南京大学

LAB 6：Reliable Communication

课程名称：计算机网络

姓名：孙文博

学号：201830210

学院：计算机科学与技术系

Email：201830210@smail.nju.edu.cn

任课教师：李文中

实验时间：2022.5.5 - 2022.5.19

一、 实验名称

Reliable Communication

二、 实验目的

- ✧ 设计一个可靠通信机制
- ✧ 巩固对滑动窗口等课内知识的理解

三、 实验内容

Task 1: Preparation

配置实验环境

Task 2: Middlebox

实现 Middlebox 功能，其会按照一个概率丢弃包，从而模拟网络中丢包的情况

Task 3: Blastee

实现 Blastee 功能，其作为接收端

Task4: Blaster

实现 Blaster 功能，其作为发送端

Task5: Running your code

测试 Middlebox，Blastee 和 Blaster 实际运行情况

四、 实验过程

Task 2: Middlebox

The features of middlebox

首先判断发来的包是哪个端口传来的，如果是从 middlebox-eth0 则说明该包是从 blaster 发来的，将包的源 mac 地址改为 middlebox-eth1 的 mac 地址，目的地址改为 blastee 的 mac 地址即可；如果该包是从 middlebox-eth1 传来的话，说明是从 blaster 发来的，与前一种情况类似。

在收到包之后，还会按照一定的丢包。其中丢包的具体实现是会在 1 到 100 产生一个随机数，比如丢包的概率为 0.19，如果这个数大于 20 则会被转发，小于 20 则会被丢弃。

Task 3: Blastee

The features of Blastee

Blastee 会收到来自 Blaster 的包，该包的结构如下：

```
        # blaster packet format:
        '''
<-------- Switchyard headers -----> <----- Your packet header(raw bytes) ------> <-- Payload in raw bytes --->
-------------------------------------------------------------------------------------------------------------
|  ETH Hdr |  IP Hdr  |  UDP Hdr  | Sequence number(32 bits) | Length(16 bits) |   Variable length payload  |
-------------------------------------------------------------------------------------------------------------
        '''
```

Blastee 的 ACK 回复的包格式应为如下结构：

```
        # ACK packet format:
        '''
------- Switchyard headers -----> <----- Your packet header(raw bytes) ------> <-- Payload in raw bytes --->
-------------------------------------------------------------------------------------------------------------
  ETH Hdr |  IP Hdr  |  UDP Hdr  |          Sequence number(32 bits)          |      Payload  (8 bytes)    |
-------------------------------------------------------------------------------------------------------------
        '''
```

所以要构造一个 Blastee 发送给 Blaster 的包，首先设置好 ETH，IP 和 UDP 包头。其中 ETH 和 IP 包头的源地址都为 Blastee 的 mac 地址和 ip 地址，目的地址为 Blaster 的 mac 地址和 ip 地址。由 Blaster 发来数据包的结构可知 packet[3] 中的第 0 到 4 字节存放着 Sequence number；第 4 到 6 字节存放着 Length；第 6 字节开始存放着 payload。所以在构造 Blastee 包的时候就要将

Sequence number 设置为 packet[3] 中的第 0 到 4 字节； Payload 设置为 packet[3] 从第 6 字节开始的 8 个字节。

Task4: Blaster

The features of Blaster

本节逻辑主要体现在是两个函数模块中，分别是 handle_packet 和 handle_no_packet 。

- ✧ handle_packet 主要处理从 Blastee 发往 Blaster 的 ACK 包
- ✧ handle_no_packet 主要处理内容是 Blaster 向 Blastee 发送新数据包，并且重新发送没有收到 ACK 的数据包

handle_packet 逻辑

读出收到包的 Sequence number 并将该序号对应的数据做好标记，表示该包已经被收到不需要被重传。并且还要及时的更新此时的 LHS 值。

handle_no_packet 逻辑

首先判断 LHS 序号对应的包是否超时，如果超时需要进行一次重传；否则需要判断目前 RHS 和 LHS 的位置判断是否超过发送窗口的大小，如果没有超过则可以发送新的包，并且更新 RHS 的值

Task5: Running your code

Running in the Mininet

在终端中输入以下指令启动 mininet:

```
$ sudo python start_mininet.py
```

在 mininet 中启动 middlebox，blastee 和 blaster 并且在各自的 xterm 中运行如下指令：

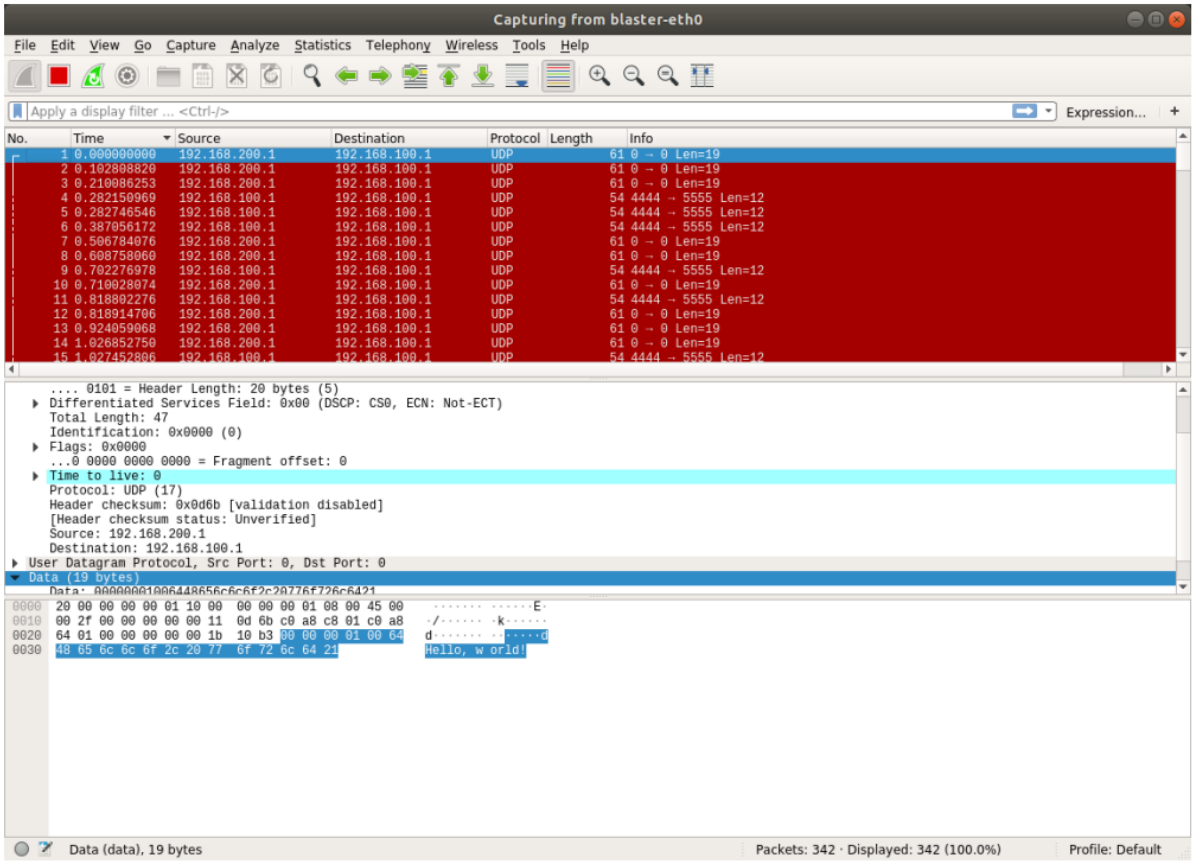
```
mininet> xterm middlebox
mininet> xterm blastee
mininet> xterm blaster
```

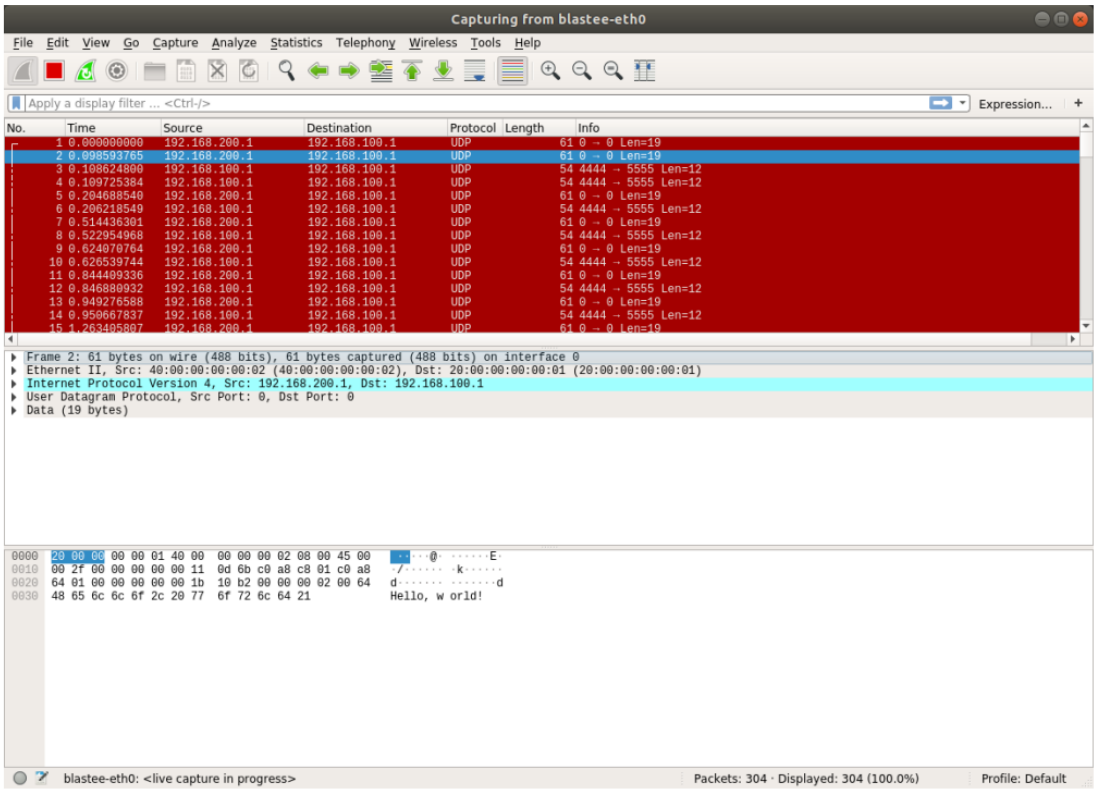
```
middlebox# swyard middlebox.py -g 'dropRate=0.19'
blastee# swyard blastee.py -g 'blasterIp=192.168.100.1 num=100'
blaster# swyard blaster.py -g 'blasteeIp=192.168.200.1 num=100 length=100
senderWindow=5 timeout=300 recvTimeout=100'
```

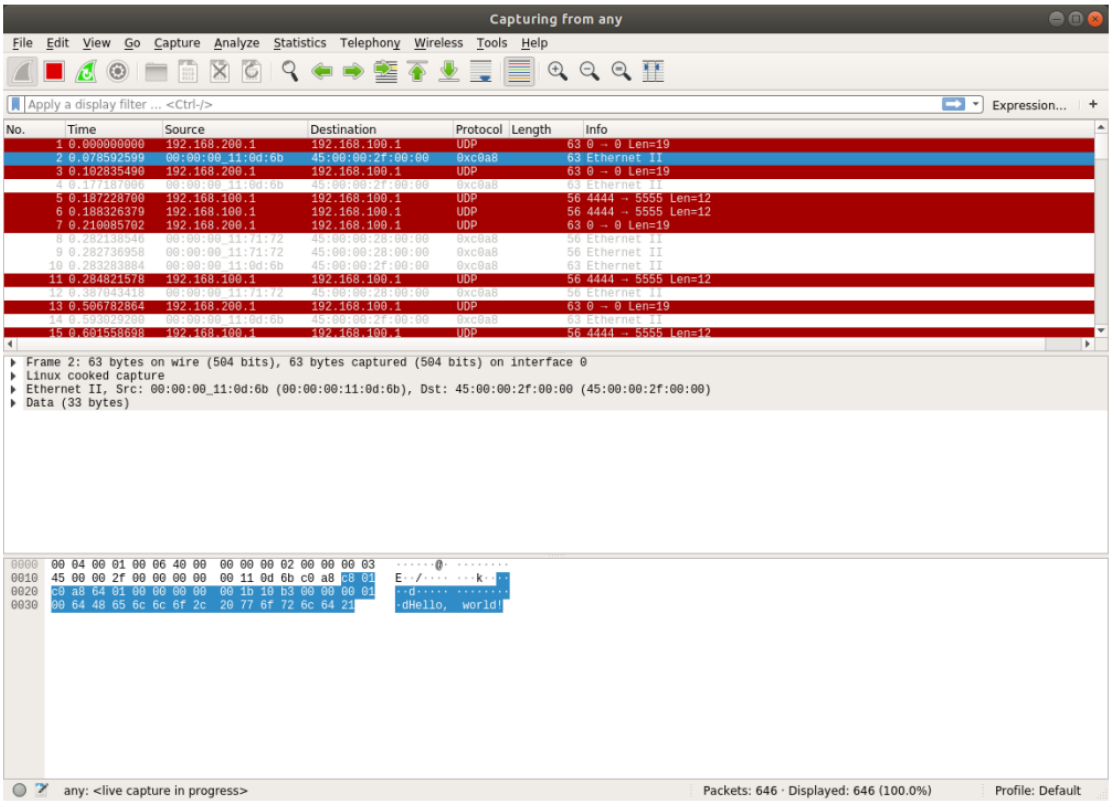
利用 wireshark 抓 middlebox , blastee 和 blaster 的包：

```
middlebox# wireshark -i middlebox
blastee# wireshark -i blastee
blaster# wireshark -i blaster
```

Wireshark 的抓包结果如下，依次是 blaster, blastee 和 middlebox：







其中 Info 中从端口 4444 到 5555 的都是 blastee 发送给 blaster 数据包；端口 0 到 0 都是 blaster 发送给 blastee 数据包。

从 blaster 的 wireshark 结果具体如下：

No1 数据包的 Sequence 为 1；No2 数据包的 Sequence 为 2；No3 数据包的 Sequence 为 1。结合 middlebox 在 xterm 的 info 和上面说过的端口 4444 到 5555 的数据包都是 blastee 发送给 blaster 等信息，可以发现 No1 被 middlebox 所丢弃，进行了一次重传。

五、 核心代码

Middlebox:

```
if fromIface == "middlebox-eth0":
    randnum = randint(1, 100)
    # drop
    if (randnum < 20):
        log_info ("middlebox drop packet")
    # modify and send
    else:
        packet[Ethernet].src = self.eth_list[middlebox_eth1_num]
        packet[Ethernet].dst = "20:00:00:00:00:01"
        log_info(f"Sending packet {packet} to blastee")
        self.net.send_packet("middlebox-eth1", packet)
    elif fromIface == "middlebox-eth1":
        packet[Ethernet].src = self.eth_list[middlebox_eth0_num]
        packet[Ethernet].dst = "10:00:00:00:00:01"
        log_info(f"Sending packet {packet} to blaster")
        self.net.send_packet("middlebox-eth0", packet)
```

重点是丢包的实现，会在 1 到 100 产生一个随机数，比如丢包的概率为 0.19，那么如果这个数大于 20 则会被转发， 小于 20 则会被丢弃。

Blastee:

```
sequence = struct.pack(">4s", packet[3].to_bytes()[0:4])
payload = struct.pack(">8s", packet[3].to_bytes()[6:14])
```

重点是 ACK 回复的生成，查阅手册及 Python 相关语法可以构建出我们想要的数据包结构（前面有提及），这里用到了 struct 的 pack 方法。

Blaster:

```
if (time.time() - self.LHS_timer) > self.timeout:
    Sequence_number = self.LHS.to_bytes(4, "big")
    Length = self.length.to_bytes(2, "big")
    Variable_length_payload = struct.pack(">13s", bytes("hello,
world!".encode('utf-8')))
    log_info (f"Retransmitting pakcet from blaster to blastee, packet info
{pkt}")
    self.net.send_packet("blaster-eth0", pkt)
elif (self.RHS - self.LHS + 1 <= self.senderWindow) and
(self.sent_pkt_flag[self.num-1] == 0):
    Sequence_number = self.RHS.to_bytes(4, "big")
    Length = self.length.to_bytes(2, "big")
    Variable_length_payload = struct.pack(">13s", bytes("Hello,
world!".encode('utf-8')))
    log_info(f"Sending packet from blaster to blastee, pkt info {pkt}")
    self.net.send_packet("blaster-eth0", pkt)
    self.sent_pkt_flag[self.RHS] = 1
    if (self.RHS - self.LHS + 1 < self.senderWindow) and (self.RHS <
self.num):
    self.RHS += 1
```

通过判断 LHS 序号对应的包是否超时，如果超时需要进行一次重传；否则需要判断目前 RHS 和 LHS 的位置判断是否超过发送窗口的大小，如果没有超过则可以发送新的包。

六、 实验总结

本次 LAB 相较前几次实验代码量有提升，难度也不低，需要对课内的相关知识有一定掌握才能理解手册中布置的任务，比如滑动窗口相关公式等等，相对而言本次实验还是比较有趣的，并且没有 testcase 的约束，只要 wireshark 的捕获结果符合我们的预期即可。这也是倒数第二次实验了，期待计网实验的完结！