Robust UDP challenge

陳侑澤

想法

- UDP – Unreliable，需要重傳機制

- Stop and wait
  - 124 packets/ 5 min.
- Go back n
  - 785 packets/ 5 min.
- Selected repeat
  - 1000 packets/ 199 sec.

想法

- Window size 跟packet的數量一樣大，最大化效率
- Loss rate = 40%
  - ACK封包小，多送的代價小，丟的代價大(必須重傳) => 一次送多個封包
- Sending/receiving buffer 開到最大，避免瓶頸
- 不單獨設定timeout，統一timeout

結果

- 參數
  - Packet size: 2500 bytes
  - Sending interval: 3000 μs
  - ACK duplicated count: 3
- 結果
  - 1000 packets/ 198sec
  - 高機率有錯誤
  - Total packet: 7210
  - Send: 62444
  - Receive: 7395

嘗試

- 更改參數
  - Packet size: 2400, 2500, 2600 bytes
  - Sending interval: 3000, 5000, 8000, 10000 μs
  - ACK duplicated count: 5~10
- 結果
  - 延長sending interval可大幅降低封包丟失率，但由於送間隔增加，整體速度下降
  - Packet size影響似乎不大?

嘗試

- 更改參數
  - Packet size: 600 bytes
- 1000 packets/26s
- 原因

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
qdisc netem ae08: dev lo root refcnt 2 limit 1000 delay 100ms  50ms loss 40% corrupt 10% rate 10Mbit
qdisc noqueue 0: dev eth0 root refcnt 2
```

優化

- 壓縮 -4s+0.6s
- Packet size = 1472 -2s
- 調整參數 -2s
  - Sending interval: 270packets / 200ms
  - ACK duplicated count: 5
- 一個packet多個ACK -2s

- 1000 packets/14.8s

Summary

1. Selected repeat
   - Data packet
     - Window size = packet count
     - Packet size = 1470 bytes data + 2 bytes sequence number
     - Sending speed: 270 packets/200ms
     - Not full packet as the identity of last packet
   - ACK packet
     - 5 ACKs in one packet
     - Send 5 packets each time
2. Sending/ receiving buffer size

```
#define MAXLINE 1470 // MTU(1500) - IP header(20) - UDP header(8) - seq.(2)
struct SendPacket {
    unsigned short id{ 0 };
    unsigned char data[MAXLINE];
    bool valid{ false };
    int length;
};
```

```
#define ACKPACKETSIZE 5
struct ACKPacket {
    unsigned short id[ACKPACKETSIZE]{ 0 };
};
```

```
int n = 500 * 1024; // 500KB
setsockopt(serverFd, SOL_SOCKET, SO_RCVBUF, &n, sizeof(n));
setsockopt(serverFd, SOL_SOCKET, SO_SNDBUF, &n, sizeof(n));
```

可能可以再優化的點

- 封包快送完時，效率大幅下降
- 壓縮時間
- 多條UDP連線?
- Bitwise ACK

Code - client

```
27  void sendPacket(int sockfd, const sockaddr* pservaddr, socklen_t servlen) {
28      connect(sockfd, (sockaddr*)pservaddr, servlen);
29
30      // record all data
31      std::map<int, SendPacket> data;
32
33      // file to packets
34      SendPacket packet;
35      FILE* file{ fopen("all.zip", "rb") };
36      int readCount;
37      while ((readCount = fread(packet.data, 1, MAXLINE, file)) != 0) {
38          packet.length = readCount + 2; // 2 is file number
39          data[packet.id] = packet;
40          packet.id++;
41      }
42      // last packet is full
43      if (packet.length == MAXLINE + 2) {
44          packet.length = 2;
45          data[packet.id] = packet;
46      }
47      fclose(file);
48
49      std::size_t packetCount{ data.size() };
50      fprintf(stderr, "[client] packet count: %lu, starting transmit.\n", packetCount);
51
52      ACKPacket ackPacket;
53      int totalSendPacketCount{ 0 };
54      while (packetCount > 0) {
```

Code - server

```
23
24  void receivePacket(int sockfd, sockaddr* pcliaddr, socklen_t clilen, std::vector<unsigned char>& zipData) {
25      std::vector<SendPacket> packetQueue(10000);
26      int firstPacketId{ 0 };
27      int totalReceivePacketCount{ 0 };
28
29      ACKPacket ackPacket;
30      int ACKCount{ 0 };
31
32      SendPacket receivePacket;
33      receivePacket.valid = true;
34      while (1) {
35          socklen_t len{ clilen };
36          int receiveCount;
37          while ((receiveCount = recvfrom(sockfd, &receivePacket, MAXLINE + 2, 0, pcliaddr, &len)) <= 0) {
38          }
39
40          totalReceivePacketCount++;
41
42          ackPacket.id[ACKCount] = receivePacket.id;
43          ACKCount++;
44          if (ACKCount == ACKPACKETSIZE) {
45              ACKCount = 0;
46              for (int i{ 0 }; i < 5; i++)
47                  sendto(sockfd, &ackPacket, sizeof(ackPacket), 0, pcliaddr, len);
48          }
49
50          //  already exist
51          if (packetQueue[receivePacket.id].valid)
52              continue;
53
```

End