Cs2520 p1 report

University of Pittsburgh

Professor: Dr. Babay

Team member: Chengzhi yong, Shixiang Long

10/01/2021

1. Local area test:

1.1 t_ncp and t_rcv

Loss_rate\ metric	Data_size	Time	rate
0	100 MB	8S	12 Mbps

1.1 ncp and rcv

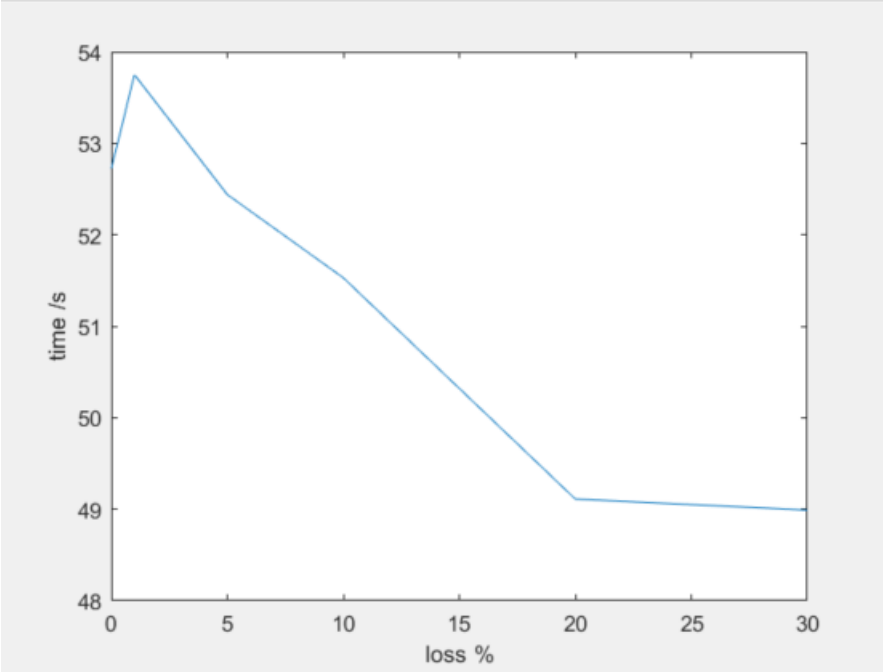
We tried tune the window size from 100~200, the rate is same

TIMEOUT 1S.

We find that Data_size are totally same, so it doesn't resend the data. Therefore, reduce time out time

Loss_rate\ metric	Data_size	Time	Rate(include junk)
0	143.423 M	52.72 s	1.896 Mbps
1%	144.74 M	53.75 s	1.860 Mbps
5%	145.83 M	52.44 s	1.969 Mbps
10%	148.44 M	51.53 s	2.10 Mbps
20%	154.58 M	49.11 s	2.035 Mbps
30%	161.10 M	48.99 s	2.0412 Mbps

Graph:



1.2 ncp/rcv and t_ncp/t_rcv at the same time

program	Data_size	Time	Rate(include junk)
ncp/rcv	136 MB	42.30 s	2.37 Mbps
t_ncp/t_rcv	100 MB	37s	2.1 Mbps

our program will preempt TCP's bandwidth.

2. wide area test

There exists 40 ms between two nodes.

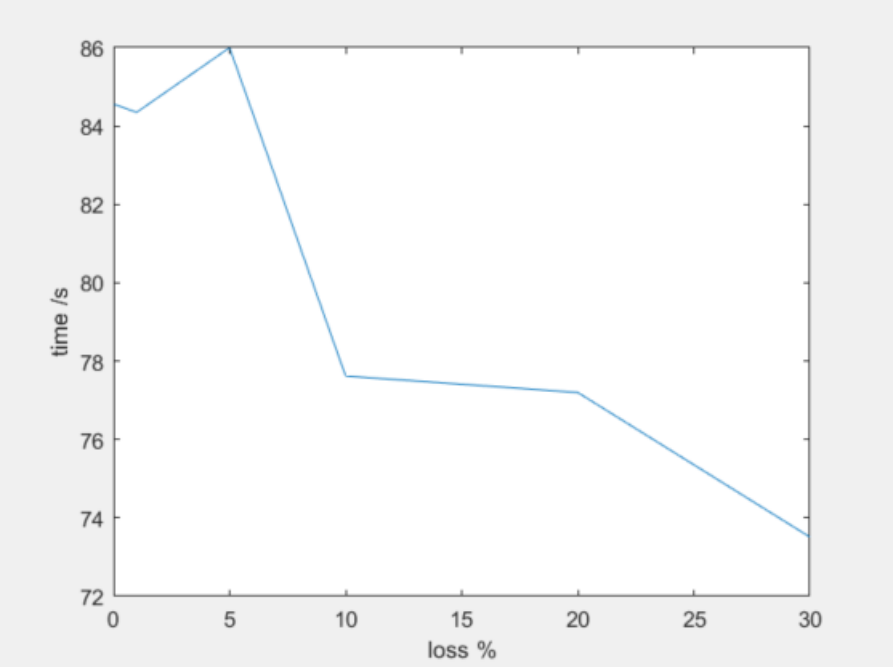
2.1 t_ncp and t_rcv

Loss_rate\ metric	Data_size	Time	rate
0	100M	9S	11 M/s

1.2 ncp and rcv

Loss_rate\ metric	Data_size	Time	rate
0	146.792 M	84.56 s	1.182 M/s
1%	146.41 M	84.35 s	1.185 M/s
5%	150.00 M	81.86 s	1.221 M/s
10%	151.69 M	77.62 s	1.288 M/s
20%	158.00 M	77.20 s	1.295 M/s
30%	164.26 M	73.51 s	1.36 M/s

Graph:



1.2 ncp/rcv and t_ncp/t_rcv at the same time

program	Data_size	Time	Rate(include junk)
ncp/rcv	150.50MB	85.328 s	1.17 Mbps
t_ncp/t_rcv	100 MB	91s	1.6 Mbps

our program will preempt TCP's bandwidth.

The way to tune the parameters:

For timeout, we set timeout = 200. Since RTT = 40, I think 120 is a appropriate timeout on the sender side. Actually, later, I choose 1s to test the program, the result is same, which is weird. For the window size, I tried different window size from 50 to 200. The best window size is 100. Actually, 200 and 100 have the same performance. I think 100 is the maximum size.

Besides, there is another timeout on the receiver side. The problem we faced was that when the sender sends 1,2,3···.60 packages, the order is different. Maybe the receiver would receive 60 first, so it immediately send NACK 1 2 3···5 to the sender. However, the packages are just in flight, which are not lost. In this case, the sender receives the NACK, and resend the 1 2 3···5, it wastes the bandwidth but not have any contribution of the transmitting. So we regulate that if the receiver timeout, it send the ack and NACK to the sender. This timeout is 60, which is 1.5*RTT(40ms). Besides, we tried 70 120, the results are same.

The result is interesting. You could see the data_size increase along with the loss increasing. It is understandable, since the sender should resend more packages. However, the funny thing is that the transmitting rate increase and the time decrease, which is totally different from my understanding. We spend many hours but can't understand why.

Besides, from the contrast between TCP and our program running at the same time, we find that our program would preempt TCP's bandwidth.