گزارش کار پروژه NS2

سید علی امام زاده ۸۱۰۱۹۹۳۷۷، محمدحسین عقیلی ۸۱۰۱۹۹۵۷۶

مقدمه

نحوه کارکرد ۳ پروتوکل newreno,vegas,tahoe :

Tahoe:

Loss Recovery : تاهو از یک مکانیسم اساسی برای بازیابی از دست دادن بسته استفاده می کند. وقتی بسته ای لاست می شود، تاهو آن را نشانه ازدحام شبکه می داند و باعث ارسال مجدد سریع می شود. سپس وارد یک دوره زمانی می شود که در طی آن اندازه پنجره ازدحام خود را به میزان قابل توجهی کاهش می دهد.

Congestion Avoidance: پس از دوره تایم اوت، تاهو وارد فاز شروع آهسته می شود و به تدریج اندازه پنجره ازدحام را افزایش می دهد تا به سطح قبل از ضرر برسد. سپس به مرحله congestion avoidance منتقل می شود.

Vegas:

Loss Detection : وگاس TCP بر روی تشخیص تراکم با نظارت بر زمان رفت و برگشت (RTT) و حفظ برآورد اندازه پنجره تراکم ایده آل تمرکز می کند. از تفاوت بین RTT واقعی و مورد انتظار برای تشخیص تراکم شبکه استفاده می کند.

Congestion Avoidance: هدف وگاس این است که شبکه را در نقطه بهینه نگه دارد، نه خیلی کمتر از ظرفیت و نه باعث تراکم شود. هنگامی که تراکم را تشخیص می دهد، اندازه پنجره خود را به نصف کاهش می دهد تا مشکل را کاهش دهد. این روش به وگاس کمک می کند تا بدون تجربه نوسانات بزرگ، حالت ثابت خود را حفظ کند.

:NewReno

بهبود یافته از الگوریتم Tahoe است. با پرداختن به موضوعی که به عنوان "مشکل جزئی ACK" شناخته می شود، مکانیسم بازیابی ضرر را بهبود می بخشد.

همینطور بازیابی سریع دارد و هنگامی که NewReno یک ACK جزئی (تأیید برای بسته ای با شماره توالی بالاتر از آخرین بسته تایید شده) دریافت می کند، چنین استنباط می کند که برخی از بسته ها با موفقیت دریافت شده اند.

NewReno به جای وارد شدن به یک بازه زمانی مانند tahoe، یک ارسال مجدد سریع انجام می دهد و به مرحله اجتناب از تراکم باز می گردد، و از تأخیر غیرضروری تایم اوت جلوگیری می کند.

توضیح بخش های مهم فایل های pyhon, tcl :

در این بخش ابتدا دو رنگ را برای شبیه سازی ست میکنیم.

```
$ns color 1 Blue
$ns color 2 Red
```

فایل خروجی را میسازیم و یک عدد رندوم برای دیلی از بین ۵ تا ۲۵ در نظر خواهیم گرفت(برای رنو و وگاس هم به همین شکل):

```
set namefile [open tahoe.nam w]
$ns namtrace-all $namefile
set tracefilel [open tahoeTrace.tr w]
$ns trace-all $tracefilel  ;

set rnd [new RNG]
$rnd seed 2

set randVar [new RandomVariable/Uniform]
$randVar use-rng $rnd
$randVar set min_ 5.0
$randVar set max_ 25.0

set rndVar1 [$randVar value]
set rndVar2 [$randVar value]
```

تعریف ۶ نود:

```
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
```

ارتباطات بین نود ها طبق توپولوژی شبکه:

```
$ns duplex-link $n1 $n3 100Mb 5ms DropTail
$ns duplex-link $n2 $n3 100Mb [expr {double(round(100*$rndVar1))/100}]ms DropTail
$ns duplex-link $n3 $n4 100Kb 1ms DropTail
$ns duplex-link $n4 $n5 100Mb 5ms DropTail
$ns duplex-link $n4 $n6 100Mb [expr {double(round(100*$rndVar2))/100}]ms DropTail
$ns queue-limit $n3 $n4 10
$ns queue-limit $n4 $n3 10
```

ست کردن صف در روتر ها (در لینک خروجی هر روتر):

```
$ns queue-limit $n3 $n4 10
$ns queue-limit $n4 $n3 10
```

در این بخش دو سورس میسازیم و با استفاده از سینک نود ۱ را به ۵ و نود ۲ را به ۴ متصل میکنیم. طبق صورت پروژه ttl را به این دو میدهیم.

همینطور یک سری از مشخصات مانند acl, rtt, cwnd را که برای انالیز به انها نیاز پیدا خواهیم کرد را مشخص میکنیم که در فایل خروجی trace باشند.

برای newreno و vegas هم به همین شکل عمل میکنیم و فقط تفاوت در نوع tcp هست.

```
set source1 [new Agent/TCP]
$source1 set class_ 2
$source1 set ttl_ 64
$ns attach-agent $n1 $source1
set sink1 [new Agent/TCPSink]
```

```
$ns attach-agent $n5 $sink1
$ns connect $source1 $sink1
$source1 set fid_ 1

set source2 [new Agent/TCP]
$source2 set class_ 1
$source2 set ttl_ 64
$ns attach-agent $n2 $source2
set sink2 [new Agent/TCPSink]
$ns attach-agent $n6 $sink2
$ns connect $source2 $sink2
$source2 set fid_ 2

$source1 attach $tracefilel
$source1 tracevar cwnd_
$source1 tracevar ack_
$source1 tracevar rtt_

$source2 attach $tracefilel
$source2 tracevar cwnd_
$source2 tracevar ack_
$source2 tracevar rtt_
```

در اخر هم تابع ۱۰۰۰ ثانیه شبیه سازی را اجرا میکنیم و سپس تابع finish را صدا میزنیم.

```
set myftp1 [new Application/FTP]
$myftp1 attach-agent $source1

set myftp2 [new Application/FTP]
$myftp2 attach-agent $source2

$ns at 0.0 "$myftp2 start"
$ns at 0.0 "$myftp1 start"

$ns at 1000.0 "finish"
```

بررسی فایل انالیز داده ها که توسط پایتون و کتابخانه matplotlib نوشته شده:

تابع هایی برای تشخیص و جداکردن داده ها از فایل های trace تعریف کرده ایم و توجه به شماره نود انها را جدا میکنیم

```
def splitCWND(data):
    cwnds1_5 = [-1] * size
    cwnds2_6 = [-1] * size
    for line in data:
        if "cwnd_" in line:
            if line[1] == '0':
                cwnds1_5[ceil(float(line[0]))] = float(line[6])
            else:
                cwnds2_6[ceil(float(line[0]))] = float(line[6])
    cwnds1_5 = arrange(cwnds1_5, -1)
    cwnds2_6 = arrange(cwnds2_6, -1)
    return cwnds1_5, cwnds2_6

def splitRtt(data):
    rtt1_5 = [-1] * size
    rtt2_6 = [-1] * size
    for line in data:
        if "rtt_" in line:
            if line[1] == '0':
                rtt1_5[ceil(float(line[0]))] = float(line[-1])
            else:
                rtt2_6[ceil(float(line[0]))] = float(line[-1])
    return arrange(rtt1_5, -1), arrange(rtt2_6, -1)


def splitAcks(data):
    acks1_5 = ['none'] * size
    acks2_6 = ['none'] * size
    for line in data:
        if "ack_" in line:
            if line[1] == '0':
                acks1_5[ceil(float(line[0]))] = float(line[-1])
            else:
                acks2_6[ceil(float(line[0]))] = float(line[-1])
    return arrange(acks1_5, 'none'), arrange(acks2_6, 'none')

def splitLost(data):
    lost1_5 = [-1] * size
    lastlost1_5 = 0
    lost2_6 = [-1] * size
    lastlost2_6 = 0
    for line in data:
```

```
        if line[0] == 'd':
            if line[-4][0] =='0':
                lastlost1_5 +=1
                lost1_5[ceil(float(line[1]))] = lastlost1_5
            elif line[-4][0] =='1':
                lastlost2_6 +=1
                lost2_6[ceil(float(line[1]))] = lastlost2_6
    return arrange(lost1_5, -1), arrange(lost2_6, -1)
```

سپس داده ها را با این توابع به متغیر هایمان اضافه میکنیم:

```
def addCwndDatas(newRenoData, vegasData, tahoeData):
    global cwndDict1_5, cwndDict2_6
    newRenoDataCwnd1_5, newRenoDataCwnd2_6 = splitCWND(newRenoData)
    vegasDataCwnd1_5, vegasDataCwnd2_6 = splitCWND(vegasData)
    tahoeDataCwnd1_5, tahoeDataCwnd2_6 = splitCWND(tahoeData)

    for i in range(size):
        cwndDict1_5["newreno"][i] += newRenoDataCwnd1_5[i]
        cwndDict1_5["vegas"][i] += vegasDataCwnd1_5[i]
        cwndDict1_5["tahoe"][i] += tahoeDataCwnd1_5[i]
        cwndDict2_6["newreno"][i] += newRenoDataCwnd2_6[i]
        cwndDict2_6["vegas"][i] += vegasDataCwnd2_6[i]
        cwndDict2_6["tahoe"][i] += tahoeDataCwnd2_6[i]


def addGoodputDatas(newRenoData, vegasData, tahoeData):
    global goodputDict1_5, goodputDict2_6
    newRenoDataAcks1_5, newRenoDataAcks2_6 = splitAcks(newRenoData)
    vegasDataAcks1_5, vegasDataAcks2_6 = splitAcks(vegasData)
    tahoeDataAcks1_5, tahoeDataAcks2_6 = splitAcks(tahoeData)

    for i in range(size):
        goodputDict1_5["newreno"][i] += newRenoDataAcks1_5[i]
        goodputDict1_5["vegas"][i] += vegasDataAcks1_5[i]
        goodputDict1_5["tahoe"][i] += tahoeDataAcks1_5[i]
        goodputDict2_6["newreno"][i] += newRenoDataAcks2_6[i]
        goodputDict2_6["vegas"][i] += vegasDataAcks2_6[i]
        goodputDict2_6["tahoe"][i] += tahoeDataAcks2_6[i]

def addRttDatas(newRenoData, vegasData, tahoeData):
    global rttDict1_5, rttDict2_6
    newRenoDataRtt1_5, newRenoDataRtt2_6 = splitRtt(newRenoData)
    vegasDataRtt1_5, vegasDataRtt2_6 = splitRtt(vegasData)
    tahoeDataRtt1_5, tahoeDataRtt2_6 = splitRtt(tahoeData)
```

```
    for i in range(size):
        rttDict1_5["newreno"][i] += newRenoDataRtt1_5[i]
        rttDict1_5["vegas"][i] += vegasDataRtt1_5[i]
        rttDict1_5["tahoe"][i] += tahoeDataRtt1_5[i]
        rttDict2_6["newreno"][i] += newRenoDataRtt2_6[i]
        rttDict2_6["vegas"][i] += vegasDataRtt2_6[i]
        rttDict2_6["tahoe"][i] += tahoeDataRtt2_6[i]

def addLostDatas(newRenoData, vegasData, tahoeData):
    global lostDict1_5, lostDict2_6
    newRenoDataLost1_5, newRenoDataLost2_6 = splitLost(newRenoData)
    vegasDataLost1_5, vegasDataLost2_6 = splitLost(vegasData)
    tahoeDataLost1_5, tahoeDataLost2_6 = splitLost(tahoeData)

    for i in range(size):
        lostDict1_5["newreno"][i] += newRenoDataLost1_5[i]
        lostDict1_5["vegas"][i] += vegasDataLost1_5[i]
        lostDict1_5["tahoe"][i] += tahoeDataLost1_5[i]
        lostDict2_6["newreno"][i] += newRenoDataLost2_6[i]
        lostDict2_6["vegas"][i] += vegasDataLost2_6[i]
        lostDict2_6["tahoe"][i] += tahoeDataLost2_6[
```

هر کدام از شبیه سازی ها را ۱۰ بار اجرا میکنیم و با تابع average میانگین را محاسبه میکنیم

```
def run():
    for i in range(10):
        print("run Num #",i+1,)
        runOnce()
    Avrage()
```

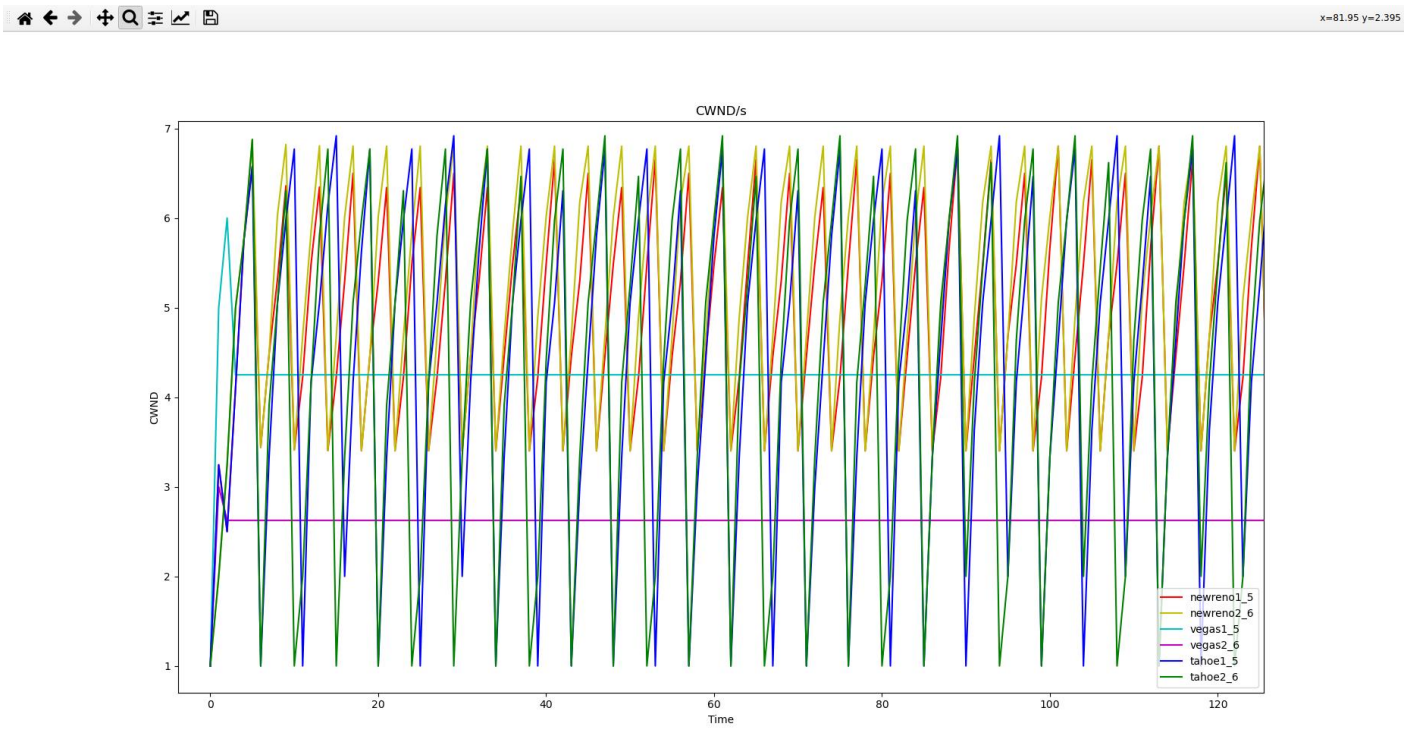
```
def Avrage():
    global cwndDict1_5, cwndDict2_6, goodputDict1_5, goodputDict2_6
    for key in cwndDict1_5:
        for i in range(size):
            cwndDict1_5[key][i] /= 10
            cwndDict2_6[key][i] /= 10
            goodputDict1_5[key][i] /= 10
            goodputDict2_6[key][i] /= 10
            rttDict1_5[key][i] /= 10
            rttDict2_6[key][i] /= 10
            lostDict1_5[key][i] /= 10
```

در اخر هم با matplotlib دیاگرام ها را رسم میکنیم (به همین شکل برای rtt,loss,goodput هم نمودار رسم خواهیم کرد)

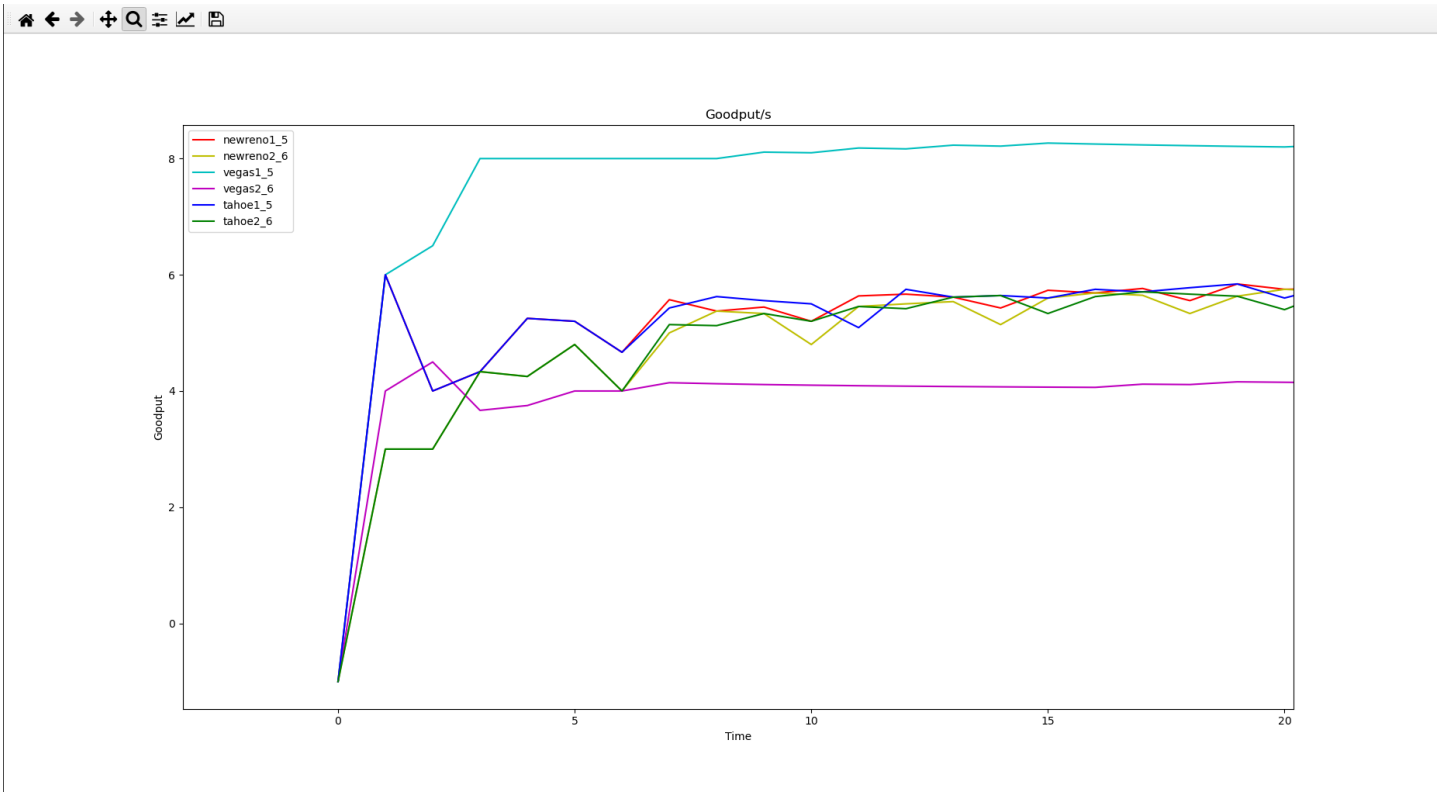
```
def CwndDiagram():
    global cwndDict1_5, cwndDict2_6
    colors = ['g', 'b', 'm', 'c', 'y', 'r']
    for key in cwndDict1_5.keys():
        plt.plot(range(size), cwndDict1_5[key], label=key+'1_5', c=colors[-1])
        colors.pop()
        plt.plot(range(size), cwndDict2_6[key], label=key+'2_6', c=colors[-1])
        colors.pop()

    plt.xlabel("Time")
    plt.ylabel("CWND")
    plt.title("CWND/s")
    plt.legend()
    plt.show()
```

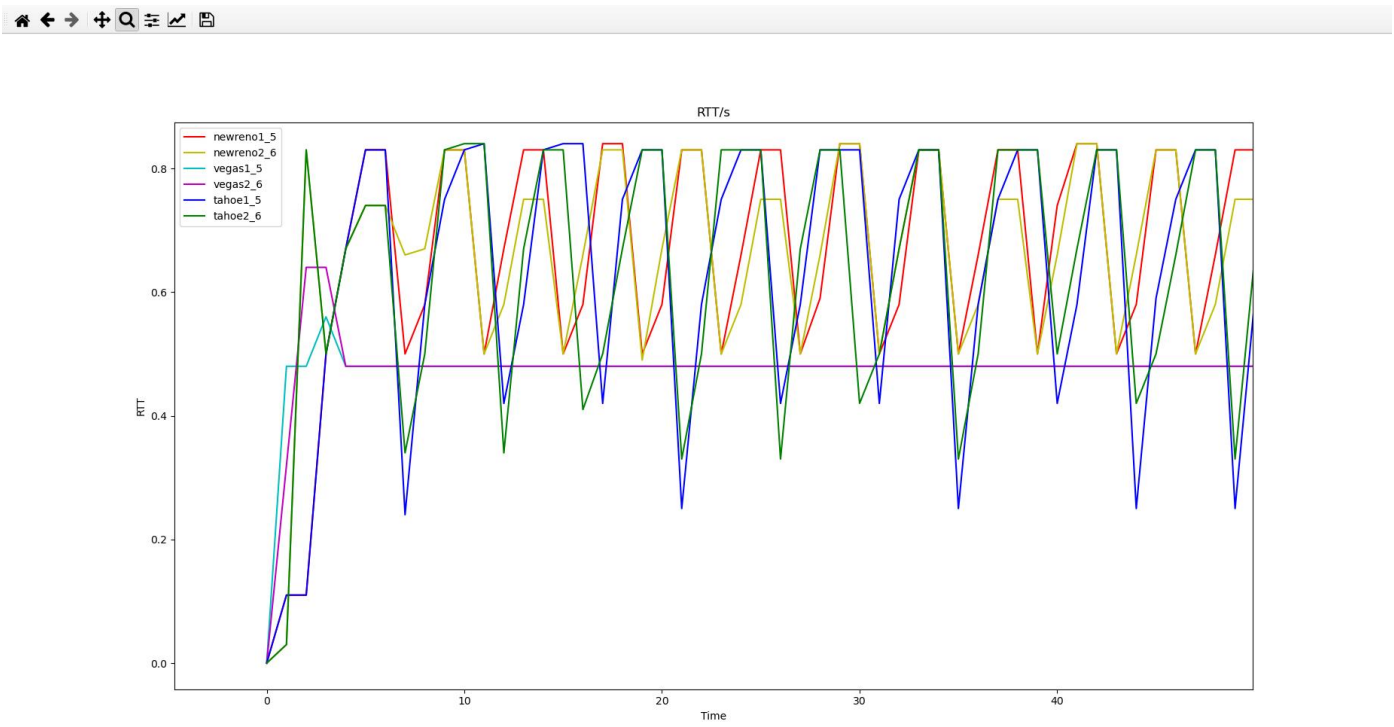
تحلیل خروجی ها:



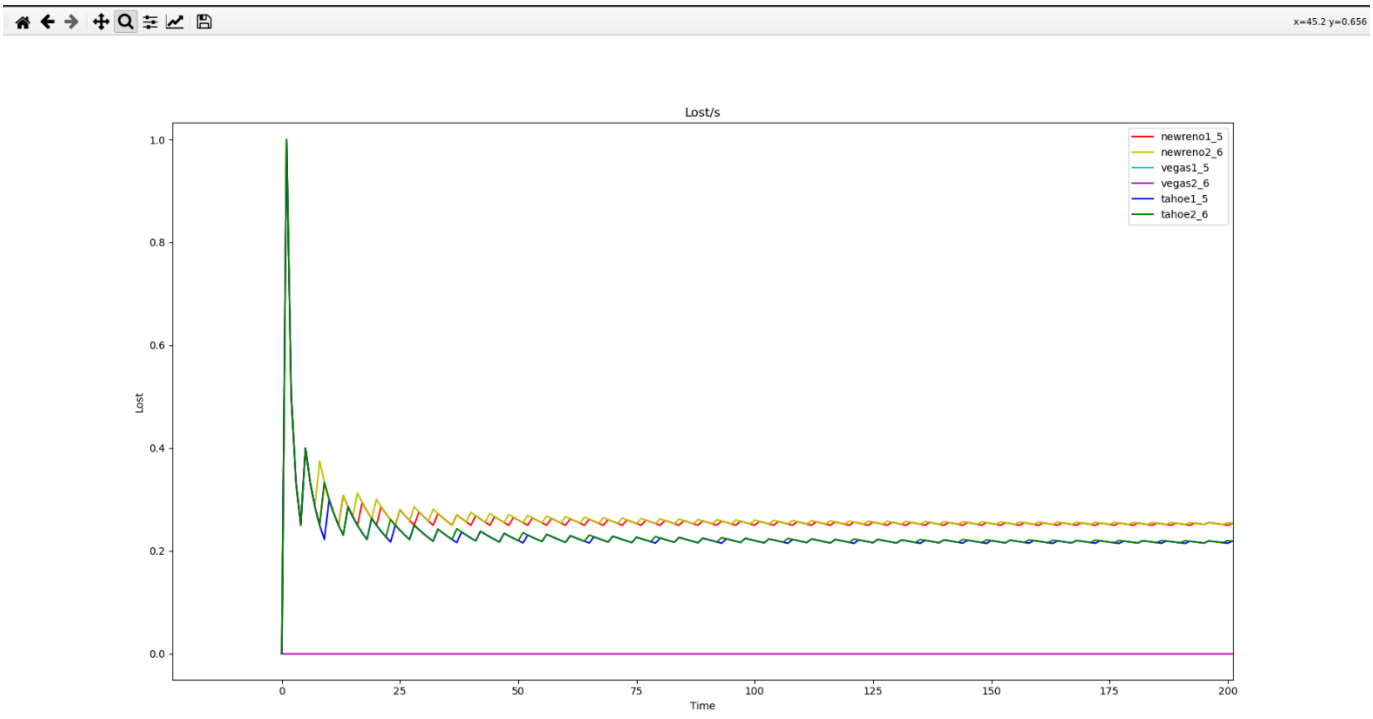
همانطور که مشاهده می شود برای vegas بعد از iteration های اول که محاسبات را انجام داده و میانگین را بدست آورده است، cwnd تقریبا ثابت شده است و برای newreno وtahoe نوسان را دارد.



همانطور که مشاهده می شود Goodput برای vegas متفاوت از newreno و tahoe است.



همانطور که دردیاگرام می بینیم، rtt برای stable vegas شده است ولی در newreno و tahoe نوسان دارد.



همانطور که مشاهده می شود، vegas، packet loss ندارد ولی در newreno و tahoe در ابتدا یک پیک داریم و در ادامه با یک نرخ ثابت ادامه می دهند.