南京大学本科生实验报告

课程名称：计算机网络　　任课教师：李文中　　助教：

学院	计算机科学与技术系	专业（方向）	计算机科学与技术
学号	**221220143**	姓名	陈雨檬
Email	**221220143@smail.nju.edu.cn**	开始/完成日期	**2024.5.20/2024.5.27**

1. 实验名称

Lab6:Reliable Communication

2. 实验目的

通过简单的模拟网络结构模拟出可靠通信的范式内容。

3. 实验内容

Task1：Preparation

Task2：Middlebox

How: 接收来自 blaster 的数据包，通过 random.uniform(0,1) 生成一个小数，如果结果小于规定的参数 dropRate 则认为该包丢失，否则将包转发给 blastee；接收来自 blastee 的 ack 包，将其转发给 blaster。

Task3：Blastee

How: 接收 Middlebox 转发而来的 blaster 的数据包，回复具有其相同 sequencenumber 和 8 字节相同内容的 ack 包。 如果发现 num 以内的 sequencenumber 序号均已回复过则关闭自己。

Task4：Blaster

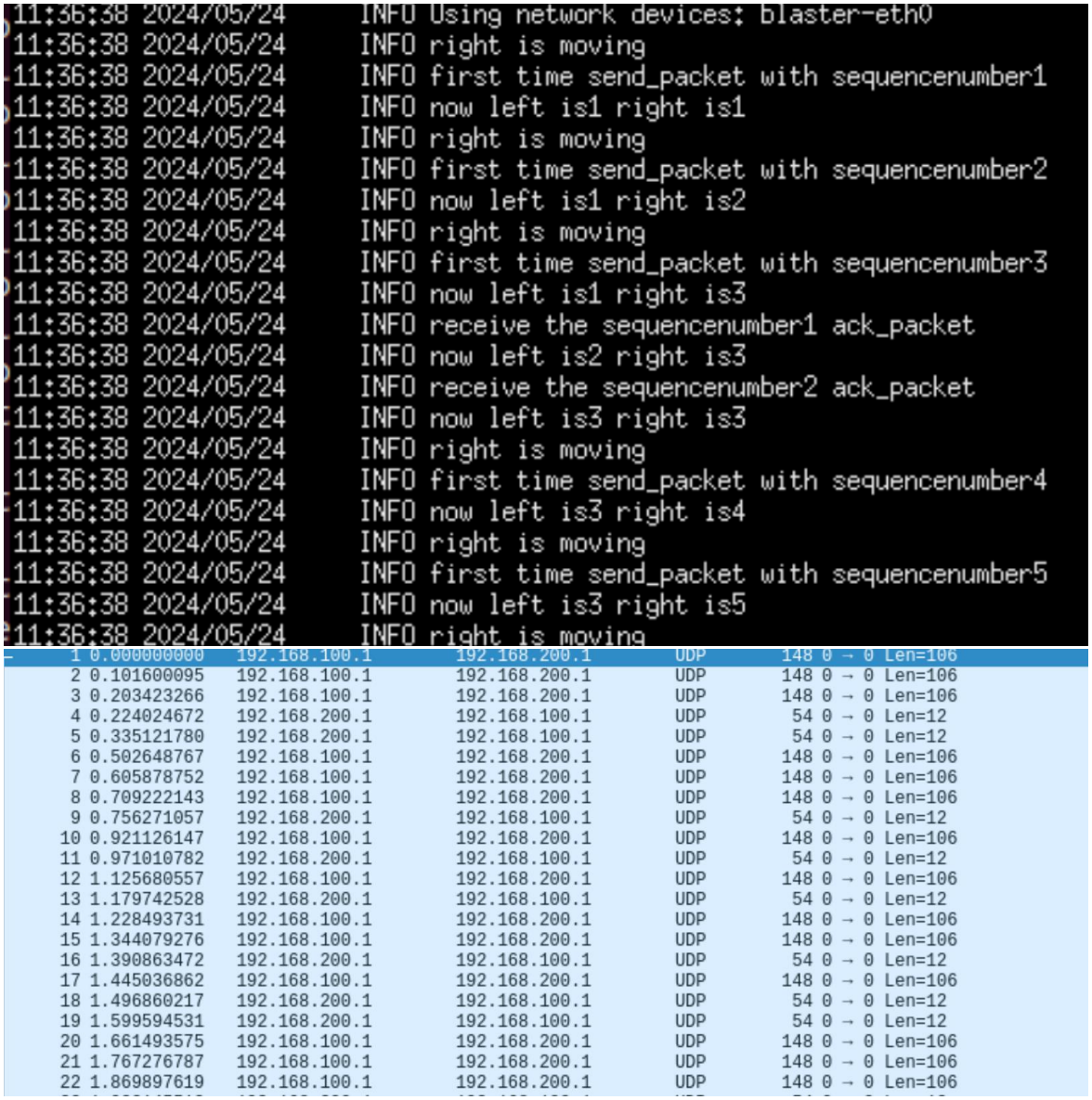
How: 向 blastee 发送数据包，通过维护一个窗口来确保通信是可靠的，只有收到窗户第一个数据包对应的 ack 窗口左指针才会移动并且更新时间戳；如果一直没有收到则认为产生了丢包，进行重发；如果重发队列不为空则说明还有包需要重发，进行重发；否则将当前窗口队列仍未收到 ack 的包加入重发队列进行重发，每次只重发一个包。

Task5：Run the code

How: 通过传入不同的参数，进行部署测试和抓包，确保任何情况下各端口都能成功运行

4. 实验结果

进行部署测试首先测试默认参数情况：



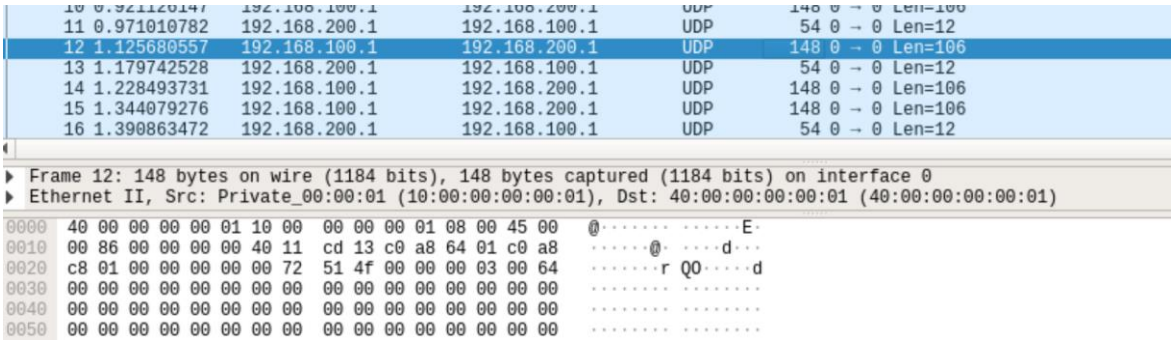
（见 lab_6_1_blaster.pcapng）

长度为 106 的即为 blaster 发出的数据包（4+2+100），长度为 12（4+8）的即为接收到的 ack 数据包。前一图中 xterm 的打印结果也显示出该过程。

```
11:36:38 2024/05/24     INFO transmit the sequencenumber1 packet to blastee
11:36:38 2024/05/24     INFO transmit the sequencenumber2 packet to blastee
11:36:38 2024/05/24     INFO transmit the sequencenumber1 ack_packet to blaster
11:36:38 2024/05/24     INFO Drop the sequencenumber3 packet
11:36:38 2024/05/24     INFO transmit the sequencenumber2 ack_packet to blaster
11:36:38 2024/05/24     INFO transmit the sequencenumber4 packet to blastee
11:36:38 2024/05/24     INFO Drop the sequencenumber5 packet
11:36:38 2024/05/24     INFO transmit the sequencenumber4 ack_packet to blaster
11:36:38 2024/05/24     INFO transmit the sequencenumber6 packet to blastee
11:36:39 2024/05/24     INFO transmit the sequencenumber6 ack_packet to blaster
11:36:39 2024/05/24     INFO transmit the sequencenumber7 packet to blastee
11:36:39 2024/05/24     INFO transmit the sequencenumber7 ack_packet to blaster
```

该 middlebox 中的打印结果显示出第 3 个和第 5 个序列号的包均被认为丢弃。

因此



（见 lab_6_1_blaster.pcapng）

表示 blaster 对没有收到 ack 的队列中的数据包进行了重发

```
11:37:00 2024/05/24     INFO totaltime22.0231 throughput794.6213 goodput454.0693
11:37:00 2024/05/24     INFO retrantimes75 timeouttimes27
11:37:00 2024/05/24     INFO allbytes17500 allbytesonce10000
11:37:00 2024/05/24     INFO all down!
```

该图显示出发完 num 个 packet 并收到 ack 后得到的该次部署实验的结果数据。

再测试丢包率为 1 的特殊情况

1	0.000000000	192.168.100.1	192.168.200.1	UDP	148 0 → 0 Len=106
2	0.105064867	192.168.100.1	192.168.200.1	UDP	148 0 → 0 Len=106
3	0.207149908	192.168.100.1	192.168.200.1	UDP	148 0 → 0 Len=106
4	0.308311092	192.168.100.1	192.168.200.1	UDP	148 0 → 0 Len=106
5	0.411313714	192.168.100.1	192.168.200.1	UDP	148 0 → 0 Len=106
6	0.633494930	192.168.100.1	192.168.200.1	UDP	148 0 → 0 Len=106
7	0.734761981	192.168.100.1	192.168.200.1	UDP	148 0 → 0 Len=106
8	0.841109834	192.168.100.1	192.168.200.1	UDP	148 0 → 0 Len=106
9	0.944571999	192.168.100.1	192.168.200.1	UDP	148 0 → 0 Len=106
10	1.046750568	192.168.100.1	192.168.200.1	UDP	148 0 → 0 Len=106
11	1.250908337	192.168.100.1	192.168.200.1	UDP	148 0 → 0 Len=106
12	1.355429406	192.168.100.1	192.168.200.1	UDP	148 0 → 0 Len=106
13	1.457757614	192.168.100.1	192.168.200.1	UDP	148 0 → 0 Len=106
14	1.558960031	192.168.100.1	192.168.200.1	UDP	148 0 → 0 Len=106
15	1.660571039	192.168.100.1	192.168.200.1	UDP	148 0 → 0 Len=106
16	1.872677502	192.168.100.1	192.168.200.1	UDP	148 0 → 0 Len=106
17	1.974886127	192.168.100.1	192.168.200.1	UDP	148 0 → 0 Len=106
18	2.076030211	192.168.100.1	192.168.200.1	UDP	148 0 → 0 Len=106
19	2.177089889	192.168.100.1	192.168.200.1	UDP	148 0 → 0 Len=106
20	2.284047162	192.168.100.1	192.168.200.1	UDP	148 0 → 0 Len=106
21	2.499783814	192.168.100.1	192.168.200.1	UDP	148 0 → 0 Len=106
22	2.601171867	192.168.100.1	192.168.200.1	UDP	148 0 → 0 Len=106
23	2.701870319	192.168.100.1	192.168.200.1	UDP	148 0 → 0 Len=106
24	2.802534427	192.168.100.1	192.168.200.1	UDP	148 0 → 0 Len=106

见（lab_6_2_blaster.pcapng）

看到 blaster 始终在重传序号 1 到 5 的数据包；再测试丢包率为 0

的情况

```
11:47:30 2024/05/24     INFO totaltime13.3745 throughput747.6924 goodput747.6924
11:47:30 2024/05/24     INFO retrantimes0 timeouttimes0
11:47:30 2024/05/24     INFO allbytes10000 allbytesonce10000
```

如图所有包都只发送了一次，没有重发

5. 核心代码

Middlebox 核心代码

```
if fromIface == "middlebox-eth0":
    log_debug("Received from blaster")
    '''
    Received data packet
    Should I drop it?
    If not, modify headers & send to blastee
    '''
    p=random.uniform(0,1)
    if(p<=self.dropRate):
        log_info('Drop the sequencenumber{} packet'.format(sequencenumber))
        return
    port=self.net.interface_by_name("middlebox-eth1")
    packet[0].src=port.ethaddr
    packet[0].dst=EthAddr('20:00:00:00:00:01')
    log_info('transmit the sequencenumber{} packet to blastee'.format(sequencenumber))
    self.net.send_packet("middlebox-eth1", packet)
```

将从 **blaster** 接收到的包转发到 **blastee**

```
elif fromIface == "middlebox-eth1":
    log_debug("Received from blastee")
    '''
    Received ACK
    Modify headers & send to blaster. Not dropping ACK packets!
    net.send_packet("middlebox-eth0", pkt)
    '''
    port=self.net.interface_by_name("middlebox-eth0")
    packet[0].src=port.ethaddr
    packet[0].dst=EthAddr('10:00:00:00:00:01')
    log_info('transmit the sequencenumber{} ack_packet to blaster'.format(sequencenumber))
    self.net.send_packet("middlebox-eth0", packet)
```

将从 **blastee** 接收到的 **ack** 包转发到 **blaster**

Blastee 核心代码

```
rawbytes=packet[3].to_bytes()
rawsequencenumber=rawbytes[0:4]
sequencenumber = struct.unpack('!I', rawsequencenumber)[0]
contents=rawbytes[6:]
contents=contents[:8]
ack=Ethernet() + IPv4(protocol = IPProtocol.UDP) + UDP()
ack[0].src=EthAddr("20:00:00:00:00:01")
ack[0].dst=EthAddr("40:00:00:00:00:02")
ack[1].ttl=64
ack[1].src=IPv4Address("192.168.200.1")
ack[1].dst=self.blasterIp
ack+=RawPacketContents(rawsequencenumber)
ack+=RawPacketContents(contents)
log_info('receive and reply the sequencenumber{} packet'.format(sequencenumber))
self.net.send_packet("blastee-eth0", ack)
flag=False
self.dic[str(sequencenumber)]=True
for i in range(1,self.num+1):
    if self.dic[str(i)]==False:
        flag=True
if not flag:
    self.shutdown()
```

收到的数据包内容填写 **ack** 包的内容并发送回 **blaster**，**如果已经收到序列号为 num 的包则关闭自己。**

Blaster 核心代码

```
starttime=0# first packet time
totaltime=0# first to end time = endtime-starttime
retrantimes=0# retransmit times
timeouttimes=0# timeout times
allbytes=0# all packte bytes
allbytesonce=0# packte bytes without retransmit
throughput=0# allbytes / totaltime
goodput=0# allbytesonce / totaltime
```

首先定义全局变量如上

在 **blaster** 中定义一个 **self.queue**， 其元素是自定义元素

Item，定义如下

```
class Item:
    def __init__(self,pkt,sequencenumber,time):
        self.pkt=pkt
        self.sequencenumber=sequencenumber
        self.acked=False
```

在 **blaster** 中初始化自定义变量如下

```
self.net = net
# TODO: store the parameters
self.blasteeIp=IPv4Address(blasteeIp)
self.num=int(num)
if self.num<=0:
    log_info("error parameter 'num'")
    return
self.length=int(length)
if not (self.length>=0 and self.length<=65535):
    log_info("error parameter 'length'")
    return
self.senderWindow=int(senderWindow)
self.timeout=float(int(timeout)/1000)
self.recvTimeout=float(int(recvTimeout)/1000)
self.queue=[]
self.timestamp=0
self.retranqueue=[]
self.left=1
self.right=0
```

再进行收到 **ack** 包的处理

```
rawbytes  = packet[3].to_bytes()
rawsequencenumber = rawbytes[0:4]
sequencenumber = struct.unpack('!I', rawsequencenumber)[0]
```

首先获取 **ack** 包中的序列号

```
if self.left>self.right:
    log_info("the ack_packet is repetited")
    return
if sequencenumber==self.queue[0].sequencenumber:
    self.queue[0].acked=True
    i=0
    while i<len(self.queue) and self.queue[i].acked:
        i+=1
        self.left+=1
        self.timestamp=time.time()
        self.retranqueue=[]
    if i<len(self.queue):
        self.queue=self.queue[i:]
    else:
        self.queue=[]
else:
    for item in self.queue:
        if item.sequencenumber==sequencenumber:
            item.acked=True
            break
log_info('now left is{} right is{}'.format(self.left,self.right))
```

再判断此时队列是否为空，为空则表示是重复的 **ack** 包，直接返回；否则为队列进行更新，如果移动了 **left** 则更新时间戳。

```
if self.left==self.num+1:
    totaltime=time.time()-starttime
    throughput=allbytes/totaltime
    goodput=allbytesonce/totaltime
    log_info("totaltime{:.4f} throughput{:.4f} goodput{:.4f}".format(totaltime,throughput,goodput))
    log_info("retrantimes{} timeouttimes{}".format(retrantimes,timeouttimes))
    log_info("allbytes{} allbytesonce{}".format(allbytes,allbytesonce))
    log_info("all down!")
    self.shutdown()
    return
```

再判断是否已经结束了，如果结束则打印各种信息。

然后进行发包函数的处理 。

首先判断是否已经结束，如果没有则继续。

再判断是否存在还没有重发的包；如果存在则说明重发还没有结束，继续重发。

```
if len(self.retranqueue)!=0:
    log_info('going to retransmit')
    log_info('retransmit the sequencenumber{} packt'.format(self.retranqueue[0].sequencenumber))
    self.net.send_packet('blaster-eth0',self.retranqueue[0].pkt)
    retrantimes+=1
    allbytes+=self.length
    self.retranqueue.remove(self.retranqueue[0])
    return
```

再判断此时是否到达最大窗口大小，如果没有则创建包并进行发送 。

```
if self.right<self.num and self.right-self.left+1<self.senderWindow:
    # Creating the headers for the packet
    pkt = Ethernet() + IPv4() + UDP()
    pkt[1].protocol = IPProtocol.UDP
    pkt[1].src=IPv4Address("192.168.100.1")
    pkt[1].dst=IPv4Address("192.168.200.1")
    pkt[1].ttl=64
    pkt[0].src=EthAddr("10:00:00:00:00:01")
    pkt[0].dst=EthAddr("40:00:00:00:00:01")
    log_info('right is moving')
    self.right+=1
    sequencenumber=self.right
    rawsequencenumber=sequencenumber.to_bytes(4,byteorder='big')
    rawlength=self.length.to_bytes(2,byteorder='big')
    rawcontent=bytes([0]*self.length)
    pkt+=RawPacketContents(rawsequencenumber)
    pkt+=RawPacketContents(rawlength)
    pkt+=RawPacketContents(rawcontent)
    log_info('first time send_packet with sequencenumber{}'.format(sequencenumber))
    self.net.send_packet('blaster-eth0',pkt)
    if sequencenumber==1:
        starttime=time.time()
    self.queue.append(Item(pkt,sequencenumber,starttime))
    allbytes+=self.length
    allbytesonce+=self.length
```

否则最后再判断此时是否超时，如果已经粗略超时则把需要重发的包加入重发队列进行重发。

```
elif time.time()-self.timestamp>=self.timeout:
    log_info("timeout!!!")
    timeouttimes+=1
    for item in self.queue:
        if item.acked==False:
            self.retranqueue.append(item)
    log_info('timeout and retransmit the sequencenumber{} packt'.format(self.retranqueue[0].sequencenumber))
    self.net.send_packet('blaster-eth0',self.retranqueue[0].pkt)
    retrantimes+=1
    allbytes+=self.length
    self.retranqueue.remove(self.retranqueue[0])
```

6. 思考与感受

本次实验相当复杂，难度较大，希望自己已经用最简洁的代码实现了功能。