**

**break;**

**case** DATA\_TIMEOUT**:**

dbg\_event**(**"---- DATA %d timeout\n"**,** arg**);**

next\_frame\_to\_send **=** ack\_expected**;**

**for** **(**int i **=** 1**;** i **<=** nbuffered**;** i**++)**

**{**//重发错帧之后所有帧

send\_data\_frame**(**next\_frame\_to\_send**,** frame\_expected**,** buffer**);**

inc**(**next\_frame\_to\_send**);**

**}**

**break;**

**}**

**if** **(**nbuffered **<** MAX\_SEQ **&&** phl\_ready**)**//物理层ready且缓存未满

enable\_network\_layer**();**

**else**

disable\_network\_layer**();**

**}**

**}**

## Selective协议

Datalink.h//头文件

/\* FRAME kind \*/

#define FRAME\_DATA 1

#define FRAME\_ACK 2

#define FRAME\_NAK 3

Selective.cpp//cpp文件主程序

#include <stdio.h>

#include <string.h>

#include<stdbool.h>

//改了c++常规调试信息格式最后一个

#include "protocol.h"

#include "datalink.h"

#define MAX\_SEQ 31

#define NR\_BUFS ((MAX\_SEQ+1)/2)

#define inc(k) if (k < MAX\_SEQ) k++; else k = 0

**typedef** unsigned char seq\_nr**;**

**typedef** struct **{** unsigned char data**[**PKT\_LEN**];** **}**packet**;**

bool no\_nak **=** true**;**

seq\_nr oldest\_frame **=** MAX\_SEQ **+** 1**;**

seq\_nr frame\_expected**;**

**typedef** struct **{**

unsigned char kind**;** // FRAME\_DATA

seq\_nr seq**;**

seq\_nr ack**;**

packet info**;**

unsigned int padding**;**//Stack of s was currupted调试的时候出错，加上这个就没错了

**}**frame**;**

#define DATA\_TIMER 3000

#define ACK\_TIMER 1000

static int phl\_ready **=** 0**;**

static void put\_frame**(**unsigned char **\***frame**,** int len**)**

**{**

**\*(**unsigned int **\*)(**frame **+** len**)** **=** crc32**(**frame**,** len**);**

send\_frame**(**frame**,** len **+** 4**);**

phl\_ready **=** 0**;**

**}**

static bool between**(**seq\_nr a**,** seq\_nr b**,** seq\_nr c**)**

**{**

**return((**a **<=** b**)** **&&** **(**b **<** c**))** **||** **((**c **<** a**)** **&&** **(**a **<=** b**))** **||** **((**b **<** c**)** **&&** **(**c **<** a**));**

**}**

void send\_data**(**unsigned char fk**,** seq\_nr frame\_nr**,** seq\_nr frame\_expected**,** packet buffer**[])**

**{**

frame s**;**

s**.**kind **=** fk**;**

s**.**seq **=** frame\_nr**;**

s**.**ack **=** **(**frame\_expected **+** MAX\_SEQ**)** **%** **(**MAX\_SEQ **+** 1**);**

**if** **(**fk **==** FRAME\_DATA**)** **{**

//s.info = buffer[frame\_nr % NR\_BUFS];//??

memcpy**(**s**.**info**.**data**,** **&**buffer**[**frame\_nr **%** NR\_BUFS**].**data**,** PKT\_LEN**);**

dbg\_frame**(**"Send DATA %d %d, ID %d\n"**,** s**.**seq**,** s**.**ack**,** **\*(**short **\*)**s**.**info**.**data**);**

put\_frame**((**unsigned char **\*)&**s**,** 3 **+** PKT\_LEN**);**

start\_timer**(**frame\_nr**%**NR\_BUFS**,** DATA\_TIMER**);**

**}**

**if** **(**fk **==** FRAME\_NAK**)** **{**

no\_nak **=** false**;**

dbg\_frame**(**"Send NAK %d\n"**,** s**.**ack**);**

put\_frame**((**unsigned char **\*)&**s**,** 3 **+** PKT\_LEN**);**

**}**

**if** **(**fk **==** FRAME\_ACK**)**

**{**

dbg\_frame**(**"Send ACK %d\n"**,** s**.**ack**);**

put\_frame**((**unsigned char **\*)&**s**,** 3 **+** PKT\_LEN**);**

**}**

phl\_ready **=** 0**;**

stop\_ack\_timer**();**

**}**

int main**(**int argc**,** char **\*\***argv**)**

**{**

int arg**,** len **=** 0**;**

protocol\_init**(**argc**,** argv**);**

lprintf**(**"Designed by Hu Xiaoyan, build: " \_\_DATE\_\_" "\_\_TIME\_\_"\n"**);**

seq\_nr ack\_expected**;**//发送窗口下界

seq\_nr next\_frame\_to\_send**;**//发送窗口上界

seq\_nr frame\_expected**;**//接受窗口下界

seq\_nr too\_far**;**//接受窗口上界

int i**;**

frame r**;**

packet out\_buf**[**NR\_BUFS**];**

packet in\_buf**[**NR\_BUFS**];**

bool arrived**[**NR\_BUFS**];**

seq\_nr nbuffered**;**

int event**;**

ack\_expected **=** 0**;**

next\_frame\_to\_send **=** 0**;**

frame\_expected **=** 0**;**

too\_far **=** NR\_BUFS**;**

nbuffered **=** 0**;**

**for** **(**i **=** 0**;** i **<** NR\_BUFS**;** i**++)** arrived**[**i**]** **=** false**;**

enable\_network\_layer**();**

**while** **(**true**)**

**{**

event **=** wait\_for\_event**(&**arg**);**

**switch** **(**event**)**

**{**

**case** NETWORK\_LAYER\_READY**:**//允许网络层发送数据帧

nbuffered**++;**//缓冲区数据帧数加一

get\_packet**(**out\_buf**[**next\_frame\_to\_send**%**NR\_BUFS**].**data**);**//从网络层接收数据

send\_data**(**FRAME\_DATA**,** next\_frame\_to\_send**,** frame\_expected**,** out\_buf**);**//发送数据帧

inc**(**next\_frame\_to\_send**);**//发送窗口上界下移

dbg\_frame**(**"Send DATA %d %d, ID %d\n"**,** r**.**seq**,** r**.**ack**,** r**.**info**);**

**break;**

**case** PHYSICAL\_LAYER\_READY**:**//物理层空闲

phl\_ready **=** 1**;**

**break;**

**case** FRAME\_RECEIVED**:**//接受方收到一个帧

len **=** recv\_frame**((**unsigned char **\*)&**r**,** **sizeof** r**);**//帧长

**if** **(**len **<** 5 **||** crc32**((**unsigned char **\*)&**r**,** len**)** **!=** 0**)** **{**

dbg\_event**(**"\*\*\*\* Receiver Error, Bad CRC Checksum\n"**);**

**if** **(**no\_nak**)** send\_data**(**FRAME\_NAK**,** 0**,** frame\_expected**,** out\_buf**);**

**break;**

**}**

**if** **(**r**.**kind **==** FRAME\_DATA**)**

**{**

dbg\_frame**(**"Recv DATA %d %d, ID %d\n"**,** r**.**seq**,** r**.**ack**,** r**.**info**);**

**if** **((**r**.**seq **!=** frame\_expected**)** **&&** no\_nak**)**//未按顺序到达

send\_data**(**FRAME\_NAK**,** 0**,** frame\_expected**,** out\_buf**);**//返回nak

**else** start\_ack\_timer**(**ACK\_TIMER**);**//开启ack定时器

**if** **(**between**(**frame\_expected**,** r**.**seq**,** too\_far**)** **&&** **(**arrived**[**r**.**seq**%**NR\_BUFS**]** **==** false**))** **{**

arrived**[**r**.**seq**%** NR\_BUFS**]** **=** true**;**

in\_buf**[**r**.**seq**%** NR\_BUFS**]** **=** r**.**info**;**

**while** **(**arrived**[**frame\_expected**%**NR\_BUFS**])** **{**//如果到达的帧落在接收窗上

put\_packet**(**in\_buf**[**frame\_expected**%**NR\_BUFS**].**data**,** len **-** 7**);**//将到达的数据帧发送到网络层

no\_nak **=** true**;**//表示对于这一帧没有发送过否定确认

arrived**[**frame\_expected**%**NR\_BUFS**]** **=** false**;**//该处缓冲空间重新记为空

inc**(**frame\_expected**);**//接收窗口后移一位

inc**(**too\_far**);**

start\_ack\_timer**(**ACK\_TIMER**);**

**}**

**}**

**}**

**if** **((**r**.**kind **==** FRAME\_NAK**)** **&&** between**(**ack\_expected**,** **(**r**.**ack **+** 1**)** **%** **(**MAX\_SEQ **+** 1**),** next\_frame\_to\_send**))** **{**

send\_data**(**FRAME\_DATA**,** **(**r**.**ack **+** 1**)** **%** **(**MAX\_SEQ **+** 1**),** frame\_expected**,** out\_buf**);**

//dbg\_frame("Recv NAK %d\n", r.ack);

**}**

**while** **(**between**(**ack\_expected**,** r**.**ack**,** next\_frame\_to\_send**))** **{**

nbuffered**--;**//缓冲区数据帧数量减一

stop\_timer**(**ack\_expected**%**NR\_BUFS**);**//停止ack计时器

inc**(**ack\_expected**);**

**}**

**break;**

**case** DATA\_TIMEOUT**:**//数据帧超时。重发

dbg\_event**(**"---- DATA %d timeout\n"**,** arg**);**

send\_data**(**FRAME\_DATA**,** ack\_expected**,** frame\_expected**,** out\_buf**);**

**break;**

**case** ACK\_TIMEOUT**:**//确认帧超时，发送确认帧

dbg\_event**(**"---- ACK %d timeout\n"**,** arg**);**

send\_data**(**FRAME\_ACK**,** 0**,** frame\_expected**,** out\_buf**);**

**break;**

**}**

**if** **(**nbuffered **<** NR\_BUFS **&&** phl\_ready**)** enable\_network\_layer**();**//接收方缓冲区未满且物理层空闲，则开启网络层

**else** disable\_network\_layer**();**//否则关闭网络层

**}**

**}**