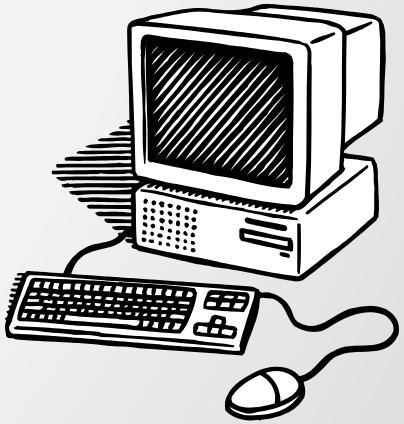
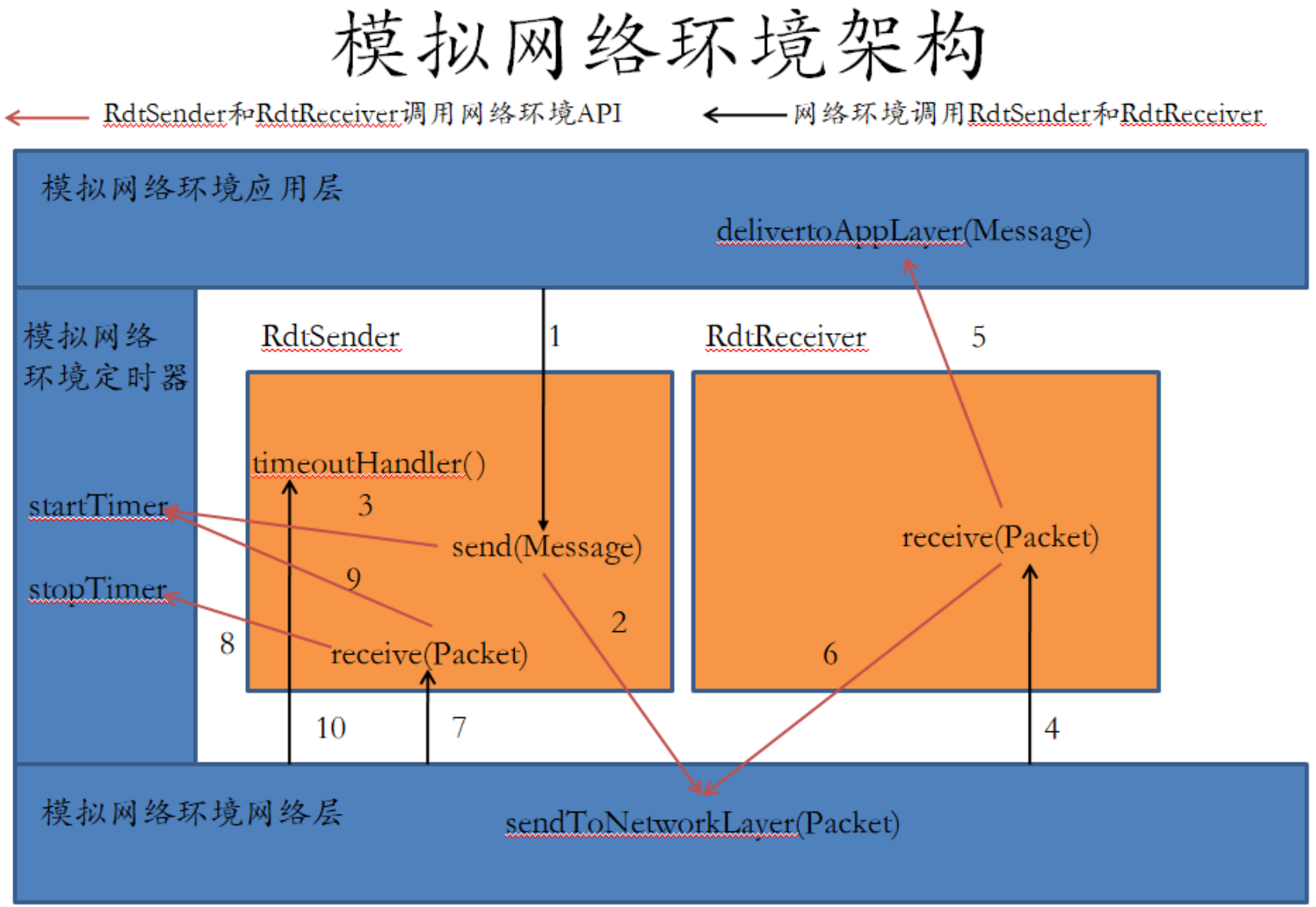
可靠传输协议设计

作者:Sukuna



模拟网络环境的介绍


模拟网络环境架构
RdtSender和RdtReceiver调用网络环境API
网络环境调用RdtSender和RdtReceiver
模拟网络环境应用层
delivertoAppLayer(Message)
模拟网络
环境定时器
RdtSender
RdtReceiver
1
5
timeoutHandler()
3
startTimer
send(Message)
receive(Packet)
9
stopTimer
2
8
receive(Packet)
6
10
7
4
模拟网络环境网络层
sendToNetworkLayer(Packet)

模拟网络环境库的装载(VS2019版本)

- 编译的链接部分非常重要,netsimlib.lib是模拟网络环境库,没有这个库我们不能完成下面的实验.
- 在VS2019中,我们只需要在stdAfx.h头文件中添加这个编译选项即可:
  - #pragma comment(lib,"您的路径\\netsimlib.lib")

```
#pragma once

#include "targetver.h"

#include <stdio.h>
#include <tchar.h>

#pragma comment (lib,"C:\\GBN\\netsimlib.lib")

#include <iostream>
using namespace std;
```

模拟网络环境库的装载(Linux_Cmake)

- 使用Cmake我们也可以方便地进行链接.注意:使用Cmake时需要将netsimlib.lib编译成静态的库netsimlib.a
- 下面对Cmake代码进行解析:
  - {PROJECT_SOURCE_DIR}可以认为是一个环境变量,当运行cmake的时候,这个值就是当前命令行挂载的文件系统位置(说人话就是执行cmake的Project根地址)
  - INCLUDE_DIERCTORIES添加了include的寻址位置,当你使用include的时候,不仅仅会在系统目录寻找头文件,还会在这里寻找.
  - FIND_LIBRARY找到一个库,并把这个库起一个别名叫NETSIM_LIB.

```
cmake_minimum_required(VERSION 3.5)
PROJECT(stop_wait)

SET(CMAKE_C_COMPTLER GCC)
add_definitions(-std=c++11)
INCLUDE_DIRECTORIES(${PROJECT_SOURCE_DIR}/include)
aux_source_directory(${PROJECT_SOURCE_DIR}/src SRC_LIST)
SET(EXECUTABLE_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/bin)
ADD_EXECUTABLE(stop_wait ${SRC_LIST})
FIND_LIBRARY(NETSIM_LIB libnetsim.a ${PROJECT_SOURCE_DIR}/lib)
TARGET_LINK_LIBRARIES(stop_wait  ${NETSIM_LIB})
```

提交代码需要提供的文件

- 在学生代码工程里，需要包含以下头文件：
  - DataStructure.h
  - Global.h
  - NetworkService.h
  - RandomEventEnum.h
  - RdtReceiver.h
  - RdtSender.h
  - Tool.h
- 另外还需要如下常规的头文件：
  - stdio.h
  - string.h
  - iostream

数据结构定义

- 第四层报文段:Packet
  - checksum 校验和
  - seqnum 序号
  - acknum ACK序号
  - payload 负载,就是装载的实际内容
  - Packet() 基本构造函数:使用new Packet() 隐式调用Packet().
  - Packet &operator=(const Packet &pkt);运算符重载,即Packet a = b.即调用operator=(b);返回值为a.

数据结构定义

- 第五层应用层报文:Message
  - data:应用层数据.
  - Message()和Message& operator=(const Message &msg);和上面的意思一样.

模拟网络环境API接口

- void startTimer(RandomEventTarget target, int timeOut,int seqNum);
  - RandomEventTarget 是定义的枚举类型，用来标识发送方和接收方。
  - timeOut是超时时间.
  - seqNum是对应的包序号.
  - 功能:启动定时器
- void stopTimer(RandomEventTarget target,int seqNum)
  - RandomEventTarget 是定义的枚举类型，用来标识发送方和接收方.
  - seqNum是对应的包序号.
  - 功能:停止定时器
  - 定时器:一个packet可以对应一个定时器
  - 重新启动一个定时器前，一定要先关闭该定时器（要注意seqNum 参数的一致性），否则模拟网络环境会提示“试图启动一个已启动的定时器”。

模拟网络环境API接口

- void sendToNetworkLayer(RandomEventTarget target, Packet pkt)
  - RandomEventTarget 是定义的枚举类型，用来标识发送方和接收方。
  - pkt是要传输的Packet.
  - 功能:将数据包发送到网络层，由RdtSender或RdtReceiver调用.
- void delivertoAppLayer(RandomEventTarget target, Message msg)
  - RandomEventTarget 是定义的枚举类型，用来标识发送方和接收方。
  - msg就是要传递的Message
  - 功能:将数据包向上递交到应用层，由RdtReceiver调用

模拟网络环境API接口

- virtual void init() = 0;
  - 初始化网络环境，在main里调用
- virtual void start() = 0;
  - 启动网络环境，在main里调用
- virtual void setRtdSender(RdtSender *ps) = 0;
  - 注入具体的发送方对象，在main里调用
- virtual void setRtdReceiver(RdtReceiver *ps) = 0;
  - 设置具体的接收对象，在main里调用
- virtual void setInputFile(const char *ifile) = 0;
  - 设置输入文件路径,就是传向Sender的输入文件.Sender从inputfile读文件.
- virtual void setOutputFile(const char *ofile) = 0;
  - 设置输出文件路径,就是Reciever传出来的输出文件.Reciever向outputfile写文件.

模拟网络环境API接口

- setRunMode函数也是增加的接口函数，用于设置网络模拟环境的运行模式。如果设置 mode=0（也是该函数的缺省参数），则为 Verbose 模式，网络模拟环境会输出很多模拟环境的运行信息，可以帮助学生观察协议的工作过程，特别是协议实现有问题时，可以帮助分析协议出现的问题；如果设置为 mode=1，则为 Silence 模式，这是会关闭掉模拟环境输出的运行信息，而控制台只会输出学生协议实现代码里打印的信息。

学生需要完成的API

- 分成三类:
- GBN
- SR
- 简化TCP
- 学生需要完成这三类的发送方和接收方共5个函数.

Sender

- 发送方
  - 使用方式:定义一个类例如GBNSender,要继承于RdtSender,并完成抽象函数.
- bool send(Message &message) = 0;
  - 发送应用层下来的Message，由NetworkService调用。
  - 如果发送方成功地将Message发送到网络层，返回true;
  - 如果因为发送方处于等待确认状态或发送窗口已满而拒绝发送Message，则返回false
- void receive(Packet &ackPkt);
  - 接受确认Ack，将被NetworkService调用
- void timeoutHandler(int seqNum) = 0;
  - Timeout handler，将被NetworkService调用
- bool getWaitingState()
  - 返回RdtSender是否处于等待状态，如果发送方正等待确认或者发送窗口已满，返回true

Reciever

- void receive(Packet &packet);
  - 接收报文，将被NetworkService调用

检查方式

- 检查的时候将会分为两部分:
- 运行检查脚本.
- 回答三个可靠传输协议的设计原理问题.

检查脚本怎么运行?(Step1.修改main函数)

- 1、修改setRunMode为1
- 2、修改InputFile和Outpitfile为存储位置.

```
#include "stdafx.h"
#include "Global.h"
#include "RdtSender.h"
#include "RdtReceiver.h"
#include "StopWaitRdtSender.h"
#include "StopWaitRdtReceiver.h"


int main(int argc, char* argv[])
{
    RdtSender *ps = new GBNSender();
    RdtReceiver * pr = new GBNReceiver();
//  pns->setRunMode(0);
    pns->setRunMode(1);
    pns->init();
    pns->setRtdSender(ps);
    pns->setRtdReceiver(pr);
    pns->setInputFile("C:\\GBN\\input.txt");
    pns->setOutputFile("C:\\GBN\\output.txt");

    pns->start();

    delete ps;
    delete pr;
    delete pUtils;
    delete pns;
    return 0;
}
```

Step2.修改bat脚本(Windows)

- 修改appname为exe文件的地址,inputname为第一步input文件的地址,outputname为第一步output文件的地址.要不然脚本会找不到文件.其他地方不可以更改!

```
1   @echo off
2
3   set appname="StopWait.exe"
4   set inputname="input.txt"
5   set outputname="output.txt"
6   set resultname="result.txt"
7
8   for /l %%i in (1,1,10) do (
9       echo Test %appname% %%i:
10      %appname% > %resultname% 2>&1
11      fc /N %inputname% %outputname%
12  )
13  pause
14
15
```

Step2.修改Shell脚本(Linux)

- 修改appname为Linux可执行文件的地址,inputname为第一步input文件的地址,outputname为第一步output文件的地址.要不然脚本会找不到文件.其他地方不可以更改!

```
#!/bin/bash
# appname 程序名称
# inputname 输入文件名
# outputname 输出文件名
# resultname 程序控制台输出结果重定向文件名

appname='stop_wait'
inputname='input.txt'
outputname='output.txt'
resultname='result.txt'

for ((i=1;i<=10;i++))
do
echo Test $appname $i
./$appname > $resultname 2>&1
cmp $inputname $outputname
echo Test $i over
done

```