华中科技大学
计算机网络实验报告

实验名称 lab03-TCP Congestion 性能测试分析

姓　名	专业	学　　号	贡献百分比	得 分
王科俨	人工智能与自动化	M202072949	100%	

注：团队成员贡献百分比之和为 1

教师评语：

一. 环境（详细说明实验运行的操作系统，网络平台，机器的配置）

操作系统：SEEDUbuntu-16.04-32bit

网络拓扑模拟：GNS3

虚拟机：VirtualBox

网络性能测试工具：iperf

二. 实验目的

1. 掌握基础 ip 网络配置；

2. 选择 TCP 的三种支持的三种拥塞控制算法 cubic、reno、westwood，通过改变 Seed-Router 的延迟和丢包，来测试三种 TCP 拥塞控制算法在不同情况下（延迟、丢包）的性能。

3. 深入理解不同 TCP 拥塞控制算法思路及原理。

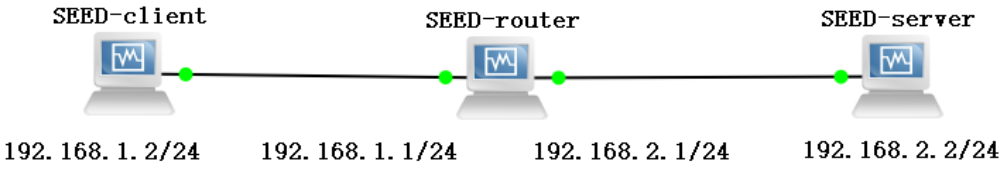
三. 实验步骤（包括主要流程和说明）

I. 使用 VirtualBox 创建虚拟机

新建一个虚拟机，名称为 SEED-client，类型为 Linux，版本选择 Ubuntu(32-bit)，内存 1G，使用实验提供的虚拟硬盘文件创建。使用 VirtualBox 的完全复制功能复制两个虚拟机，分别命名为 SEED-router 和 SEED-server。

II. 使用 GNS3 搭建网络拓扑结构

打开 GNS3，新建一个名称为 Lab03 的项目，在首选项中找到 VirtualBox VMs 处，通过 New 新建 SEED-client、SEED-router 和 SEED-server 模版，修改每个模板的网卡数量为 3，然后搭建如下的网络拓扑结构（SEED-client 与 SEED-router 的 eth1 网卡相连，SEED-router 和 SEED-server 的 eth2 网卡相连，3 个虚拟机的 eth0 网卡均在 VirtualBox 中设置为网络地址转换(NAT)以便访问互联网下载实验所需工具）：



III. 配置虚拟机 IP 地址及路由

搭建完网络拓扑结构后，运行虚拟机，使用

sudo ip address add {NETWORK/MASK} dev {ADAPTOR}

按下表配置 IP 地址：

虚拟机	IP 地址	网卡
SEED-client	192.168.1.2/24	eth1
SEED-router	192.168.1.1/24	eth1
	192.168.2.1/24	eth2
SEED-server	192.168.2.2/24	eth2

在 SEED-client 上输入命令

$ sudo ip route add 192.168.2.0/24 dev eth1 via 192.168.1.1

同理在 SEED-server 上输入命令

$ sudo ip route add 192.168.1.0/24 dev eth2 via 192.168.2.1

此时 SEED-client 路由表如下：

```
169.254.0.0/16 dev eth1  scope link  metric 1000
192.168.1.0/24 dev eth1  proto kernel  scope link  src
192.168.1.2
192.168.2.0/24 via 192.168.1.1 dev eth1
```

SEED-server 路由表如下：

```
169.254.0.0/16 dev eth2  scope link  metric 1000
192.168.1.0/24 via 192.168.2.1 dev eth2
192.168.2.0/24 dev eth2  proto kernel  scope link  src
192.168.2.2
```

在 SEED-router 上输入命令:

```
$ echo "1"> /proc/sys/net/ipv4/ip_forward
```

开启转发功能。

在 SEED-client 上输入命令：

```
$ ping 192.168.2.2
```

结果如下:

```
PING 192.168.2.2 (192.168.2.2) 56(84) bytes of data.
64 bytes from 192.168.2.2: icmp_seq=1 ttl=63 time=1.53
ms
64 bytes from 192.168.2.2: icmp_seq=2 ttl=63 time=1.35
ms
64 bytes from 192.168.2.2: icmp_seq=3 ttl=63 time=0.996
 ms
64 bytes from 192.168.2.2: icmp_seq=4 ttl=63 time=1.22
ms
64 bytes from 192.168.2.2: icmp_seq=5 ttl=63 time=1.88
ms
64 bytes from 192.168.2.2: icmp_seq=6 ttl=63 time=1.53
ms
```

至此 3 台虚拟机可以相互连接。

检查 3 台虚拟机的 eth0 网卡是否设置为 NAT，没有的话在 VirtualBox 中设置。

在 SEED-client 中输入

```
$ ip address
```

结果如下:

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdi
sc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:e6:1d:8e brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynam
ic eth0
       valid_lft 84847sec preferred_lft 84847sec
    inet6 fe80::2c09:c3ed:90fa:9ff0/64 scope link
       valid_lft forever preferred_lft forever
```

```
$ ping www.baidu.com
```

```
PING www.a.shifen.com (182.61.200.6) 56(84) bytes of da
ta.
64 bytes from 182.61.200.6: icmp_seq=1 ttl=50 time=21.5
 ms
64 bytes from 182.61.200.6: icmp_seq=2 ttl=48 time=21.2
 ms
64 bytes from 182.61.200.6: icmp_seq=3 ttl=48 time=21.7
 ms
64 bytes from 182.61.200.6: icmp_seq=4 ttl=48 time=21.9
 ms
64 bytes from 182.61.200.6: icmp_seq=5 ttl=48 time=22.1
 ms
```

可以访问外部网络，在 SEED-client 和 SEED-server 上输入命令:

```
$ sudo apt-get install iperf
```

安装网络性能测试工具 iperf，至此环境搭建完毕。

IV. cubic 算法测试

通过如下命令将 SEED-client 和 SEED-server 拥塞算法设置为 cubic

```
$ sysctl -w net.ipv4.tcp_congression_control=cubic
```

在 SEED-server 端运行下面命令，让 iperf 服务器端守护在 5001 端口:

```
$ iperf -s
```

```
[12/20/20]seed@VM:~$ sudo sysctl -w net.ipv4.tcp_conges
tion_control=cubic
net.ipv4.tcp_congestion_control = cubic
[12/20/20]seed@VM:~$ sudo iperf -s
------------------------------------------------------
-----
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------
-----
```

在 SEED-client 端运行下列命令，让 iperf 客户端运行，同时利用 Linux 内核的 tcp probe 模块监控特定连接中参数变化:

```
$ modprobe tcp_probe port=5001
```

//对端口 5001 的 tcp 连接进行监控

```
$ cat /proc/net/tcpprobe > data.txt &
```

//后台将 tcpprobe 捕捉信息写入 data.txt 中

$ pid=$!

//保存上一个读命令 pid,用于结束 tcpprobe 接口

$ iperf -c 192.168.2.2 -t 15

//使用 iperf 建立一个 TCP 流

$ kill $pid

```
[  3] local 192.168.1.2 port 55884 connected with 192.1
68.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-15.0 sec   816 MBytes   456 Mbits/sec
```

在 SEED-router 上输入如下命令，增加 0.5%的丢包

$ tc qdisc add dev eth2 root netem loss 0.5%

结果如下:

```
[  3] local 192.168.1.2 port 55886 connected with 192.1
68.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-15.0 sec   728 MBytes   407 Mbits/sec
```

修改丢包率为 2.5%结果如下:

结果如下:

```
[  3] local 192.168.1.2 port 55888 connected with 192.1
68.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-15.1 sec   238 MBytes   132 Mbits/sec
```

删除丢包设置后，在 SEED-router 上输入如下命令(modprobe tcp_probe 时加入 full=1 参数否则无法记录数据):

$ tc qdisc add dev eth2 root netem delay 10ms 3ms

增加 10±3ms 的时延，结果如下:

```
[  3] local 192.168.1.2 port 55890 connected with 192.1
68.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-15.0 sec   254 MBytes   142 Mbits/sec
```

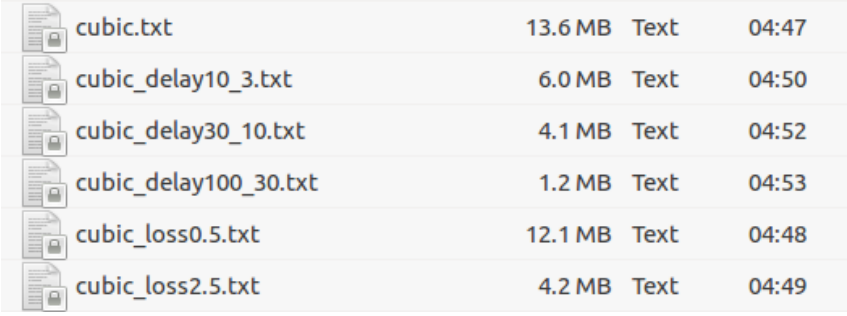
修改时延为 30±10ms，结果如下:

```
[  3] local 192.168.1.2 port 55892 connected with 192.1
68.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-15.0 sec   165 MBytes  92.1 Mbits/sec
```

修改时延为 100±30ms，结果如下:

```
[  3] local 192.168.1.2 port 55894 connected with 192.1
68.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-15.1 sec  44.8 MBytes  24.8 Mbits/sec
```

记录数据文件如下:



V. reno 算法测试

在 SEED-client 和 SEED-server 上输入命令:

```
$ sysctl -w net.ipv4.tcp_congestion_control=reno
```

将拥塞控制算法设置为 reno

```
[  3] local 192.168.1.2 port 55872 connected with 192.1
68.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-15.0 sec   817 MBytes   457 Mbits/sec
```

加上 0.5%丢包率:

```
[  3] local 192.168.1.2 port 55874 connected with 192.1
68.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-15.0 sec  729 MBytes   408 Mbits/sec
```

加上 2.5%丢包率:

```
[  3] local 192.168.1.2 port 55876 connected with 192.1
68.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-15.2 sec   294 MBytes   162 Mbits/sec
```

加上 10±3ms 的时延：

```
[  3] local 192.168.1.2 port 55878 connected with 192.1
68.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-15.0 sec   214 MBytes   120 Mbits/sec
```

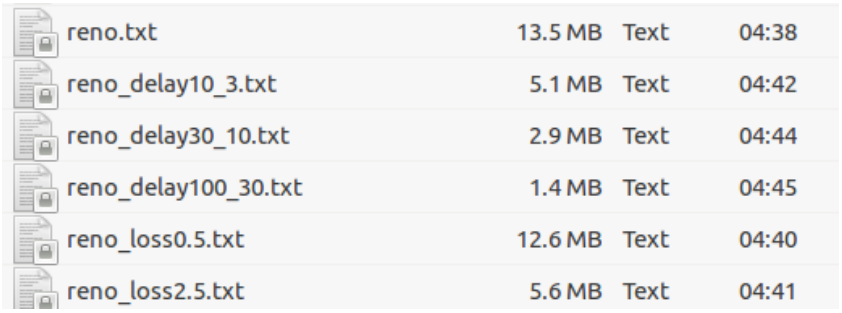
加上 30±10ms 的时延：

```
[  3] local 192.168.1.2 port 55880 connected with 192.1
68.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-15.0 sec   113 MBytes  63.0 Mbits/sec
```

加上 100±30ms 的时延：

```
[  3] local 192.168.1.2 port 55882 connected with 192.1
68.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-15.0 sec  56.5 MBytes  31.5 Mbits/sec
```

记录数据文件如下：



VI. westwood 算法测试

在 SEED-client 和 SEED-server 上输入命令：

```
$ echo "westwood" > /proc/sys/net/ipv4/tcp_congestion_control
```

将拥塞控制算法设置为 westwood

```
[  3] local 192.168.1.2 port 55864 connected with 192.1
68.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-15.0 sec   811 MBytes   454 Mbits/sec
```

加上 0.5%丢包率：

```
[  3] local 192.168.1.2 port 46174 connected with 192.1
68.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-15.0 sec   804 MBytes   450 Mbits/sec
```

加上 2.5%丢包率：

```
[  3] local 192.168.1.2 port 46176 connected with 192.1
68.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-15.0 sec   788 MBytes   441 Mbits/sec
```

加上 10±3ms 的时延：

```
[  3] local 192.168.1.2 port 55866 connected with 192.1
68.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-15.0 sec   290 MBytes   162 Mbits/sec
```

加上 30±10ms 的时延：

```
[  3] local 192.168.1.2 port 55868 connected with 192.1
68.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-15.0 sec   186 MBytes   104 Mbits/sec
```

加上 100±30ms 的时延：

```
[  3] local 192.168.1.2 port 55870 connected with 192.1
68.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-15.1 sec  72.5 MBytes  40.4 Mbits/sec
```

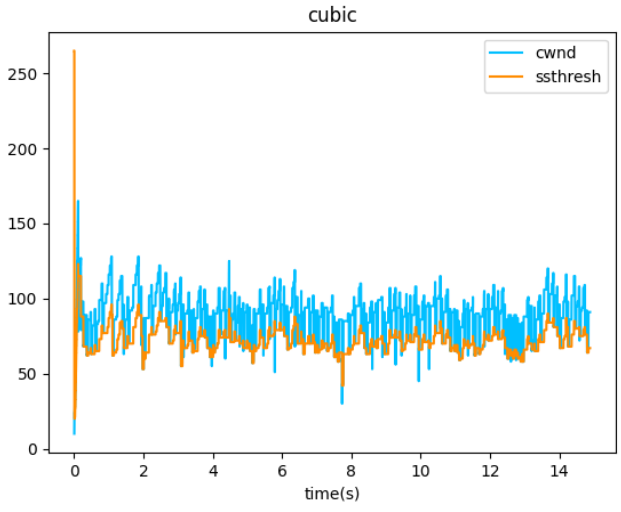
记录数据文件如下：

westwood.txt	14.4 MB	Text	04:29
westwood_delay10_3.txt	6.8 MB	Text	04:31
westwood_delay30_10.txt	4.6 MB	Text	04:32
westwood_delay100_30.txt	1.8 MB	Text	04:33
westwood_loss0.5.txt	14.0 MB	Text	04:23
"cubic_delay100_30.txt" selected (1.2 MB)	14.2 MB	Text	04:24

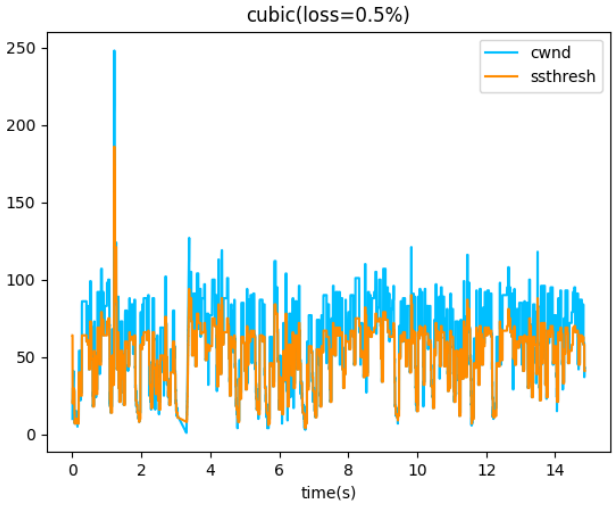
四. 实验结果和分析

I. cubic 算法数据及图表

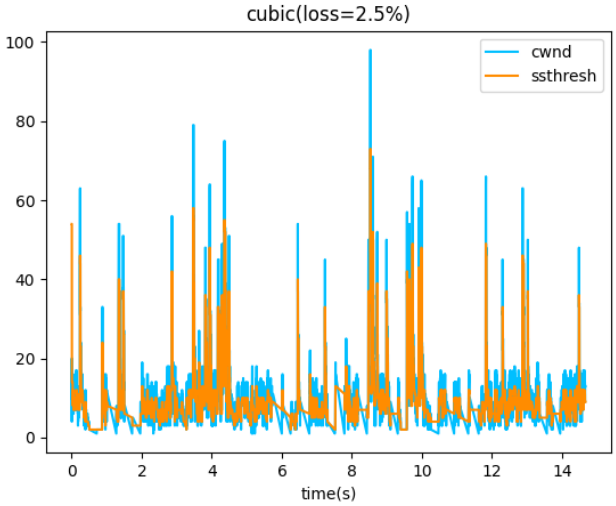
正常情况下：



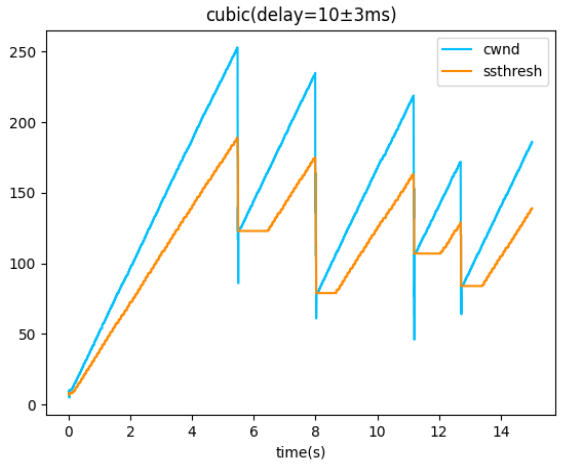
loss=0.5%时：



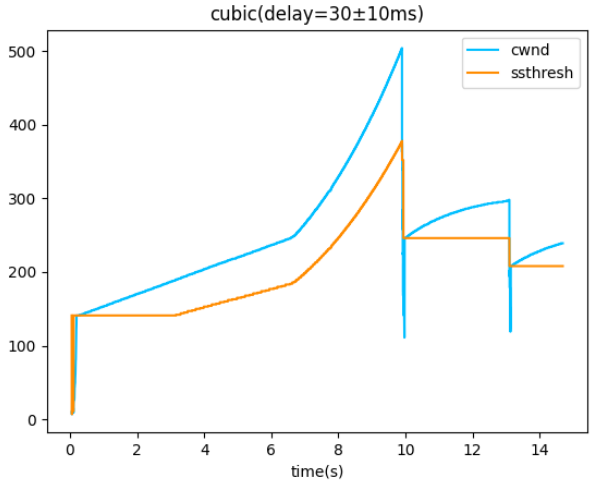
loss=2.5%时：



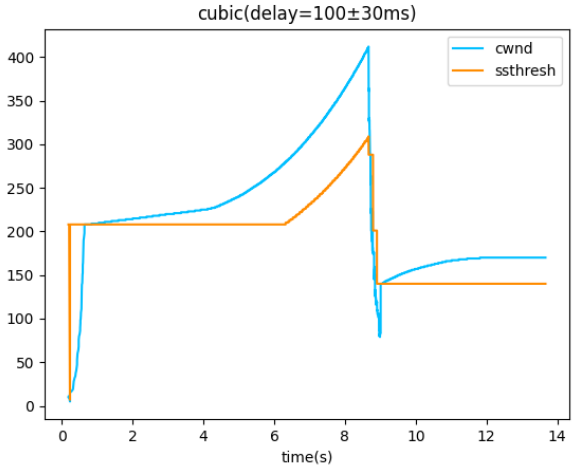
delay=10±3ms:



delay=30±10ms:

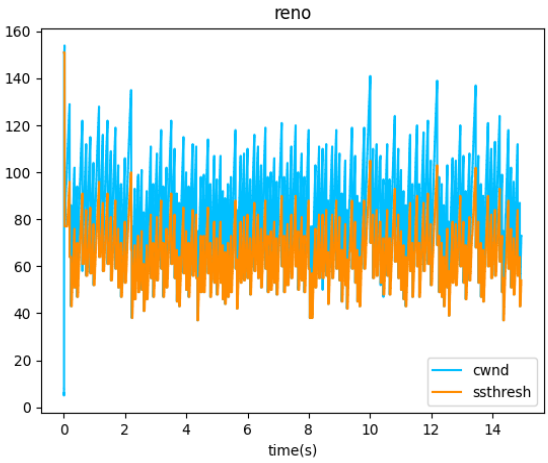


delay=100±30ms:

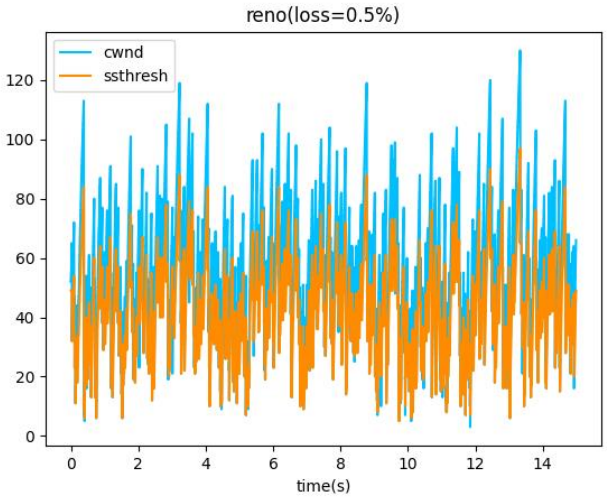


II. reno 算法数据及图表

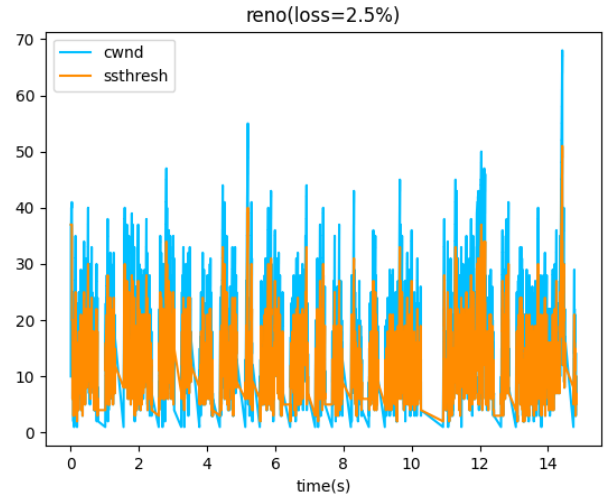
正常情况下：



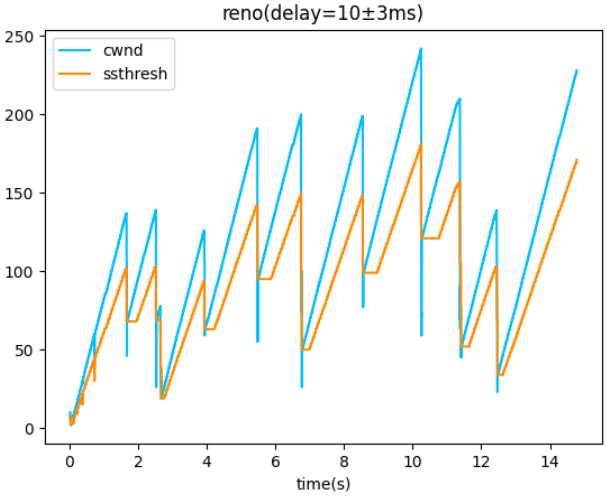
loss=0.5%：



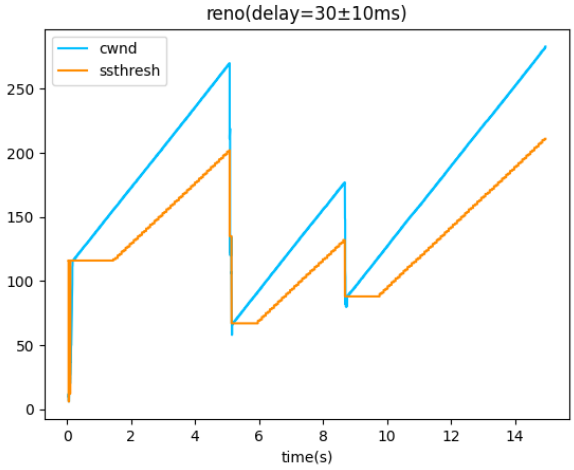
loss=2.5%：



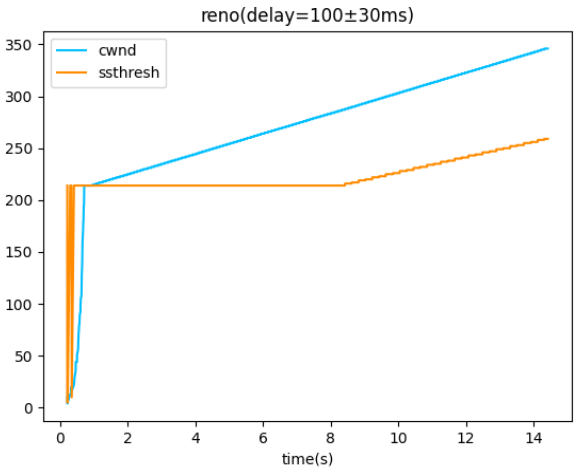
delay=10±3ms:



delay=30±10ms:

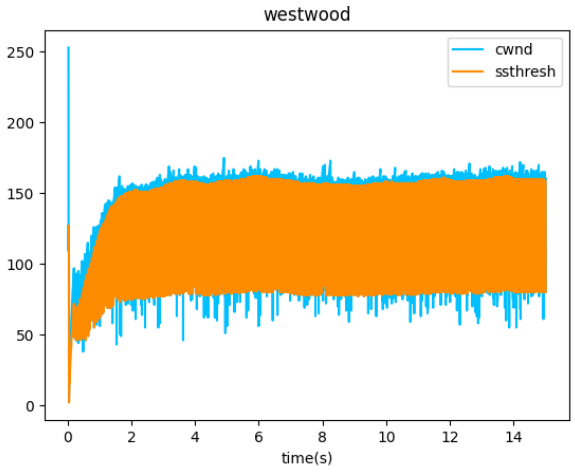


delay=100±30ms:

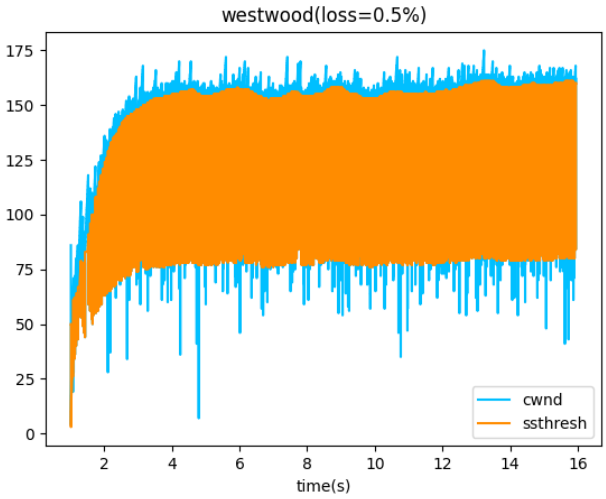


III. westwood 算法数据及图表

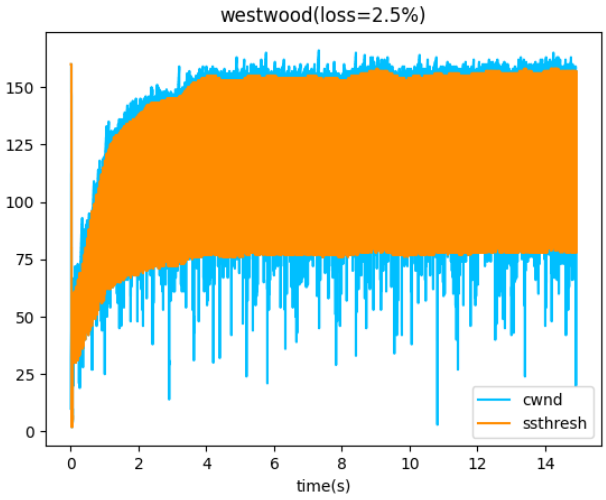
正常情况下：



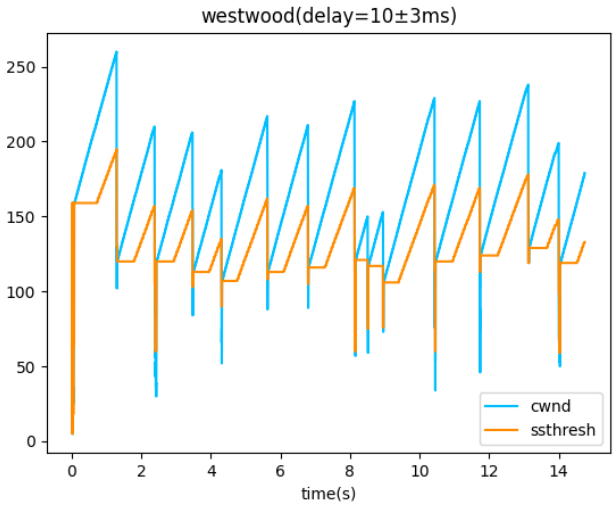
loss=0.5%：



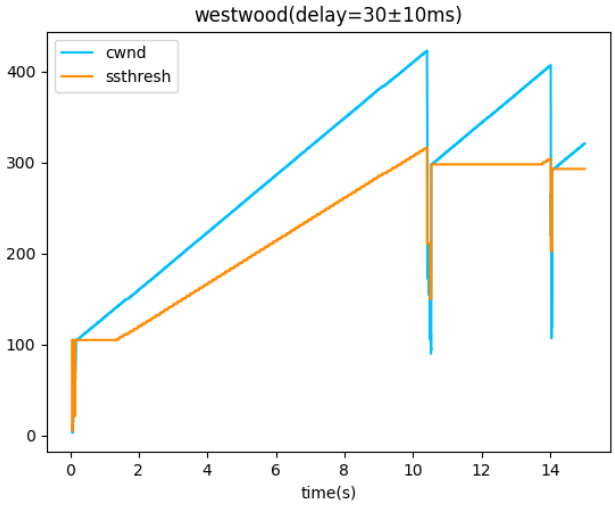
loss=2.5%：



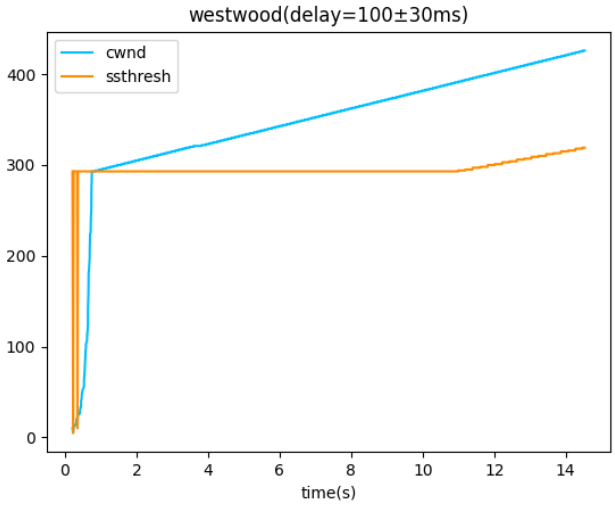
delay=10±3ms:



delay=30±10ms:



delay=100±30ms:

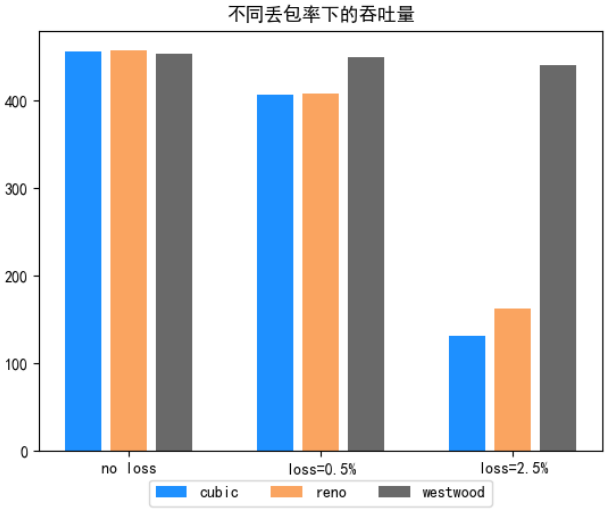


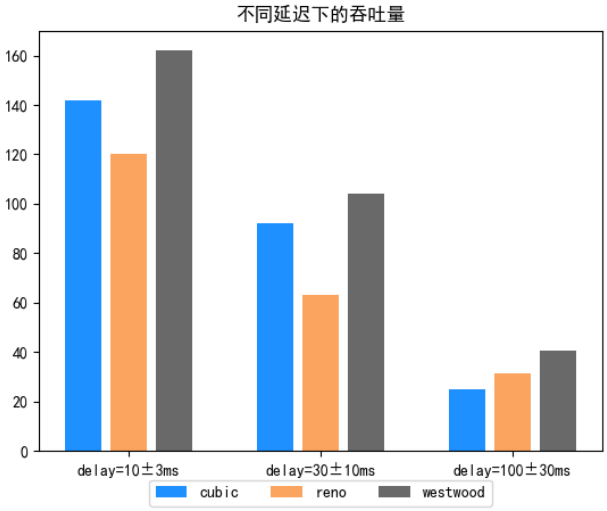
IV. 3 种拥塞控制算法比较

不同丢包率和延迟时间下的吞吐率如下表所示：

拥塞控制算法	正常情况(Mbit/s)	loss=0.5%	loss=2.5%	delay=10±3ms	delay=30±10ms	delay=100±30ms
cubic	456	407	132	142	92.1	24.8
reno	457	408	162	120	63	31.5
westwood	454	450	441	162	104	40.4

绘制如下图所示柱状图：





可以看出在不设置丢包以及丢包率为 0.5%时，3 种算法的吞吐量差不多，westwood 在 0.5%的丢包率下略胜一筹，而在 2.5%丢包率的情况下，cubic 和 reno 算法的吞吐量大幅下降，westwood 算法小幅度下降，综合来看 westwood 算法在丢包时表现较好。

随着时延的增加，3 种算法均有大幅下降，但综合来看 westwood 算法最优，在时延较低时 cubic 算法优于 reno 算法，而在时延比较高时，reno 算法略优于 cubic 算法。