

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	211220049	姓名	石璐
Email	211220049@smail.nju.edu.cn	开始/完成日期	2023 年 5 月 26 日

1. 实验名称：Reliable Communication

2. 实验目的

建立可靠通信库，包括以下功能：

- （1）在 blaster 上建立 ACK 机制
- （2）维护 blaster 上固定大小的滑动窗口
- （3）若 blaster 上发生超时，重新发送未 ACK 的数据包。

3. 实验内容

Task 1: Middlebox

✓ Coding:

```
21
22 def handle_packet(self, recv: switchyard.llnetbase.ReceivedPacket):
23     _, fromIface, packet = recv
24     if fromIface == "middlebox-eth0":
25         log_debug("Received from blaster")
26         """
27         Received data packet
28         Should I drop it?
29         If not, modify headers & send to blastee
30         """
31         if random() < self.dropRate:
32             return
33         packet[0] = Ethernet(src="40:00:00:00:00:02",dst="20:00:00:00:00:01",ethertype=EtherType.I
34         self.net.send_packet("middlebox-eth1", packet)
35     elif fromIface == "middlebox-eth1":
36         log_debug("Received from blastee")
37         """
38         Received ACK
39         Modify headers & send to blaster. Not dropping ACK packets!
40         net.send_packet("middlebox-eth0", pkt)
41         """
42         packet[0] = Ethernet(src="40:00:00:00:00:01",dst="10:00:00:00:00:01",ethertype=EtherType.I
43         self.net.send_packet("middlebox-eth0", packet)
44     else:
45         log_debug("Oops :))")
```

- 来自 blaster (eth0) 的报文，以 dropRate 的概率丢包。
- 来自 blastee (eth1) 的报文，直接转发。

Task 2: Blastee

✓ Coding:

```
24
25 def handle_packet(self, recv: switchyard.llnetbase.ReceivedPacket):
26     _, fromIface, packet = recv
27     log_debug(f"I got a packet from {fromIface}")
28     log_debug(f"Pkt: {packet}")
29
30     # construct headers
31     eth = Ethernet(src="20:00:00:00:00:01",dst="40:00:00:00:00:02",ethertype=EtherType.IPv4)
32     ip = IPv4(src="192.168.200.1",dst=self.blasterIp,protocol=IPProtocol.UDP)
33     udp = UDP()
34     seq = RawPacketContents(raw=packet[3].to_bytes()[4:])
35     pkt = eth + ip + udp + seq
36
37     payLoadLen = int.from_bytes(packet[3].to_bytes()[4:6],byteorder='big')
38     if payLoadLen >= 8:
39         pkt.add_payload(packet[3].to_bytes()[6:14])
40     else:
41         pkt.add_payload(packet[3].to_bytes()[6:]+bytes([0]*(8-payLoadLen)))
42
43     # send ACK
44     self.net.send_packet("blastee-eth0", pkt)
```

- 收到来自 blaster 的报文，需要回复对应的 ACK 报文
- 首先创建 ACK 报文：包括 Ethernet 头、IP 头、UDP 头、seq number 和 payload 五个部分，其中 seq number 与 recv_packet 中的 seq number 相同，payload 取 recv_packet 中负载的前 8 个字节(不足 8 个用 0 补齐)。
- 然后向 blaster 发送 ACK 报文

Task 3: Blaster

✓ Coding:

```
41 def handle_packet(self, recv: switchyard.llnetbase.ReceivedPacket):
42     _, fromIface, packet = recv
43     log_debug("I got a packet")
44     log_debug(f"Pkt: {packet}")
45
46     # ACK
47     seq = int.from_bytes(packet[3].to_bytes():4,byteorder='big')
48     if seq in self.packets:
49         if seq == list(self.packets.keys())[0]:
50             self.time = time.time()
51             del self.packets[seq]
52
53     if len(self.packets) > 0:
54         self.LHS = list(self.packets.keys())[0]
55     else:
56         self.LHS = self.RHS + 1
57     # print stats
58     log_debug(f"Total TX time: {self.time-self.baseTime} s")
59     log_debug(f"Number of reTX: {self.reTXnum}")
60     log_debug(f"Number of T0s: {self.T0num}")
61     log_debug(f"Throughput: {self.throughput/(self.time-self.baseTime)} Bps")
62     log_debug(f"Goodput: {self.goodput/(self.time-self.baseTime)} Bps")
63
```

- 第一部分：处理收到的 ACK 报文
 - 获取 ACK 报文的 seq
 - 查找 self.packets 中是否有 key 为 seq 的项(用 self.packets 字典结构记录未被 ACK 的报文, key 为 seq number, value 为 packet), 如果有, 说明 seq 对应的报文未被 ACK, 将该报文从 self.packets 中删除, 表示该报文被 ACK, 并将 LHS 更新为剩余报文中最小的 seq; 如果没有, 表示重复 ACK, 不做任何处理。
 - 如果 seq 为 SW 的 LHS, 重置计时器。
 - 如果当前所有报文均被 ACK, 打印统计信息。

```
66 def handle_no_packet(self):
67     log_debug("Didn't receive anything")
68
69     # process old packet
70     if len(self.packets) > 0 and (time.time() - self.time) * 1000 > float(self.timeout):
71         self.T0num += 1
72         for i in self.packets.keys():
73             if i not in self.wl:
74                 self.wl.append(i)
75
76     NRT = True # not retransmit
77     while NRT and len(self.wl) > 0:
78         seq = self.wl[0]
79         del self.wl[0]
80         if seq in self.packets:
81             self.net.send_packet("blaster-eth0", self.packets[seq])
82             NRT = False
83             self.reTXnum += 1
84             self.throughput += int.from_bytes(self.packets[seq][4].data,byteorder='big')
85             if seq == self.LHS:
86                 self.time = time.time()
87
```

- 第二部分：处理 SW 中未被 ACK 的报文

- 如果 self.packets 存在未被 ACK 的报文且发生 timeout，则将所有报文加入等待队列，并在后续循环中依次重传，并适时重置计时器。
- 同时记录 TOnum、reTXnum、throughput 等统计信息。

```
88 # send new packet
89 if NRT and self.RHS - self.LHS + 1 < int(self.senderWindow) and self.RHS < int(self.num):
90     if self.RHS == 0:
91         self.baseTime = time.time()
92
93     # Creating the headers for the packet
94     pkt = Ethernet() + IPv4() + UDP()
95     pkt[0].src="10:00:00:00:00:01"
96     pkt[0].dst="40:00:00:00:00:01"
97     pkt[0].ethertype=EtherType.IPv4
98     pkt[1].src = "192.168.100.1"
99     pkt[1].dst = self.blasteeIp
100    pkt[1].protocol = IPProtocol.UDP
101
102    # Do other things here and send packet
103    seq = self.RHS + 1
104
105    # construct packet
106    pkt += RawPacketContents(raw=(seq).to_bytes(4,byteorder='big'))
107    pkt += RawPacketContents(raw=(int(self.length)).to_bytes(2,byteorder='big'))
108
109    intstream = [randint(0,255) for _ in range(int(self.length))]
110    string = str(intstream)
111    pkt += RawPacketContents(raw=string)
112
113    # send packet
114    self.net.send_packet("blaster-eth0", pkt)
115    self.packets[seq] = pkt
116    self.RHS = seq
117    self.throughput += int.from_bytes(pkt[4].data,byteorder='big')
118    self.goodput += int.from_bytes(pkt[4].data,byteorder='big')
```

- 第三部分：发送新的报文

- 如果当前循环中未重传任何报文且 SW 未满足且发送的报文总数未达到上限 num，则可以发送新的报文。
- 在第一次发送报文时记录 baseTime。
- 创建新报文：包括 Ethernet 头、IP 头、UDP 头、seq number（4 字节）、payload len（2 字节）和 payload（可变长度）六个部分，其中 seq number 为 RHS+1，payload 填充长度为 payload len 的随机字节流。
- 向 blastee 发送新报文，并将其加入 self.packets 等待 ACK。
- 更新 RHS
- 记录 throughput、goodput 等统计信息。

Task 4: Running Code

✅ Deploying:

在 mininet 中运行，wireshark 在 middlebox-eth0 上抓包。

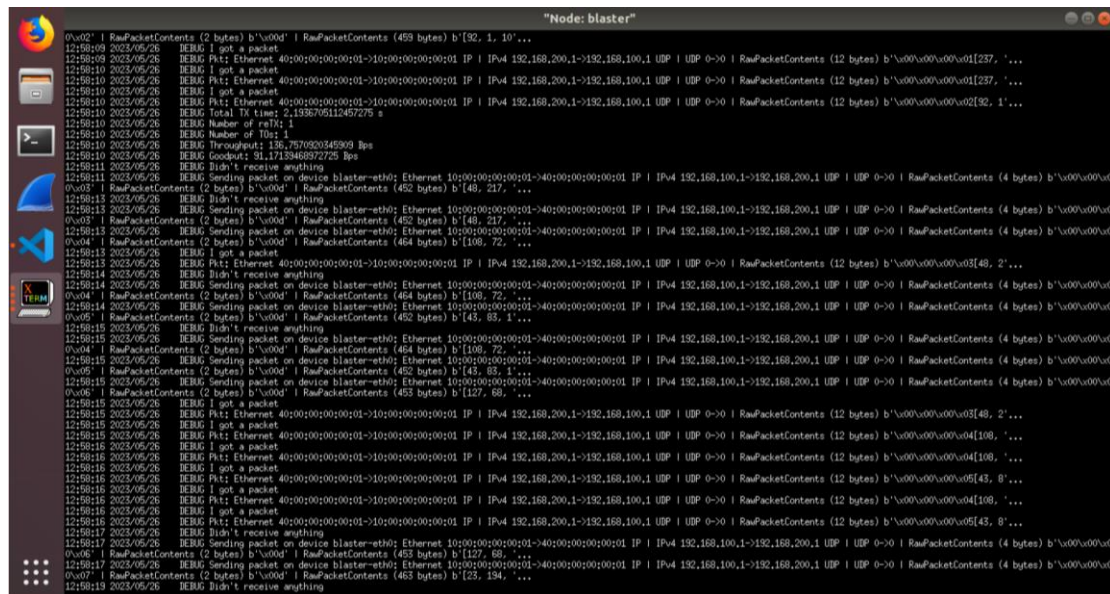
结合以下 log_debug 信息和抓包信息进行简要说明。

● log_debug 信息:

- blaster 发送 seq=1 报文，middlebox 成功转发该报文
- blastee 对 seq=1 发送 ACK 报文，middlebox 转发该 ACK 报文
- 在首次对 seq=1 的 ACK 报文到达 blaster 之前发生 timeout，因此 blaster 重传 seq=1 报文，middlebox 成功转发并发回对重传 seq=1 的 ACK
- 接着 blaster 发送 seq=2 报文，middlebox 成功转发并发回对 seq=2 的 ACK
- 三个 ACK 报文依次到达 blaster，分别为 seq=1（首次发送的 ACK）、seq=1（重传的 ACK）、seq=2，此时 blaster 所有已发送的报文均被 ACK，打印 stats。
- 接着 blaster 继续发送新的 seq=3 报文，后续过程与上述过程类似。

● 抓包信息:

- 在 middlebox 的 eth0 端口上，依次收到了首次 seq=1 报文、对首次 seq=1 的 ACK 报文、seq=1 的重传报文、seq=2 报文、对重传 seq=1 的 ACK 报文、对 seq=2 的 ACK 报文、……，符合我们在 log_debug 中分析的过程



```
"Node: blaster"
0x02' | RawPacketContents (2 bytes) b'\x00' | RawPacketContents (459 bytes) b'[32, 1, 10'...'
12:58:09 2023/05/26  DEBG I got a packet
12:58:09 2023/05/26  DEBG Pkt: Ethernet 40200:00:00:00:01->10:00:00:00:00:01 IP | IPv4 192.168.200.1->192.168.100.1 UDP | UDP 0->0 | RawPacketContents (12 bytes) b'\x00\x00\x00\x01[237, '...'
12:58:10 2023/05/26  DEBG I got a packet
12:58:10 2023/05/26  DEBG Pkt: Ethernet 40200:00:00:00:01->10:00:00:00:00:01 IP | IPv4 192.168.200.1->192.168.100.1 UDP | UDP 0->0 | RawPacketContents (12 bytes) b'\x00\x00\x00\x01[237, '...'
12:58:10 2023/05/26  DEBG I got a packet
12:58:10 2023/05/26  DEBG Pkt: Ethernet 40200:00:00:00:01->10:00:00:00:00:01 IP | IPv4 192.168.200.1->192.168.100.1 UDP | UDP 0->0 | RawPacketContents (12 bytes) b'\x00\x00\x00\x02[32, 1'...'
12:58:10 2023/05/26  DEBG Total Tx time: 2.1336705112457275 s
12:58:10 2023/05/26  DEBG Number of rx: 1
12:58:10 2023/05/26  DEBG Throughput: 136.797020345909 Bps
12:58:10 2023/05/26  DEBG Goodput: 91.1213468972725 Bps
12:58:11 2023/05/26  DEBG I didn't receive anything
12:58:11 2023/05/26  DEBG Sending packet on device blaster-eth0: Ethernet 10:00:00:00:00:01->40:00:00:00:00:01 IP | IPv4 192.168.100.1->192.168.200.1 UDP | UDP 0->0 | RawPacketContents (4 bytes) b'\x00\x00\x00'
0x03' | RawPacketContents (2 bytes) b'\x00' | RawPacketContents (452 bytes) b'[48, 217, '...'
12:58:13 2023/05/26  DEBG I didn't receive anything
12:58:13 2023/05/26  DEBG Sending packet on device blaster-eth0: Ethernet 10:00:00:00:00:01->40:00:00:00:00:01 IP | IPv4 192.168.100.1->192.168.200.1 UDP | UDP 0->0 | RawPacketContents (4 bytes) b'\x00\x00\x00'
0x03' | RawPacketContents (2 bytes) b'\x00' | RawPacketContents (452 bytes) b'[48, 217, '...'
12:58:14 2023/05/26  DEBG Sending packet on device blaster-eth0: Ethernet 10:00:00:00:00:01->40:00:00:00:00:01 IP | IPv4 192.168.100.1->192.168.200.1 UDP | UDP 0->0 | RawPacketContents (4 bytes) b'\x00\x00\x00'
0x04' | RawPacketContents (2 bytes) b'\x00' | RawPacketContents (454 bytes) b'[108, 72, '...'
12:58:15 2023/05/26  DEBG I got a packet
12:58:15 2023/05/26  DEBG Pkt: Ethernet 40200:00:00:00:01->10:00:00:00:00:01 IP | IPv4 192.168.200.1->192.168.100.1 UDP | UDP 0->0 | RawPacketContents (12 bytes) b'\x00\x00\x00\x03[48, 2'...'
12:58:15 2023/05/26  DEBG I didn't receive anything
12:58:15 2023/05/26  DEBG Sending packet on device blaster-eth0: Ethernet 10:00:00:00:00:01->40:00:00:00:00:01 IP | IPv4 192.168.100.1->192.168.200.1 UDP | UDP 0->0 | RawPacketContents (4 bytes) b'\x00\x00\x00'
0x04' | RawPacketContents (2 bytes) b'\x00' | RawPacketContents (454 bytes) b'[108, 72, '...'
12:58:16 2023/05/26  DEBG Sending packet on device blaster-eth0: Ethernet 10:00:00:00:00:01->40:00:00:00:00:01 IP | IPv4 192.168.100.1->192.168.200.1 UDP | UDP 0->0 | RawPacketContents (4 bytes) b'\x00\x00\x00'
0x05' | RawPacketContents (2 bytes) b'\x00' | RawPacketContents (452 bytes) b'[45, 83, 1'...'
12:58:16 2023/05/26  DEBG Sending packet on device blaster-eth0: Ethernet 10:00:00:00:00:01->40:00:00:00:00:01 IP | IPv4 192.168.100.1->192.168.200.1 UDP | UDP 0->0 | RawPacketContents (4 bytes) b'\x00\x00\x00'
0x06' | RawPacketContents (2 bytes) b'\x00' | RawPacketContents (453 bytes) b'[127, 68, '...'
12:58:16 2023/05/26  DEBG I got a packet
12:58:16 2023/05/26  DEBG Pkt: Ethernet 40200:00:00:00:01->10:00:00:00:00:01 IP | IPv4 192.168.200.1->192.168.100.1 UDP | UDP 0->0 | RawPacketContents (12 bytes) b'\x00\x00\x00\x03[48, 2'...'
12:58:16 2023/05/26  DEBG I got a packet
12:58:16 2023/05/26  DEBG Pkt: Ethernet 40200:00:00:00:01->10:00:00:00:00:01 IP | IPv4 192.168.200.1->192.168.100.1 UDP | UDP 0->0 | RawPacketContents (12 bytes) b'\x00\x00\x00\x04[108, '...'
12:58:16 2023/05/26  DEBG I got a packet
12:58:16 2023/05/26  DEBG Pkt: Ethernet 40200:00:00:00:01->10:00:00:00:00:01 IP | IPv4 192.168.200.1->192.168.100.1 UDP | UDP 0->0 | RawPacketContents (12 bytes) b'\x00\x00\x00\x05[43, 8'...'
12:58:16 2023/05/26  DEBG I got a packet
12:58:16 2023/05/26  DEBG Pkt: Ethernet 40200:00:00:00:01->10:00:00:00:00:01 IP | IPv4 192.168.200.1->192.168.100.1 UDP | UDP 0->0 | RawPacketContents (12 bytes) b'\x00\x00\x00\x04[108, '...'
12:58:16 2023/05/26  DEBG I got a packet
12:58:16 2023/05/26  DEBG Pkt: Ethernet 40200:00:00:00:01->10:00:00:00:00:01 IP | IPv4 192.168.200.1->192.168.100.1 UDP | UDP 0->0 | RawPacketContents (12 bytes) b'\x00\x00\x00\x05[43, 8'...'
12:58:17 2023/05/26  DEBG I didn't receive anything
12:58:17 2023/05/26  DEBG Sending packet on device blaster-eth0: Ethernet 10:00:00:00:00:01->40:00:00:00:00:01 IP | IPv4 192.168.100.1->192.168.200.1 UDP | UDP 0->0 | RawPacketContents (4 bytes) b'\x00\x00\x00'
0x06' | RawPacketContents (2 bytes) b'\x00' | RawPacketContents (453 bytes) b'[127, 68, '...'
12:58:17 2023/05/26  DEBG Sending packet on device blaster-eth0: Ethernet 10:00:00:00:00:01->40:00:00:00:00:01 IP | IPv4 192.168.100.1->192.168.200.1 UDP | UDP 0->0 | RawPacketContents (4 bytes) b'\x00\x00\x00'
0x07' | RawPacketContents (2 bytes) b'\x00' | RawPacketContents (453 bytes) b'[23, 194, '...'
12:58:19 2023/05/26  DEBG I didn't receive anything
```


Wireshark 五 12:59
Capturing from middlebox-eth0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.100.1	192.168.200.1	UDP	496	0 → 0 Len=454
2	1.043068258	192.168.200.1	192.168.100.1	UDP	54	0 → 0 Len=12
3	1.312214283	192.168.100.1	192.168.200.1	UDP	496	0 → 0 Len=454
4	1.312215805	192.168.100.1	192.168.200.1	UDP	507	0 → 0 Len=465
5	1.874177395	192.168.200.1	192.168.100.1	UDP	54	0 → 0 Len=12
6	1.874182282	192.168.200.1	192.168.100.1	UDP	54	0 → 0 Len=12
7	3.286276435	192.168.100.1	192.168.200.1	UDP	500	0 → 0 Len=458
8	4.397613964	192.168.200.1	192.168.100.1	UDP	54	0 → 0 Len=12
9	4.551536848	192.168.100.1	192.168.200.1	UDP	500	0 → 0 Len=458
10	4.566567009	192.168.100.1	192.168.200.1	UDP	512	0 → 0 Len=470
11	6.024228532	192.168.100.1	192.168.200.1	UDP	512	0 → 0 Len=470
12	6.024521066	192.168.100.1	192.168.200.1	UDP	500	0 → 0 Len=458
13	6.519072167	192.168.200.1	192.168.100.1	UDP	54	0 → 0 Len=12
14	6.525634281	192.168.200.1	192.168.100.1	UDP	54	0 → 0 Len=12
15	6.608943992	192.168.100.1	192.168.200.1	UDP	512	0 → 0 Len=470
16	6.609620349	192.168.100.1	192.168.200.1	UDP	500	0 → 0 Len=458
17	6.609816255	192.168.100.1	192.168.200.1	UDP	501	0 → 0 Len=459
18	7.497002469	192.168.200.1	192.168.100.1	UDP	54	0 → 0 Len=12
19	7.497006334	192.168.200.1	192.168.100.1	UDP	54	0 → 0 Len=12
20	7.890521283	192.168.200.1	192.168.100.1	UDP	54	0 → 0 Len=12
21	8.219669498	192.168.200.1	192.168.100.1	UDP	54	0 → 0 Len=12
22	9.336603567	192.168.100.1	192.168.200.1	UDP	501	0 → 0 Len=459
23	9.403542354	192.168.100.1	192.168.200.1	UDP	511	0 → 0 Len=469
24	11.098038699	192.168.100.1	192.168.200.1	UDP	501	0 → 0 Len=459
25	11.098285047	192.168.100.1	192.168.200.1	UDP	511	0 → 0 Len=469
26	11.153059529	192.168.200.1	192.168.100.1	UDP	54	0 → 0 Len=12
27	11.583566486	192.168.200.1	192.168.100.1	UDP	54	0 → 0 Len=12
28	11.583904353	192.168.200.1	192.168.100.1	UDP	54	0 → 0 Len=12

▶ Frame 1: 496 bytes on wire (3968 bits), 496 bytes captured (3968 bits) on interface 0
 ▶ Ethernet II, Src: Private_00:00:01 (10:00:00:00:00:01), Dst: 40:00:00:00:00:01 (40:00:00:00:00:01)
 ▶ Internet Protocol Version 4, Src: 192.168.100.1, Dst: 192.168.200.1
 ▶ User Datagram Protocol, Src Port: 0, Dst Port: 0
 ▶ Data (454 bytes)

0020 c8 01 00 00 00 00 01 ce 1a 79 00 00 00 01 00 64y.....
 0030 5b 32 33 37 2c 20 39 31 2c 20 33 36 2c 20 32 30 [237, 91, 36, 20
 0040 37 2c 20 36 32 2c 20 31 2c 20 31 35 31 2c 20 39 7, 62, 1, 151, 9
 0050 32 2c 20 31 31 32 2c 20 37 30 2c 20 39 2c 20 32 2, 112, 70, 9, 2

Data (data), 454 bytes

补充：由于丢包概率较低，上面的例子中没有出现丢包，下面的截图显示发生了丢包。

```

23:02:25.2023/05/26 IEBUG Received from blaster
23:02:26.2023/05/26 IEBUG Received from blaster
23:02:26.2023/05/26 IEBUG Sending packet on device middlebox-eth1: Ethernet 40:00:00:00:00:02->20:00:00:00:00:01 IP | IPv4 192.168.100.1->192.168.200.1 UDP | UDP 0->0 | RawPacketContents (468 bytes) b'\x00\x00
0x0f0x00d123'

```