

# 实验报告

## 一、实验名称

Lab 6: Reliable Communication

## 二、实验目的

实现以下功能以提供额外的传输保证：

- 1. blastee上对每个成功接收的数据包实施ACK机制
- 2. blaster上实现固定大小的滑动窗口
- 3. blaster上实施粗略的超时机制，以重新发送未收到ACK的数据包

## 三、实验内容

通过start\_mininet.py读出每个节点的ip和mac，然后打表

	ip	mac
blaster	192.168.100.1	10:00:00:00:00:01
middlebox(eth0 - blaster)	192.168.100.2	40:00:00:00:00:01
middlebox(eth1 - blastee)	192.168.200.2	40:00:00:00:00:02
blastee	192.168.200.1	20:00:00:00:00:01

### middlebox

对于来自blaster的包，修改ethHeader并转发到blastee，以dropRate随机丢包；对于来自blastee的ack回复，修改ethHeader并转发到blaster，不丢包。

```
def handle_packet(self, recv: switchyard.llnetbase.ReceivedPacket):
    _, fromIface, packet = recv
    if not packet.has_header(IPv4):
        log_info("Received non-IPv4 packet")
        return
    seq_num = int.from_bytes(packet[RawPacketContents].data[:4], 'big')
    if fromIface == "middlebox-eth0": # Received from blaster
        if random() < self.dropRate:
            log_info(f"Drop packet {seq_num}")
            return
        log_info(f"Forwarding packet {seq_num}")
        packet[Ethernet].src = middlebox_eth1
        packet[Ethernet].dst = blastee_eth
```

```

        self.net.send_packet("middlebox-eth1", packet)
    elif fromIface == "middlebox-eth1": # Received from blastee
        packet[Ethernet].src = middlebox_eth0
        packet[Ethernet].dst = blaster_eth
        log_info(f"Forwarding ACK {seq_num}")
        self.net.send_packet("middlebox-eth0", packet)

```

## blastee

收到来自blaster的包，创建ack回复，并发到blaster

```

def create_ack(self, packet, seq_num):
    eth = Ethernet(src=blastee_eth, dst=middlebox_eth1, ethertype=EtherType.IPv4)
    ip = IPv4(src=blastee_ip, dst=blaster_ip, protocol=IPProtocol.UDP, ttl=64)
    udp = UDP()
    blaster_payload = packet[RawPacketContents].data.ljust(8, b'\0')
    payload = blaster_payload[:8]
    raw_data = struct.pack('!I8s', seq_num, payload)
    raw_packet = RawPacketContents(raw_data)
    ack = eth + ip + udp + raw_packet
    return ack

def handle_packet(self, recv: switchyard.llnetbase.ReceivedPacket):
    _, fromIface, packet = recv
    seq_num = int.from_bytes(packet[RawPacketContents].data[:4], 'big')
    log_info(f"Received packet {seq_num}")
    ack = self.create_ack(packet, seq_num)
    self.net.send_packet("blastee-eth0", ack)
    if not self.recved[seq_num]:
        self.recved[seq_num] = True
        self.recv_num += 1

```

## blaster

收到窗口最左端lhs位置的的ack，就向右移动lhs到没有ack过的位置

```

def handle_packet(self, recv: switchyard.llnetbase.ReceivedPacket):
    _, fromIface, packet = recv
    seq_num = int.from_bytes(packet[RawPacketContents].data[:4], 'big')
    log_info(f"Received ACK {seq_num}")
    if (seq_num < self.lhs or seq_num > self.rhs):
        return
    if (not self.acked[seq_num]):
        self.acked[seq_num] = True
        self.acked_num += 1
    if (seq_num == self.lhs):
        while (self.lhs <= self.num and self.acked[self.lhs]):
            self.lhs += 1
        if self.lhs == self.num + 1:
            self.end_time = time.time()
            self.lhs_time = time.time()

```

如果rhs还能向右移动（不超过窗口大小），就发送一个rhs位置的包并将rhs右移

```
def create_packet(self, seq_num):
    eth = Ethernet(src=blaster_eth, dst=middlebox_eth0, ethertype=EtherType.IPv4)
    ip = IPv4(src=blaster_ip, dst=self.blas teeIp, protocol=IPProtocol.UDP,
ttl=64)
    udp = UDP()
    payload = bytes([randint(0, 255) for _ in range(self.length)])
    length = len(payload)
    raw_data = struct.pack('!IH', seq_num, length) + payload
    raw_packet = RawPacketContents(raw_data)
    pkt = eth + ip + udp + raw_packet
    return pkt

def handle_no_packet(self):
    # log_info(f"lhs: {self.lhs}, rhs: {self.rhs}")
    log_info(f"{list(range(self.lhs, self.rhs + 1))}")
    self.handle_timeout()
    if self.rhs < self.num and self.rhs - self.lhs + 1 < self.senderwindow:
        self.rhs += 1
        if self.rhs == 1:
            self.begin_time = time.time()
        pkt = self.create_packet(self.rhs)
        log_info(f"Sending packet{self.rhs}")
        self.net.send_packet("blaster-eth0", pkt)
```

如果lhs一直停在一个位置，就把窗口内没ack过的包全部重发

```
def handle_timeout(self):
    if(time.time() - self.lhs_time > self.timeout/1000.0):
        log_info("Timeout")
        self.timeout_num += 1
        for i in range(self.lhs, self.rhs + 1):
            if(not self.acked[i]):
                pkt = self.create_packet(i)
                log_info(f"Resending packet{i}")
                self.net.send_packet("blaster-eth0", pkt)
                self.resend_num += 1
```

最后全都ack掉了就结束，输出统计信息

```
def shutdown(self):
    self.net.shutdown()
    log_info("-----")
    log_info(f"Blaster finished sending {self.num} packets")
    # log_info(f"begin_time: {self.begin_time}")
    # log_info(f"end_time: {self.end_time}")
    log_info(f"Total time (in seconds): {self.end_time - self.begin_time}")
    log_info(f"Number of resends: {self.resend_num}")
    log_info(f"Number of timeouts: {self.timeout_num}")
    log_info(f"Throughput (Bps): {(self.num + self.resend_num) * self.length / (self.end_time - self.begin_time)}")
    log_info(f"Goodput (Bps):{self.num * self.length / (self.end_time - self.begin_time)}")
    log_info("-----")
```

## 四、实验结果

### mininet测试

The screenshot displays the results of a mininet test involving three nodes: "Node: middlebox", "Node: blaster", and "Node: blaster". The logs show the flow of packets and the completion of the test.

**Node: middlebox logs:**

```
22:14:59 2024/05/25 INFO Using network devices: middlebox-
22:19:40 2024/05/25 INFO Forwarding packet 1
22:19:40 2024/05/25 INFO Forwarding packet 2
22:19:40 2024/05/25 INFO Forwarding ACK 1
22:19:40 2024/05/25 INFO Forwarding packet 1
22:19:40 2024/05/25 INFO Drop packet 2
22:19:40 2024/05/25 INFO Forwarding packet 3
22:19:40 2024/05/25 INFO Forwarding ACK 2
22:19:40 2024/05/25 INFO Forwarding packet 1
22:19:40 2024/05/25 INFO Forwarding packet 2
22:19:40 2024/05/25 INFO Forwarding packet 3
22:19:40 2024/05/25 INFO Forwarding packet 4
22:19:40 2024/05/25 INFO Forwarding ACK 1
22:19:40 2024/05/25 INFO Forwarding ACK 3
22:19:40 2024/05/25 INFO Forwarding packet 1
22:19:40 2024/05/25 INFO Forwarding packet 2
22:19:40 2024/05/25 INFO Forwarding packet 3
22:19:40 2024/05/25 INFO Forwarding packet 4
22:19:40 2024/05/25 INFO Forwarding packet 5
22:19:40 2024/05/25 INFO Forwarding ACK 1
22:19:40 2024/05/25 INFO Forwarding ACK 2
22:19:40 2024/05/25 INFO Forwarding ACK 3
22:19:40 2024/05/25 INFO Forwarding ACK 4
```

**Node: blaster logs:**

```
22:15:46 2024/05/25 INFO Saving iptables state and installin
22:19:40 2024/05/25 INFO Using network devices: blaster-eth0
22:19:40 2024/05/25 INFO Received packet 1
22:19:40 2024/05/25 INFO Received packet 2
22:19:40 2024/05/25 INFO Received packet 1
22:19:40 2024/05/25 INFO Received packet 1
22:19:40 2024/05/25 INFO Received packet 2
22:19:40 2024/05/25 INFO Received packet 3
22:19:40 2024/05/25 INFO Received packet 4
22:19:40 2024/05/25 INFO Received packet 1
22:19:40 2024/05/25 INFO Received packet 2
22:19:40 2024/05/25 INFO Received packet 3
22:19:40 2024/05/25 INFO Received packet 4
22:19:40 2024/05/25 INFO Received packet 5
22:19:40 2024/05/25 INFO Received packet 6
22:19:40 2024/05/25 INFO Received packet 7
22:19:40 2024/05/25 INFO Received packet 8
22:19:40 2024/05/25 INFO Received packet 9
22:19:40 2024/05/25 INFO Received packet 10
22:19:40 2024/05/25 INFO Received packet 11
22:19:40 2024/05/25 INFO Received packet 12
22:19:40 2024/05/25 INFO Received packet 13
```

**Node: blaster logs (continued):**

```
22:19:39 2024/05/25 INFO [
22:19:39 2024/05/25 INFO Sending packet1
22:19:40 2024/05/25 INFO [1]
22:19:40 2024/05/25 INFO Sending packet2
22:19:40 2024/05/25 INFO [1, 2]
22:19:40 2024/05/25 INFO Timeout
22:19:40 2024/05/25 INFO Resending packet1
22:19:40 2024/05/25 INFO Resending packet2
22:19:40 2024/05/25 INFO Sending packet3
22:19:40 2024/05/25 INFO [1, 2, 3]
22:19:40 2024/05/25 INFO Timeout
22:19:40 2024/05/25 INFO Resending packet1
22:19:40 2024/05/25 INFO Resending packet2
22:19:40 2024/05/25 INFO Resending packet3
22:19:40 2024/05/25 INFO Sending packet4
22:19:40 2024/05/25 INFO [1, 2, 3, 4]
22:19:40 2024/05/25 INFO Timeout
22:19:40 2024/05/25 INFO Resending packet1
22:19:40 2024/05/25 INFO Resending packet2
22:19:40 2024/05/25 INFO Resending packet3
22:19:40 2024/05/25 INFO Resending packet4
22:19:40 2024/05/25 INFO Sending packet5
22:19:40 2024/05/25 INFO Received ACK1
22:19:40 2024/05/25 INFO Received ACK2
```

**Packet Capture Analysis:**

The packet capture analysis shows the flow of packets between the nodes. The "Node: blaster" capture shows the final state of the test, including the total time, number of resends, and throughput.

**Node: blaster logs (final summary):**

```
22:19:54 2024/05/25 INFO Sending packet98
22:19:54 2024/05/25 INFO Received ACK95
22:19:54 2024/05/25 INFO Received ACK96
22:19:54 2024/05/25 INFO [97, 98]
22:19:54 2024/05/25 INFO Sending packet99
22:19:54 2024/05/25 INFO Received ACK97
22:19:54 2024/05/25 INFO [98, 99]
22:19:54 2024/05/25 INFO Sending packet100
22:19:54 2024/05/25 INFO Received ACK98
22:19:54 2024/05/25 INFO [99, 100]
22:19:54 2024/05/25 INFO Received ACK99
22:19:54 2024/05/25 INFO [100]
22:19:54 2024/05/25 INFO Received ACK100
22:19:54 2024/05/25 INFO -----
22:19:54 2024/05/25 INFO Blaster finished sending 100 packets
22:19:54 2024/05/25 INFO Total time (in seconds): 14.471242427825928
22:19:54 2024/05/25 INFO Number of resends: 104
22:19:54 2024/05/25 INFO Number of timeouts: 43
22:19:54 2024/05/25 INFO Throughput (Bps): 1409.6923675864903
22:19:54 2024/05/25 INFO Goodput (Bps): 691.0256703855343
22:19:54 2024/05/25 INFO -----
22:19:54 2024/05/25 INFO Restoring saved iptables state
```

共发了100个包，有43次timeout和104个重传，总用时14秒