Performance results

Transfer time vs error probability

data unit: 50

error probability	transfer time (ms)	Throughput (Kbytes/s)
0	48.709	1227.55
0.01	143.356	417.09
0.05	535.723	111.61
0.1	1048.418	57.03
0.15	1519.931	39.34
0.2	1987.192	30.09
0.25	2484.817	24.06
0.3	2988.496	20.01
0.35	3490.334	17.13
0.4	3963.453	15.08
0.45	4468.471	13.38
0.5	4925.426	12.14
0.6	5879.368	10.17
0.7	6865.727	8.71
0.8	7846.120	7.62
0.9	8805.227	6.79

Transfer time vs data unit size

error probability = 0

data units	transfer time (ms)	Throughput (Kbytes/s)
50	48.709	1227.55
100	24.907	2400.65
200	13.437	4449.88
500	5.789	10328.73
1000	3.829	15615.83
1500	2.906	20575.70
2000	2.247	26610.14
2500	1.807	33089.65
3000	1.535	38953.09
3500	1.299	46030.02
4000	1.365	43804.39
4500	1.491	40102.61
5000	1.080	55363.88

error probability = 0.1

data units	transfer time (ms)	Throughput (Kbytes/s)
50	1048.418	57.03
100	563.812	106.05
200	274.320	217.96
500	114.620	521.66
1000	60.51	988.15

error probability = 0.2

data units	transfer time (ms)	Throughput (Kbytes/s)
50	1987.192	30.09
100	1061.070	56.35
200	550.491	108.61
500	214.000	279.41
1000	105.471	566.91

error probability = 0.3

data units	transfer time (ms)	Throughput (Kbytes/s)
50	2988.496	20.01
100	1646.682	36.31
200	835.489	71.56
500	348.027	171.80
1000	167.564	356.83

Code explain

UDP server

main function

- Two input parameter
  - First is error probability, second is data units.

```
error_pro = atof(argv[1]);  // transfer char to float
DATALEN = atoi(argv[2]);
```

flow control

- Stop-wait flow control:

  After server receive a packet, it will return an ACK.

- Two copies of same frame

  Alternate between ACK0 and ACK1 to solve the above problem

```
if ((flag = memcmp(compare, recvs, n)) == 0)
{
    printf("two same packets\n");
    if ((flag = sendto(sockfd, &ack, 2, 0, (struct sockaddr *)&addr,
len)) == -1)
    {
        printf("send error!");
        exit(1);
    }
    continue;
}
else
{
    memcpy(compare, recvs, n);
    if (ack.num == 1)
        ack.num = 2;
    else
        ack.num =1 ;
}
```

- Error probability

- Calculate how many packet/ACK will be lost:

```
float rand_num = 1.0; //a random number. If it smaller than err_pro, indicator that the packet/ARQ is lost
long data_num = 59793;
float erro_num = data_num/DATALEN * err_pro;
```

- If packet/ACK is lost, ACK won't be transmit

```
// if(rand_num <= err_pro)
if (erro_num > 0 && suc_time >10)
{
    // printf("A random number: %f\nerr_pro is: %f\n",rand_num,err_pro);
    err_times += 1;
    erro_num -= 1;
    continue;
}
```

UDP Client

main function

- Two input parameter

```
if ((sh=gethostbyname(argv[1]))==NULL) {            //get host's information
    printf("error when gethostbyname");
    exit(0);
}

DATALEN = atoi(argv[2]);
```

flow control

- Stop-wait flow control

  After transmit, client will wait for ACK

```
if ((n = sendto(sockfd, &sends, slen, 0, addr, addrlen)) == -1)
{
    printf("send error!\n");
    n = errno;
    printf("error is %d\n",n);
    exit(1);
}
gettimeofday(&rtt_in, NULL);
setsockopt(sockfd, SOL_SOCKET, SO_RCVTIMEO, &rtt_set, sizeof(rtt_set));
if ((n= recvfrom(sockfd, &ack_rec, 2, 0, (struct sockaddr *)&addr_sto, &addrlen_sto))==-1)
{
    n = errno;
    if (n == 11)
    {
        // printf("packet lost, retransmit\n");
```

```
        err_times += 1;
        continue;
    }
    printf("error when receiving\n");
    printf("error is %d\n",n);
    exit(1);
}
```

- Estimate RTT

  Assume that RTT = weight * RTT(original) + (1-weight) * RTT(new). RTT initial is 1ms

```
tv_sub(&rtt_out, &rtt_in);
R0 = (rtt_out.tv_sec)*1000.0 + (rtt_out.tv_usec)/1000.0;
RTT = weight * RTT + (1-weight) * R0;
// printf("new RTT is: %f\nR0 is: %f", RTT,R0);
rtt_set.tv_sec = 0;
rtt_set.tv_usec = RTT * 1000;
```

- ARQ

  Set waiting time. If ACK not received, packet will be retransmit

```
setsockopt(sockfd, SOL_SOCKET, SO_RCVTIMEO, &rtt_set, sizeof(rtt_set));
if ((n= recvfrom(sockfd, &ack_rec, 2, 0, (struct sockaddr *)&addr_sto,
&addrlen_sto))==-1)
{
    n = errno;
    if (n == 11)
    {
        // printf("packet lost, retransmit\n");
        err_times += 1;
        continue;
    }
    printf("error when receiving\n");
    printf("error is %d\n",n);
    exit(1);
}
```