

计网 LAB6 实验报告

姓名：徐佳美

学号：181860117

专业：计算机科学与技术系

任课老师：李文中

邮箱：181860117@smail.nju.edu.cn

开始/完成日期：5.17-5.20

一实验名称

可靠通信。

二实验目的

实现 ACK 机制，滑动窗口和超时重传。

三实验内容和核心代码

Task2: middlebox

(1) 读取参数：

Getprop 函数从文件中读取丢包率和包的数目。

(2) 转发：

根据收到包的接口的不同，从 blaster 收到包，获取一个 0-1 之间的随机数，如果小于丢包率，丢弃；否则，从另一个接口转发该包；从 blastee 收到包，直接转发，并记录 ACK 的数目，如果等于包数目，退出循环。

```
randNum=random()
if (randNum> compNum):
    pkt[Ethernet].dst=blasteeMac
    net.send_packet("middlebox-eth1", pkt)
#
net.send_packet("middlebox-eth0", pkt)
acknum+=1
if(acknum>=num):
    break
```

Task3: blastee

(1) 读取参数：

Getpara 函数从文件中读取 blaster 的 IP 地址和待 ACK 的包的数目。

(2) 构造 ACK 包：

Mk_ack 函数，传入发送方和接收方的 IP 和 MAC 地址，ttl，构造以太网，IPV4 和 UDP 包头，并根据传入的参数写入相应内容。

从收到的 pkt 中提取 RawPacketContents 包头，前四个字节为序列号，从第七位开始提取八个字节作为 payload，并加入构造的包中。

```
将 ACK 包发出，并记录已经 ack 的数目，等于总的包数目时，退出循环。
ackpkt=mk_ack(blasteeMac,blasterMac,blasteeIp,blasterIp,64)
```

```

xpkt=pkt[RawPacketContents]
log_info("xpkt={}".format(xpkt))
seqNum=xpkt._raw[:4]
log_info("seq={}".format(seqNum))
payLoad=pkt[RawPacketContents]._raw[6:14]
ackpkt=ackpkt+seqNum+payLoad
net.send_packet(dev,ackpkt)
acknum+=1
if(acknum>=num):
    break

```

Task4: blaster

(1) 读取参数:

Getpara 函数从文件中读取 blaste 的 IP 地址, 包的数目, 负载长度, SW, 超时时间, 阻塞时间。

(2) 构造要发送的包:

Initpkt 函数, 传入要构造的包的数目, 发送方和接收方的 IP 和 MAC, 负载长度, ALLpkt[] 存储所有要发送的包: 第一项 pkt 存储包的前三个包头, 以太网, IPV4 和 UDP 包头, 第二项到第四项分别存储转成 byte 格式的序列号, 负载长度, 负载; 后三项存储 ACK 标记, 发送标记, 上次发送时间来辅助之后的操作。

```

payload='a pkt send from blaster to blaste'
payload=payload[0:length]
plraw=RawPacketContents(payload)
lenraw=(length).to_bytes(2,'big')
TotalPkt=[]
for i in range(1,num+1):
    seqraw=(i).to_bytes(4,'big') # i is the num,4 is bytenum,big-endian
    temp=[]
    temp.append(pkt) #0:
    temp.append(seqraw) #1:seqNum
    temp.append(lenraw) #2
    temp.append(plraw) #3
    temp.append(False) # 4: acked or not
    temp.append(False) #5 :has been sent or not
    temp.append(0) #6 :send time
    TotalPkt.append(temp)
return TotalPkt

```

(3) 发包:

首先判断当前是否能发包, 然后调用 sendPkt 函数:

该函数检查序列号从 LHS 到 LHS+SW 之间的包, 若发现有超时还没有收到 ACK 的包, 优先发送这个包, 记录重传次数; 否则, 发送当前窗口中未发送过的序列号最小的包, RHS+1; 返回 RHS 和重传次数。

```

def sendPkt(self,SW,left,right,timeout,net,num,renum):
    for i in range(left,left+SW): #seqnum begin 1,but i store it begin 0
        if i<=num and self[i-1][4]==False: #not been acked,total num pkts
            j=i-1
            if self[i-1][5]==True and time.time()-self[i-1][6]>timeout: #the
timeout pkt is priority
                pkt=self[j][0]+self[j][1]+self[j][2]+self[j][3]
                net.send_packet("blaster-eth0",pkt)
                self[j][6]=time.time()
                renum+=1
                log_info("resend seq={}".format(i))
            elif self[j][5]==False:
                self[j][5]=True
                self[j][6]=time.time()
                pkt=self[j][0]+self[j][1]+self[j][2]+self[j][3]
                net.send_packet("blaster-eth0",pkt)
                right+=1
                log_info("send seq={}".format(i))
    return right,renum

```

(4) 收到 ACK 包:

Getpkt 函数, 从 pkt 中提取出序列号, 判断该包是否被 ACK 过, 如果没有, 更改 ACK 标记, 并将 acknum 加 1, 如果序列号等于 LHS, 将 LHS 加 1。

```

def getPkt(self,pkt,timeOut,left,ackednum):
    log_info("getPkt")
    xpkt = pkt[RawPacketContents]._raw
    seqNum=xpkt[0:4]
    log_info("seq={}".format(seqNum))
    seq=int.from_bytes(seqNum,'big')
    if time.time()-self[seq-1][6]<=timeOut:
        if(self[seq-1][4])==False:
            ackednum+=1
            self[seq-1][4]=True #change ack tag
            if(seq==left):
                left+=1
    return left,ackednum

```

(5) 打印状态:

比较已经 ACK 的包的数目和总的数目, 根据相关变量记录或计算, 打印状态信息, 退出循环。

N=7,丢包率 0.2 打印信息如下:

```

23:31:44 2020/05/19 INFO Using network devices: blaster-eth0
23:31:44 2020/05/19 INFO getpara
23:31:44 2020/05/19 INFO initpkt
23:31:44 2020/05/19 INFO send seq=1
23:31:44 2020/05/19 INFO send seq=2
23:31:44 2020/05/19 INFO send seq=3
23:31:44 2020/05/19 INFO send seq=4
23:31:44 2020/05/19 INFO send seq=5
23:31:45 2020/05/19 INFO getPkt
23:31:45 2020/05/19 INFO seq=b'\x00\x00\x00\x01'
23:31:45 2020/05/19 INFO getPkt
23:31:45 2020/05/19 INFO seq=b'\x00\x00\x00\x02'
23:31:45 2020/05/19 INFO getPkt
23:31:45 2020/05/19 INFO seq=b'\x00\x00\x00\x03'
23:31:45 2020/05/19 INFO getPkt
23:31:45 2020/05/19 INFO seq=b'\x00\x00\x00\x05'
23:31:45 2020/05/19 INFO send seq=6
23:31:45 2020/05/19 INFO send seq=7
23:31:45 2020/05/19 INFO getPkt
23:31:45 2020/05/19 INFO seq=b'\x00\x00\x00\x06'
23:31:45 2020/05/19 INFO getPkt
23:31:45 2020/05/19 INFO seq=b'\x00\x00\x00\x07'
23:31:54 2020/05/19 INFO resend seq=4
23:31:55 2020/05/19 INFO getPkt
23:31:55 2020/05/19 INFO seq=b'\x00\x00\x00\x04'
23:31:55 2020/05/19 INFO total time=10.820921182632446
23:31:55 2020/05/19 INFO retransmitted num=1
23:31:55 2020/05/19 INFO timeout num=1
23:31:55 2020/05/19 INFO through put bps=5.914468733283064
23:31:55 2020/05/19 INFO good put bps=5.175160141622681
23:31:55 2020/05/19 INFO Restoring saved iptables state

(syenv) root@njucs-VirtualBox:~/switchyard#

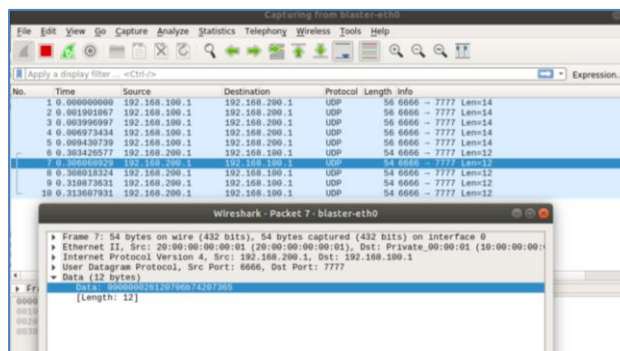
```

Task5: 运行和验证

n=5, SW=5, 丢包率 0.2

此时没有发生重传:

Blaster 1-5 为发送的包, 6-10 为收到的 ACK 包, 序号为 1-5;



Blastee 显示发出了五个包的 ACK;

```

root@njucs-VirtualBox:~/switchyard# source ./syenv/bin/activate
(syenv) root@njucs-VirtualBox:~/switchyard# ./switchyard/seqarg.py lab_6/blastee
e.py
21:39:33 2020/05/20 INFO Saving iptables state and installing switchyard rule
21:39:33 2020/05/20 INFO Using network devices: blastee-eth0
21:39:33 2020/05/20 INFO ip:192.168.100.1,num:5
21:39:33 2020/05/20 INFO seqb='\x00\x00\x00\x01'
21:39:33 2020/05/20 INFO seqb='\x00\x00\x00\x02'
21:39:33 2020/05/20 INFO seqb='\x00\x00\x00\x03'
21:39:33 2020/05/20 INFO seqb='\x00\x00\x00\x04'
21:39:33 2020/05/20 INFO seqb='\x00\x00\x00\x05'
21:39:33 2020/05/20 INFO Restoring saved iptables state

(syenv) root@njucs-VirtualBox:~/switchyard#

```

Middlebox 的两个端口各自收到了十个包, 其中, 前五个为从 blaster 收到的序号 1-5 的包,

后五个为从 blastee 收到的序号为 1-5 的 ACK 包。

